



GT-96100A

Advanced Communication Controller

Datasheet
Revision 1.0
3 October, 2000

Please contact Galileo Technology for possible updates before finalizing a design.

FEATURES

- Integrated communication controller and system controller with PCI interface for high-performance embedded control applications.
- Eight Multi-Protocol Serial Controllers (MPSCs):
 - Support HDLC, BISYNC, UART and Transparent protocols.
 - Bit rate of up to 55Mbit/s on multiple channels simultaneously.
 - Can drive dedicated pins or use TDMs.
 - Dedicated DPLL for clock recovery and data encoding/decoding.
 - Supports NRZ, NRZI, FM0, FM1, Manchester and Differential Manchester.
 - Hardware support for HDLC over asynchronous channel in UART mode.
- Four FlexTDM channels:
 - Time slot assigner for serial and control channels.
 - Supports up to four Basic Rate ISDN interfaces (2B+D) in GCI mode.
 - Fully programmable via dual-port memory.
- Two 10/100Mbps Fast Ethernet MAC controllers:
 - MII/RMII interface.
 - Full duplex and flow-control support.
 - Programmable perfect filtering of 1/2K or 8K MAC addresses (both physical and multicast).
 - 2 Queues for Tx Priority queueing
 - 4 Queues for Priority queueing based on IP DSCP field or 802.1q tag or MAC address.
 - IGMP and BPDU packet trapping.
- Twenty Serial DMA (SDMA) channels to support the communications and Ethernet controllers.
 - Moves data between communications controllers and SDRAM/PCI.
 - Buffer chaining via a linked list of descriptors.
- Eight baud rate generators with multiple clock sources.
- 64-bit CPU bus interface:
 - Supports all 64-bit bus MIPS CPUs: RM5260, RM5270/1 and RM7000 from QED, RV4600 through RV5000 from IDT and R5000 compatibles from various vendors.
 - 100MHz bus frequency.
 - 3.3V bus interface.
 - Support for multiple GT-96100A devices on the same SysAD bus (up to 4).
 - 8x64-bit (64 byte) CPU write posting buffer accepts CPU writes with zero wait-states.
 - CPU address remapping to resources.
 - Zero wait state secondary cache support (L2 of R4xxx and R5000, L3 of R7000).
 - Backward Software Compatibility with GT-64010A, GT-64011 and GT-64120.

- SDRAM controller:
 - 3.3V (5V tolerant).
 - 4GB address space.
 - Supports 16/64/128/256/512Mbit SDRAM devices.
 - Supports 64-bit registered SDRAM.
 - Supports 2-way & 4-way SDRAM bank interleaving.
 - Up to 4GB bank address space, 1MB granularity.
 - 1 to 4 banks supported.
 - 64-bit data width.
 - ECC support for 64-bit SDRAM.
 - Zero wait-state interleaved burst accesses at 100MHz.
 - Supports the VESA Unified Memory Architecture (VUMA) Standard - allows for external masters access to SDRAM directly.
- Device controller:
 - 5 chip selects.
 - Programmable timing for each chip select.
 - Supports many types of standard memory and I/O devices.
 - Up to 4GB address space.
 - Optional external wait-state support.
 - 8-, 16-, 32- and 64-bit width device support.
 - Support for boot ROMs.
- Four Independent DMA (IDMA) channels:
 - Chaining via linked-lists of records.
 - Byte address boundary for source and destination.
 - Moves data between PCI, memory, and devices.
 - Two 64-byte internal FIFOs.
 - Alignment of source and destination addresses.
 - DMAs can be initiated by the CPU writing to a register, external request via DMAReq* pin, or an internal timer/counter.
 - Termination of DMA transfer on each channel.
 - Descriptor ownership transfer to CPU.
 - Fly-By support for local data bus.
 - Override capability of source/destination/record address mapping.

- Two 32-bit or one 64-bit high-performance PCI 2.1 compliant devices:
 - Dual mode PCI interface can be used as two independent 32-bit interfaces (synchronous or asynchronous to each other) or as a single 64-bit interface.
 - 192-bytes of posted write and read prefetch buffers for each PCI interface.
 - 32/64-bit PCI master and target operations.
 - PCI bus speed of up to 66MHz with zero wait states.
 - Universal PCI buffers (each 32-bit PCI use a different voltage).
 - Operates either synchronous or asynchronous to the CPU clock.
 - Burst transfers used for efficient data movement.
 - Doorbell interrupts provided between CPU and PCI.
 - Supports flexible byte swapping through PCI interface.
 - Synchronization barrier support for PCI side.
 - PCI address remapping to resources.
- Host to PCI bridge:
 - Translates CPU cycles into PCI I/O or Memory cycles.
 - Generates PCI Configuration, Interrupt Acknowledge, and Special cycles on PCI bus.
- PCI to Main Memory bridge:
 - Supports fast back-to-back transactions.
 - Supports memory and I/O transactions to internal configuration registers.
 - Supports locked operations.
- I₂O and Plug and Play Support:
 - Industry Standard I₂O messaging unit on primary 32-bit PCI interface (also available in 64-bit mode).
 - Plug and Play compatible configuration registers.
 - PCI configuration header can be loaded from boot PROM.
 - PCI configuration registers are accessible from both CPU and PCI bus.
 - Expansion ROM support.
- PCI Hot-Plug and CompactPCI Hot-Swap capable compliant.
- Two programmable PCI Arbiter functions:
 - Supports up to 9 external agents in addition to PCI_0 and PCI_1 internal devices.
 - Two level priority arbitration capability
 - each request can be assigned either high or low priority.
- Two-stage watchdog timer (NMI, Reset).
- One 32-bit wide timer/counter, Three 24-bit wide timer/counters.
- Eighty-eight pins dedicated for peripheral functions and general purpose I/Os.
 - Each pin can be configured independently as peripheral or General Purpose I/O.
 - Supports simple I/O and LED control.
 - Inputs can generate a maskable interrupt.
- 2.5V Core Supply Voltage, 3.3V I/O Supply Voltage (PCI and Peripherals).
 - All inputs are 5V tolerant.
- JTAG Boundary Scan.
- 492 pin PBGA package.
- Advanced 0.25 micron CMOS process.

Part Number: GT-96100A
Publication Revision: 1.0

©Galileo Technology, Inc.

No part of this datasheet may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of Galileo Technology, Inc.

Galileo Technology, Inc. retains the right to make changes to these specifications at any time, without notice.

Galileo Technology, Inc. makes no warranty of any kind, expressed or implied, with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for any particular purpose. Galileo Technology, Inc. further does not warrant the accuracy or completeness of the information, text, graphics, or other items contained within these materials. Galileo Technology, Inc. makes no commitment to update nor to keep current the information contained in this document.

Galileo Technology, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in Galileo Technology, Inc. products. No other circuit patent licenses are implied.

Galileo Technology, Inc. products are not designed for use in life support equipment or applications in which if the product failed it would cause a life threatening situation. Do not use Galileo Technology, Inc. products in these types of equipment or applications.

Contact your local sales office to obtain the latest specifications before finalizing your product.

Galileo Technology, Inc.
142 Charcot Avenue
San Jose, California 95131
Phone: 1 408 367-1400
Fax: 1 (408) 367-1401
E-mail: info@galileot.com
www.galileoT.com



Other brands and names are the property of their respective owners.

TABLE OF CONTENTS

1. Overview	19
1.1 Communication Unit Description	19
1.2 CPU Interface	21
1.3 SDRAM and Device Interface	21
1.4 PCI Interface	21
1.5 Independent DMA (IDMA) Engines	22
1.6 Peripheral Configurations	23
2. Pin Information	25
3. Address Space Decoding	56
3.1 Two Stage Decoding Process	57
3.2 Disabling Address Decoders	63
3.3 DMA Unit Address Decoding	63
3.4 Address Space Decoding Errors	63
3.5 Default Memory Map	64
3.6 Address Remapping	67
3.7 Using the CPU PCI Override	70
3.8 Using the DMA to PCI Bypass	71
4. CPU Interface Description	72
4.1 CPU Interface Signals	72
4.2 SysAD, SysADC, and SysCmd Buses	73
4.3 Operation of WrRdy* and the Internal Write Posting Queues	79
4.4 CPU Write Modes and Write Patterns Supported	79
4.5 CPU Interface Endianness	80
4.6 Burst Order	80
4.7 MIPS L2 Cache Support	80
4.8 Multiple GT-96100A Support	81
4.9 CPU Interface Restrictions	84
4.10 CPU Interface Control Registers	84
5. Memory Controller	95
5.1 SDRAM Controller	98
5.2 Connecting the Address Bus to the SDRAM	106
5.3 Programmable SDRAM Parameters	108
5.4 SDRAM Performance	110
5.5 SDRAM Bank Interleaving	113
5.6 Unified Memory Architecture (UMA) Support	113
5.7 Device Controller	118
5.8 Programming the ADP lines for other Functions	125
5.9 Memory Controller Restrictions	126
5.10 Registered SDRAM Interface Restrictions	128
5.11 Memory Interface Control Registers	128
6. Data Integrity	143
6.1 SDRAM ECC	143
6.2 PCI Parity Support	147

6.3	Parity Support for Devices	147
6.4	CPU Parity Support	147
6.5	Data Integrity Flow	148
6.6	Register Information	150
6.7	CPU Errors Report Registers	151
7.	PCI Interfaces	152
7.1	Reset Configuration	152
7.2	PCI Master Operation	152
7.3	PCI Target Interface	157
7.4	PCI Synchronization Barriers	161
7.5	PCI Master Configuration	162
7.6	Target Configuration and Plug and Play	163
7.7	PCI Bus/Device Bus/CPU Clock Synchronization	166
7.8	64-bit PCI Configuration	167
7.9	Retry Enable	167
7.10	Locked Cycles	167
7.11	Hot-Swap Support	168
7.12	PCI Power Management Support	168
7.13	PCI Interface Restrictions	169
7.14	PCI Control and Configuration Registers	169
8.	Intelligent I/O (I2O) Standard Support	201
8.1	Overview	201
8.2	I2O Registers	201
8.3	Enabling I2O Support	203
8.4	Register Map Compatibility with the i960Rx Family	203
8.5	Message Registers	203
8.6	Doorbell Registers	204
8.7	Circular Queues	204
8.8	I2O Support Registers	209
9.	Independent DMA Controllers (IDMA Controllers)	219
9.1	DMA Channel Registers	219
9.2	DMA Channel Control Register (0x840 - 0x84c)	221
9.3	Restarting a Disabled Channel	224
9.4	Reprogramming an Active Channel	224
9.5	Arbitration	225
9.6	Current Descriptor Pointer Registers	225
9.7	Design Information	225
9.8	Initiating a DMA from a Timer/Counter	230
9.9	DMA Restrictions	230
9.10	DMA Control Registers	231
10.	PCI Arbiter	241
10.1	Interface	241
10.2	Arbitration Scheme	242
10.3	Arbitration Parking	243
10.4	PCI Arbiter Configuration Register	243

11. Communication Interface Unit (CIU)	246
11.1 CIU Connectivity	247
11.2 Address Decoding and PCI Override (MASTER)	248
11.3 Arbitration Scheme	248
11.4 CIU Arbiter Configuration Register	251
12. 10/100Mb Ethernet Unit	253
12.1 Functional Overview	253
12.2 Port Features	254
12.3 Operational Description	255
12.4 Ethernet Port	275
12.5 Internal Control Registers	283
12.6 Ethernet MIB Counters	301
13. Serial DMA (SDMA)	307
13.1 Overview	307
13.2 SDMA Descriptors	308
13.3 SDMA Configuration Register (SDC)	311
13.4 SDMA Command Register (SDCMx)	313
13.5 SDMA Group Configuration Register	315
13.6 SDMA Descriptor Pointer Registers	316
13.7 Transmit SDMA	316
13.8 Receive SDMA	318
13.9 SDMA Interrupt and Mask register (SDI and SDM)	319
13.10 SDMA in Auto Mode	319
13.11 SDMA Registers	320
14. Multi Protocol Serial Controller (MPSC)	326
14.1 DPLL	326
14.2 MPSCx Main Configuration Register (MMCRx)	328
14.3 MPSCx Protocol Configuration Registers (MPCRx)	337
14.4 Channel Registers (CHxRx)	337
14.5 HDLC Mode	337
14.6 BISYNC Mode	347
14.7 UART Mode	361
14.8 Transparent Protocol	372
15. FlexTDM Units (FTDM)	379
15.1 FlexTDM Architecture	380
15.2 FlexTDM DPRAM	380
15.3 FlexTDM Programing Modes	383
15.4 FlexTDM Configuration Register (TCR)	384
15.5 FlexTDM Synchronization	386
15.6 IOM (GCI) Mode	387
15.7 PCM Highway Mode	388
15.8 Data Rate Adoption	388
15.9 FlexTDM Auxiliary Channels A and B	388
15.10 IOM Programing	391
15.11 FlexTDM Registers	394

16.	Baud Rate Generators (BRGs)	397
16.1	BRG Inputs and Outputs	397
16.2	BRG Baud Tuning	397
16.3	BRG Registers	398
17.	Watchdog Timer	401
17.1	Watchdog Registers	401
17.2	Watchdog Operation	402
18.	Timers/Counters	403
18.1	Timer / Counter Registers	403
19.	General Purpose Ports	405
19.1	Overview	405
19.2	General Purpose Control Registers	405
20.	Physical Signal Routing	414
20.1	Signal Routing	414
20.2	Clock Routing	418
21.	Interrupt Controller	424
21.1	Interrupt Cause Registers	424
21.2	Interrupt Mask Registers	425
21.3	Interrupt Summaries	426
21.4	Interrupt Select Registers	426
21.5	Interrupt Registers Tables	427
22.	Reset Configuration	452
23.	Connecting the Memory Controller to SDRAM and Devices	455
23.1	SDRAM	455
23.2	Devices	456
24.	JTAG Interface	460
24.1	IEEE Standard 1149.1	460
24.2	TAP Controller	460
24.3	Instruction Register (IR)	461
24.4	Bypass Register (BR)	461
24.5	JTAG Scan Chain	461
24.6	ID Register	462
25.	Big and Little Endian	463
25.1	Background	463
25.2	Configuring a System for Big and Little Endian	465
26.	Using the GT-96100A Without the CPU Interface	466
27.	Using the GT-96100A in Different PCI Configurations	467
28.	Phased Locked Loop (PLL) Application Notes	473
28.1	PLL Power Supply	473
28.2	PLL Characteristics	474

29. System Configurations	475
29.1 Minimal System Configuration	475
29.2 Typical System Configuration	476
29.3 High Performance System.	477
30. Register Tables	478
30.1 Access to On-Chip PCI Configuration Space Registers	478
30.2 Register Maps	479
31. DC Characteristics	508
31.1 DC Electrical Characteristics Over Operating Range	509
31.2 Thermal Data.	512
32. AC Timing	513
32.1 TCik/PCIk Restrictions	516
32.2 Serial (Communication) Clock Domain AC Characteristic	518
32.3 MPSC Waveforms	525
32.4 MII Waveforms	529
32.5 JTAG AC Characteristics.	530
32.6 Additional Delay Due to Capacitive Loading	531
33. Pinout Table, 492 Pin BGA	533
34. 492 BGA Package Mechanical Information	542
35. GT-96100A Part Numbering	543
35.1 Standard Part Number.	543
35.2 Valid Part Numbers	543
36. Abbreviations	544
37. Revision History	545

List of Tables

1. Overview	19
Table 1: GT-96100A Serial Performance	20
Table 2: GT-96100A Port Configurations	23
Table 3: GT-96100A Peripheral Configurations	24
2. Pin Information	25
Table 4: CPU Interface Pin Assignments	26
Table 5: Secondary Cache Interface Pin Assignments	26
Table 6: PCI Bus 0 Pin Assignments	27
Table 7: PCI Bus 1 Pin Assignments	29
Table 8: SDRAM and Devices Pin Assignments	31
Table 9: Local Address and Data Bus Pin Assignments	32
Table 10: DMA Pin Assignments	35
Table 11: WAN Pin Assignments	36
Table 12: LAN Pin Assignments	45
Table 13: GPP Pin Assignments	51
Table 14: Interrupt Interface Pin Assignments	53
Table 15: Watchdog Interface Pin Assignments	54
Table 16: Test Interface Pin Assignments	54
Table 17: Clock/Control Interface Pin Assignments	54
3. Address Space Decoding	56
Table 18: CPU and Device Decoder Mappings	57
Table 19: PCI_0 Base Address Register and Device Decoder Mappings	58
Table 20: PCI_1 Base Address Register and Device Decoder Mappings	58
Table 21: CPU and Device Decoder Default Address Mapping	64
Table 22: PCI Function 0 and Device Decoder Default Address Mapping	65
Table 23: PCI Function 1 (Byte Order Swap) and Device Decoder Default Address Mapping	66
Table 24: PCI Address Remapping Example	69
4. CPU Interface Description	72
Table 25: CPU Interface Signals	72
Table 26: SysCmd Bit Summary	73
Table 27: Address Phase SysCmd[8:0] Encodings (driven by CPU)	74
Table 28: Read Response SysCmd[8:0] Encodings (driven by the GT-96100A)	75
Table 29: CPU Write SysCmd[8:0] Encodings (driven by local master)	76
Table 30: SysAD Read Phases	76
Table 31: SysAD Write Phases	78
Table 32: Pin Strapping the GT-96100A ID	81
Table 33: WrRdy*, ValidIn*, and ScDOE* Signal Multiple GT-96100A Functionality	82
Table 34: Initializing a Multiple GT-96100A System	83
Table 35: CPU Interface Register Map	84
5. Memory Controller	95
Table 71: DMAReq*, Ready* and BypOE* Functionality	101

Table 72:	SysAD/PCI Address Decoding for 32-bit SDRAM, 16 Mbit	104
Table 73:	SysAD/PCI Address Decoding for 64-bit SDRAM, 256/512 Mbit	104
Table 74:	SysAD/PCI Address Decoding for 32-bit SDRAM, 64 Mbit	105
Table 75:	SysAD/PCI Address Decoding for 64-bit SDRAM, 64/128 Mbit	105
Table 76:	SysAD/PCI Address Decoding for 64-bit SDRAM, 256 Mbit	106
Table 77:	Programmable SDRAM Parameters	109
Table 78:	CPU SDRAM Performance on Reads	110
Table 79:	Events Determining PCI Read Performance from SDRAM	111
Table 80:	GT-96100A Sync. Modes	111
Table 81:	SDRAM Performance Summary PCI Read Accesses	112
Table 82:	UMA AC Timing Parameters	114
Table 83:	ADP[7:0] Pin Functionality	125
Table 84:	32-bit Device Limitations	127
Table 85:	Memory Interface Register Map	128
6.	Data Integrity	143
Table 123:	ECC Code Matrix	143
Table 124:	Registers for Implementing Parity and ECC	150
7.	PCI Interfaces	152
Table 130:	DevNum to IdSel Mapping	162
Table 131:	PCI_0 Registers Loaded at RESET	165
Table 132:	PCI_1 Registers Loaded at RESET	165
Table 133:	PCI Control and Configuration Register Map	169
8.	Intelligent I/O (I2O) Standard Support	201
Table 209:	I2O PCI and CPU Offsets	202
Table 210:	Register Differences Between the GT-96100A and i960Rx	203
Table 211:	Circular Queue Starting Addresses	205
Table 212:	I2O Circular Queue Functional Summary	209
Table 213:	I2O Support Register Map	210
9.	Independent DMA Controllers (IDMA Controllers)	219
Table 237:	Location of Source Address, SLP	223
Table 238:	Location of Destination Address, DLP	223
Table 239:	Location of Record Address, RLP	223
Table 240:	IDMA Controller Design Information Terms and Definitions	225
Table 241:	Source and Data Transfer Examples	227
Table 242:	Fly-By Bits	229
Table 243:	DMA Control Register Map	231
10.	PCI Arbiter	241
Table 269:	PCI Arbiter's Interface	241
11.	Communication Interface Unit (CIU)	246
12.	10/100Mb Ethernet Unit	253
Table 273:	Ethernet TX Descriptor - Command/Status word	260
Table 274:	Ethernet TX Descriptor - Byte Count	261

Table 275: Ethernet TX Descriptor - Buffer Pointer	261
Table 276: Ethernet TX Descriptor - Next Descriptor Pointer	261
Table 277: Ethernet RX Descriptor - Command/Status word	264
Table 278: Ethernet RX Descriptor - Buffer Size / Byte Count	266
Table 279: Ethernet RX Descriptor - Buffer Pointer	266
Table 280: Ethernet RX Descriptor - Next Descriptor Pointer	266
Table 281: Hash Table Entry Fields	270
Table 282: Packet Filtering Status.	275
Table 283: MII Management Frame Format	281
Table 284: Bit Transmission Parts.	281
Table 285: Ethernet Unit Register Map	283
Table 300: IP Differentiated Services CodePoint to Priority0 low (DSCP2P0L)	299
Table 301: IP Differentiated Services CodePoint to Priority0 high (DSCP2P0H)	299
Table 302: IP Differentiated Services CodePoint to Priority1 low (DSCP2P1L)	299
Table 303: IP Differentiated Services CodePoint to Priority1 high (DSCP2P1H)	299
Table 304: VLAN Priority Tag to Priority (VPT2P).	299
Table 305: Writing IP DSCP Priority Example.	300
Table 306: Writing VLAN Priority Example	300
Table 307: Writing IP DSCP and VLAN Priority Example	301
Table 308: Writing IP DSCP and VLAN Priority Register mapping Example	301
Table 309: Terms Used in MIB Counters Descriptions	301
13. Serial DMA (SDMA)	307
Table 311: SDMA Descriptor - Command/Status word	309
Table 312: SDMA Descriptor - Buffer Size, Byte Count (Rx Descriptor).	310
Table 313: SDMA Descriptor - Byte Count, Shadow Byte Count (Tx Descriptor)	310
Table 314: SDMA Descriptor - Buffer Pointer	310
Table 315: SDMA Descriptor - Next Descriptor Pointer	311
Table 319: SDMA Definitions	318
Table 320: SDMA Group 0 Register Map	320
Table 321: SDMA Group 1 Register Map	323
14. Multi Protocol Serial Controller (MPSC).....	326
Table 323: TIDL/RTSM Relationship.	333
Table 325: SDMAx Command/Status Field for HDLC Mode.	338
Table 337: BISYNC Receiver Operating Modes	348
Table 338: SDMAx Command/Status Field for BISYNC Mode	350
Table 341: BISYNC Control Character Register Format.	358
Table 342: Auto Transparent Programming	359
Table 343: CPU Controlled Operation.	359
Table 345: SDMAx Command/Status Field for UART Mode.	362
Table 347: UART Stop Bit Reception and Framing Error	365
Table 349: UART Control Character Register Format.	371
Table 351: SDMAx Command/Status Field for Transparent Mode	374
Table 352: Transparent Mode Synchronization Options.	376
Table 353: Transmitter Mode Synchronization Options	376

15. FlexTDM Units (FTDM)	379
Table 356: Flex TDM DPRAM Entry.	380
Table 358: Monitor Channel Handshaking Process	390
Table 359: IOM-1 Programming	391
Table 360: IOM2-TE Programming.	392
Table 361: IOM2-LC (connected to channel 3) GCI.	393
Table 362: FlexTDM Register Map.	394
16. Baud Rate Generators (BRGs)	397
Table 363: BRG Registers Map.	398
Table 364: BRGx Configuration Register (BCR)	399
Table 365: BRGx Baud Tuning register (BTR)	400
17. Watchdog Timer	401
18. Timers/Counters	403
19. General Purpose Ports	405
Table 373: Control Registers	405
Table 374: GPP Registers Map	406
20. Physical Signal Routing	414
21. Interrupt Controller	424
Table 390: Interrupt Registers Map	427
22. Reset Configuration	452
Table 420: Reset Configuration	452
23. Connecting the Memory Controller to SDRAM and Devices	455
Table 421: 64-bit SDRAM.	455
Table 422: 32-bit SDRAM.	456
Table 423: 64-bit Devices.	456
Table 424: 32-bit Devices.	457
Table 425: 16-bit Devices.	458
Table 426: 8-bit Devices.	459
24. JTAG Interface	460
Table 427: Supported JTAG Instructions	461
Table 428: IDCODE Register Map	462
25. Big and Little Endian	463
Table 429: Nomenclature	464
Table 430: Configuring for Big and Little Endian	465
26. Using the GT-96100A Without the CPU Interface	466
Table 431: CPU-less Pin Strapping	466
27. Using the GT-96100A in Different PCI Configurations	467
Table 432: No PCI Interface	467
Table 433: PCI_0 as 32-bit PCI Only	468
Table 434: PCI_0 as 32-bit PCI and PCI_1 as 32-bit PCI	470

Table 435: PCI_0 as 64-bit PCI Only	471
28. Phased Locked Loop (PLL) Application Notes	473
29. System Configurations	475
30. Register Tables	478
Table 436: CPU Registers Map	479
Table 437: SDRAM Registers Map	480
Table 438: DMA Registers Map	482
Table 439: Timer/Counter Registers Map	483
Table 440: PCI Registers Map	483
Table 441: Interrupts Registers Map	486
Table 442: I2O Support Registers Map	488
Table 443: Communication Unit Register Map	489
31. DC Characteristics	508
Table 444: Absolute Maximum Ratings	508
Table 445: Recommended Operating Conditions	508
Table 446: Pin Capacitance	508
Table 447: DC Electrical Characteristics Over Operating Range	509
Table 448: Driving Pad Characteristics	510
Table 449: Thermal data for GT-96100A in BGA 492	512
32. AC Timing	513
Table 450: AC Timing Measurement Formulas	513
Table 451: AC Commercial Grade Timing	513
Table 452: TCik/PCIk Restrictions	517
Table 453: Flex-TDM Receive Timing - Normal Clock	518
Table 454: Flex-TDM Transmit Timing - Normal Speed Clock	519
Table 455: Flex-TDM Receive Timing - Double Speed Clock	520
Table 456: Flex-TDM Transmit Timing - Double Speed Clock	521
Table 457: MPSC Receive Timing	523
Table 458: MPSC Transmit Timing	524
Table 459: MII Transmit Timing	529
Table 460: MII Receive Timing	529
Table 461: JTAG AC Characteristics	530
Table 462: Btyp Values	532
33. Pinout Table, 492 Pin BGA	533
Table 463: GT-96100A Pinout Table	533
34. 492 BGA Package Mechanical Information.....	542
35. GT-96100A Part Numbering.....	543
36. Abbreviations	544
37. Revision History	545
Table 464: Document History	545

List of Figures

Figure 1: Pin List	25
Figure 2: Two Stage Address Decoding- Conceptual View	56
Figure 3: CPU-Side Resource Group Decode Function and Example	60
Figure 4: Device Sub-Decode Function and Example	61
Figure 5: Bank Size Register Function Example (16Meg Decode)	62
Figure 6: CPU Address Remapping To Resources	68
Figure 7: Double Word (8 bytes) Read by CPU With Parity Check Bits	77
Figure 8: Four Word (16 bytes) Burst Read by CPU	78
Figure 9: CPU Four Word Burst Write	79
Figure 10: R5000 L2 Read Miss Example	81
Figure 11: Memory Controller Default Arbitration	96
Figure 12: Memory Controller Modified Arbitration	97
Figure 13: Non-Staggered Refresh Waveform	99
Figure 14: Staggered Refresh Waveform	99
Figure 15: Read Modify Write Transaction by the SDRAM Controller	100
Figure 16: 512 Mbit/64-bit SDRAM Connection to Memory Bus Using x8 Devices	108
Figure 17: VUMA Device and The GT-96100A sharing SDRAM	114
Figure 18: Handing the Bus Over	115
Figure 19: MREQ* Requests from the VUMA Device	116
Figure 20: Waveform Showing Device Read Parameters	120
Figure 21: Waveform Showing Device Write Parameters	121
Figure 22: Ready* Extending AccToFirst on Read Cycle	122
Figure 23: Ready* Extending AccToNext on Read Cycle	123
Figure 24: Extending WrActive Parameter on Write Cycle	124
Figure 25: PCI Master FIFOs in Single 64-bit Mode	154
Figure 26: PCI Master FIFOs in Dual 32-bit Mode	155
Figure 27: PCI Target Interface “Ping-Pong” FIFOs	158
Figure 28: PCI Target Interface FIFOs Operational Example	159
Figure 29: PCI Configuration Header	164
Figure 30: Power Management Registers	168
Figure 31: I2O Circular Queue Operation	206
Figure 32: Chained Mode DMA	224
Figure 33: PCI Arbiter’s Interface Diagram	241
Figure 34: PCI Arbitration Flow	242
Figure 35: CIU Connection Diagram	247
Figure 36: Arbiter Connectivity	249
Figure 37: MASTER Arbitration Flow	250

Figure 38: Ethernet Descriptors and Buffers	255
Figure 39: Ethernet Packet Transmission Example	257
Figure 40: Ethernet TX Descriptor	259
Figure 41: Ethernet TX Buffer Alignment Restrictions (5 byte payload)	259
Figure 42: Ethernet RX DMA Descriptor.	264
Figure 43: Type of Service Queueing Algorithm.	268
Figure 44: Ethernet Hash Table Entry	270
Figure 45: Address Chain	272
Figure 46: Address Filtering Process	274
Figure 47: RMIIDi-Bit Stream.	279
Figure 48: MII Transmit Signal Timing	279
Figure 49: MII Receive Signal Timing.	280
Figure 50: MDIO Output Delay	282
Figure 51: MDIO Setup and Hold Time	282
Figure 52: SDMA Descriptor Format	308
Figure 53: SDMAx Configuration Register (SDCx).	311
Figure 54: SDMA Command Register (SDCMx)	313
Figure 55: SDMA Descriptor Pointer Registers	316
Figure 56: Using Auto Mode to Create Idle Loop	320
Figure 57: MPSC DPLL Encoding/Decoding Schemes	327
Figure 58: MPSC Main Configuration Register (MMCRx)	328
Figure 59: Typical HDLC Frame	337
Figure 60: Typical LocalTalk Frame	337
Figure 61: MPSCx Protocol Configuration Register (MPCRx) for HDLC	339
Figure 62: Channel Registers (CHxRx) for HDLC	341
Figure 63: Typical BISYNC/MonoSYNC Frames	347
Figure 64: MPSCx Protocol Configuration Register (MPCRx) for BISYNC	351
Figure 65: Channel Registers (CHxRx) for BISYNC.	353
Figure 66: BISYNC Control Character Register Format.	357
Figure 67: Typical UART Frame	361
Figure 68: MPSCx Protocol Configuration Register (MPCRx) for UART Mode	363
Figure 69: Channel Registers (CHxRx) for UART Mode	367
Figure 70: UART Control Character Register Format.	370
Figure 71: Channel Registers (CHxRx) for Transparent Mode.	375
Figure 72: FlexTDM Architecture	379
Figure 73: Typical IOM Structures	387
Figure 74: Auxiliary Channel A Control Registers	389
Figure 75: Channel B Control Register.	390
Figure 76: Baud Rate Generator Block Diagram	397

Figure 77: Watchdog Register Map	401
Figure 78: Filtering Circuit	473
Figure 79: Resistor and Capacitor Values for VccPLL and VssPLL	473
Figure 80: Minimal System Configuration	475
Figure 81: Typical System Configuration	476
Figure 82: High Performance System	477
Figure 83: Power vs. Operating Frequency	511
Figure 84: TClk = PClk, in Sync Mode = 1, Skew Requirement	517
Figure 85: Flex-TDM Receive Timing - Normal Clock Waveform	518
Figure 86: Flex-TDM Transmit Timing - Normal Speed Clock Waveform	519
Figure 87: Flex-TDM Receive Timing - Double Speed Clock Waveform.	520
Figure 88: Flex-TDM Receive Timing - Double Speed Clock Waveform.	521
Figure 89: Flex-TDM Transmit Timing - Double Speed Clock Waveform	522
Figure 90: Flex-TDM Transmit Timing - Double Speed Clock Waveform	522
Figure 91: MPSC Receive Timing	523
Figure 92: MPSC Transmit Timing.	524
Figure 93: Output Delay From RTS*, Asynchronous CTS* (CTSS=0 in MMCRLx) Waveform	525
Figure 94: Output Delay From RTS*, Synchronous CTS* (CTSS=1 in MMCRLx) Waveform	525
Figure 95: Output Delay From CTS*, Asynchronous CTS* (CTSS=0 in MMCRLx) Waveform	526
Figure 96: Output Delay From CTS*, Synchronous CTS* (CTSS=1 in MMCRLx) Waveform	526
Figure 97: CTS* Loss In Synchronous Protocol: Start of Frame Waveform With CTS Lost	526
Figure 98: CTS* Loss In Synchronous Protocol: Start of Frame Waveform Without CTS Lost	527
Figure 99: CTS* Loss In Synchronous Protocol, Synchronous CTS* (CTSS=1 in MMCRLx) Waveform.	527
Figure 100: CTS* Loss In Synchronous Protocol, Asynchronous CTS* (CTSS=0 in MMCRLx) Waveform.	527
Figure 101: Reception Control Using CD* Waveform	527
Figure 102: External Sync (RSYL=0 in MMCRHx), CD* Pulse Mode (CDM=0 in MMCRLx) Waveform	528
Figure 103: Transmit Synchronize to Receive (TSYN=1 in MMCRLx), External Sync (RSYL=0 in MMCRHx) Waveform.	528
Figure 104: Transmit Synchronize to Receive (TSYN=1 in MMCRLx), External Sync (RSYL=0 in MMCRHx), CD* and CTS* Pulse Mode (CTSM=1 and CDM=1 in MMCRLx), Synchronous CTS* (CTSS=1 in MMCRLx) Waveform.	528
Figure 105: MII Port Transmit Signals Timing	529

Figure 106:MII Port Receive Signals Timing530
Figure 107:JTAG AC Timing530
Figure 108:GT-96100A Pinout Map (top view, left side).540
Figure 109:GT-96100A Pinout Map (top view, right side).541
Figure 110:492 BGA Package Mechanical Information.542
Figure 111:Sample Part Number.543

1. OVERVIEW

The GT-96100A offers a single-chip solution for designers building communication systems using any high performance 64-bit MIPS CPUs.

CPUs compatible with the GT-96100A include:

- The RM5260, RM5270/1 and RM7000 from QED.
- The RV4600 through RV5000 from IDT.
- Other R5000 compatibles from various vendors.

The GT-96100A integrates a system controller with a communication unit that handles a wide range of serial communication protocols, such as Ethernet, Fast Ethernet, and HDLC. Its architecture supports several system implementations for different applications and cost/performance points. Also, it is possible to design a powerful system with minimal glue logic, or add commodity logic (controlled by the GT-96100A) for differentiated system architectures that attain higher performance.

The GT-96100A has a three or four bus architecture:

- A 64-bit interface to the CPU bus (SysAD bus)
- A 64-bit interface to the memory and device subsystem
- Two independent 32-bit PCI interfaces or one 64-bit PCI interface

The three/four buses are de-coupled from each other in most accesses, enabling concurrent operation of the CPU bus, PCI devices, and accesses to memory. For example, the GT-96100A can simultaneously support a CPU bus writing to the on-chip write buffer, an IDMA agent moving data from SDRAM to its own buffers, and a PCI device writing into an on-chip FIFO.

1.1 Communication Unit Description

The heart of the GT-96100A device is a high-performance WAN communications unit.

This unit includes:

- Eight multi-protocol serial controllers.
- Four FlexTDM time slot assigners.
- Two perfect filtering 10/100 Ethernet controllers.
- Twenty SDMA channels.

The GT-96100A can directly support several WAN interfaces including Basic Rate ISDN (two channels), frame relay, non channelized T1/E1/T3, xDSL (HDSL, VDSL etc.), HSSI, and others.

1.1.1 Multi-protocol Serial Controllers

The eight multi-protocol serial controllers (MPSCs) integrated onto the GT-96100A support UART, HDLC, BISYNC, and transparent protocols. The MPSCs are implemented in the hardware. Hardware implementation allows for superior performance when compared to microcoded implementations.

In HDLC mode, the MPSCs perform all framing operations such as bit stuffing/stripping and flag generation, and part of the data link operations (e.g. address recognition functions). The MPSCs directly support common HDLC protocols including those used by ISDN and frame relay. Each MPSC can communicate over dedicated package pins or through one of four FlexTDM time slot assigners.

1.1.2 FlexTDM Time Slot Assigners

There are four FlexTDM (time slot assigners) in the GT-96100A.

The FlexTDMs support PCM Highway, IOM1, and IOM2 (GCI) formats to allow connections to most WAN framer and PHY devices. The FlexTDMs are fully programmable and can be configured to support almost any proprietary TDM bus. They can also be programmed to interface voice CODECs and MVIP bus peripherals.

The FlexTDM unit includes two auxiliary channels that can be multiplexed onto the TDM highway with data from the eight MPSCs. They are optimized for supporting GCI bus Monitor and C/I channels.

1.1.3 10/100 Ethernet Controllers

There are two 10/100-Mbps full duplex Ethernet ports in GT-96100A. Each port is fully compliant with the IEEE 802.3 and 802.3u standards and integrates MAC function and a dual speed MII interface.

The ports' speed (10 or 100Mb/s) and duplex mode (half or full duplex) is auto-negotiated through the PHY and does not require user intervention. The ports' logic also supports 802.3x flow-control mode for full-duplex and back-pressure mode for half-duplex.

The GT-96100A's Ethernet ports includes Galileo's advanced address filtering capability and can be programmed to accept or reject packets based on MAC addresses, thus providing hardware acceleration to complicated tasks such as bridging, routing, and firewall. The ports' can also filter up to 8,000 individual MAC addresses.

1.1.4 SDMA Channels

The GT-96100A offers 20 SDMA channels to support the eight MPSCs and two Fast Ethernet controllers. The SDMA channels are used to transfer data from the various serial ports to the SDRAM (and vice versa) or over the PCI. The SDMA channels use linked chain of descriptors and buffers to reduce CPU overhead.

[Table 1](#) summarizes guaranteed throughput of the MPSCs in HDLC mode when the two Fast-Ethernet ports run at 100Mbit/s full wire speed in full duplex mode.

Table 1: GT-96100A Serial Performance

No.	Operational Mode	Aggregate Bandwidth		
		Serial	Ethernet	Total
1	4 ports @55 Mbps simultaneously	440Mbit/s	400Mbit/s	840Mbit/s
2	6 ports @45 Mbps simultaneously	540Mbit/s	400Mbit/s	940Mbit/s
3	All the 8 ports @30 Mbps simultaneously	480Mbit/s	400Mbit/s	880Mbit/s

1.2 CPU Interface

The GT-96100A's SysAD bus allows the CPU and other local bus masters to access the PCI and memory/device buses.

The SysAD bus protocol supports byte, sub-word, 32-bit word, and 64-bit word operations with burst lengths of up to eight words (sub-word, two word, and four word burst length are also supported). With a maximum frequency of 100MHz, the CPU can transfer in excess of 800 Mbytes/sec.

The GT-96100A allows up to four GT-96100A devices, or GT-64120 system controllers, to share the same CPU interface. This significantly increases the address space, number of communication channels, and flexibility of system design.¹

The GT-96100A supports CPU address remapping to resources and can operate in little or big endian mode.

1.3 SDRAM and Device Interface

The GT-96100A integrates a SDRAM controller with a 64-bit interface.

The SDRAM controller supports 16, 64, 128, 256 and 512Mbit SDRAMs. It is 3.3V and 5V tolerant, operates at frequencies of up to 100MHz, and can address up to 4GBytes. Up to four SDRAM banks can be connected to the controller and it supports 2 bank interleaving for 16 Mbit SDRAMs and 2 or 4 bank interleaving for 64/128/256/512 Mbit SDRAMs.

The SDRAM controller also supports a UMA feature that enables external masters to arbitrate for direct access to SDRAM. This feature enhances system performance and gives flexibility when designing shared memory systems.

The GT-96100A device controller supports different types of memory and I/O devices. It has the control signals and timing programmability to support devices such as Flash, EPROMs, FIFOs, and I/O controllers. Device widths from 8-bits to 64-bits are also supported.

ECC generation and checking is supported both internally and externally and is optional for each bank of SDRAM.

1.4 PCI Interface

The GT-96100A interfaces directly to the PCI bus. The PCI interface can be configured to function as either:

- Two 32-bit PCI devices (PCI_0 and PCI_1)
- A single 64-bit PCI device (PCI_0) operating at a maximum frequency of 66MHz.

Each of the GT-96100A's PCI interface can either be a master initiating PCI bus transaction or a target responding to a PCI bus operation.

The GT-96100A incorporates 192-bytes of posted write and read prefetch buffers per PCI device for efficient data transfer between the CPU bus/DMA to PCI and PCI to main memory.

1. The increased loading will only have a small effect on the system's maximum operating frequency.

The GT-96100A becomes a PCI bus master when the CPU interface unit or the internal DMA engine initiates a bus cycle to a PCI device. The following PCI bus cycles are supported:

- Memory Read/Write
- Interrupt Acknowledge
- Special
- I/O Read/Write
- Configuration Read/Write
- Locked Reads/Writes (only for PCI_0 slave).

The GT-96100A acts as a target when a PCI device initiates a memory access (or an I/O access in the case of internal registers). It responds to all memory read/write accesses, as well as to all configuration and I/O cycles in the case of internal registers. It is possible to program the PCI slave function to retry all PCI transactions targeted to the GT-96100A. The PCI slave performs PCI address remapping to resources.

The GT-96100A includes all required PCI configuration registers. All internal registers, including PCI configuration registers, are accessible from both the CPU bus and the PCI bus. GT-96100A configuration register set is PC Plug-and-Play compatible, with industry standard I₂O support.

The GT-96100A supports PCI Hot-Plug and CompactPCI Hot Swap Capable requirements.

The GT-96100A can also act as a PCI to Memory bridge and PCI communication peripheral, even without the presence of a CPU.

1.5 Independent DMA (IDMA) Engines

The GT-96100A incorporates four high performance IDMA engines.

Each IDMA engine has the capability to transfer data between PCI devices, between PCI devices and main memory, or between devices residing on the 64-bit device/memory bus. The IDMA uses two internal 64-byte FIFOs for temporary storage of IDMA data. These pair of FIFOs allows two IDMA channels to be working concurrently with each channel utilizing one FIFO. For example, while channel 0 is reading data from SDRAM into one FIFO, channel 2 can write data from the other FIFO to the PCI bus.

Source and destination addresses can be nonaligned on any byte address boundary. The IDMA channels can be controlled from the CPU or PCI interfaces or via a linked list of records without CPU bus intervention. This linked list is loaded by the IDMA controller into the channel's working set when a IDMA transaction ends. The IDMA supports increment/decrement/hold on source and destination addresses independently and alignment of addresses towards source and destination. In addition, the GT-96100A provides an override capability of source/destination/record address mapping to force access to PCI_0 or PCI_1.

IDMA can be initiated by the CPU writing to a register, an external request via IDMAReq* pin, or an internal timer/counter. Four End-of-Transfer pins, which act as inputs to the GT-96100A, allow ending a IDMA transfer on a certain channel. In case of chained mode, it is possible to transfer the descriptor to CPU ownership after the transfer has ended. The CPU then calculates the number of remaining bytes in the buffer associated with the closed descriptor.

Fly-by is also supported. This mode allows data to be transferred directly between two residents on the device/memory bus without having to go into an IDMA FIFO.

1.6 Peripheral Configurations

The GT-96100A provides 88 pins to configure either as peripheral function pins or as general purpose I/Os. These pins consist of the following ports:

- Six WAN ports (A, B, C, D, E, F) with seven pins allocated per port (total of 42 pins).
- Two LAN port (MII0 and MII1) with 15 pins allocated per port (total of 30 pins).
- One GPP port with 16 pins allocated to it.

Each of the ports listed above supports multiple internal functions. [Table 2](#) shows the configuration options supported for each port.

Table 2: GT-96100A Port Configurations

Port	Port Configuration Options
A	<ol style="list-style-type: none"> 1. Physical interface for MPSC0 2. PCI_0 arbiter signals 3. General Purpose Port
B	<ol style="list-style-type: none"> 1. Physical interface for MPSC1 2. PCI_1 arbiter signals 3. General Purpose Port
C	<ol style="list-style-type: none"> 1. Physical interface for MPSC2 2. Physical interface for FlexTDM0 3. General Purpose Port
D	<ol style="list-style-type: none"> 1. Physical interface for MPSC3 2. Physical interface for FlexTDM1 3. General Purpose Port
E	<ol style="list-style-type: none"> 1. Physical interface for MPSC4 2. Physical interface for FlexTDM2 3. General Purpose Port
F	<ol style="list-style-type: none"> 1. Physical interface for MPSC5 2. Physical interface for FlexTDM3 3. General Purpose Port
MII0	<ol style="list-style-type: none"> 1. MII interface for Ethernet0 2. Physical interface for MPSC6, MPSC7 3. General Purpose Port
MII1	<ol style="list-style-type: none"> 1. MII interface for Ethernet1 2. RMII interface for both Ethernet0 and Ethernet1 3. General Purpose Port

Table 3 shows typical peripheral configurations supported by the GT-96100A.

Table 3: GT-96100A Peripheral Configurations

Peripheral Configuration Option	Port A	Port B	Port C	Port D	Port E	Port F	Port MII0	Port MII1
<ul style="list-style-type: none"> • Two serial ports • Two TDM ports • One Ethernet port 	Mpsc0	Mpsc1	TDM0	TDM1	GPP	GPP	Ether0	GPP
<ul style="list-style-type: none"> • Six serial ports • Two Ethernet ports 	Mpsc0	Mpsc1	Mpsc2	Mpsc3	Mpsc4	Mpsc5	Ether0	Ether1
<ul style="list-style-type: none"> • Two PCI arbiters • Two serial port • Two TDM ports • Two Ethernet ports 	PCI_0 arbiter	PCI_1 arbiter	Mpsc2	Mpsc3	TDM2	TDM3	Ether0	Ether1
<ul style="list-style-type: none"> • One PCI arbiter • Three serial ports • Four TDM ports • One Ethernet port 	PCI_0 arbiter	Mpsc1	TDM0	TDM1	TDM2	TDM3	Mpsc6 and Mpsc7	Ether1
<ul style="list-style-type: none"> • Eight Serial ports • Two Ethernet ports 	Mpsc0	Mpsc1	Mpsc2	Mpsc3	Mpsc4	Mpsc5	Mpsc6 and Mpsc7	Ether0 and Ether1

2. PIN INFORMATION

Figure 1: Pin List

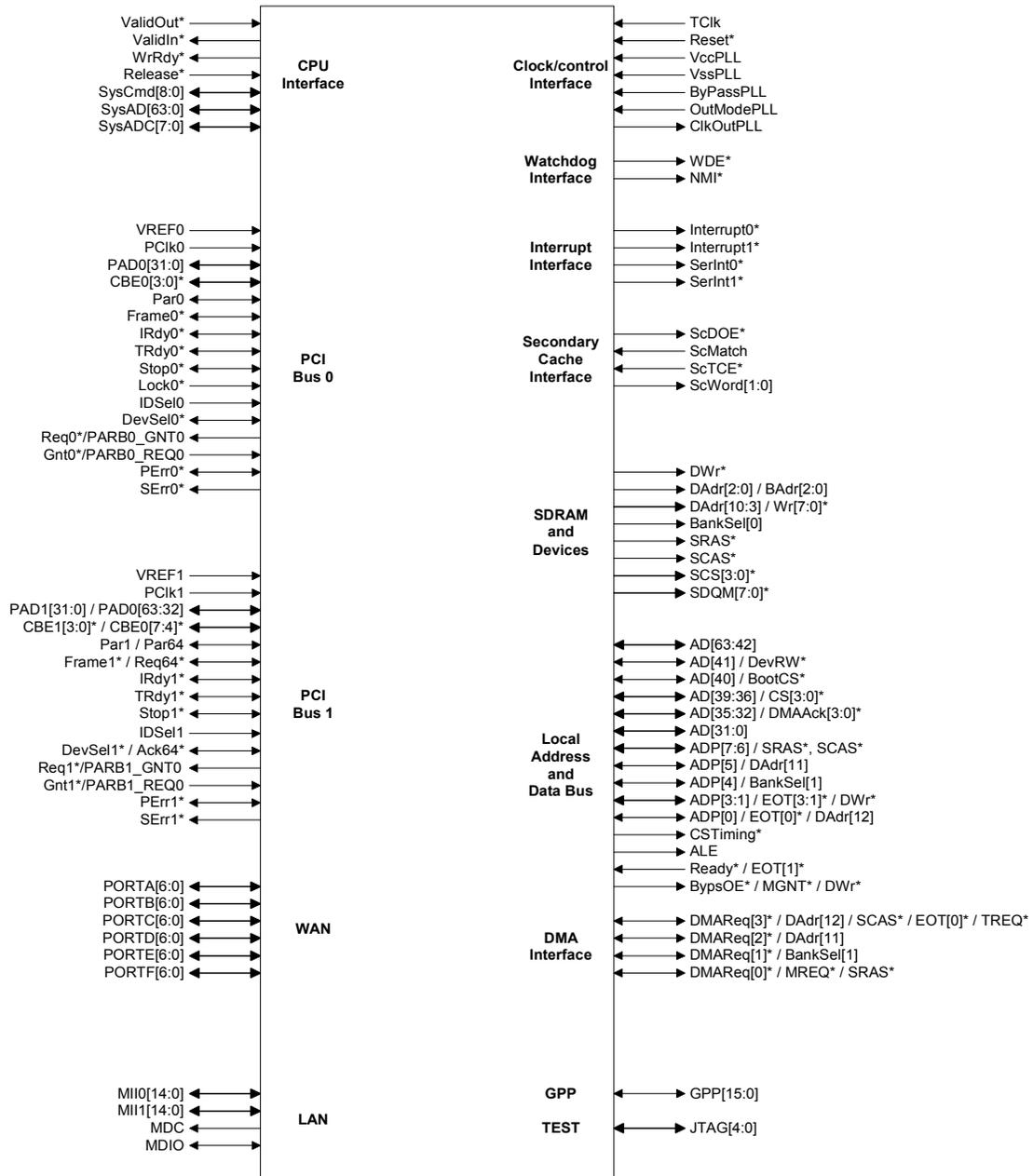


Table 4: CPU Interface Pin Assignments

Pin Name	Type	Full Name	Description
ValidOut*	I	Valid Output	Driven by the CPU to signal a valid address or data on the SysAD bus and a valid command or data identifier on the SysCmd bus.
ValidIn*	O	Valid Input	Driven by the GT-96100A to signal that it is driving valid data on the SysAD bus and a valid data identifier on the SysCmd bus.
WrRdy*	O	Write Ready	Driven by the GT-96100A to signal that it can accept a CPU write request from the CPU (i.e. there is room in the write post-ing FIFO).
Release*	I	Release	Driven by the CPU to signal that it has released the SysAD and SysCmd buses for completion of a read request.
SysCmd[8:0]	I/O	System Command/Data Identifier Bus	9-bit bus used for command and data identifier transmission between the CPU and GT-96100A.
SysAD[63:0]	I/O	System Address/Data Bus	64-bit bus used as multiplexed address and data bus for communication between the CPU, the GT-96100A, and the L2 cache.
SysADC[7:0]	I/O	System Address/Data Check	8-bit bus used as parity for the SysAD bus. SysADC is valid on data cycles only.
CPU Interface Total: 85			

Table 5: Secondary Cache Interface Pin Assignments

Pin Name	Type	Full Name	Description
ScDOE*	O	Secondary Cache Data RAM Output Enable	Asserted by the GT-96100A to cause the data RAM to drive data onto their I/O pins. This signal is monitored by the processor to determine when to drive the data RAM write enable in a secondary cache miss refill sequence. This pin must be left unconnected if secondary cache is not used.
ScMatch	I	Secondary Cache Tag Match	Asserted by the cache Tag RAM when a match occurs between the value on its data inputs and the contents of its RAM at the value of its address inputs. This pin must be pulled LOW through a 4.7KOhm resistor if secondary cache is not used.

Table 5: Secondary Cache Interface Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
ScTCE*	I	Secondary Cache Tag RAM Chip Enable	Indicates that a secondary cache access is occurring. This pin must be pulled HIGH through a 4.7KOhm resistor if secondary cache is not used.
ScWord[1:0]	O	Secondary Cache Double Word Index	Driven by the GT-96100A on cache miss refills. These pins must be left unconnected if secondary cache is not used.
Secondary Cache Total: 5			

Table 6: PCI Bus 0 Pin Assignments

Pin Name	Type	Full Name	Description
VREF0	I	PCI_0 Voltage Reference	Must be connected directly to the 3.3V or the 5V power plane, depending on which voltage level PCI_0 supports. NOTE: VREF0 and VREF1 can be completely independent voltage levels.
PCIk0	I	PCI_0 Clock	Provides the timing for the PCI_0 transactions. The PCI_0 clock range is between 0 and 66MHz. NOTE: The PCIk0 cycle must be higher than the TClk cycle by at least 1ns. See Section 32.1 “TClk/PCIk Restrictions” on page 516 .
PAD0[31:0]	I/O	PCI_0 Address/Data	32-bit multiplexed PCI_0 address and data lines. During the first clock of the transaction, PAD0[31:0] contains a physical byte address (32 bits). During subsequent clock cycles, this contains data.
CBE0[3:0]*	I/O	PCI_0 Command/Byte Enable	During the address phase of the transaction, CBE0[3:0]* provides the PCI_0 bus command. During the data phase, these lines provide the byte enables.
Par0	I/O	PCI_0 Parity	Calculated by the GT-96100A as an even parity bit for the PAD0[31:0] and CBE0[3:0]* lines.
Frame0*	I/O	PCI_0 Frame	Asserted by the GT-96100A to indicate the beginning and duration of a master transaction. Frame0* asserts to indicate the beginning of the cycle. While asserted, data transfer continues. Frame0* deasserts to indicate that the next data phase is the final data phase transaction. Frame0* is monitored by the GT-96100A when it acts as a PCI target.
IRdy0*	I/O	PCI_0 Initiator Ready	Asserted to indicate the bus master’s ability to complete the current data phase of the transaction. A data phase is completed on any clock when both IRdy0* and TRdy0* are asserted. Wait cycles are inserted until TRdy0* and IRdy0* are asserted together.

Table 6: PCI Bus 0 Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
TRdy0*	I/O	PCI_0 Target Ready	Asserted to indicate the target agent's ability to complete the current data phase of the transaction. A data phase is completed on any clock when both TRdy0* and IRdy0* are asserted. Wait cycles are inserted until TRdy0* and IRdy0* are asserted together.
Stop0*	I/O	PCI_0 Stop	Asserted to indicate that current target is requesting the bus master to stop the current transaction. As a master, the GT-96100A responds to the assertion of Stop0* by disconnecting, retrying, or aborting. As a target, the GT-96100A asserts Stop0* to retry or disconnect.
Lock0*	I	PCI_0 Lock	Asserted to indicate an automatic operation that may require multiple transactions to complete. When the GT-96100A is a PCI_0 target, Lock0* is sampled on the rising edge of PClk0 when Frame0* is asserted. If Lock0* is sampled asserted, the GT-96100A enters a locked state and remains in this state until Lock0* is sampled deasserted on the following rising edge of PClk0, when Frame0* is sampled asserted.
IDSel0	I	PCI_0 Initialization Device Select	Asserted to indicate a chip select during PCI_0 configuration read and write transactions.
DevSel0*	I/O	PCI_0 Device Select	Asserted by the target of the current access. When the GT-96100A is bus master, it expects the target to assert DevSel0* within five bus cycles, confirming the access. If the target does not assert DevSel0* within this time window, the GT-96100A aborts the cycle. As a target, when the GT-96100A recognizes that it is the target of a transaction, it asserts DevSel0* at medium speed (two cycles after assertion of Frame0*).
Req0*/ PARB0_GNT1	O	PCI_0 Bus Request	If the internal arbiter for PCI_0 is disabled, this signal is asserted by the GT-96100A to indicate to the PCI_0 bus arbiter that it requires use of the PCI_0 bus.
		PCI_0 arbiter output grant 1	If the internal arbiter for PCI_0 is enabled, this pin functions as the arbiter's grant 1 output signal.
Gnt0*/ PARB0_REQ1	I	PCI_0 Bus Grant	If the internal arbiter for PCI_0 is disabled, this signal is asserted by the external PCI_0 bus arbiter to indicate that access to the PCI_0 bus is granted to the GT-96100A.
		PCI_0 arbiter input request 1	If the internal arbiter for PCI_0 is enabled, this pin functions as the arbiter's request 1 input signal.

Table 6: PCI Bus 0 Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
PErr0*	I/O STS	PCI_0 Parity Error	Asserted when a data parity error is detected. This pin features a sustained tristate output.
SErr0*	OD	PCI_0 System Error	Asserted when a serious system error (not necessarily a PCI_0 error) is detected. SErr0* behavior in the GT-96100A is programmable (refer to PCI section for details). This pin features an open-drain output.
PCI Bus 0 Total: 50			

Table 7: PCI Bus 1 Pin Assignments

Pin Name	Type	Full Name	Description
VREF1	I	PCI_1 Voltage Reference	Must be connected directly to the 3.3V or the 5V power plane depending on which voltage level PCI_1 supports. NOTE: VREF0 and VREF1 can be completely independent voltage levels.
PCIk1	I	PCI_1 Clock	Provides the timing for PCI_1 transactions. The PCI_1 clock range is between 0 and 66MHz. Runs independently of PCIk0. Active only when PCI_1 is enabled. NOTE: The PCIk0 cycle must be higher than the TClk cycle by at least 1ns. This clock frequency can be independent of both TClk and PCIk0. See Section 32.1 "TClk/PCIk Restrictions" on page 516.
PAD1[31:0]/ PAD0[63:32]	I/O	PCI_1 Address/Data	During the first clock of the transaction, PAD1[31:0] contains a physical byte address (32 bits). During subsequent clock cycles, PAD1[31:0] contains data.
		PCI_0 (64 bit) Address/Data	If PCI_0 is configured for 64 bit, these pins function as PAD0[63:32] and carry the most significant 32 bits of data for PCI_0 transactions.
CBE1[3:0]*/ CBE0[7:4]*	I/O	PCI_1 Command/Byte Enable	During the address phase of the transaction, CBE1[3:0]* provide the PCI_1 bus command. During the data phase, these lines provide the byte enables.
		PCI_0 (64 bit) Byte Enable	If PCI_0 is configured for 64 bit, these pins function as CBE0[7:4]* and carry byte enables for the most significant 32 bits of PCI_0 data.
Par1/Par64	I/O	PCI_1 Parity	Calculated by the GT-96100A as an even parity bit for PAD1[31:0] and CBE1[3:0]* lines.
		PCI_0 (64 bit) Parity	If PCI_0 is configured for 64 bit, this pin functions as Par64 and carries even parity bit for PAD0[63:32] and CBE0[7:4]*.

Table 7: PCI Bus 1 Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Frame1*/ Req64*	I/O	PCI_1 Frame	Asserted by the GT-96100A to indicate the beginning and duration of a master transaction. Frame1* asserts to indicate the beginning of the cycle. While asserted, data transfer continues. Deasserts to indicate that the next data phase is the final data phase transaction. Frame1* is monitored by the GT-96100A when it acts as a target.
		PCI_0 (64 bit) Request 64	If PCI_0 is configured for 64 bit, this pin functions as Req64* and functions as a request for a 64-bit transaction. Req64* has the same timing as Frame0*.
	SoR	Sampled on RESET to configure the GT-96100A prior to boot-up. See Section 22. "Reset Configuration" on page 452 for more information.	
IRdy1*	I/O	PCI_1 Initiator Ready	Asserted to indicate the bus master's ability to complete the current data phase of the transaction. A data phase is completed on any clock when both IRdy1* and TRdy1* are asserted. Wait cycles are inserted until TRdy1* and IRdy1* are asserted together.
TRdy1*	I/O	PCI_1 Target Ready	Asserted to indicate the target agent's ability to complete the current data phase of the transaction. A data phase is completed on any clock when both TRdy1* and IRdy1* are asserted. Wait cycles are inserted until TRdy1* and IRdy1* are asserted together.
Stop1*	I/O	PCI_1 Stop	Asserted to indicate the current target is requesting the bus master to stop the current transaction. As a master, the GT-96100A responds to the assertion of Stop1* by disconnecting, retrying or aborting. As a target, the GT-96100A asserts Stop1* to retry or disconnect.
IDSel1	I	PCI_1 Initialization Device Select	Asserted to indicate a chip select during PCI_1 configuration read and write transactions.
DevSel1*/ Ack64*	I/O	PCI_1 Device Select	Asserted by the target of the current access. When the GT-96100A is bus master, it expects the target to assert DevSel1* within 5 bus cycles, confirming the access. If the target does not assert DevSel1* within this time window, the GT-96100A aborts the cycle. As a target, when the GT-96100A recognizes that it is the target of a transaction, it asserts DevSel1* at medium speed (two cycles after assertion of Frame1*).
		PCI_1 (64 bit) Acknowledge 64	If PCI_0 is configured for 64 bit, this signal functions as Ack64*. When actively driven by the PCI target, it indicates that the target is willing to accept 64 bit data. Ack64* has the same timing as DevSel0*.

Table 7: PCI Bus 1 Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Req1*/ PARB1_GNT1	O	PCI_1 Bus Request	If the internal arbiter for PCI_1 is disabled, this signal is asserted by the GT-96100A to indicate to the PCI_1 bus arbiter that it requires use of the PCI_1 bus.
		PCI_1 arbiter output grant 1	If the internal arbiter for PCI_1 is enabled, this pin functions as the PCI_1 arbiter's grant 1 output signal.
Gnt1*/ PARB1_REQ1	I	PCI_1 Bus Grant	If the internal arbiter for PCI_1 is disabled, this signal is asserted by the external PCI_1 bus arbiter to indicate that access to the PCI_1 bus is granted to the GT-96100A.
		PCI_1 arbiter input request 1	If the internal arbiter for PCI_1 is enabled, this pin functions as the PCI_1 arbiter's request 1 input signal.
PErr1*	I/O STS	PCI_1 Parity Error	Asserted when a data parity error is detected. This pin features a sustained tristate output.
SErr1*	OD	PCI_1 System Error	Asserted when a serious system error (not necessarily a PCI_1 error) is detected. SErr1* behavior in the GT-96100A is programmable (refer to PCI section for details). This pin features an open-drain output.
PCI Bus 1 Total: 49			

Table 8: SDRAM and Devices Pin Assignments

Pin Name	Type	Full Name	Description
DWr*	O	SDRAM Write	Asserted low when the GT-96100A performs a write transaction to the SDRAM.
DAdr[2:0]/ BAdr[2:0]	O	SDRAM Address [2:0]	When accessing a SDRAM bank, these pins function as SDRAM address bits [2:0].
		Burst Address [2:0]	In write and read accesses from devices these pins function as burst address bits [2:0]. See Section 23.2 "Devices" on page 456 for more information on how to connect these address bits to various devices.
	SoR	NOTE: Sampled on RESET to configure the GT-96100A prior to boot-up. See Section 22. "Reset Configuration" on page 452 for more information.	
DAdr[10:3]/ Wr[7:0]*	O	SDRAM Address [10:3]	When accessing a SDRAM bank, these pins function as SDRAM address bits.
		Byte Write [7:0]	In write and accesses to devices these pins function as byte write enable indications for bytes [7:0]. See Section 23.2 "Devices" on page 456 for more information on how to connect these address bits to various devices.
	SoR	NOTE: Sampled on RESET to configure the GT-96100A prior to boot-up. See Section 22. "Reset Configuration" on page 452 for more information.	

Table 8: SDRAM and Devices Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
BankSel[0]	O	SDRAM Bank Select [0]	In SDRAM accesses, this pin functions as bank select bit [0].
	SoR	NOTE: Sampled on RESET to configure the GT-96100A prior to boot-up. See Section 22. "Reset Configuration" on page 452 for more information.	
SRAS*	O	SDRAM Row Address Strobe	Asserted to indicate that an active ROW address is driven on DAdr lines.
SCAS*	O	SDRAM Column Address Strobe	Asserted to indicate that an active COLUMN address is driven on DAdr lines.
SCS[3:0]*	O	SDRAM Chip Selects	SDRAM chip selects for up to 4 banks.
SDQM[7:0]*	O	SDRAM Byte Enables	In SDRAM write transaction these pins function as byte enable signals.
	SoR	NOTE: SDQM[3:0] are sampled on RESET to configure the GT-96100A prior to boot-up. See Section 22. "Reset Configuration" on page 452 for more information.	
SDRAM/Devices Total: 27			

Table 9: Local Address and Data Bus Pin Assignments

Pin Name	Type	Full Name	Description
AD[63:42]	I/O	Address/Data [63:42]	In SDRAM accesses, these pins function as part of the data to be read/written from/to the SDRAMs. In Device accesses, these pins function as data during the data phase.
AD[41]/DevRW*	I/O	Address/Data [41]	In SDRAM/Device data phase, this pin functions as data bit [41].
		Device Read-Write	In Device address phase, this pin indicates if an access to a device is read ('1') or write ('0'). Latching is done via ALE.
AD[40]/BootCS*	I/O	Address/Data [40]	In SDRAM/Device data phase, this pin functions as data bit [40].
		Boot Chip Select	In Device address phase, this pin functions as the boot device chip select. Latching is done via ALE.
AD[39:36]/CS[3:0]*	I/O	Address/Data [39:36]	In SDRAM/Device data phase, these pins function as data bits [39:36].
		Chip Select [3:0]	In Device address phase, these pins function as Device Chip Selects and are valid (and should be latched). The Chip Selects need to be qualified with the CSTiming* signal. Latching is done via ALE.

Table 9: Local Address and Data Bus Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
AD[35:32]/ DMAAck [3:0]*	I/O	Address/Data [35:32]	In SDRAM/Device data phase, these pins function as data bits [35:32].
		DMA Acknowl- edge[3:0]	In Device address phase, these pins function as DMA Acknowl-edges and are valid (and should be latched). They need to be qualified with the CSTiming* signal. Latching is done via ALE.
AD[31:0]	I/O	Address/ Data[31:0]	Multiplexed address and data bus to the SDRAM (data only) and Devices (address and data).
ADP[7:6]/ SRAS*/ SCAS*	I/O	SDRAM data ECC [7:6]	If the GT-96100A is configured for ECC mode, then in SDRAM accesses, these pins serve as bits [7:6] of the ECC for data bits [63:0]. ECC is generated by the GT-96100A for 64-bit SDRAM writes, and read from SDRAM ECC bank for 64-bit SDRAM reads.
		SDRAM Row Address Strobe	ADP[7:6] can be configured to function as SRAS* on RESET. See Section 22. "Reset Configuration" on page 452.
		SDRAM Col- umn Address Strobe	ADP[7:6] can be configured to function as SCAS* on RESET. See Section 22. "Reset Configuration" on page 452.
ADP[5]/ DAdr[11]	I/O	SDRAM data ECC [5]	If the GT-96100A is configured for ECC mode, then in SDRAM accesses this pin serve as bit [5] of the ECC for data bits [63:0]. ECC is generated by the GT-96100A for 64 bit SDRAM writes, and read from SDRAM ECC bank for 64 bit SDRAM reads.
		SDRAM Address [11]	If the GT-96100A is configured to non-ECC mode, then in SDRAM accesses this pin functions as SDRAM address bit[11].
ADP[4]/Bank Sel[1]	I/O	SDRAM data ECC [4]	If the GT-96100A is configured for ECC mode, then in SDRAM accesses this pin serve as bit [4] of the ECC for data bits [63:0]. ECC is generated by the GT-96100A for 64 bit SDRAM writes, and read from SDRAM ECC bank for 64 bit SDRAM reads.
		SDRAM Bank Select [1]	If the GT-96100A is configured to non-ECC mode, then in SDRAM accesses, this pin functions as bank select bit[1].
ADP[3:1]/ EOT[3:1]*/ DWr*	I/O	SDRAM data ECC [3:1]	If the GT-96100A is configured for ECC mode, then in SDRAM accesses, these pins serve as bits [3:1] of the ECC for data bits [63:0]. ECC is generated by the GT-96100A for 64 bit SDRAM writes, and read from SDRAM ECC bank for 64 bit SDRAM reads.
		End of DMA Transfer [3:1]	If the GT-96100A is configured to non-ECC mode, then in SDRAM accesses, these pins serve as End Of Transfer indications for the DMA channels.
		SDRAM Write	ADP[3] can be configured to function as DWr* on RESET. See Section 22. "Reset Configuration" on page 452.

Table 9: Local Address and Data Bus Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
ADP[0]/ EOT[0]*/ DAdr[12]	I/O	SDRAM data ECC[0]	If the GT-96100A is configured for ECC mode, then in SDRAM accesses, this pin serves as bit [0] of the ECC for data bits [63:0]. ECC is generated by the GT-96100A for 64 bit SDRAM writes, and read from SDRAM ECC bank for 64 bit SDRAM reads.
		End of DMA Transfer [0]	If the GT-96100A is configured to non-ECC mode, then in SDRAM accesses, this pin serves as End Of Transfer indication for DMA channel 0.
		SDRAM Address [12]	ADP[0] can be configured to function as SDRAM Address [12] on RESET. See Section 22. "Reset Configuration" on page 452.
CSTiming*	O	Chip Select Timing	This signal is active (asserted low) for the number of cycles that the device currently being accessed is programmed to. Used to qualify CS[3:0]*, BootCS and DMAAck[3:0]* signals.
	SoR	NOTE: Sampled on RESET to configure the GT-96100A prior to boot-up. See Section 22. "Reset Configuration" on page 452 for more information.	
ALE	O	Address Latch Enable	This signal is asserted in the Device address phase and must be used to latch the Address, BootCS*, CS[3:0]*, DevRW* and DMAAck[3:0]* pins from the AD bus.
Ready*/ EOT[1]*	I	Ready	This input signal is used as a cycle extender NOTE: When inactive during device access, the access is extended until Ready* is asserted.
		End Of Trans- fer [1]	Ready* can be programmed to function as EOT[1]*. See Section 5.1.2.3 "DMA End of Transfer Pins Functionality" on page 101.
	SoR	NOTE: Sampled on RESET to configure the GT-96100A prior to boot-up. See Section 22. "Reset Configuration" on page 452 for more information.	
BypsOE*/ MGNT*/DWr*	O	Bypass Out- put Enable	If bypass mode is enabled, this signal controls the output enable for bypass latches/buffers/switches. The bypass can be used when a 64-bit read transaction is executed from the CPU. Read data will be transferred directly to the CPU bus. See Table 77.
		Memory (AD) bus Grant	If the GT-96100A is configured (at RESET) for UMA support, this pin functions as Memory Grant. It is asserted in response to MREQ*.
		SDRAM Write	This pin can be programmed to function as DWr*. See Section 5.1.2.1 "Duplicating SDRAM Control Lines" on page 100.
Local Address Total: 76			

Table 10: DMA Pin Assignments

Pin Name	Type	Full Name	Description
DMAReq[3]*/ DAdr[12]/ EOT[0]*/ SCAS*/TREQ*	I/O	DMA Request[3]	DMA request by external devices to IDMA channel 3.
		SDRAM Address [12]	This pin can be configured to function as DAdr[12]. See Section 5.1.2.4 “Multiplexing DAdr[12]” on page 102.
		UMA Internal Request	For UMA operation, DMAReq[3]* can be programmed to indicate that there is a pending internal request in SDRAM and Device interface that requires the GT-96100A ownership of the AD bus. See Section 5.6.6 “Total Request” on page 117.
		End of DMA Transfer[0]	DMAReq[3]* can be programmed to function as EOT[0]*. See Section 5.1.2.3 “DMA End of Transfer Pins Functionality” on page 101.
	SDRAM Column Address Strobe	DMAReq[3]* can be programmed to function as SCAS*. See Section 5.1.2.1 “Duplicating SDRAM Control Lines” on page 100.	
	SoR	NOTE: Sampled on RESET to configure the GT-96100A prior to boot-up. See Section 22. “Reset Configuration” on page 452 for more information.	
DMAReq[2]*/ DAdr[11]	I/O	DMA Request[2]	DMA request by external devices to IDMA channel 2.
		SDRAM Address [11]	This pin can be configured to function as SDRAM address bit[11]. See Section 5.1.2.2 “Duplicating DAdr[11] and BankSel[1] on DMAReq[2:1]” on page 101.
	SoR	NOTE: Sampled on RESET to configure the GT-96100A prior to boot-up. See Section 22. “Reset Configuration” on page 452 for more information.	
DMAReq[1]*/ BankSel[1]	I/O	DMA Request[1]	DMA request by external devices to IDMA channel 1.
		SDRAM Bank Select [1]	This pin can be configured to function as BankSel[1]. See Section 5.1.2.2 “Duplicating DAdr[11] and BankSel[1] on DMAReq[2:1]” on page 101.
	SoR	NOTE: Sampled on RESET to configure the GT-96100A prior to boot-up. See Section 22. “Reset Configuration” on page 452 for more information.	
DMAReq[0]*/ MREQ*/ SRAS*	I/O	DMA Request [0]	DMA request by external devices to IDMA channel 0.
		Memory Bus Request	If the GT-96100A is configured (at RESET) for UMA support, this pin functions as Memory Request.
		SDRAM Row Address Strobe	This pin can be programmed to function as SRAS*. See Section 5.1.2.1 “Duplicating SDRAM Control Lines” on page 100.
DMA Total: 4			

Table 11: WAN Pin Assignments

Pin Name	Type	Full Name	Description
Port A NOTE: Port A can be connected to MPSC0. See Section 20. “Physical Signal Routing” on page 414 for more details. When not connected to MPSC0, Port A can be connected to the PCI Arbiter or used as GPP. See Section 19. “General Purpose Ports” on page 405 for more details.			
PORTA[0]	I/O	RXD0 (Input)	Serial receive data input to MPSC0.
		PARB0_GNT2 (output)	PCI_0 arbiter output grant 2.
		GPP16 (I/O)	General Purpose Pin 16.
PORTA[1]	I/O	PARB0_REQ2 (Input)	PCI_0 arbiter input request 2.
		TXD0 (Output)	Serial transmit data output from the MPSC0.
		GPP17 (I/O)	General Purpose Pin 17.
PORTA[2]	I/O	PARB0_REQ3 (Input)	PCI_0 arbiter input request 3.
		RTS0* (Output)	Request to Send output from MPSC0. Indicates that MPSC0 is ready to transmit data.
		GPP18 (I/O)	General Purpose Pin 18.
PORTA[3]	I/O	CTS0* (Input)	Clear to Send input to MPSC0. Indicates to MPSC0 that data transmission may begin.
		PARB0_GNT3 (Output)	PCI_0 arbiter output grant 3.
		GPP19 (I/O)	General Purpose Pin 19.
PORTA[4]	I/O	CD0 (Input)	Carrier Detect input to MPSC0. Indicates to MPSC0 that it can begin reception of data.
		PARB0_GNT4 (Output)	PCI_0 arbiter output grant 4.
		GPP20 (I/O)	General Purpose Pin 20.
PORTA[5]	I/O	SCLK0 (Input)	Input clock to MPSC0. Can be used as both transmit and receive clock. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		PARB0_REQ4 (Input)	PCI_0 arbiter input request 4.
		OSCLK0 (Output)	Output clock from MPSC0. Can be used when SCLK0 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs).
		GPP21 (I/O)	General Purpose Pin 21.

Table 11: WAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port A (Continued)			
PORTA[6]	I/O	TSCLK0 (Input)	Input clock to MPSC0. Can be used by the MPSC transmitter when separate receive and transmit clocks are needed. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		PARB0_REQ5 (Input)	PCI_0 arbiter input request 5.
		OTSCLK0 (Output)	Output Tx clock from MPSC0. Can be used when TSCLK0 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs, or when there is no need for a separate Tx clock).
		GPP22 (I/O)	General Purpose Pin 22.
Port B			
<p>NOTE: Port B can be connected to MPSC1. See Section 20. "Physical Signal Routing" on page 414 for more details.</p> <p>When not connected to MPSC1, Port B can be connected to the PCI Arbiter or used as GPP. See Section 19. "General Purpose Ports" on page 405 for more details.</p>			
PORTB[0]	I/O	RXD1 (Input)	Serial receive data input to MPSC1.
		PARB1_GNT2 (Output)	PCI_1 arbiter output grant 2.
		GPP24 (I/O)	General Purpose Pin 24.
PORTB[1]	I/O	PARB1_REQ2 (Input)	PCI_1 arbiter input request 2.
		TXD1 (Output)	Serial transmit data output from the MPSC1.
		GPP25 (I/O)	General Purpose Pin 25.
PORTB[2]	I/O	PARB1_REQ3 (Input)	PCI_1 arbiter input request 3.
		RTS1* (Output)	Request to Send output from MPSC1. Indicates that MPSC1 is ready to transmit data.
		GPP26 (I/O)	General Purpose Pin 26.
PORTB[3]	I/O	CTS1* (Input)	Clear to Send input to MPSC1. Indicates to MPSC1 that data transmission may begin.
		PARB1_GNT3 (Output)	PCI_1 arbiter output grant 3.
		PP27 (I/O)	General Purpose Pin 27.

Table 11: WAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port B (Continued)			
PORTB[4]	I/O	CD1 (Input)	Carrier Detect input to MPSC1. Indicates to MPSC1 that it can begin reception of data.
		PARB0_GNT6 (Output)	PCI_0 arbiter output grant 6.
		PARB1_GNT4 (Output)	PCI_1 arbiter output grant 4.
		GPP28 (I/O)	General Purpose Pin 28.
PORTB[5]	I/O	SCLK1 (Input)	Input clock to MPSC1. Can be used as both transmit and receive clock. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		PARB0_REQ6 (Input)	PCI_0 arbiter input request6.
		PARB1_REQ4 (Input)	PCI_1 arbiter input request 4.
		OSCLK1 (Output)	Output clock from MPSC1. Can be used when SCLK1 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs).
		GPP29 (I/O)	General Purpose Pin 29.
PORTB[6]	I/O	TSCLK1 (Input)	Input clock to MPSC1. Can be used by the MPSC transmitter when separate receive and transmit clocks are needed. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OTCLK1 (Output)	Output Tx clock from MPSC1. Can be used when TSCLK1 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs, or when there is no need for a separate Tx clock).
		PARB0_GNT5 (Output)	PCI_0 arbiter output grant 5.
		GPP30 (I/O)	General Purpose Pin 30.

Table 11: WAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port C			
NOTE: Port C can be connected to MPSC2. See Section 20. “Physical Signal Routing” on page 414 for more details.			
When not connected to MPSC2, Port C can be connected to TDM0 or used as GPP. See Section 19. “General Purpose Ports” on page 405 for more details.			
PORTC[0]	I/O	RXD2 (Input)	Serial receive data input to MPSC2.
		TRXD0 (Input)	Serial receive data input to TDM channel0.
		TTXD0 (Output)	Serial transmit data from TDM channel0.
		GPP32 (I/O)	General Purpose Pin 32.
PORTC[1]	I/O	TXD2 (Output)	Serial transmit data output from the MPSC2.
		TTXD0 (Output)	Serial transmit data from TDM channel0.
		TRXD0 (Input)	Serial receive data input to TDM channel0.
		GPP33 (I/O)	General Purpose Pin 33.
PORTC[2]	I/O	RTS2* (Output)	Request to Send output from MPSC2. Indicates that MPSC2 is ready to transmit data.
		TDSTRB0 (Output)	TDM channel0 strobe output signal, which can be used to gate clocks to external devices that do not have a built in TDM.
		GPP34 (I/O)	General Purpose Pin 34.
PORTC[3]	I/O	CTS2* (Input)	Clear to Send input to MPSC2. Indicates to MPSC2 that data transmission may begin.
		TTSYNC0 (Input)	Transmit Frame Sync input to TDM channel0.
		GPP35 (I/O)	General Purpose Pin 35.
PORTC[4]	I/O	CD2 (Input)	Carrier Detect input to MPSC2. Indicates to MPSC2 that it can begin reception of data (input to MPSC2).
		TRSYNC0 (Input)	Receive Frame Sync input to TDM channel0.
		GPP36 (I/O)	General Purpose Pin 36.

Table 11: WAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port C (Continued)			
PORTC[5]	I/O	SCLK2 (Input)	Input clock to MPSC2. Can be used as both transmit and receive clock. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OSCLK2 (Output)	Output clock from MPSC2. Can be used when SCLK2 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs).
		TRCLK0 (Input)	Receive input clock to TDM channel0.
		GPP37 (I/O)	General Purpose Pin 37.
PORTC[6]	I/O	TSCLK2 (Input)	Input clock to MPSC2. Can be used by the MPSC transmitter when separate receive and transmit clocks are needed. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OTSCLK2 (Output)	Output Tx clock from MPSC2. Can be used when TSCLK2 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs, or when there is no need for a separate Tx clock).
		TTCLK0 (Input)	Transmit input clock to TDM channel0.
		GPP38 (I/O)	General Purpose Pin 38.
Port D NOTE: Port D can be connected to MPSC3. See Section 20. “Physical Signal Routing” on page 414 for more details. When not connected to MPSC3, Port D can be connected to TDM1 or used as GPP. See Section 19. “General Purpose Ports” on page 405 for more details.			
PORTD[0]	I/O	RXD3 (Input)	Serial receive data input to MPSC3.
		TRXD1 (Input)	Serial receive data input to TDM channel1.
		TTXD1 (Output)	Serial transmit data from TDM channel1.
		GPP40 (I/O)	General Purpose Pin 40.
PORTD[1]	I/O	TXD3 (Output)	Serial transmit data output from the MPSC3.
		TTXD1 (Output)	Serial transmit data from TDM channel1.
		TRXD1 (Input)	Serial receive data input to TDM channel1.
		GPP41 (I/O)	General Purpose Pin 41.

Table 11: WAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port D (Continued)			
PORTD[2]	I/O	RTS3* (Output)	Request to Send output from MPSC3. Indicates that MPSC3 is ready to transmit data.
		TDSTRB1 (Output)	TDM channel1 strobe output signal, which can be used to gate clocks to external devices that do not have a built in TDM.
		GPP42 (I/O)	General Purpose Pin 42.
PORTD[3]	I/O	CTS3* (Input)	Clear to Send input to MPSC3. Indicates to MPSC3 that data transmission may begin.
		TTSYNC1 (Input)	Transmit Frame Sync input to TDM channel1.
		GPP43 (I/O)	General Purpose Pin 43.
PORTD[4]	I/O	CD3 (Input)	Carrier Detect input to MPSC3. Indicates to MPSC3 that it can begin reception of data (input to MPSC3).
		TRSYNC1 (Input)	Receive Frame Sync input to TDM channel1.
		GPP44 (I/O)	General Purpose Pin 44.
PORTD[5]	I/O	SCLK3 (Input)	Input clock to MPSC3. Can be used as both transmit and receive clock. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OSCLK3 (Output)	Output clock from MPSC3. Can be used when SCLK3 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs).
		TRCLK1 (Input)	Receive input clock to TDM channel1.
		GPP45 (I/O)	General Purpose Pin 45.
PORTD[6]	I/O	TSCLK3 (Input)	Input clock to MPSC3. Can be used by the MPSC transmitter when separate receive and transmit clocks are needed. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OTCLK3 (Output)	Output Tx clock from MPSC3. Can be used when TSCLK3 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs, or when there is no need for a separate Tx clock).
		TTCLK1 (Input)	Transmit input clock to TDM channel1.
		GPP46 (I/O)	General Purpose Pin 46.

Table 11: WAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port E NOTE: Port E can be connected to MPSC4. See Section 20. “Physical Signal Routing” on page 414 for more details. When not connected to MPSC4, Port E can be connected to TDM2 or used as GPP. See Section 19. “General Purpose Ports” on page 405 for more details.			
PORTE[0]	I/O	RXD4 (Input)	Serial receive data input to MPSC4.
		TRXD2 (Input)	Serial receive data input to TDM channel2.
		TTXD2 (Output)	Serial transmit data from TDM channel2.
		GPP48 (I/O)	General Purpose Pin 48.
PORTE[1]	I/O	TXD4 (Output)	Serial transmit data output from the MPSC4.
		TTXD2 (Output)	Serial transmit data from TDM channel2.
		TRXD2 (Input)	Serial receive data input to TDM channel2.
		GPP49 (I/O)	General Purpose Pin 49.
PORTE[2]	I/O	RTS4* (Output)	Request to Send output from MPSC4. Indicates that MPSC4 is ready to transmit data.
		TDSTRB2 (Output)	TDM channel2 strobe output signal, which can be used to gate clocks to external devices that do not have a built in TDM.
		GPP50 (I/O)	General Purpose Pin 50.
PORTE[3]	I/O	CTS4* (Input)	Clear to Send input to MPSC4. Indicates to MPSC4 that data transmission may begin.
		TTSYNC2 (Input)	Transmit Frame Sync input to TDM channel2.
		GPP51 (I/O)	General Purpose Pin 51.
PORTE[4]	I/O	CD4 (Input)	Carrier Detect input to MPSC4. Indicates to MPSC4 that it can begin reception of data (input to MPSC4).
		TRCLK2 (Input)	Receive input clock to TDM channel2.
		GPP52 (I/O)	General Purpose Pin 52.

Table 11: WAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port E (Continued)			
PORTE[5]	I/O	SCLK4 (Input)	Input clock to MPSC4. Can be used as both transmit and receive clock. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OSCLK4 (Output)	Output clock from MPSC4. Can be used when SCLK4 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs).
		TRSYNC2 (Input)	Receive Frame Sync input to TDM channel2.
		GPP53 (I/O)	General Purpose Pin 53.
PORTE[6]	I/O	TSCLK4 (Input)	Input clock to MPSC4. Can be used by the MPSC transmitter when separate receive and transmit clocks are needed. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OTCLK4 (Output)	Output Tx clock from MPSC4. Can be used when TSCLK4 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs, or when there is no need for a separate Tx clock).
		TTCLK2 (Input)	Transmit input clock to TDM channel2.
		GPP54 (I/O)	General Purpose Pin 54.
Port F			
NOTE: Port F can be connected to MPSC5. See Section 20. "Physical Signal Routing" on page 414 for more details. When not connected to MPSC5, Port F can be connected to TDM3 or used as GPP. See Section 19. "General Purpose Ports" on page 405 for more details.			
PORTF[0]	I/O	RXD5 (Input)	Serial receive data input to MPSC5.
		TRXD3 (Input)	Serial receive data input to TDM channel3.
		TTXD3 (Output)	Serial transmit data from TDM channel3.
		GPP56 (I/O)	General Purpose Pin 56.
PORTF[1]	I/O	TXD5 (Output)	Serial transmit data output from the MPSC5.
		TTXD3 (Output)	Serial transmit data from TDM channel3.
		TRXD3 (Input)	Serial receive data input to TDM channel3.
		GPP57 (I/O)	General Purpose Pin 57.

Table 11: WAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port F (Continued)			
PORTF[2]	I/O	RTS5* (Output)	Request to Send output from MPSC5. Indicates that MPSC5 is ready to transmit data.
		TDSTRB3 (Output)	TDM channel3 strobe output signal, which can be used to gate clocks to external devices that do not have a built in TDM.
		GPP58 (I/O)	General Purpose Pin 58.
PORTF[3]	I/O	CTS5* (Input)	Clear to Send input to MPSC5. Indicates to MPSC5 that data transmission may begin.
		TTSYNC3 (Input)	Transmit Frame Sync input to TDM channel3.
		GPP59 (I/O)	General Purpose Pin 59.
PORTF[4]	I/O	CD5 (Input)	Carrier Detect input to MPSC5. Indicates to MPSC5 that it can begin reception of data (input to MPSC5).
		TRCLK3 (Input)	Receive input clock to TDM channel3.
		GPP60 (I/O)	General Purpose Pin 60.
PORTF[5]	I	SCLK5 (Input)	Input clock to MPSC5. Can be used as both transmit and receive clock. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OSCLK5 (Output)	Output clock from MPSC5. Can be used when SCLK5 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs).
		TRSYNC3 (Input)	Receive Frame Sync input to TDM channel3.
		GPP61 (I/O)	General Purpose Pin 61.
PORTF[6]	I/O	TSCLK5 (Input)	Input clock to MPSC5. Can be used by the MPSC transmitter when separate receive and transmit clocks are needed. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OTCLK5 (Output)	Output Tx clock from MPSC5. Can be used when TSCLK5 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs, or when there is no need for a separate Tx clock).
		TTCLK3 (Input)	Transmit input clock to TDM channel3.
		GPP62 (I/O)	General Purpose Pin 62.
WAN total: 42			

Table 12: LAN Pin Assignments

Pin Name	Type	Full Name	Description
<p>Port MII0</p> <p>NOTE: Port MII0 can be connected to MPSC6 and/or MPSC7. See Section 20. "Physical Signal Routing" on page 414 for more details.</p> <p>When not connected to MPSC6 and MPSC7, Port MII0 is connected to Ethernet 0 or used as GPP. See Section 19. "General Purpose Ports" on page 405 for more details.</p>			
MII0[0]	I/O	MTXEN0 - MII0 Transmit Enable (Output)	Indicates that a packet is being transmitted to the PHY. MTXEN0 is synchronous to MTXCLK0.
		GPP64 (I/O)	General Purpose Pin 64.
MII0[1]	I/O	MTXCLK0 - MII0 Transmit Clock (Input)	Provides the timing reference for the transfer of the MTXEN0, MTXD0 signals. It operates at either 25 MHz (100Mbps) or 2.5 MHz (10Mbps).
		TSCLK6 (Input)	Input clock to MPSC6. Can be used by the MPSC transmitter when separate receive and transmit clocks are needed. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OTSCLK6 (Output)	Output Tx clock from MPSC6. Can be used when TSCLK6 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs, or when there is no need for a separate Tx clock).
		GPP65 (I/O)	General Purpose Pin 65.
MII0[2]	I/O	MTXD0[3] - MII0 Transmit Data Bit [3] (Output)	Data nibble bit [3] output to the external PHY device. Synchronous to MTXCLK0.
		RTS6* (Output)	Request to Send output from MPSC6. Indicates that MPSC6 is ready to transmit data.
		GPP66 (I/O)	General Purpose Pin 66.
MII0[3]	I/O	MTXD0[2] - MII0 Transmit Data Bit [2] (Output)	Data nibble bit [2] output to the external PHY device. Synchronous to MTXCLK0.
		TXD6 (Output)	Serial transmit data output from the MPSC6.
		GPP67 (I/O)	General Purpose Pin 67.

Table 12: LAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port MII0 (Continued)			
MII0[4]	I/O	MTXD0[1] - MII0 Transmit Data Bit [1] (Output)	Data nibble bit [1] output to the external PHY device. Synchronous to MTXCLK0.
		RTS7* (Output)	Request to Send output from MPSC7. Indicates that MPSC7 is ready to transmit data.
		GPP68 (I/O)	General Purpose Pin 68.
MII0[5]	I/O	MTXD0[0] - MII0 Transmit Data Bit [0] (Output)	Data nibble bit [0] output to the external PHY device. Synchronous to MTXCLK0.
		TXD7 (Output)	Serial transmit data output from the MPSC7.
		GPP69 (I/O)	General Purpose Pin 69.
MII0[6]	I/O	MCOLO - MII0 Collision detect (Input)	Indicates that a collision has been detected on the wire. NOTE: This input is ignored in half - and full-duplex mode when MTxEn0 is LOW. MCoI0 is asynchronous.
		TSCLK7 (Input)	Input clock to MPSC7. Can be used by the MPSC transmitter when separate receive and transmit clocks are needed. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OTCLK7 (Output)	Output Tx clock from MPSC7. Can be used when TSCLK7 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs, or when there is no need for a separate Tx clock).
		GPP70 (I/O)	General Purpose Pin 70.
MII0[7]	I/O	MRXD0[3] - MII0 Receive Data Bit [3] (Input)	Data nibble bit [3] input from external PHY. Synchronous to MRXCLK0.
		CTS6* (Input)	Clear to Send input to MPSC6. Indicates to MPSC6 that data transmission may begin.
		GPP71 (I/O)	General Purpose Pin 71.
MII0[8]	I/O	MRXD0[2] - MII0 Receive Data Bit [2] (Input)	Data nibble bit [2] input from external PHY. Synchronous to MRXCLK0.
		RXD6 (Input)	Serial receive data input to MPSC6.
		GPP72 (I/O)	General Purpose Pin 72.

Table 12: LAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port MII0 (Continued)			
MII0[9]	I/O	MRXD0[1] - MII0 Receive Data Bit [1] (Input)	Data nibble bit [1] input from external PHY. Synchronous to MRXCLK0.
		CTS7* (Input)	Clear to Send input to MPSC7. Indicates to MPSC7 that data transmission may begin.
		GPP73 (I/O)	General Purpose Pin 73.
MII0[10]	I/O2	MRXD0[0] - MII0 Receive Data Bit [0] (Input)	Data nibble bit [0] input from external PHY. Synchronous to MRXCLK0.
		RXD7 (Input)	Serial receive data input to MPSC7.
		GPP74 (I/O)	General Purpose Pin 74.
MII0[11]	I/O	MRXER0 - MII0 Receive Error (Input)	Indicates that an error was detected in the received frame. This input is ignored when MRXDV0 is inactive.
		CD6 (Input)	Carrier Detect input to MPSC6. Indicates to MPSC6 that it can begin reception of data.
		GPP75 (I/O)	General Purpose Pin 75.
MII0[12]	I/O	MRXCLK0 (Input)	Provides the timing reference for the transfer of the MRXDV0, MRXD0 and MRXER0 signals. Operates at either 25 MHz (100Mbps) or 2.5 MHz (10Mbps). NOTE: The nominal frequency of MRXCLK0 must match the nominal frequency of MTXCLK0.
		SCLK6 (Input)	Input clock to MPSC6. Can be used as both transmit and receive clock. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OSCLK6 (Output)	Output clock from MPSC6. Can be used when SCLK6 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMS).
		GPP76 (I/O)	General Purpose Pin 76.

Table 12: LAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port MII0 (Continued)			
MII0[13]	I/O	MRXDV0 - MII0 Receive Data Valid (Input)	Indicates that valid data is present on RXD0 lines. Synchronous to MRXCLK0.
		SCLK7 (Input)	Input clock to MPSC7. Can be used as both transmit and receive clock. NOTE: This clock also serves as one of the input clocks to the Baud Rate Generators.
		OSCLK7 (Output)	Output clock from MPSC7. Can be used when SCLK7 is not needed (i.e. the MPSC is programmed to use one of the Baud Rate Generators as its clock source or is connected to one of the TDMs).
		GPP77 (I/O)	General Purpose Pin 77.
MII0[14]	I/O	MCRS0 - MII0 Carrier Sense (Input)	In half duplex mode, indicates that either the transmit or receive medium is non-idle. NOTE: MCRS0 is ignored in full-duplex.
		CD7 (Input)	Carrier Detect input to MPSC7. Indicates to MPSC7 that it can begin reception of data.
		GPP78 (I/O)	General Purpose Pin 78.
Port MII1			
MII1[0]	I/O	MTXEN1 - MII1 Transmit Enable (Output)	Indicates that a packet is being transmitted to the PHY. MTXEN1 is synchronous to MTXCLK1.
		GPP80 (I/O)	General Purpose Pin 80.
		RMTXEN1 (Output)	Transmit enable for Ethernet port 1.
MII1[1]	I/O	MTXCLK1 - MII1 Transmit Clock (Input)	Provides the timing reference for the transfer of the MTXEN1, MTXD1 signals. It operates at either 25 MHz (111Mbps) or 2.5 MHz (11Mbps). NOTE: Port MII1 can be connected to Ethernet 1. See Section 20. “Physical Signal Routing” on page 414 for more details. When not connected to Ethernet 1, Port MII1 is connected to Ethernet 0 and Ethernet 1 RMII interfaces or used as GPP. See Section 19. “General Purpose Ports” on page 405 for more details.
		GPP81 (I/O)	General Purpose Pin 81.
		RM50CLK (input)	RMII clock input (50MHz)

Table 12: LAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port MII1 (Continued)			
MII1[2]	I/O	MTXD1[3] - MII1 Transmit Data Bit [3] (Output)	Data nibble bit [3] output to the external PHY device. Synchronous to MTXCLK1.
		GPP82 (I/O)	General Purpose Pin 82.
		RMTXD0[1] (Output)	RMI1 transmit data bit [1] for port 0.
MII1[3]	I/O	MTXD1[2] - MII1 Transmit Data Bit [2] (Output)	Data nibble bit [2] output to the external PHY device. Synchronous to MTXCLK1.
		GPP83 (I/O)	General Purpose Pin 83.
		RMTXD0[0] (Output)	RMI1 transmit data bit [0] for port 0.
MII1[4]	I/O	MTXD1[1] - MII1 Transmit Data Bit [1] (Output)	Data nibble bit [1] output to the external PHY device. Synchronous to MTXCLK1.
		GPP84 (I/O)	General Purpose Pin 84.
		RMTXD1[1] (Output)	RMI1 transmit data bit [1] for port 1.
MII1[5]	I/O	MTXD1[0] - MII1 Transmit Data Bit [0] (Output)	Data nibble bit [0] output to the external PHY device. Synchronous to MTXCLK1.
		RMTXD1[0] (Output)	RMI1 transmit data bit [0] for port 1.
		GPP85 (I/O)	General Purpose Pin 85.
MII1[6]	I/O	MCOL1 - MII1 Collision detect (Input)	Indicates that a collision has been detected on the wire. NOTE: This input is ignored in half- and full-duplex mode when MTxEn1 is LOW. MCol1 is asynchronous.
		GPP86 (I/O)	General Purpose Pin 86.
		RMTXEN0 (Output)	RMI1 transmit enable for Ethernet port 0.

Table 12: LAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port MII1 (Continued)			
MII1[7]	I/O	MRXD1[3] - MII1 Receive Data Bit [3] (Input)	Data nibble bit [3] input from external PHY. Synchronous to MRXCLK1.
		GPP87 (I/O)	General Purpose Pin 87.
		RMRXD0[1] (Input)	RMI1 receive data bit [1] for port 0.
MII1[8]	I/O	MRXD1[2] - MII1 Receive Data Bit [2] (Input)	Data nibble bit [2] input from external PHY. Synchronous to MRXCLK1.
		GPP88 (I/O)	General Purpose Pin 88.
		RMRXD0[0] (Input)	RMI1 receive data bit [0] for port 0.
MII1[9]	I/O	MRXD1[1] - MII1 Receive Data Bit [1] (Input)	Data nibble bit [1] input from external PHY. Synchronous to MRXCLK1.
		GPP89 (I/O)	General Purpose Pin 89.
		RMRXD1[1] (Input)	RMI1 receive data bit [1] for port 1.
MII1[10]	I/O	MRXD1[0] - MII1 Receive Data Bit [0] (Input)	Data nibble bit [0] input from external PHY. Synchronous to MRXCLK1.
		GPP90 (I/O)	General Purpose Pin 90.
		RMRXD1[0] (Input)	RMI1 receive data bit [0] for port 1.
MII1[11]	I/O	MRXER1 - MII1 Receive Error (Input)	Indicates that an error was detected in the received frame. NOTE: This input is ignored when MRXDV1 is inactive.
		GPP91 (I/O)	General Purpose Pin 91.

Table 12: LAN Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
Port MII1 (Continued)			
MII1[12]	I/O	MRXCLK1 - MII1 Receive Clock (Input)	Provides the timing reference for the transfer of the MRXDV1, MRXD1 and MRXER1 signals. Operates at either 25 MHz (111Mbps) or 2.5 MHz (11Mbps). NOTE: The nominal frequency of MRXCLK1 must match the nominal frequency of MTXCLK1.
		GPP92 (I/O)	General Purpose Pin 92.
MII1[13]	I/O	MRXDV1 - MII1 Receive Data Valid (Input)	Indicates that valid data is present on RXD1 lines. Synchronous to MRXCLK1.
		GPP93 (I/O)	General Purpose Pin 93.
		RMCRSDV1 (Input)	RMII CRS_DV for port 1.
MII1[14]	I/O	MCRS1 - MII1 Carrier Sense (Input)	In half-duplex mode, indicates that either the transmit or receive medium is non-idle. MCRS1 is ignored in full-duplex.
		GPP94 (I/O)	General Purpose Pin 94.
		RMCRSDV0 (Input)	RMII CRS_DV for port 0.
Port MDC and MDIO			
MDC	O	MII Management Interface Clock Signal	MII management serial data transfers are clocked by this clock output.
MDIO	I/O	MII Management Interface Data Signal	MII management serial data to and from the PHYs is transmitted on this line.
LAN total: 32			

Table 13: GPP Pin Assignments

Pin Name	Type	Full Name	Description
GPP[0]	I/O	BCLK0 (Input)	One of the input clocks that can be used for the Baud Rate Generators.
		GPP0 (I/O)	General purpose I/O pin for system use.

Table 13: GPP Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
GPP[1]	I/O	BCLK1 (Input)	One of the input clocks that can be used for the Baud Rate Generators.
		GPP1 (I/O)	General purpose I/O pin for system use.
GPP[2]	I/O	BRGO0 (Output)	Baud Rate Generator's output (from Baud Rate Generator 0).
		GPP2 (I/O)	General purpose I/O pin for system use.
GPP[3]	I/O	BRGO1 (Output)	Baud Rate Generator's output (from Baud Rate Generator 1).
		GPP3 (I/O)	General purpose I/O pin for system use.
GPP[4]	I/O	TRCLK0 (Input)	Input receive clock to TDM channel0.
		GPP4 (I/O)	General purpose I/O pin for system use.
GPP[5]	I/O	TRCLK1 (Input)	Input receive clock to TDM channel1.
		GPP5 (I/O)	General purpose I/O pin for system use.
GPP[6]	I/O	TRCLK2 (Input)	Input receive clock to TDM channel2.
		OTSCLK4 (Output)	Output Tx clock from MPSC4.
		GPP6 (I/O)	General purpose I/O pin for system use.
GPP[7]	I/O	TRCLK3 (Input)	Input receive clock to TDM channel3.
		OTSCLK5 (Output)	Output Tx clock from MPSC5.
		GPP7 (I/O)	General purpose I/O pin for system use.
GPP[8]	I/O	TOCLK0 (Output)	Output clock from TDM channel0.
		GPP8 (I/O)	General purpose I/O pin for system use.
GPP[9]	I/O	TOCLK1 (Output)	Output clock from TDM channel1.
		GPP9 (I/O)	General purpose I/O pin for system use.
GPP[10]	I/O	TOCLK2 (Output)	Output clock from TDM channel2.
		OTSCLK2 (Output)	Output Tx clock from MPSC2.
		GPP10 (I/O)	General purpose I/O pin for system use.

Table 13: GPP Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
GPP[11]	I/O	TOCLK3 (Output)	Output clock from TDM channel3.
		OTSCLK3 (Output)	Output Tx clock from MPSC3.
		GPP11 (I/O)	General purpose I/O pin for system use.
GPP[12]	I/O/OD		A general purpose I/O pin for system use. When configured as functional output, this pin features an open-drain output. NOTE: See Section 19. "General Purpose Ports" on page 405 for details about GPP configuration options.
GPP[13]	I/O/OD		A general purpose I/O pin for system use. When configured as functional output, this pin features an open-drain output. NOTE: See Section 19. "General Purpose Ports" on page 405 for details about GPP configuration options.
GPP[14]	I/O		A general purpose I/O pin for system use.
GPP[15]	I/O		A general purpose I/O pin for system use.
GPP Total: 16			

Table 14: Interrupt Interface Pin Assignments

Pin Name	Type	Full Name	Description
Interrupt0*	I/O	Interrupt0	Driven by the GT-96100A to signal that one (or more) of the internal (unmasked) interrupt sources within the GT-96100A is set.
Interrupt1*	OD	Interrupt1	Driven by the GT-96100A to signal that one (or more) of the internal (unmasked) interrupt sources within the GT-96100A is set. This pin features an open-drain output.
SerInt0*	I/O	Serial Interrupt0	Driven by the GT-96100A to signal that one (or more) of the internal (unmasked) SERIAL interrupt sources within the GT-96100A is set. This signal is dedicated only to interrupt events that occur within the communication unit.
SerInt1*	I/O	Serial Interrupt1	Driven by the GT-96100A to signal that one (or more) of the internal (unmasked) SERIAL interrupt sources within the GT-96100A is set. This signal is dedicated only to interrupt events that occur within the communication unit.
Interrupt Total: 4			

Table 15: Watchdog Interface Pin Assignments

Pin Name	Type	Full Name	Description
WDE*	I/O	Watchdog Expired (Output)	Asserts to indicate that the watchdog timer in the GT-96100A expired.
NMI*	I/O	Non Maskable Interrupt (Output)	Asserted by the GT-96100A to indicate that a non maskable interrupt is pending.
Watchdog Total: 2			

Table 16: Test Interface Pin Assignments

Pin Name	Type	Full Name	Description
JTAG[0]	I	TCLK - JTAG Clock	Input clock for test logic. TMS and TDI are received on the rising edge, TDO is driven from the falling edge. This signal determines the shifting rate.
JTAG[1]	I	TMS - JTAG Mode Select	A broadcast test signal that controls test logic operation.
JTAG[2]	I	TDI - JTAG Data In	Serial data input.
JTAG[3]	O	TDO - JTAG Data Out	Serial data output. Tristate changes on negative change of JTCLK.
JTAG[4]	I	TRST - JTAG RESET	Asynchronous reset to the JTAG controller.
TEST Total: 5			

Table 17: Clock/Control Interface Pin Assignments

Pin Name	Type	Full Name	Description
TCIk	I	Clock	Master clock input to the GT-96100A (up to 100MHz). TCIk is used for the SysAD interface and the Device interface. TCIk must be driven for ALL applications.
Reset*	I	Reset	Must be asserted for any reset sequence.
VccPLL	I	Vcc for PLL	Used for supplying a low-noise power to the internal PLL.
VssPLL	I	Vss for PLL	Used for supplying a low-noise ground to the internal PLL.
ByPassPLL	I	ByPass PLL	This is used for PLL bypass mode: 0 - PLL is not bypassed (normal mode) 1 - PLL is bypassed

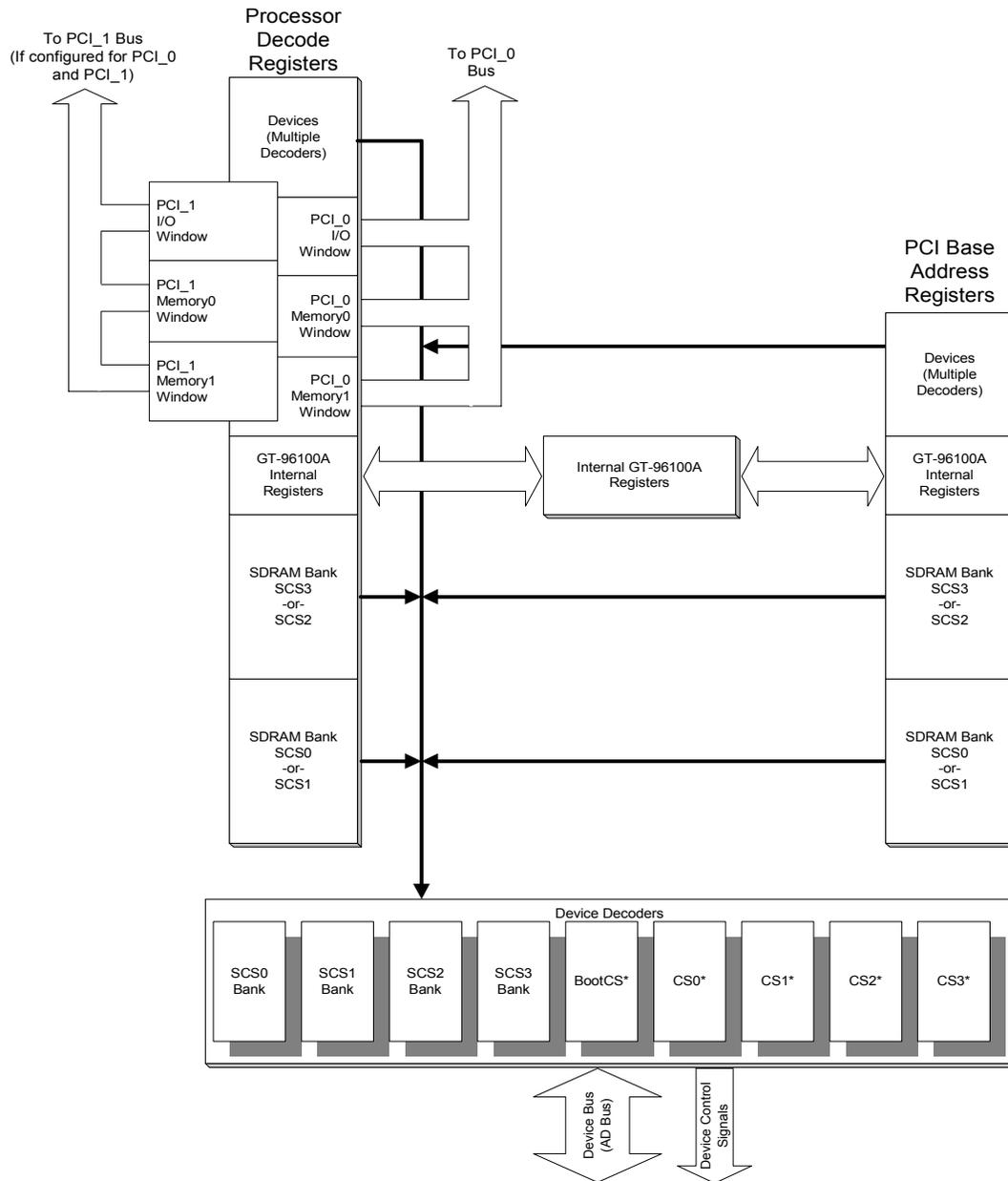
Table 17: Clock/Control Interface Pin Assignments (Continued)

Pin Name	Type	Full Name	Description
OutModePLL	I	Output Mode PLL Clock	This pin controls the clock output function of the PLL: 0 - PLL generated clock is driven on ClkOutPLL pin. 1 - PLL generated clock is not driven out (i.e. ClkOutPLL pin is in hi-z state).
ClkOutPLL	O	Clock Output for PLL	This pin is used for driving the PLL generated clock. To enable this pin, the OutModePLL pin must be pulled low.
Clock/Control Total: 7			

3. ADDRESS SPACE DECODING

The GT-96100A has a fully programmable address map. Two address spaces exist: the CPU address space and the PCI address space (see Figure 2.) Both address maps use a two-stage decoding process where major device regions are decoded first, then the individual devices are subdecoded.

Figure 2: Two Stage Address Decoding- Conceptual View



3.1 Two Stage Decoding Process

The system resources are divided into the following groups:

- SCS[1:0]*
- SCS[3:2]*
- CS[2:0]*
- CS[3] & BootCS*
- Internal Registers
- PCI_0 I/O
- PCI_0 Memory0/1
- PCI_1 I/O
- PCI_1 Memory0/1

NOTE: PCI_1 I/O and PCI_1 Memory0/1 will only exist if the GT-96100A is configured for both PCI_0 and PCI_1.

Each group can have a minimum of 2 Mbytes and a maximum of 4 Gbytes of address space. The individual devices in the device groups are further sub decoded to 1 Mbyte resolution. [Table 18](#) shows the CPU decode and device sub-decode associations. [Table 19](#) shows the same process for PCI_0 and [Table 20](#) shows the process for PCI_1.

Table 18: CPU and Device Decoder Mappings

CPU Decoder	Associated Device Sub-Decoders
SCS[1:0]*	SCS[0]* SCS[1]*
SCS[3:2]*	SCS[2]* SCS[3]*
CS[2:0]*	CS0* CS1* CS2*
BootCS*/CS3*	BootCS* CS3*
PCI_0 I/O	None. Accesses decoded for PCI_0 I/O are bridged to PCI_0 I/O transfers.
PCI_0 Memory 0/1	None. Accesses decoded for PCI_0 Memory 0/1 are bridged to PCI Memory transfers.
PCI_1 I/O	None. Accesses decoded for PCI_1 I/O are bridged to PCI_1 I/O transfers.
PCI_1 Memory 0/1	None. Accesses decoded for PCI_1 Memory 0/1 are bridged to PCI_1 Memory transfers.
Internal	None. Decodes to the GT-96100A internal registers.

Table 19: PCI_0 Base Address Register and Device Decoder Mappings

PCI Base Address Register (BAR) Decoder¹	Associated Device Sub-Decoders
SCS[1:0]* - BAR 0 at 0x10	SCS0* SCS1*
SCS[3:2]* - BAR 1 at 0x14	SCS2* SCS3*
CS[2:0]* - BAR 2 at 0x18	CS0* CS1* CS2*
BootCS*/CS3* - BAR 3 at 0x1C	BootCS* Cs3*
Internal Registers (Memory) - BAR 4 at 0x20	None Decodes PCI_0 memory accesses to the GT-96100A internal registers.
Internal Registers (I/O) - BAR 5 at 0x24	None. Decodes PCI_0 I/O accesses to the GT-96100A internal registers.
Expansion ROM - BAR at 0x30	None. Decodes directly to CS3*.

1. This mapping also applies to the swap BARs located in PCI function 1, if enabled.

Table 20: PCI_1 Base Address Register and Device Decoder Mappings

PCI Base Address Register (BAR) Decoder¹	Associated Device Sub-Decoders
SCS[1:0]* - BAR 0 at 0x90	SCS0* SCS1*
SCS[3:2]* - BAR 1 at 0x94	SCS2* SCS3*
CS[2:0]* - BAR 2 at 0x98	CS0* CS1* CS2*
BootCS*/CS3* - BAR 3 at 0x9C	BootCS* Cs3*
Internal Registers (Memory) - BAR 4 at 0xa0	None. Decodes PCI_1 memory accesses to the GT-96100A internal registers.

Table 20: PCI_1 Base Address Register and Device Decoder Mappings (Continued)

PCI Base Address Register (BAR) Decoder¹	Associated Device Sub-Decoders
Internal Registers (I/O) - BAR 5 at 0xa4	None. Decodes PCI_1 I/O accesses to the GT-96100A internal registers.

1. This mapping also applies to the swap BARs located in PCI function 1, if enabled.

3.1.1 CPU Side Decoding Process

Decoding on the CPU side starts with the SysAD address being compared with the values in the various CPU Low and High decoder registers. For example, the SCS[1:0]* CPU High and Low decoder registers set the address range in which the SCS0* and SCS1* signals are active (i.e. where DRAM banks 0 and 1 are located.) The comparison works as follows:

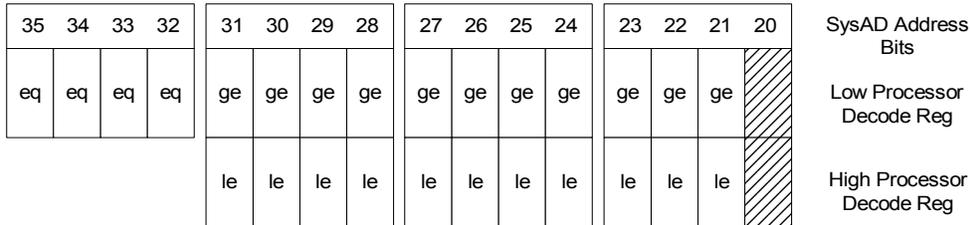
1. Bits 35:32 of the SysAD address are compared against bits 14:11 in the various CPU Low decode registers. These values must match exactly. This effectively sets a 4 Gbyte “page” for the resource group.
2. Bits 31:21 of the SysAD address are compared against bits 10:0 in the various CPU Low decode registers. The value of the SysAD bits must be greater than or equal to the Low decode value. This sets the lower boundary for the region.
3. Bits 31:21 of the SysAD address are compared against the High decode registers. The value of the SysAD bits must be less than or equal to this value. This sets the upper bound for the region.
4. If all of the above are true, then the resource group is selected and a subdecode is performed to determine the specific resource.

An example of the CPU resource group decode process is shown in [Figure 3](#).

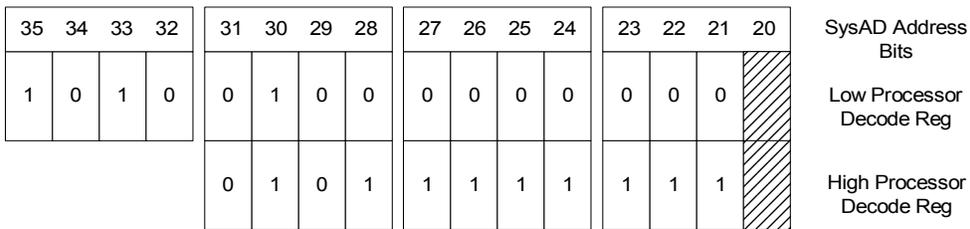
Figure 3: CPU-Side Resource Group Decode Function and Example

'eq' equal to
 'ge' greater or equal to
 'le' less or equal to

If the SysAD address is between the Low and the High decode addresses, then the access is passed to the Device Unit for sub-decode.



Example: Set up a SysAD decode region that starts at 0xA.4000.0000 and is 512Mbytes in length (0xA.4000.0000 to 0xA.5FFF.FFFF):

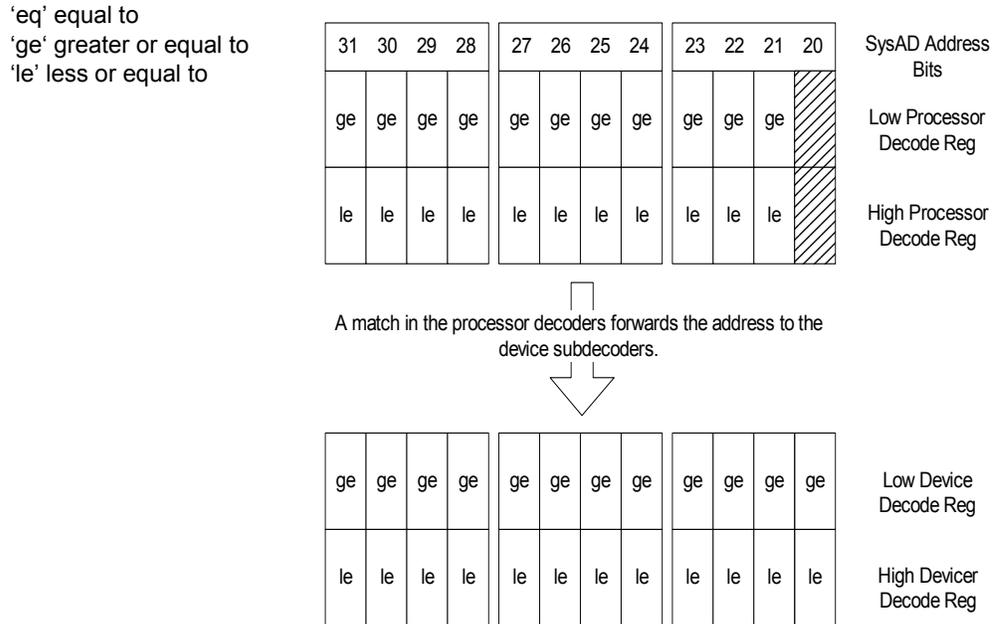


Once a CPU resource group has been decoded, it must be subdecoded to determine which physical device should be accessed within that group. This decoding is controlled by the device Low and High decode registers. The comparison works as follows:

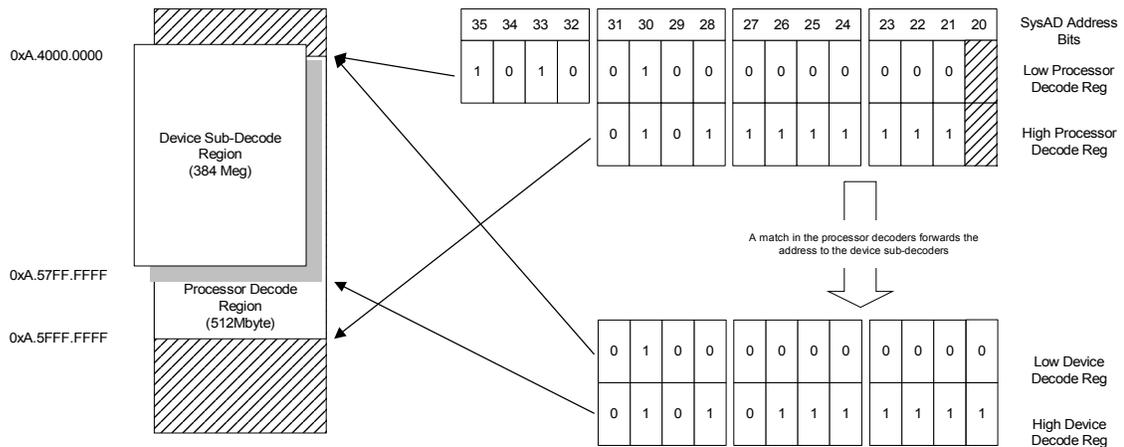
1. Bits 31:20 of the SysAD address are compared against the relevant device Low decode registers. The value of the SysAD bits must be greater than or equal to the Low decode value. This sets the lower boundary for the sub-decode region.
2. Bits 31:20 of the SysAD address are compared against the relevant device High decode registers. The value of the SysAD bits must be less than or equal to this value. This sets the upper bound for the sub-decode region.
3. If all of the above are true, then the specific device is selected and an access to that device is performed.

Figure 4 illustrates the device decode process that occurs after the CPU resource group has been decoded.

Figure 4: Device Sub-Decode Function and Example



Example: Using the previous CPU decode example (0xA.4000.0000 to 0xA.5FFF.FFFF), place a device sub-decode in the first 384 Meg (0xA.4000.0000 to 0xA.57FF.FFFF).



3.1.2 PCI Side Decoding Process

Decoding on the PCI side starts with the PCI address being compared with the values in the various Base Address Registers. For example, the SCS[1:0]* Base Address register sets the PCI base address range in which the SCS0* and SCS1* signals are active (i.e., where DRAM banks 0 and 1 are located in PCI space).

Once a resource group has been decoded by a BAR, it must be subdecoded to determine which physical device should be accessed within that group. This decoding is controlled by the Device Low and High decode registers.

NOTE: These registers are the same ones used for CPU-side decoding. This means that the PCI and SysAD memory maps are coupled at the device decoders. Address bits 31:20 (the bits compared by the Device decoders) for any given device overlap in both the PCI and SysAD maps.

The sub-decoding comparison works as follows:

1. Bits 31:20 of the PCI address are compared against bits the relevant device Low decode registers. The value of the PCI address bits must be greater than or equal to the Low decode value. This sets the lower boundary for the sub-decode region.
2. Bits 31:20 of the PCI address are compared against the relevant device's High decode registers. The value of the PCI address bits must be less than or equal to this value. This sets the upper bound for the sub-decode region.
3. If all of the above are true, the specific device is selected and an access to that device is performed.

Figure 5: Bank Size Register Function Example (16Meg Decode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	Bank Size Reg
=	=	=	=	=	=	=	=	x	x	x	x	x	x	x	x	x	x	x	x	Comparison against PCI address

'=' must match exactly
 'x' don't care

3.2 Disabling Address Decoders

CPU interface address decoders can be disabled by setting the Low decoder value higher than the High decoder value.

Device sub-decoder can be disabled by setting the Low decoder value higher than the High decoder value.

PCI address decoders can be disabled by setting the BAR's corresponding bit in Base Address registers' Enable to 1.

3.3 DMA Unit Address Decoding

The IDMA controller uses the address mapping of the CPU interface.

NOTE: CPU interface address mapping to determine whether the source address is located in one of the SDRAM banks, Device banks, PCI_0 or PCI_1. The same is true for destination address and next record address. DMA address decoding is only up to address bit [31]. Bits [35:32] of CPU address decoding registers are ignored.

3.4 Address Space Decoding Errors

When the CPU tries to access an address from the SysAD that is not supported, the GT-96100A:

- Latches the address into the Bus Error Address registers (offsets 0x70,0x78).
- Issues a bus error (over SysCmd[5]), if the access was a read access.
- Issues an interrupt, if the access was a read or write access.

This feature is useful during software debugging, when errant code can cause fetches from unsupported addresses.

When SysAD matches one of CPU interface address spaces, but misses the associated subdecoders, the GT-96100A:

- Issues a bus error (over SysCmd[5]), if the access was a read access.
- Sets the MemOut bit in the Interrupt Cause register.

When a PCI access hits in a Base Address register then misses in the associated subdecoders:

- Random data is returned on a read and write data is discarded.
- Latches the address into the Address Decode Error register (offset 0x470).
- The MemOut bit in the Interrupt Cause register is also set.

Accesses that miss all of the GT-96100A BARs result in no response at all from the GT-96100A.

When a DMA accesses an unmapped address, DMAOut bit in the interrupt Cause register is set.

NOTE: Never program address space decoders to overlap. Programming address space decoders to overlap results in unpredictable behavior.

3.5 Default Memory Map

The default CPU memory map that is valid following RESET is shown in [Table 21](#). The default PCI map and BAR sizing information is shown in [Table 22](#) and [Table 23](#).

Table 21: CPU and Device Decoder Default Address Mapping

CPU Decode Range and Size	Resource Group	Device Decode Range and Size	Device Selected
0x0 to 0x0.00FF.FFFF 16 Megabytes	SCS[1:0]*	0x0 to 0x0.007F.FFFF 8 Megabytes	SCS0*
		0x0.0080.0000 to 0x0.00FF.FFFF 8 Megabytes	SCS1*
0x0.0100.0000 to 0x0.01FF.FFFF 16 Megabytes	SCS[3:2]*	0x0.0100.0000 to 0x0.017F.FFFF 8 Megabytes	SCS2*
		0x0.0180.0000 to 0x0.01FF.FFFF 8 Megabytes	SCS3*
0x0.1400.0000 to 0x0.1400.0FFF 4 Kbytes	Internal Registers	No subdecode. Access bridged directly to the GT-96100A internal registers	Internal Registers
0x0.1000.0000 to 0x0.11FF.FFFF 32 Megabytes	PCI0 I/O	No subdecode. Access bridged directly to PCI I/O space	PCI0
0x0.1200.0000 to 0x0.13FF.FFFF 32 Megabytes	PCI0 Mem0	No subdecode. Access bridged directly to PCI memory space	PCI0
0x0.1C00.0000 to 0x0.1E1F.FFFF 34 Megabytes	CS[2:0]	0x0.1C00.0000 to 0x0.1C7F.FFFF 8 Megabytes	CS0*
		0x0.1C80.0000 to 0x0.1CFF.FFFF 8 Megabytes	CS1*
		0x0.1D00.0000 to 0x0.1DFF.FFFF 16 Megabytes	CS2*
		0x0.1E00.0000 to 0x0.1E1F.FFFF not accessible	
0x0.1F00.0000 to 0x0.1FFF.FFFF 16 Megabytes	CS[3] and BootCS*	0x0.1F00.0000 to 0x0.1FBF.FFFF 12 Megabyte	CS3*
		0x0.1FC0.0000 to 0x0.1FFF.FFFF 4 Megabytes	BootCS*

Table 21: CPU and Device Decoder Default Address Mapping (Continued)

CPU Decode Range and Size	Resource Group	Device Decode Range and Size	Device Selected
0x0.F200.0000 to 0x0.F3FF.FFFF 32 Megabytes	PCI0 Mem1	No subdecode. Access bridged directly to PCI memory space	PCI0
0x0.2000.0000 to 0x0.21FF.FFFF 32 Megabytes	PCI1 I/O	No subdecode Access bridged directly to PCI I/O space	PCI1
0x0.2200.0000 to 0x0.23FF.FFFF 32 Megabytes	PCI1 Mem0	No subdecode. Access bridged directly to PCI memory space	PCI1
0x0.2400.0000 to 0x0.25FF.FFFF 32 Megabytes	PCI1 Mem1	No subdecode. Access bridged directly to PCI memory space	PCI1

Table 22: PCI Function 0 and Device Decoder Default Address Mapping

PCI Function 0 Decode Range and Size	Resource Group	Device Decode Range and Size	Device Selected
0x0 to 0x0.00FF.FFFF 16 Megabytes in Memory Space	SCS[1:0]*	0x0 to 0x0.007F.FFFF 8 Megabytes	SCS0*
		0x0.0080.0000 to 0x0.00FF.FFFF 8 Megabytes	SCS1*
0x0.0100.0000 to 0x0.01FF.FFFF 16 Megabytes in Memory Space	SCS[3:2]*	0x0.0100.0000 to 0x0.017F.FFFF 8 Megabytes	SCS2*
		0x0.0180.0000 to 0x0.01FF.FFFF 8 Megabytes	SCS3*
0x0.1400.0000 to 0x0.1400.0FFF 4 Kbytes in Memory Space	Internal Registers	No subdecode	Internal Registers
0x0.1400.0000 to 0x0.1400.0FFF 4 Kbytes in I/O Space	Internal Registers	No subdecode.	Internal Registers

Table 22: PCI Function 0 and Device Decoder Default Address Mapping (Continued)

PCI Function 0 Decode Range and Size	Resource Group	Device Decode Range and Size	Device Selected
0x0.1C00.0000 to 0x0.1DFF.FFFF 32 Megabytes in Memory Space	CS[2:0]	0x0.1C00.0000 to 0x0.1C7F.FFFF 8 Megabytes	CS0*
		0x0.1C80.0000 to 0x0.1CFF.FFFF 8 Megabytes	CS1*
		0x0.1D00.0000 to 0x0.1DFF.FFFF 16 Megabytes	CS2*
0x0.1F00.0000 to 0x0.1FFF.FFFF 16 Megabytes in Memory Space	CS[3] and BootCS*	0x0.1F00.0000 to 0x0.1FBF.FFFF 12 Megabyte	CS3*
		0x0.1FC0.0000 to 0x0.1FFF.FFFF 4 Megabytes	BootCS*
0x0.1F00.000 to 0x0.1FFF.FFFF 16 Megabytes (uses CS[3] and BootCS* size register)	PCI Expansion ROM	No subdecode. This decoder is used only during PC BIOS initialization.	CS3*

Table 23: PCI Function 1 (Byte Order Swap) and Device Decoder Default Address Mapping

PCI Function 0 Decode Range and Size	Resource Group	Device Decode Range and Size	Device Selected
0x0 to 0x0.00FF.FFFF 16 Megabytes in Memory Space	SCS[1:0]*	0x0 to 0x0.007F.FFFF 8 Megabytes	SCS0*
		0x0.0080.0000 to 0x0.00FF.FFFF 8 Megabytes	SCS1*
0x0.0100.0000 to 0x0.01FF.FFFF 16 Megabytes in Memory Space	SCS[3:2]*	0x0.0100.0000 to 0x0.017F.FFFF 8 Megabytes	SCS2*
		0x0.0180.0000 to 0x0.01FF.FFFF 8 Megabytes	SCS3*

Table 23: PCI Function 1 (Byte Order Swap) and Device Decoder Default Address Mapping (Continued)

PCI Function 0 Decode Range and Size	Resource Group	Device Decode Range and Size	Device Selected
0x0.1F00.0000 to 0x0.1FFF.FFFF 16 Megabytes in Memory Space	CS[3]* and BootCS*	0x0.1F00.0000 to 0x0.1FBF.FFFF 12 Megabyte	CS3*
		0x0.1FC0.0000 to 0x0.1FFF.FFFF 4 Megabytes	BootCS*

3.6 Address Remapping

The GT-96100A supports address remapping on both the CPU interface and the PCI interfaces.

NOTE: Although the IDMA controllers use the CPU address decode registers, the source and destination DMA addresses are NEVER remapped.

3.6.1 CPU Address Remapping

The resources that can be addressed by the CPU are the following:

- SDRAM banks (SCS[1:0]*, SCS[3:2]*)
- Devices (CS[2:0]*, CS[3]* & BootCS*)
- PCI_0 IO
- PCI_0 Memory0/1
- PCI_1 IO
- PCI_1 Memory0/1

NOTE: PCI_1 IO and PCI_1 Memory0/1 are only addressable if the device is configured for both PCI_0 and PCI_1 on RESET.

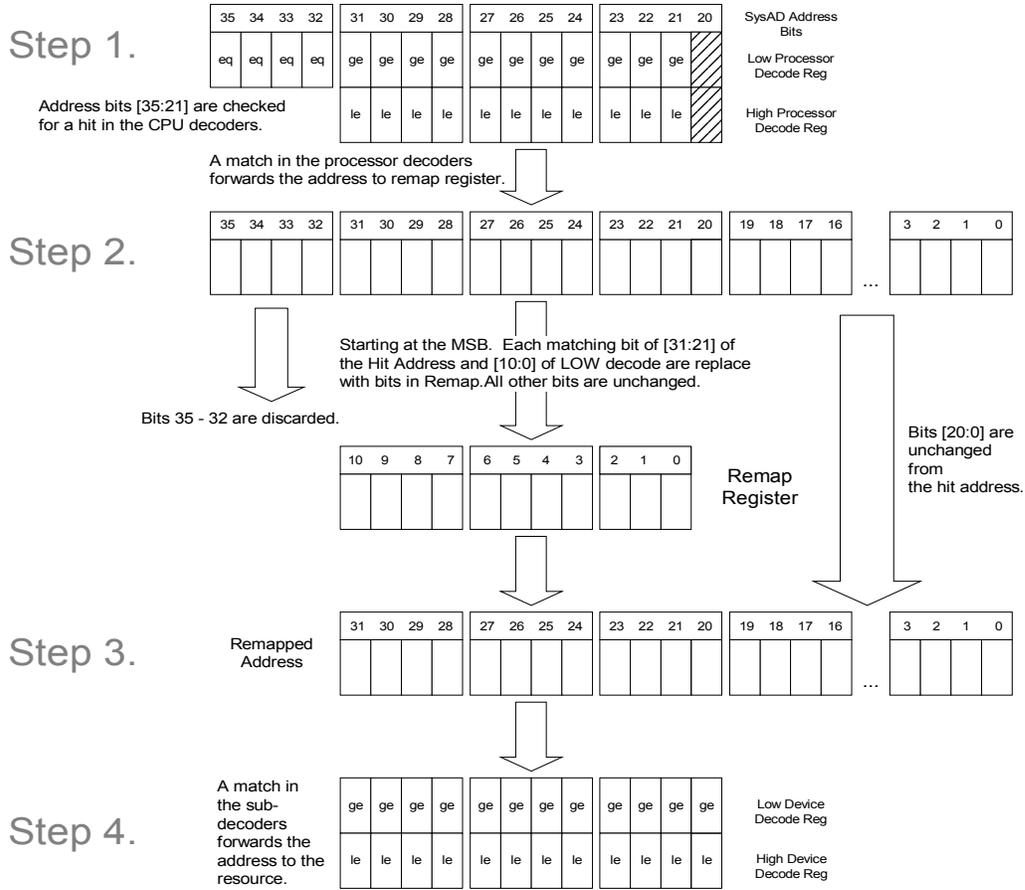
Each resource addressed by the CPU has a Remap register associated with it. These registers are listed in [Section 4.10.2 “CPU Address Decode Registers” on page 87](#).

An address presented on the SysAD bus by the CPU is decoded with the following steps:

1. Address bits [35:21] are checked for a hit in the CPU decoders.
2. Assuming there is a hit in the CPU decoders, the HIT address will have bits 35:32 discarded. Bits 20:0 are left unchanged. Bits[31:21] are remapped as follows: Going from the most significant bit (MSB) to least significant bit (LSB) of the HIT address bits [31:21], any bit found matching to its respective bit in the LOW decode register’s bits [10:0] will cause the according bit in the remap register to REPLACE the original address bit. Upon first mismatch, all remaining LSBs of address bits[31:21] are unchanged.
3. Address bits [31:20] of the *remapped address* are checked to be a hit in the Device decoders.
4. Assuming there is a hit in the Device decoders, the HIT address will be transferred to the resource.

See [Figure 6](#) outlining this address remapping procedure.

Figure 6: CPU Address Remapping To Resources



3.6.2 Writing to Decode Registers

When a LOW decode register is written to, the least significant 11 bits are written to the associated remap register, simultaneously.

When a remap register is written to, only its contents are affected.

Following RESET, the default value of a remap register is equal to its associated LOW Decode register bits [10:0].

Unless a specific write operation to a remap register takes place, a 1:1 mapping is maintained. Also, changing a LOW Decode register's contents automatically returns its associated space to a 1:1 mapping. This allows for backward software compatibility with other Galileo Technology devices such as the GT-64010A and the GT-64011 core logic devices.

3.6.3 PCI Address Remapping

The PCI slave interface has the ability to remap addresses of PCI transactions to memory using PCI remap registers. There are seven registers for PCI_0 and seven registers for PCI_1, if implemented.

These registers correspond to the following PCI Base Address Registers:

- SCS[1:0]*
- SCS[3:2]*
- CS[2:0]*
- CS[3]* & BootCS*
- Swapped SCS[1:0]*
- Swapped SCS[3:2]*
- Swapped CS[3]* and BootCS*

These registers are listed in the Register Section as part of PCI Internal Registers, [Section 7.14.1 “PCI Internal Registers” on page 172](#). The offsets for these registers are 0xc48 - 0xc64. Each MAP register is 32-bits wide. Each MAP register is 32-bits wide, where bits [12:0] are Read Only.

When an address is presented on the PAD lines, the address decoder in the PCI slave compares the PCI address to its base/size registers. If there is a HIT in one of the seven Base Address registers listed above, then the address undergoes remapping in accordance to the right Remap register in the non-masked address bits (by size register). An example of this is summarized in [Table 24](#).

Table 24: PCI Address Remapping Example

PCI address	0x1D98.7654
SCS[1:0]* BAR	0x1F00.0000
SCS[1:0]* Size	0x03FF.FFFF
SCS[1:0]* Remap Register	0x3F00.0000
Remapped PCI Address Presented to SDRAM	0x3D98.7654

NOTE: The size register is programmed to 0x03FF.FFFF. This indicates that this BAR requires a hit in the six MSB (bits 31:26) bits of the PCI address for their to be a hit in the BAR. Therefore, the PCI address 0x1DXX.XXXX is a hit in a BAR programmed to 0x1FXX.XXXX as bits 31-26 of both of these addresses is 0b0001.11.

Then according to the Remap register, these same bit locations will be remapped to 6b111111. The rest of the PCI address bits (i.e. [25:0]) remain unchanged. This means that the final PCI slave address will be 0x3D987654.

3.6.4 Writing to Decode Registers

When a BAR register is written to, the associated remap register is written to, simultaneously. When a remap register is written to, only its contents are affected.

Following RESET, the default value of a remap register is equal to its associated BAR decode register.

Unless a specific write operation to a remap register takes place, a 1:1 mapping is maintained. Also, changing a BAR register's contents automatically returns its associated space to a 1:1 mapping. This allows for backward software compatibility with other Galileo Technology devices such as the GT-64010A and the GT-64011 core logic devices.

3.7 Using the CPU PCI Override

In default, the CPU interface supports 512Mbyte PCI memory address space (256Mbyte on PCI_0 Mem0, 256Mbyte on PCI_0 Mem1). If configured to both PCI_0 and PCI_1, it supports 512Mbyte also on PCI_1. The CPU PCI override feature enables larger PCI memory address space.

The CPU configuration register includes four PCI override bits - two bits per PCI_0 and two bits per PCI_1. Each bit pair controls whether the PCI window is 2Gbyte, 1Gbyte, or the default.

Setting the PCI override bits are set to '01', and SysAD bits[31:30] match bits [10:9] of the PCI Mem0 Low decode address register, the transaction is directed to PCI Mem0. This effectively sets a 1Gbyte window in the PCI memory address space.

NOTE: If Bits[31:30] do not match bits [10:9] of PCI Mem0 Low decode address register, the address is compared against all other address decode registers.

When PCI override bits are set to '10', and SysAD bit[31] matches bit [10] of the PCI Mem0 Low decode address register, the transaction is directed to PCI Mem0. This effectively sets a 2Gbyte window to PCI.

NOTE: If bit[31] does not match bit [10] of PCI Mem0 Low decode address register, the address is compared against all other address decode registers.

If PCI override bits are set to '00' there is no PCI override (default address decoding).

NOTE: When PCI override is enabled, there is no address remapping from the CPU to the PCI.

3.8 Using the DMA to PCI Bypass

In default, the IDMA controller uses CPU interface address decoding. However, the IDMA controller supports direct access to PCI bus, bypassing this address decoding.

In each of the four DMA channel control registers, there are six bits:

- Two bits per source address.
- Two bits per destination address.
- Two bits per next record address.

Each bit pair controls whether the address should be directed to PCI_0 memory space, to PCI_1 memory space, or run through CPU address decoding.

4. CPU INTERFACE DESCRIPTION

The GT-96100A SysAD bus interface allows the CPU to gain access to the GT-96100A's internal registers, PCI interface and the memory/device bus (AD bus). The SysAD bus supports accesses from one to 32 bytes in length.

The SysAD bus on the GT-96100A is a slave-only interface. The GT-96100A will never master the SysAD bus.

4.1 CPU Interface Signals

The CPU interface incorporates the following signals:

Table 25: CPU Interface Signals

Signal	Type	Description
SysAd[63:0] - Master Address/Data	I/O	Transfers multiplexed address/data.
SysCmd[8:0] - Master Port Command	I/O	Transfers information about the access (read/write, size) and the data (good/bad, last word).
SysADC[7:0] - Master Data Check	I/O	An 8-bit bus containing parity for the SysAD bus. SysADC is valid on data cycles only.
ValidOut*	I	Indicates that the local master is driving valid address/data/command on the SysAD bus.
ValidIn*	O	Indicates that the GT-96100A is driving valid data/command on the SysAD bus.
WrRdy* ¹	O	Indicates that the GT-96100A is capable of accepting a write transaction up to eight 32-bit words in length.
Release*	I	Indicates to the GT-96100A that the local master will not drive the SysAD after the current clock cycle. For example, the local master is floating the SysAD and SysCmd bus for completion of a read.
Interrupt*	O	An "OR" of all the internal interrupt sources on the GT-96100A.
ScMatch	I	L2 cache Tag RAM hit indication.
TcDOE*	O	L2 cache data RAM output enable. Asserted by the GT-96100A on L2 read hit.
TcTCE*	I	L2 cache Tag RAM chip enable. Sampled by the GT-96100A to identify L2 access.
TcWord[1:0]	O	L2 cache word index. Driven by the GT-96100A during L2 read miss.

1. There is no RdRdy* signal output from the GT-96100A. This signal should be tied LOW on the CPU as the GT-96100A is always ready to accept a read command.

The SysAD bus is synchronous with respect to TC1k and is locked with respect to the AD bus. The SysAD bus may be asynchronous with respect to the PCI bus or locked to the PCI bus for lower synchronization latency.

4.2 SysAD, SysADC, and SysCmd Buses

The SysAD and SysCmd bus protocol implemented by the GT-96100A is completely compatible with the 64-bit Orion bus protocol used by the IDT R4xxx, R5000, and R7000 processors. The GT-96100A extends this protocol to support bursts less than four 64-bit words. These extensions can be used by DMA engines on the SysAD bus for more efficient use of the interface.

The SysAD[63:0] bus is a 64-bit multiplexed address/data bus. The local CPU drives address for a single cycle then either drives data (for a write) or floats the bus in anticipation of returned data (for a read.)

SysADC[7:0] is valid during data cycles only. It provides parity information for data on the SysAD bus. SysADC has the same timing as SysAD.

The SysCmd[8:0] bus conveys the following information about the transaction:

- The direction (read/write).
- The size (byte, short, word, multi-word).
- The status of the data (good/bad/last.).

SysCmd is driven by the CPU (or other local master) during the address phase of a transaction (with direction/size information) and for the duration of a write (with good/bad/last information.) The GT-96100A drives SysCmd during the data phase of read transactions.

The encodings for SysCmd[8:0] are shown in the tables below.

NOTE: Many encodings are not defined. These encodings are reserved and must not be used. A summary of bit usage is shown below.

Table 26: SysCmd Bit Summary

SysCmd Bit	Function
SysCmd[8]	0 = Transaction information (read/write/size) 1 = Data information (good/bad/last)
SysCmd[7]	Indicates last data/not last data during data cycles. NOTE: Must be 0 for address cycles.
SysCmd[6]	0 = Read transaction during address cycles 1 = Write transaction during address cycles NOTE: Must be 0 for data cycles.
SysCmd[5]	Indicates error status for data cycles. NOTE: Must be 0 for address cycles.

Table 26: SysCmd Bit Summary (Continued)

SysCmd Bit	Function
SysCmd[4]	0 = Check the Data & Check-bits during data cycles 1 = Do not check Data during data cycles NOTE: Must be 1 for address cycles.
SysCmd[3:0]	Encoded to indicate size of the transfer during address cycles. Reserved during data cycles.

Table 27: Address Phase SysCmd[8:0] Encodings (driven by CPU)

SysCmd[8:0] Encoding ¹									Command Mnemonic	Command Description
8	7	6	5	4	3	2	1	0		
0	0	0	0	1	1	0	0	0	RdByte	Read a single byte.
0	0	0	0	1	1	0	0	1	RdShort	Read 16 bits.
0	0	0	0	1	1	0	1	0	RdTriByte	Read 3 bytes.
0	0	0	0	1	1	0	1	1	RdWord	Read 4 bytes (single word).
0	0	0	0	1	1	1	0	0	Rd5Byte	Read 5 bytes.
0	0	0	0	1	1	1	0	1	Rd6Byte	Read 6 bytes.
0	0	0	0	1	1	1	1	0	Rd7Byte	Read 7 bytes.
0	0	0	0	1	1	1	1	1	RdDWord	Read a double-word (64 bits).
0	0	0	0	1	0	X	X	0	Rd4Words	Not supported.
0	0	0	0	1	0	X	0	1	Rd8Words	Read eight words (32 bytes) in a 4-DW burst.
0	0	0	0	0	X	X	X	X	Invalid	Reserved.
0	0	1	0	1	1	0	0	0	WrByte	Write a single byte.
0	0	1	0	1	1	0	0	1	WrShort	Write 16 bits.
0	0	1	0	1	1	0	1	0	WrTriByte	Write 3 bytes.
0	0	1	0	1	1	0	1	1	WrWord	Write 4 bytes (single word).
0	0	1	0	1	1	1	0	0	Wr5Byte	Read 5 bytes.
0	0	1	0	1	1	1	0	1	Wr6Byte	Read 6 bytes.
0	0	1	0	1	1	1	1	0	Wr7Byte	Read 7 bytes.
0	0	1	0	1	1	1	1	1	Wr2Words	Write a double word (8 bytes).
0	0	1	0	1	0	X	X	0	Wr4Words	Not supported.
0	0	1	0	1	0	X	0	1	Wr8Words	Write eight words (32 bytes) in a 4-DW burst.

Table 27: Address Phase SysCmd[8:0] Encodings (driven by CPU) (Continued)

SysCmd[8:0] Encoding ¹									Command Mnemonic	Command Description
8	7	6	5	4	3	2	1	0		
0	0	1	1	0	0	X	X	X	NullReq	Null Request command.
0	0	1	1	X	1	X	X	X	Invalid	Reserved.

1. 'X' denotes "don't care" but 'X' signals must be driven to a valid 0/1.

Table 28: Read Response SysCmd[8:0] Encodings (driven by the GT-96100A)

SysCmd[8:0] Encoding ¹									Command Mnemonic	Command Description
8	7	6	5	4	3	2	1	0		
1	1	0	E	0	X	X	X	X	RD &Chk	Indicates valid data and data-check-bit within a burst. E = 0 Data is good E = 1 Data is erroneous
1	1	0	E	1	X	X	X	X	RD &NoChk	Indicates valid data within a burst (no check-bit). E = 0 Data is good E = 1 Data is erroneous
1	0	0	E	0	X	X	X	X	REOD &Chk	Indicates last valid data and data-checkbit within a burst. E = 0 Data is good E = 1 Data is erroneous
1	0	0	E	1	X	X	X	X	REOD &NoChk	Indicates last valid data in a burst (no check-bit). E = 0 Data is good E = 1 Data is erroneous

1. 'X' denotes "don't care" but 'X' signals are driven to a valid 0/1 by the GT-96100A.

Table 29: CPU Write SysCmd[8:0] Encodings (driven by local master)

SysCmd[8:0] Encoding ¹									Command Mnemonic	Command Description
8	7	6	5	4	3	2	1	0		
1	1	1	E	0	X	X	X	X	WD &Chk	Indicates valid data and data-check-bit within a burst. E = 0 Data is good E = 1 Data is erroneous
1	1	1	E	1	X	X	X	X	WD &NoChk	Indicates valid data within a burst (No check-bit). E = 0 Data is good E = 1 Data is erroneous
1	0	1	E	0	X	X	X	X	WEOD &Chk	Indicates valid data and data-check-bit within a burst. E = 0 Data is good E = 1 Data is erroneous
1	0	1	E	1	X	X	X	X	WEOD &NoChk	Indicates last valid data in a burst (No check-bit). E = 0 Data is good E = 1 Data is erroneous

1. 'X' denotes "don't care" but 'X' signals are driven to a valid 0/1 by the GT-96100A.

4.2.1 SysAD Read Protocol

SysAD reads occur in three phases:

Table 30: SysAD Read Phases

Phase	Description
Address	Address information is driven on the SysAD bus and command information is driven on SysCmd.
Mid burst-data	The GT-96100A drives data on the SysAD bus and a read response on SysCmd.
Last burst-data	The GT-96100A drives data on the SysAD bus and a read end-of-data (REOD) response on SysCmd.

NOTE: If the read command requires parity, then check bits will be driven by the GT-96100A together with SysAD data for every data transfer.

The address phase for all transactions begins with the assertion of ValidOut* to the GT-96100A. Valid address and command information must be present on SysAD and SysCmd during this phase. Release* must also be asserted to the GT-96100A to indicate that the local master is releasing mastership of the SysAD/SysCmd/SysADC buses to the GT-96100A for completion of the read. ValidOut* is deasserted at the end of the phase since the CPU is no longer driving information on SysAD/SysCmd.

For transactions longer than 64 bits, the mid-burst data phase is entered next. The GT-96100A:

- Drives valid data on SysAD.
- Drives bits on SysADC[7:0] for parity.
- Drives a valid read response (mnemonic = RD) on SysCmd.
- Assert ValidIn* to qualify the SysAD, SysADC, and SysCmd buses (see [Figure 8](#)).

The GT-96100A transitions to the last-burst data phase on the last datum of the transfer. This state is differentiated by from the mid-burst state by the REOD command driven on the SysCmd bus. The last-burst data phase is also entered for the datum returned for a double word, single word, or sub-word, read.

On the clock cycle following REOD, the GT-96100A floats the SysAD, SysADC, and SysCmd buses, returning ownership to the CPU.

Figure 7: Double Word (8 bytes) Read by CPU With Parity Check Bits

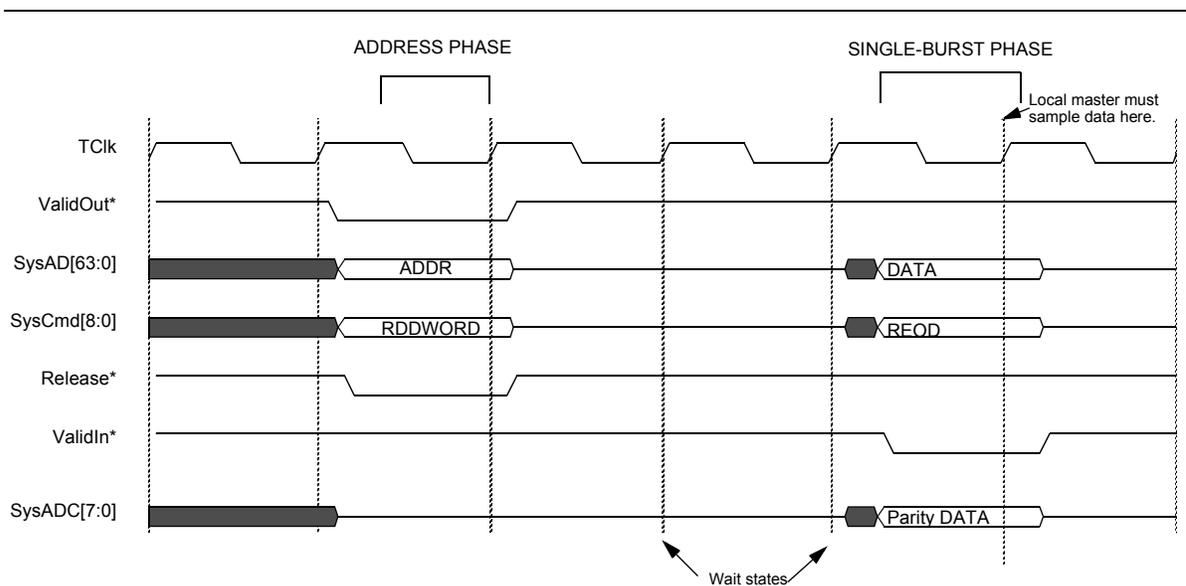
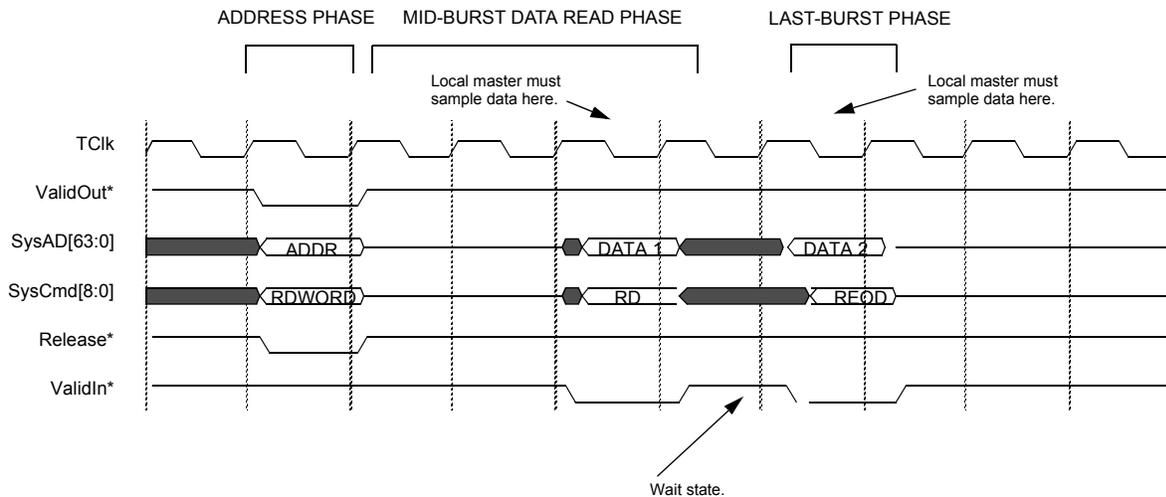


Figure 8: Four Word (16 bytes) Burst Read by CPU



4.2.2 SysAD Write Protocol

CPU writes occur in three phases:

Table 31: SysAD Write Phases

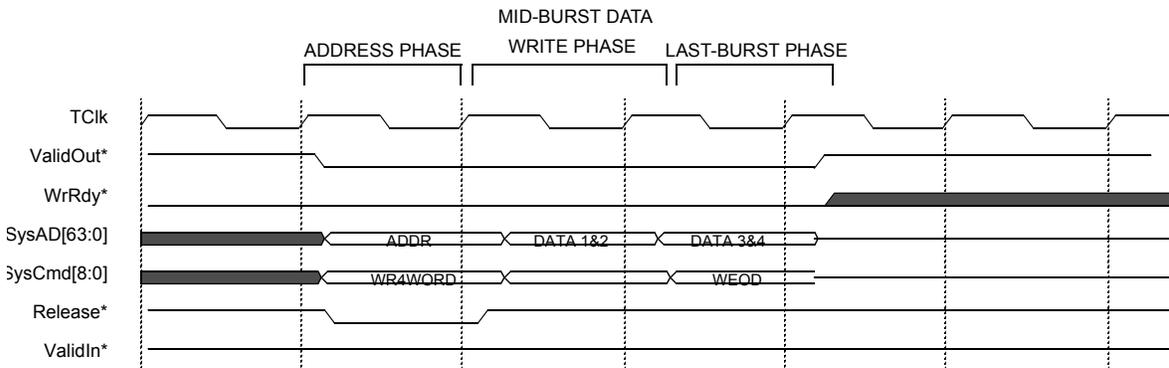
Phase	Description
Address	Address information is driven on the SysAD bus and command information is driven on SysCmd.
Mid-burst write data	The Local Master drives data on the SysAD bus, possibly Parity data on SysADC[7:0], and a write command (mnemonic = WD) on SysCmd.
Last-burst write data	The Local Master drives data on the SysAD bus and a write end-of-data (WEOD) command on SysCmd.

NOTE: If the write command requires parity, then check bits are driven by the Local Master together with SysAD data for every data transfer.

The address phase for write transactions begins with the assertion of ValidOut* to the GT-96100A. Valid address and command information must be present on the SysAD and SysCmd busses during this phase. Release* remains high for write transactions since the Local Master is not relinquishing ownership of the bus. ValidOut* remains asserted throughout a write transaction as the CPU is always driving valid information on SysAD/SysADC/SysCmd.

For transactions longer than 64 bits, the mid-burst data write phase is entered next. The CPU drives valid data on SysAD, a valid write command (mnemonic = WD) on SysCmd (see [Figure 9](#)).

Figure 9: CPU Four Word Burst Write



The GT-96100A transitions to the last-burst write data phase on the last datum of the transfer. This state is differentiated from the mid-burst state by the WEOD command driven on the SysCmd bus. The last-burst data phase is also entered for the datum written for a single word, or sub-word, write. On the clock cycle following WEOD, the GT-96100A returns to the idle state.

NOTE: CPU writes cannot be issued as long as WrRdy* is deasserted (HIGH). If WrRdy* is high and an CPU write is attempted, data from previous write cycles may be corrupted (see [Section 4.3 “Operation of WrRdy* and the Internal Write Posting Queues” on page 79.](#)) All MIPs compliant processors follow this protocol. Only DMA engines on the SysAD bus need to be concerned with sampling WrRdy* before initiating a write.

4.3 Operation of WrRdy* and the Internal Write Posting Queues

The GT-96100A’s CPU interface includes a write posting queue that absorbs local CPU writes at zero wait-states. This is required per the MIPs SysAD bus write protocol.

The write posting queue has four address entries and eight 64-bit data entries. The GT-96100A signals if there is room in the CPU write posting queue by asserting WrRdy*. If WrRdy* is asserted, the CPU may issue a write of up to eight words or four double words.

MIPs compliant processors such as the R4XXX/R5000/R7000 sample WrRdy* automatically before issuing a write.

4.4 CPU Write Modes and Write Patterns Supported

The GT-96100A supports both pipelined and R4XXX/R5000/R7000 compatible write modes (with two dead cycles between consecutive writes). The default mode is pipelined. However, the R4XXX mode can be selected in the CPU Interface Configuration Register.

The CPU interface supports only DDDD and DXDXDXDX write patterns. One of these two write patterns must be selected via the CPU serial initialization bitstream during the CPU reset process. Bit 16 of the CPU Interface Configuration register (0x000) must be programmed according to the write pattern programming of the CPU.

NOTE: In the above explanation, ‘D’ represents data and ‘X’ is a wait state.

4.5 CPU Interface Endianness

The GT-96100A provides the capability to swap the data transferred to or from the internal registers and to or from the PCI interface.

NOTE: Data written to or from the memory controller is NEVER swapped.

The GT-96100A interface endianness to the CPU is programmed on RESET by sampling the Interrupt* pin, see [Section 22. “Reset Configuration” on page 452](#). The setting of this pin programs the Endianness bit in the CPU Configuration register at 0x000. When accessing the internal registers, the endianness of the data will be determined by the Interrupt* pin’s setting.

NOTE: If set to BIG endian, data is swapped.

The setting of the ByteSwap bit in the PCI Internal Command Register, bit 0 of 0xc00, determines how data transactions from the CPU to/from PCI are handled along with the setting of bit 12 in the CPU Configuration Register, 0x000. Both of these bits are set to the same value as the pin strapping of the Interrupt*, but can be re-programmed after RESET.

The setting of MByteSwap bit and MWordSwap bit in the PCI Internal Command register, determines how data transactions from the CPU to or from the PCI are handled along with the setting of the Endianness bit in the CPU Configuration register. Both MByteSwap and Endianness bits are set to the same value as the pin strapping of the Interrupt* (resulting PCI interface working in little-endian mode). These bits can be re-programmed after RESET.

4.6 Burst Order

The GT-96100A supports only the sub-block ordered bursts used by Orion MIPS processors. Sub-block ordered bursts are optimized for the burst patterns used by most SDRAMs.

4.7 MIPS L2 Cache Support

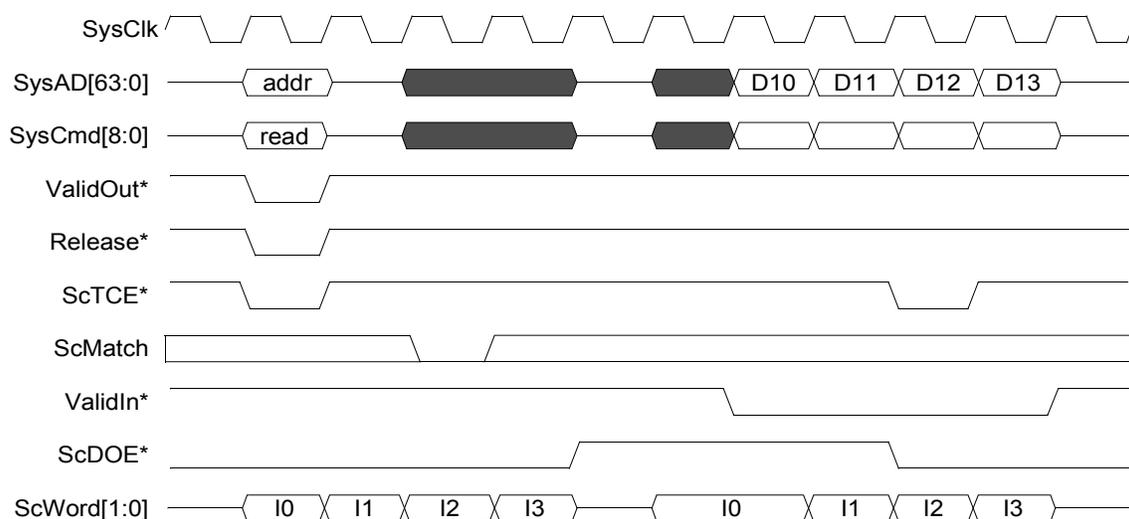
The GT-96100A supports second level cache placed on the SysAD bus. It does not include L2 cache controller, but it supports L2 required signaling, as defined in the R5000 specification.

GT-96100A samples the ScMatch signal. If a CPU access hits the L2 cache line (Tag RAM asserts ScMatch signal), the GT-96100A ignores the transaction, enabling the CPU to complete the transaction against L2 cache.

If the CPU initiates a block read transaction with ScTCE* asserted (indicating a L2 read request), and ScMatch is asserted two cycles after issue cycle (indicating a L2 hit), the GT-96100A ignores the transaction, but keeps ScDOE* asserted, enabling L2 data RAM drive read data on the SysAD bus. ScDOE[1:0] word index is driven by the R5000 L2 cache controller.

In case cache miss (ScMatch deasserted two cycles after block read issue cycle), the GT-96100A responds to the transaction. It also deasserts ScDOE* preventing L2 data RAM from driving the bus, and drives ScWord[1:0] for the L2 data RAM to load the data that the GT-96100A returns to CPU. An example of L2 read miss is shown in [Figure 10](#).

Figure 10: R5000 L2 Read Miss Example



4.8 Multiple GT-96100A Support

Up to four GT-96100A devices can be connected to the CPU System Interface without the need for any glue logic. This capability increases the address space and adds significant flexibility for system design.

Enable multiple GT-96100A devices by sampling the value of DAdr[10] on RESET, see [Section 22. “Reset Configuration” on page 452](#). If this pin is sampled HIGH, multiple GT-96100A devices are enabled. The value of DAdr[10] sampled on RESET will also set bit 18, MultiGT, of the CPU Configuration register at 0x000. This bit is programmable after RESET.

NOTE: This pin must be tied LOW if there is only one GT-96100A device.

If multiple GT-96100A devices are enabled, the values sampled on Ready* and CSTiming* determine the ID of the particular GT-96100A device, as shown in [Table 32](#). This sampled values also program the MultiGTAct bits [1:0] in the MultiGTID Register, 0x120. These bits are programmable after RESET.

Table 32: Pin Strapping the GT-96100A ID

Pin	Configuration Function
Ready*, CSTiming*	Multi-GT-96100A Address ID
00 -	GT responds to SysAD[26,25]=00
01 -	GT responds to SysAD[26,25]= 01
10 -	GT responds to SysAD[26,25]=10
11 -	GT responds to SysAD[26,25]= 11
NOTE: Boot GT-96100A must be programmed to 11	

4.8.1 Hardware Connections

When Multiple GT-96100A devices are enabled, the WrRdy*, ValidIn*, and ScDOE* signals have slightly different functionality versus when only one GT-96100A is enabled.

Table 33: WrRdy*, ValidIn*, and ScDOE* Signal Multiple GT-96100A Functionality

Signal	Multiple GT-96100A Functionality
WrRdy*	An open-source output requiring a 4.7 KOhm pull-down resistor. All WrRdy* outputs from the GT-96100A devices must be tied together to drive the CPU WrRdy* input. WrRdy* is driven LOW for one cycle before floating the output.
ValidIn*	An open-drain output requiring a 4.7 KOhm pull-up resistor. All ValidIn* outputs from the GT-96100A devices must be tied together to drive the CPU ValidIn* input. ValidIn* is driven HIGH for one cycle before floating the output.
ScDOE*	An open-source output requiring a 4.7 KOhm pull-down resistor. All ScDOE* outputs from the GT-96100A devices must be tied together to drive the CPU and Secondary cache inputs. ScDOE* is driven LOW for one cycle before floating the outputs.

4.8.2 MultiGT Bit In The CPU Configuration Register

When the MultiGT bit is SET, the CPU Interface address decoding reduces to:

- If (SysAD[26:25] == ID) AND (it's a WRITE), the access is directed to the internal space of the CPU Interface registers. Bits[11:0] define the specific register offset.
- If (SysAD[26:25] == ID) AND (it's a READ) AND (SysAD[27] == 0), the access is directed to the internal space of the CPU Interface registers. Bits[11:0] define the specific register offset.
- If (SysAD[26:25] == ID) AND (it's a READ) AND (SysAD[27] == 1), the access is directed to BootCS*. Since 0x0.1FC0.0000 implies SysAD[26:25] == 3, the GT 96100A holding the boot device should be strapped to ID = 3.

NOTE: As long as MultiGT bit is SET, there is no access to PCI, SDRAM and Devices and DMA internal registers. Access is available only to the CPU interface internal registers and to boot ROM.

- When the MultiGT bit is CLEARED, the CPU Interface resumes normal address decoding.

NOTE: After the MultiGT bit is CLEARED, WrReady*, ValidIn*, and ScDOE* signals still operate as sustained 3-state (STS) outputs.

4.8.3 Initializing a Multiple GT-96100A System

This section contains an example of connecting two GT-96100A devices to the same CPU. In actuality, it is possible to connect up to four GT-96100A devices to the same CPU.

Use the following procedure for initializing a system with two GT-96100A devices attached to the same CPU.

NOTE: Assuming that the two GT-96100A devices are called GT-1 and GT-2, respectively; both devices have DAdr[10] pulled to VCC (enabling MultiGT mode). GT-1 has Ready* and CSTiming* tied to 11 (boot GT-96100A). GT-2 has Ready* and CSTiming* tied to 00. GT-1 has the BootRom.

Table 34: Initializing a Multiple GT-96100A System

Initialization Steps	Description
1. Access GT-1's BootROM and reconfigure GT-2's CPU Interface address space registers.	After reset, the processor executes from the BootROM on GT-1 because the address on SysAD is 0x0.1FCx.xxxx where SysAD[27:25] = 111 and it's a read cycle. Registers on GT-1 are accessible via address SysAD[26:25]=11, [20:0]=offset]. Registers on GT-2 are accessible via address [SysAD[26:25]=00, [20:0]=offset].
2. Access GT-1's BootROM and reconfigure GT-1's CPU Interface address space registers.	Reconfigures ALSO, the Internal space address decode register, so that later (once Multi-GT mode is disabled) the user can distinguish between internal accesses to GT-1 or GT-2.
3. Lower GT-2 BootCS* high decode register BELOW 0x0.1FCx.xxxx (i.e. 0x0.1FBx.xxxx).	Causes GT-2 to ignore accesses to 0x0.1FCx.xxxx once taken out of MultiGT mode. Further, the address mapping of register and memory space in the GT-96100A and on their interfaces must be unique. In other words, the four PCI address ranges, two SDRAM ranges, I/O space, and internal GT-96100A register spaces of both system controllers must be different.
4. Clear GT-2 Multi-GT mode bit.	
5. Clear GT-1 Multi-GT mode bit.	

Both GT-96100A devices will now resume NORMAL operation with USUAL address decoding.

NOTE: In MultiGT mode, the GT-96100A does not support address mismatch in the CPU Interface decode. In other words, if the CPU attempts a READ of which the address is not mapped in ANY of the GT-96100A devices in the system, ValidIn* is not returned to the CPU and the system will halt.

4.8.4 Multi-GT Restrictions

1. Due to System Interface loading, maximum operating frequency will decrease as the number of GT-96100A devices increase.
2. When Multi-GT support is enabled and if the CPU and GT-96100A are in Pipeline Writes mode (bit 11, WriteMode, in the CPU Interface Configuration register - 0x0, reset to 0), a contention for one clock cycle on the WrRdy* signal may occur. As a result, only R4000 mode (no Pipeline writes - 1 wait state between transactions) is allowed When Multi-GT support is enabled.

4.9 CPU Interface Restrictions

1. The GT-96100A does not support access of more than 4 bytes to internal space.

4.10 CPU Interface Control Registers

Table 35: CPU Interface Register Map

Description	Offset	Page Number
<i>CPU Configuration</i>		
CPU Interface Configuration	0x000	page 85
Multi-GT Register	0x120	page 87
<i>CPU Address Decode</i>		
SCS[1:0]* Low Decode Address	0x008	page 87
SCS[1:0]* High Decode Address	0x010	page 87
SCS[3:2]* Low Decode Address	0x018	page 88
SCS[3:2]* High Decode Address	0x020	page 88
CS[2:0]* Low Decode Address	0x028	page 88
CS[2:0]* High Decode Address	0x030	page 88
CS[3]* & Boot CS* Low Decode Address	0x038	page 88
CS[3]* & Boot CS* High Decode Address	0x040	page 89
PCI_0 I/O Low Decode Address	0x048	page 89
PCI_0 I/O High Decode Address	0x050	page 89
PCI_0 Memory 0 Low Decode Address	0x058	page 89
PCI_0 Memory 0 High Decode Address	0x060	page 89
PCI_0 Memory 1 Low Decode Address	0x080	page 90
PCI_0 Memory 1 High Decode Address	0x088	page 91
PCI_1 I/O Low Decode Address	0x090	page 90
PCI_1 I/O High Decode Address	0x098	page 90
PCI_1 Memory 0 Low Decode Address	0x0a0	page 90
PCI_1 Memory 0 High Decode Address	0x0a8	page 91
PCI_1 Memory 1 Low Decode Address	0x0b0	page 91
PCI_1 Memory 1 High Decode Address	0x0b8	page 91
Internal Space Decode	0x068	page 91

Table 35: CPU Interface Register Map (Continued)

Description	Offset	Page Number
SCS[1:0]* Address Remap	0x0d0	page 91
SCS[3:2]* Address Remap	0x0d8	page 92
CPU Address Decode (Continued)		
CS[2:0]* Remap	0x0e0	page 92
CS[3]* & Boot CS* Remap	0x0e8	page 92
PCI_0 I/O Remap	0x0f0	page 92
PCI_0 Memory 0 Remap	0x0f8	page 92
PCI_0 Memory 1 Remap	0x100	page 93
PCI_1 I/O Remap	0x108	page 93
PCI_1 Memory 0 Remap	0x110	page 93
PCI_1 Memory 1 Remap	0x118	page 93
CPU Sync Barrier		
PCI_0 Sync Barrier Virtual Register	0x0c0	page 94
PCI_1 Sync Barrier Virtual Register	0x0c8	page 94

4.10.1 CPU Configuration Registers

Table 36: CPU Interface Configuration, Offset: 0x000

Bits	Field Name	Function	Initial Value
8:0	Reserved	Cache Operation Mapping Indicates which address bits the GT-64012 uses for cache flush and cache invalidate operations. Bits [8:0] correspond to SysAD[35:27]. Must be 0	0x0
9	Reserved	Secondary Cache support 0 - GT-64012 not present 1 - GT-64012 present Must be 0	0x0
10	Reserved	Reserved Must be 0	0x0

Table 36: CPU Interface Configuration, Offset: 0x000 (Continued)

Bits	Field Name	Function	Initial Value
11	WriteMode	Write mode 0 - Pipelined writes mode 1 - R4000 mode There must be at least two dead-cycles minimum between consecutive address-phase.	0x0
12	Endianness	Byte orientation 0 - Big endian 1 - Little endian NOTE: Affects only the internal registers and the PCI Configuration data register.	Sampled at RESET via the Interrupt* pin.
13	Reserved	Must be 0.	0x0
14	R5KL2_present	Second level cache present 0 - R5KL2 not present 1 - R5KL2 present	0x0
15	External_Hit_Delay	Register second level cache ScMatch signal ¹ . 0 - Not sampled inside the GT-96100A. 1 - Sampled inside the GT-96100A.	0x0
16	CPU WriteRate	CPU Data Write Rate 0 - DXDXDXDX 1 - DDDD	0x0
17	Stop_Retry	Relevant only if PCI Retry was enabled (DAdr[6] was sampled 0 at reset). 0 - Continue to Retry all PCI transactions targeted to the controller's PCI slave 1 - Stop Retry of PCI transactions	0x0
18	MultiGT	Multiple GT-96100A support 0 - Not Supported 1 - Supported	Sampled at RESET via the DAdr[10] pin
19	SysADCValid	GT-96100A to CPU SysADC Connection 0 - Not connected (no parity) 1 - Connected	0x0
21:20	PCI_0_Override	00 - Normal address decoding 01 - 1Gbyte PCI_0 Mem0 space 10 - 2Gbyte PCI_0 Mem0 space 11 - Reserved	0x0
23:22	Reserved		0x0

Table 36: CPU Interface Configuration, Offset: 0x000 (Continued)

Bits	Field Name	Function	Initial Value
25:24	PCI_1_Override	00 - Normal address decoding 01 - 1Gbyte PCI_1 Mem0 space 10 - 2Gbyte PCI_1 Mem0 space 11 - Reserved	0x0
31:26	Reserved		0x0

1. TagRAMs used with L2/L3 cache output the ScMatch signal registered. If for some reason, the TagRAM in use outputs the ScMatch signal non-registered, this bit must be set to 1 to maintain timing relationship between the cache and the GT-96100A.

Table 37: Multi-GT register, Offset: 0x120

Bits	Field Name	Function	Initial Value
1:0	MultiGTAct	Multi-GT Activity bits These bits represent the ID to which the GT-96100A responds with activity.	Value sampled at reset on Ready* and CSTiming*.
31:2	Reserved		0x0

4.10.2 CPU Address Decode Registers

Table 38: SCS[1:0]* Low Decode Address, Offset: 0x008

Bits	Field Name	Function	Initial Value
14:0	Low	SDRAM banks 1 and 0 are accessed when the decoded addresses are between Low and High.	0x0000
31:15	Reserved		0x0

Table 39: SCS[1:0]* High Decode Address, Offset: 0x010

Bits	Field Name	Function	Initial Value
10:0	High	SDRAM banks 1 and 0 are accessed when the decoded addresses are between Low and High.	0x07
31:11	Reserved		0x0

Table 40: SCS[3:2]* Low Decode Address, Offset: 0x018

Bits	Field Name	Function	Initial Value
14:0	Low	SDRAM banks 3 and 2 are accessed when the decoded addresses are between Low and High.	0x0008
31:15	Reserved		0x0

Table 41: SCS[3:2]* High Decode Address, Offset: 0x020

Bits	Field Name	Function	Initial Value
10:0	High	SDRAM banks 3 and 2 are accessed when the decoded addresses are between Low and High.	0x0F
31:11	Reserved		0x0

Table 42: CS[2:0]* Low Decode Address, Offset: 0x028

Bits	Field Name	Function	Initial Value
14:0	Low	Device banks 2, 1, and 0 are accessed when the decoded addresses are between Low and High.	0x00e0
31:15	Reserved		0x0

Table 43: CS[2:0]* High Decode Address, Offset: 0x030

Bits	Field Name	Function	Initial Value
10:0	High	Device banks 2, 1, and 0 are accessed when the decoded addresses are between Low and High.	0xF0
31:11	Reserved		0x0

Table 44: CS[3]* & Boot CS* Low Decode Address, Offset: 0x038

Bits	Field Name	Function	Initial Value
14:0	Low	Device bank 3 and the boot bank are accessed when the decoded addresses are between Low and High.	0x00f8
31:15	Reserved		0x0

Table 45: CS[3]* & Boot CS* High Decode Address, Offset: 0x040

Bits	Field Name	Function	Initial Value
10:0	High	Device bank 3 and the boot bank are accessed when the decoded addresses are between Low and High.	0xFF
31:11	Reserved		0x0

Table 46: PCI_0 I/O Low Decode Address, Offset: 0x048

Bits	Field Name	Function	Initial Value
14:0	Low	The PCI_0 I/O address space is accessed when the decoded addresses are between Low and High.	0x0080
31:15	Reserved		0x0

Table 47: PCI_0 I/O High Decode Address, Offset: 0x050

Bits	Field Name	Function	Initial Value
10:0	High	The PCI_0 I/O address space is accessed when the decoded addresses are between Low and High.	0x8F
31:11	Reserved		0x0

Table 48: PCI_0 Memory 0 Low Decode Address, Offset: 0x058

Bits	Field Name	Function	Initial Value
14:0	Low	The PCI_0 memory address space is accessed when the decoded addresses are between Low and High.	0x0090
31:15	Reserved		0x0

Table 49: PCI_0 Memory 0 High Decode Address, Offset: 0x060

Bits	Field Name	Function	Initial Value
10:0	High	The PCI_0 memory address space is accessed when the decoded addresses are between Low and High.	0x9F
31:11	Reserved		0x0

Table 50: PCI_0 Memory 1 Low Decode Address, Offset: 0x080

Bits	Field Name	Function	Initial Value
14:0	Low	The PCI_0 memory address space is accessed when the decoded addresses are between Low and High.	0x0790
31:15	Reserved		0x0

Table 51: PCI_0 Memory 1 High Decode Address, Offset: 0x088

Bits	Field Name	Function	Initial Value
10:0	High	The PCI_0 memory address space is accessed when the decoded addresses are between Low and High.	0x79F
31:11	Reserved		0x0

Table 52: PCI_1 I/O Low Decode Address, Offset: 0x090

Bits	Field Name	Function	Initial Value
14:0	Low	The PCI_1 I/O address space is accessed when the decoded addresses are between Low and High.	0x0100
31:15	Reserved		0x0

Table 53: PCI_1 I/O High Decode Address, Offset: 0x098

Bits	Field Name	Function	Initial Value
10:0	High	The PCI_1 I/O address space is accessed when the decoded addresses are between Low and High.	0x10F
31:11	Reserved		0x0

Table 54: PCI_1 Memory 0 Low Decode Address, Offset: 0x0a0

Bits	Field Name	Function	Initial Value
14:0	Low	The PCI_1 memory address space is accessed when the decoded addresses are between Low and High.	0x0110
31:15	Reserved		0x0

Table 55: PCI_1 Memory 0 High Decode Address, Offset: 0x0a8

Bits	Field Name	Function	Initial Value
10:0	High	The PCI_1 memory address space will be accessed when the decoded addresses are between Low and High.	0x11F
31:11	Reserved		0x0

Table 56: PCI_1 Memory 1 Low Decode Address, Offset: 0x0b0

Bits	Field Name	Function	Initial Value
14:0	Low	The PCI_1 memory address space is accessed when the decoded addresses are between Low and High.	0x0120
31:15	Reserved		0x0

Table 57: PCI_1 Memory 1 High Decode Address, Offset: 0x0b8

Bits	Field Name	Function	Initial Value
10:0	High	The PCI_1 memory address space is accessed when the decoded addresses are between Low and High.	0x12F
31:11	Reserved		0x0

Table 58: Internal Space Decode, Offset: 0x068

Bits	Field Name	Function	Initial Value
14:0	IntDecode	Registers inside the GT-96100A are accessed when SysAD bits 35:21 match the value programmed in bits 14:0.	0x00a0
31:15	Reserved		0x0

Table 59: SCS[1:0]* Address Remap, Offset: 0x0d0

Bits	Field Name	Function	Initial Value
10:0	SCS[1:0]*_Remap	CPU address remap to resources for SDRAM 0 region.	0x0
31:11	Reserved		0x0

Table 60: SCS[3:2]* Address Remap, Offset: 0x0d8

Bits	Field Name	Function	Initial Value
10:0	SCS[3:2]*_Remap	CPU address remap to resources of SDRAM 1 region.	0x008
31:11	Reserved		0x0

Table 61: CS[2:0]* Address Remap, Offset: 0x0e0

Bits	Field Name	Function	Initial Value
10:0	CS[2:0]*_Remap	CPU address remap to resources of Device 0 region.	0x0e0
31:11	Reserved		0x0

Table 62: CS[3]* & Boot CS* Address Remap, Offset: 0x0e8

Bits	Field Name	Function	Initial Value
10:0	CS[3]*_ & Boot CS*_Remap	CPU address remap to resources of Device 1 region.	0x0f8
31:11	Reserved		0x0

Table 63: PCI_0 IO Address Remap, Offset: 0x0f0

Bits	Field Name	Function	Initial Value
10:0	PCI_0_IO_Remap	CPU address remap to resources of PCI_0 IO region.	0x080
31:11	Reserved		0x0

Table 64: PCI_0 Memory 0 Address Remap, Offset: 0x0f8

Bits	Field Name	Function	Initial Value
10:0	PCI_0_Mem0_ Remap	CPU address remap to resources of PCI_0 Memory 0 region.	0x090
31:11	Reserved		0x0

Table 65: PCI_0 Memory 1 Address Remap, Offset: 0x100

Bits	Field Name	Function	Initial Value
10:0	PCI_0_Mem1_ Remap	CPU address remap to resources of PCI_0 Memory 1 region.	0x790
31:11	Reserved		0x0

Table 66: PCI_1 IO Address Remap, Offset: 0x108

Bits	Field Name	Function	Initial Value
10:0	PCI_1_IO_ Remap	CPU address remap to resources of PCI_1 IO region.	0x100
31:11	Reserved		0x0

Table 67: PCI_1 Memory 0 Address Remap, Offset: 0x110

Bits	Field Name	Function	Initial Value
10:0	PCI_1_Mem0_ Remap	CPU address remap to resources of PCI_1 Memory 0 region.	0x110
31:11	Reserved		0x0

Table 68: PCI_1 Memory 1 Address Remap, Offset: 0x118

Bits	Field Name	Function	Initial Value
10:0	PCI_1_Mem1_ Remap	CPU address remap to resources of PCI_1 Memory 1 region.	0x120
31:11	Reserved		0x0

4.10.3 CPU Sync Barrier

Table 69: PCI_0 Sync Barrier Virtual Register, Offset: 0x0c0

Bits	Field Name	Function	Initial Value
31:0	SyncBarrier_0	A CPU read from this register creates a synchronization barrier cycle. When ValidIn* is returned to the CPU, both of the PCI_0 slave FIFOs are guaranteed to be empty. The read data returned to the CPU is random and should be ignored. This register is READ ONLY.	0x0

Table 70: PCI_1 Sync Barrier Virtual Register, Offset: 0x0c8

Bits	Field Name	Function	Initial Value
31:0	SyncBarrier_1	A CPU read from this register creates a synchronization barrier cycle. When ValidIn* is returned to the CPU, both of the PCI_1 slave FIFOs are guaranteed to be empty. The read data returned to the CPU is random and should be ignored. This register is READ ONLY.	0x0

5. MEMORY CONTROLLER

The GT-96100A's Memory Controller consists of an integrated SDRAM controller and a device controller. The SDRAM controller has a 15-bit address bus (DAdr[12:0],BankSel[1:0]) and shares the 64-bit address/data (AD) bus for data transfers. The device controller uses the 64-bit muxed AD bus for both address and data transfers.

All memory and I/O devices in a GT-96100A system are connected to the AD bus (the SysAD bus is used primarily as a point-to-point connection between the CPU and the GT-96100A system.)

The memory controller only MASTER reads and writes transactions to SDRAM or devices. It receives the instructions for these transactions from the CPU, IDMA controller, or a PCI device on the PCI interface.

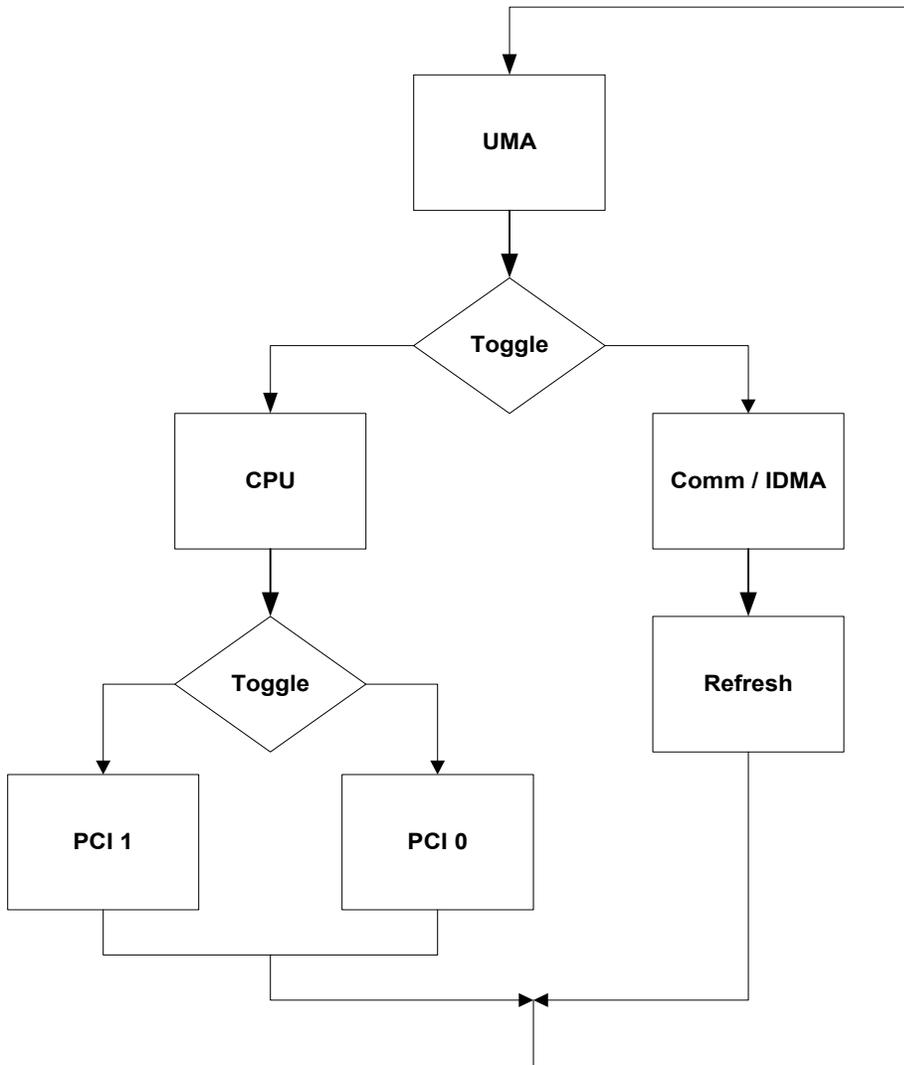
NOTE: A device may not master transactions via the GT-96100A's memory controller.

The GT-96100A's memory controller supports both 32 or 64-bit SDRAM as well as 8-, 16-, 32-, and 64-bit devices.

NOTE: Whenever this datasheet refers to 64-bit SDRAM, it means 64-bits of data plus eight additional bits for ECC.

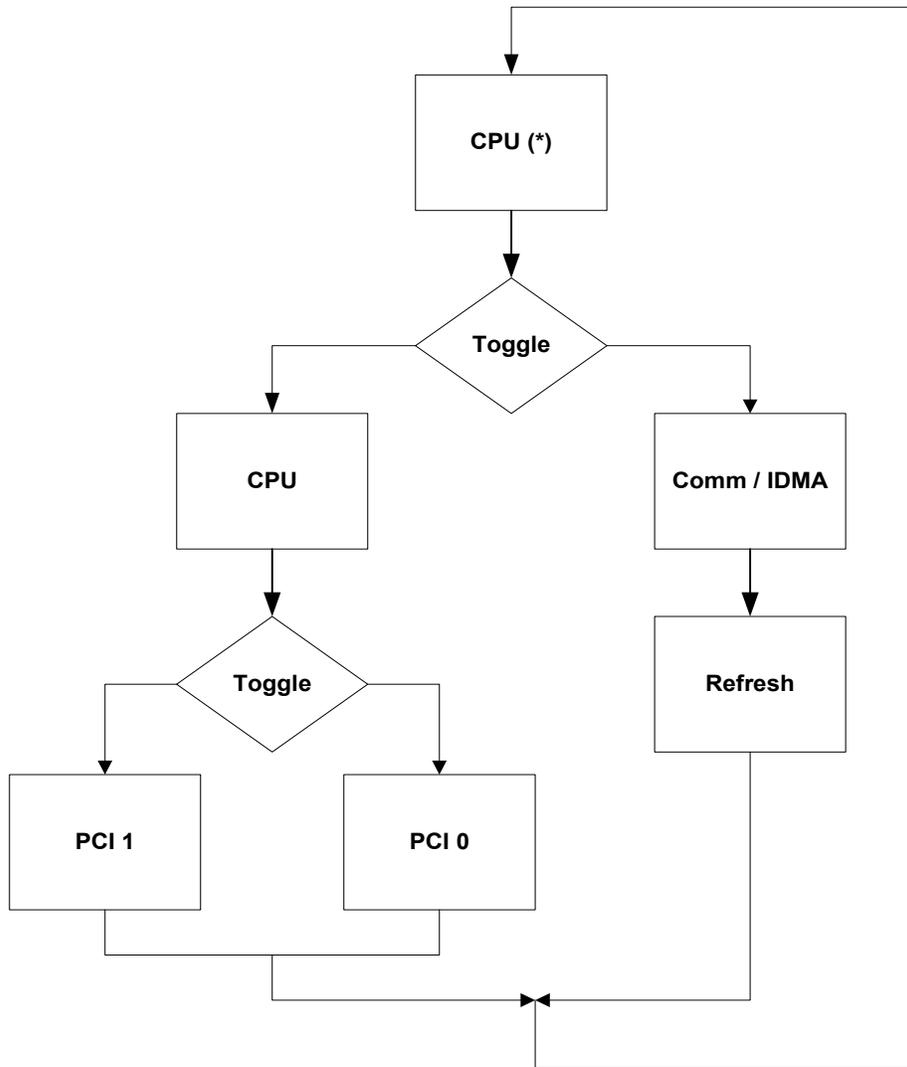
The GT-96100A supports two arbitration schemes for memory controller requests. The default arbitration is shown in [Figure 11](#), while a modified arbitration scheme is shown in [Figure 12](#).

Figure 11: Memory Controller Default Arbitration



NOTE: Only if the memory controller is idle, a Low Priority UMA request is granted.

Figure 12: Memory Controller Modified Arbitration



NOTE: Only if the memory controller is idle, a Low Priority UMA request is granted.

The modified arbitration reduces the latency associated with CPU memory read transactions.¹

1. Programming the memory controller's arbitration mode is done using bit 10 of the SDRAM Burst Mode register (see [Table 107, "SDRAM Burst Mode, Offset: 0x478,"](#) on page 135).

5.1 SDRAM Controller

The SDRAM controller supports up to four banks of SDRAMs.

The SDRAM configuration register (0x448) contains configuration information which is valid for the four banks. Various access parameters can be programmed on a per bank basis as each bank has its own parameters register (0x44c - 0x458).

The supported address depth of the SDRAM can vary for each bank, depending on whether 16, 64, 128 or 256Mbit SDRAMs are used (see [Section 5.2 “Connecting the Address Bus to the SDRAM” on page 106](#) for more information). Up to 256 Mbytes can be addressed by each SCS for a total SDRAM address space of 512 Mbytes by the GT-96100A system.

5.1.1 SDRAM Configuration Register (0x448)

The SDRAM Configuration Register contains parameters which are used for all of the SDRAM banks used with the GT-96100A.

5.1.1.1 Refresh Rates

The GT-96100A implements standard SCAS before SRAS refreshing.

The refresh rate for the SDRAM banks is programmable using the RefIntCnt field in the SDRAM Configuration Register, see [Table 105](#). For example, the default value of RefIntCnt is 0x200. If TClk is 100 MHz, then a refresh sequence will occur every 5us. This is derived from 100MHz (=10ns) * 0x200 (512d) = 5.12us.

Every instance that the refresh counter in the GT-96100A device reaches its terminal count, a refresh request is sent to the Memory Controller. This request enters the arbiter. Once the AD bus is idle and the last SDRAM or Device transaction has finished, the refresh cycle begins.

NOTE: If a UMA transaction is being serviced, the external SDRAM master is responsible for refreshing the SDRAM. See [Section 5.6 “Unified Memory Architecture \(UMA\) Support” on page 113](#).

5.1.1.2 Non-staggered and staggered Refresh

Non-staggered or staggered refresh for each bank can be programmed according to StagRef in the SDRAM configuration register.

In non-staggered refresh, SCS[3:0]* and SRAS* and SCAS* simultaneously asserts refreshing all banks at the same time as shown in [Figure 13](#).

If the SDRAM Controller is programmed to perform staggered refresh (default), SCS[3:0]* will not simultaneously assert LOW together with SRAS*, following the low-going SCAS*. Rather, SCS[0]* will first go LOW for 1 cycle, followed by SCS[1]* on the next TClk, and so on. After the last SCS[3]* has asserted LOW for 1 cycle, SCAS* and SRAS* will go HIGH again. Staggered Refresh is useful for load balancing, shown in [Figure 14](#).

Figure 13: Non-Staggered Refresh Waveform

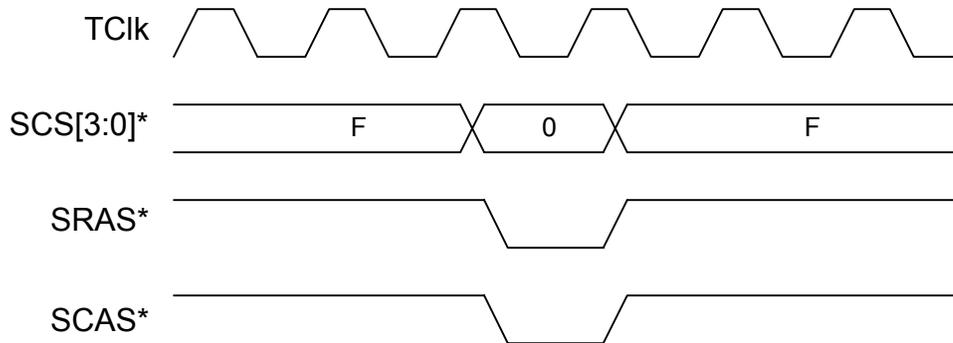
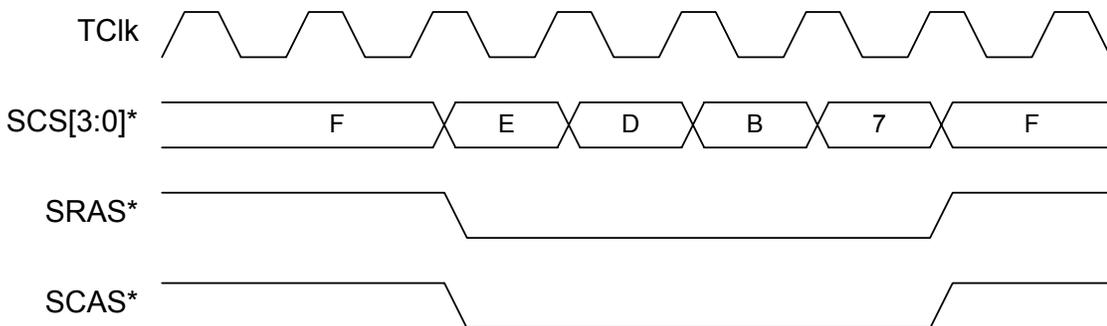


Figure 14: Staggered Refresh Waveform



5.1.1.3 Read Modify Write Enable/ECC

The GT-96100A supports Error Checking and Correction of 64-bit wide SDRAMs.

NOTE: See [Section 6. “Data Integrity” on page 143](#) for more information about ECC.

For 64-bit SDRAMs with ECC enabled, ECC is generated and written to the ADP[7:0] lines on 64-bit writes during the same cycle that the data is written.

Bit 15 enables or disables read modify write protocol to SDRAM. If ECC is enabled in any of the DRAM banks, this bit must be set to 1 to enable read modify write.

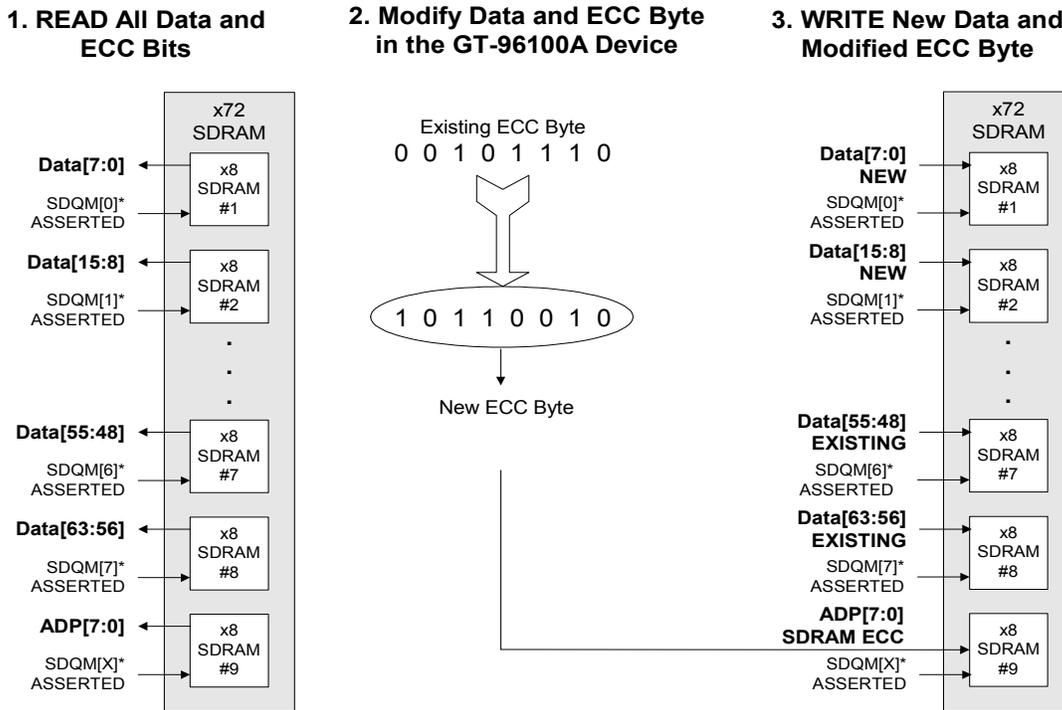
ECC checking and generation requires a 72-bit DIMM to store the ECC information. In order to generate the ECC on partial writes, the current ECC bits must first be read and then modified during the partial write. The protocol for the read modify write transaction is as follows:

1. Read the existing data and ECC information. On this read, all SDQM* lines are asserted (LOW). This means that the BE (byte enable) for the ECC byte can be connected to any of the SDQM[7:0]* outputs. The ECC data is read on the ADP[7:0] inputs.
2. Modify the ECC information based on the data that is to be written. The modification of the ECC byte is done in the GT-96100A system.

- Write the new data and new ECC byte.

Figure 15 illustrates the procedure used to generate ECC in a partial write to SDRAM.

Figure 15: Read Modify Write Transaction by the SDRAM Controller



5.1.2 Duplicating Signals

Some systems require using duplicate signals due to loading requirements. The following sections outlines which signals can be duplicated by setting the appropriate bit in the SDRAM Configuration Register.

5.1.2.1 Duplicating SDRAM Control Lines

SRAS*, SCAS*, and DWr* are the control lines for SDRAM. These signals can be duplicated on different pins for loading considerations.

Setting bit 19 to 0 means these SRAS*, SCAS*, and DWr* signals are not duplicated on other pins (default).

Setting bit 19 to 1 means SRAS*, SCAS*, and DWr* signals are duplicated on DMAReq[0]*/MREQ*, DMAReq[3]*, and BypsOE*/MGNT, respectively. These pins are no longer usable as DMAReq[0]*/MREQ*, DMAReq[3]*, and MGNT*/BypOE* when bit 19 is set to 1. Regardless of the pin strapping of DAdr[7] sampled on RESET (UMA enable), when bit 19 is set to 1, the duplication of SRAS*, SCAS*, and DWr* on the DMAReq[0]*, DMAReq[3]*, and BypsOE* pins takes priority over setting DMAReq[0]* to MREQ* and Byp-sOE* to MGNT.

5.1.2.2 Duplicating DAdr[11] and BankSel[1] on DMAReq[2:1]*

DAdr[11] and BankSel[1] are used for 64/128/256 Mbit SDRAM. These signals can be duplicated on different pins for loading considerations.

Setting bit 20 to 0 means these pins are not duplicated on other pins (default).

Setting bit 20 to 1 means DAdr[11] and BankSel[1] signals are duplicated on DMAReq[2]* and DMAReq[1]*, respectively. These pins are no longer usable as DMAReq[2]* and DMAReq[1]* when bit 20 is set to 1.

NOTE: If ECC is implemented in the system, ADP[5:4] cannot be used as DAdr[11] and BankSel[1]. Therefore, to use DAdr[11] and BankSel[1], program bit 20 to 1 and use DMAReq[2:1]* as DAdr[11] and BankSel[1].

5.1.2.3 DMA End of Transfer Pins Functionality

The IDMA controllers use the End of Transfer pins to give an external device the ability to terminate a current DMA transfer. See [Table 236](#) for more information about this feature.

Bit 21 controls the function of DMAReq[3]*. Setting this bit to 0 means DMAReq[3]* functions as DMA Request for channel 3. Setting this bit to 1 means DMAReq[3]* functions as End of Transfer for channel 0, EOT0.

Likewise, setting bit 22 to 0 means Ready* functions as Ready. And, setting this bit to 1 means Ready* functions as End of Transfer for channel 1, EOT[1].

Table 71: DMAReq*, Ready* and BypsOE* Functionality

Primary Signal Name	Secondary Signal Name (Programmed on RESET)	Bit 19 = 1	Bit 20 = 1	Bit 21 = 1	Bit 22 = 1
DMAReq[0]*	MREQ*	SRAS*			
DMAReq[1]*			BankSel[1]		
DMAReq[2]*			DAdr[11]		
DMAReq[3]*		SCAS*		EOT[0]	
Ready*					EOT[1]
BypsOE*	MGNT*	DWr*			

5.1.2.4 Multiplexing DAdr[12]

If any of the SDRAM banks is configured to 256Mbit, an additional DRAM address bit is required. If bit 24 is set to 1, DAdr[12] is driven on DMAReq[3]* pin. If bit 24 is set to 0, it is driven on ADP[0] pin.

NOTE: If ECC is implemented in the system, ADP[0] cannot be used as DAdr[12]. To use DAdr[12], program bit 24 to 1 and use DMAReq[3]* as DAdr[12].

5.1.3 Registered SDRAM Support

The GT-96100A SDRAM controller can be configured to interface registered SDRAM DIMMs.

Setting bit 23 to 1 means the registered SDRAM is enabled and the SDRAM controller drives and samples SDRAM signals accordingly. The SDRAM controller compensates the clock cycle needed for the DIMM samples SDRAM control signals. Only 64-bit registered SDRAMs are supported.

5.1.4 SDRAM Operation Mode Register (0x474)

The SDRAM Operation Mode Register is a 3-bit register used to execute commands other than standard memory reads and writes to the SDRAM. These operations include:

- Normal SDRAM Mode (0x0)
- NOP Commands (0x1)
- Precharge All Banks (0x2)
- Writing to the SDRAM Mode Register (0x3)
- Force a Refresh Cycle (0x4)

In order to execute one of the above commands on the SDRAM, the following procedure must occur:

1. SDRAM Operation Mode Register must be written the corresponding value. Either the CPU or a PCI device can master this transaction.
2. This write must be followed by a dummy word (32-bit) write to the corresponding SDRAM.
3. To complete the command, the SDRAM Operation Mode Register must be written 0x0 to place it back into Normal SDRAM Mode.

5.1.4.1 Normal SDRAM Mode

The SDRAM Operation Mode Register must be written 0x0 to enable normal reading and writing to the SDRAM.

5.1.4.2 NOP Commands

The NOP command is used to perform a NOP to an SDRAM which is selected by the SDRAM Chip Select (SCS[3:0]*). This prevents unwanted commands from being registered during idle or wait states.

5.1.4.3 Precharge All Banks

The Precharge Bank command is used to deactivate the open row in a particular bank or the open row in both banks.

Once a bank has been precharged, it is in the idle state and must be activated prior to any read or write commands being issued to that bank.

5.1.4.4 Writing to the SDRAM's Parameter Register

Each SDRAM has its own Mode Register. The Mode Register defines the specific operation mode for the SDRAM. This definition includes the selection of a burst length, SCAS latency, operating mode, etc.

NOTE: Refer to the SDRAM data sheet for more information about this register.

Typically, the Mode Register of each SDRAM is initialized on boot-up of the system and is kept static. The GT-96100A has the flexibility to allow the CPU or a PCI Master to update the SDRAM's Mode Register at any time during the operation.

The parameters that the GT-96100A can change are the CAS latency and the burst length. To change these parameters in the SDRAM's Mode Register:

1. Update the corresponding SDRAM Bank Parameters Register (0x44c - 0x458) with the correct values.
2. The SDRAM Operation Mode Register must be written to 0x3. This indicates a Write Command to the SDRAM Mode Register.
3. This write must be followed by a dummy word (32-bit) write to the corresponding SDRAM whose Mode Register must be updated.
4. Finally, the SDRAM Operation Mode Register must be written 0x0 to place it back into Normal SDRAM Mode.

The GT-96100A uses the following procedure to automatically initialize the SDRAM on boot up.

NOTE: This default initialization can be easily overwritten by the procedure described above.

1. SRAS* and DWr* are asserted with DAdr[10] HIGH and SCS[3:0] = 0000. This indicates a Precharge to all SDRAM Banks.
2. SRAS* and SCAS* are asserted with SCS[3:0] = 0000. This indicates a CBR (CAS before RAS) refresh to all SDRAM Banks. This occurs twice in a row.
3. SRAS*, SCAS*, and DWr* are asserted four times in a row.
 - Once with SCS[3:0] = 1110.
 - Once with SCS[3:0] = 1101.
 - Once with SCS[3:0] = 1011.
 - And, once with SCS[3:0] = 0111.

This command programs each of the SDRAM Mode Registers by activating each of the four chip selects (SCS[3:0]) individually.

The GT-96100A automatically initializes the SDRAM on boot up to Sub-block burst ordering as required for MIPS CPUs block reads.

NOTE: The GT-96100A always programs the SDRAM's mode register to burst in sub-block order which support the MIPS CPU burst order. The other modes that are programmed following boot up are the default values of the SDRAM control and parameters register.

Set the GT-96100A to initialize to linear ordering by programing SDRAM Burst Mode register to 0x9 and then following the above procedure. Initializing to linear ordering is only possible with a linear burst read type CPU.

5.1.4.5 Force Refresh

The Force Refresh Command is used to execute a refresh cycle on the particular bank that is accessed.

5.1.5 SDRAM Address Decode Register (0x47c)

The Address Decode Register is a three bit register which determines how bits of an address, presented on the SysAD or PCI bus, are translated to row and column address bits on DAdr[12:0] and BankSel[1:0]. This flexibility allows the designer to choose the address decode setting. This gives the software the best chance This improves the software's capability of interleaving and enhancing overall system performance.

NOTE: The row and column address translation is different for 16 Mbit, 64/128 Mbit, 256 Mbit SDRAMs, 32-bit and 64-bit SDRAM banks. The address decoding depends on the setting of AddrDecode. See [Table 76](#) for the SRAS* and SCAS* address translation from the SysAD interface and PCI.

Table 72: SysAD/PCI Address Decoding for 32-bit SDRAM, 16 Mbit

AddrDecode, 0x47c	SysAD/PCI Bits used for SRAS* on BankSel[0], DAdr[10:0]	SysAD/PCI Bits used for SCAS* on BankSel[0], DAdr[10:0]
000	4, 21-11	4, "0", 23-22, 10-5, 3-2
001	5, 21-11	5, "0", 23-22, 10-6, 4-2
010	11, 21-12, 10	11, "0", 23-22, 9-2
011	12, 21-13, 11-10	12, "0", 23-22, 9-2
100	20, 21, 19-10	20, "0", 23-22, 9-2
101	21, 20-10	21, "0", 23-22, 9-2
110	22, 21-11	22, "0", 23, 10-2 (only for x4 & x8)
111	23, 21-11	23, "0", 22, 10-2 (only for x4)

Table 73: SysAD/PCI Address Decoding for 64-bit SDRAM, 256/512 Mbit

AddrDecode,	SysAD/PCI Bits used for SRAS* on BankSel[0], BankSel[1], DAdr[12:0]	SysAD/PCI Bits used for SCAS* on BankSel[0], BankSel[1], DAdr[12:0]
000	Illegal setting for 64, 128Mbit and 256Mbit SDRAM	
001	6, 7, 25-13	6, 7, 29-28, "0", 27-26, 12-8, 5-3
010	11, 12, 25-13	11, 12, 29-28, "0", 27-26, 10-3
011	13, 14, 25-15, 12-11	13, 14, 29-28, "0", 27-26, 10-3
100	21, 22, 25-23, 20-11	21, 22, 29-28, "0", 27-26, 10-3
101	23, 24, 25, 22-11	23, 24, 29-28, "0", 27-26, 10-3
110	24, 25, 23-11	24, 25, 29-28, "0", 27-26, 10-3
111	25, 26, 27, 22-11	25, 26, 29-28, "0", 24-23, 10-3

Table 74: SysAD/PCI Address Decoding for 32-bit SDRAM, 64 Mbit

AddrDecode, 0x47c	SysAD/PCI Bits used for SRAS* on BankSel[0], BankSel[1], DAdr[11:0]	SysAD/PCI Bits used for SCAS* on BankSel[0], BankSel[1], DAdr[11:0]
000	Illegal setting for 64, 128Mbit and 256Mbit SDRAM	
001	5, 6, 23-12	5, 6, "00", 25-24, 11-7, 4-2
010	11, 12, 23-13, 10	11, 12, "00", 25-24, 9-2
011	12, 13, 23-14, 11-10	12, 13, "00", 25-24, 9-2
100	20, 21, 23-22, 19-10	20, 21, "00", 25-24, 9-2
101	22, 23, 21-10	22, 23, "00", 25-24, 9-2
110	23, 24, 21-10	23, 24, "00", 25, 22, 9-2 (only for x4 & x8)
111	24, 25, 21-10	24, 25, "00", 26, 22, 9-2 (Only for x4)

Table 75: SysAD/PCI Address Decoding for 64-bit SDRAM, 64/128 Mbit

AddrDecode, 0x47c	SysAD/PCI Bits used for SRAS* on BankSel[0], BankSel[1], DAdr[11:0]	SysAD/PCI Bits used for SCAS* on BankSel[0], BankSel[1], DAdr[11:0]
000	Illegal setting for 64, 128Mbit and 256Mbit SDRAM	
001	6, 7, 24-13	6, 7, 27, "0", 26-25, 12-8, 5-3
010	11, 12, 24-13	11, 12, 27, "0", 26-25, 10-3
011	13, 14, 24-15, 12-11	13, 14, 27, "0", 26-25, 10-3
100	21, 22, 24-23, 20-11	21, 22, 27, "0", 26-25, 10-3
101	23, 24, 22-11	23, 24, 27, "0", 26-25, 10-3
110	24, 25, 22-11	24, 25, 27, "0", 26, 23, 10-3 (Only for x4 & x8)
111	25, 26, 22-11	25, 26, 27, "0", 24-23, 10-3 (Only for x4)

Table 76: SysAD/PCI Address Decoding for 64-bit SDRAM, 256 Mbit

AddrDecode, 0x47c	SysAD/PCI Bits used for SRAS* on BankSel[0], BankSel[1], DAdr[12:0]	SysAD/PCI Bits used for SCAS* on BankSel[0], BankSel[1], DAdr[12:0]
000	Illegal setting for 64, 128Mbit and 256Mbit SDRAM	
001	6, 7, 25-13	6, 7, 29-28, "0", 27-26, 12-8, 5-3
010	11, 12, 25-13	11, 12, 29-28, "0", 27-26, 10-3
011	13, 14, 25-15, 12-11	13, 14, 29-28, "0", 27-26, 10-3
100	21, 22, 25-23, 20-11	21, 22, 29-28, "0", 27-26, 10-3
101	23, 24, 25, 22-11	23, 24, 29-28, "0", 27-26, 10-3
110	24, 25, 23-11	24, 25, 29-28, "0", 27-26, 10-3
111	25, 26, 27, 22-11	25, 26, 29-28, "0", 24-23, 10-3

5.2 Connecting the Address Bus to the SDRAM

Connecting the address bus to SDRAM is very simple with The GT-96100A. The SDRAM controller has its own address bus and its depends on whether a 16 Mbit or 64 Mbit SDRAMs are being used.

5.2.1 16 MBit SDRAMs

For 16 Mbit SDRAMs, DAdr[10:0] and BankSel[0] are outputs of the GT-96100A and must be directly connected to address bits 10-0 and Bank Select of the actual SDRAM.

NOTE: DAdr[11] and BankSel[1] are not used when connecting to 16 Mbit SDRAMs. These lines are in HIGH-Z state when accessing 16Mbit SDRAMs.

During a SRAS cycle, a valid row address is placed on the DAdr[10:0] and BankSel[0] lines. During the SCAS cycle, a valid column address is placed on DAdr[9:0] (10-bit). DAdr[10] is used as the auto-precharge select bit and is always written 0 during SCAS cycles. BankSel[0] is held constant from the SRAS cycle.

With 16 MBit SDRAMs, the GT-96100A supports a maximum of 4M addresses, 12 address bits for SRAS and 10 address bits for SCAS.

5.2.2 64/128 Mbit SDRAMs

For 64/128 MBit SDRAMs, DAdr[11:0] and BankSel[1:0] are outputs of the GT-96100A and must be directly connected to address bits 11-0 and Bank Select of the actual SDRAM.

During a SRAS cycle, a valid row address is placed on the DAdr[11:0] and BankSel lines. During the SCAS cycle, a valid column address is placed on DAdr[11,9:0] (11-bit). DAdr[10] is used as the auto-precharge select bit and is always written 0 during SCAS cycles. BankSel is held constant from the SRAS cycle.

With 64 MBit SDRAMs, the GT-96100A supports a maximum of 16M addresses, 14 address bits for SRAS and 10 address bits for SCAS (DAdr[11] is ignored and is in HIGH-Z state when accessing 64 Mbit SDRAMs).

With 128 MBit SDRAMs, the GT-96100A supports a maximum of 32M addresses, 14 address bits for SRAS and 11 address bits for SCAS.

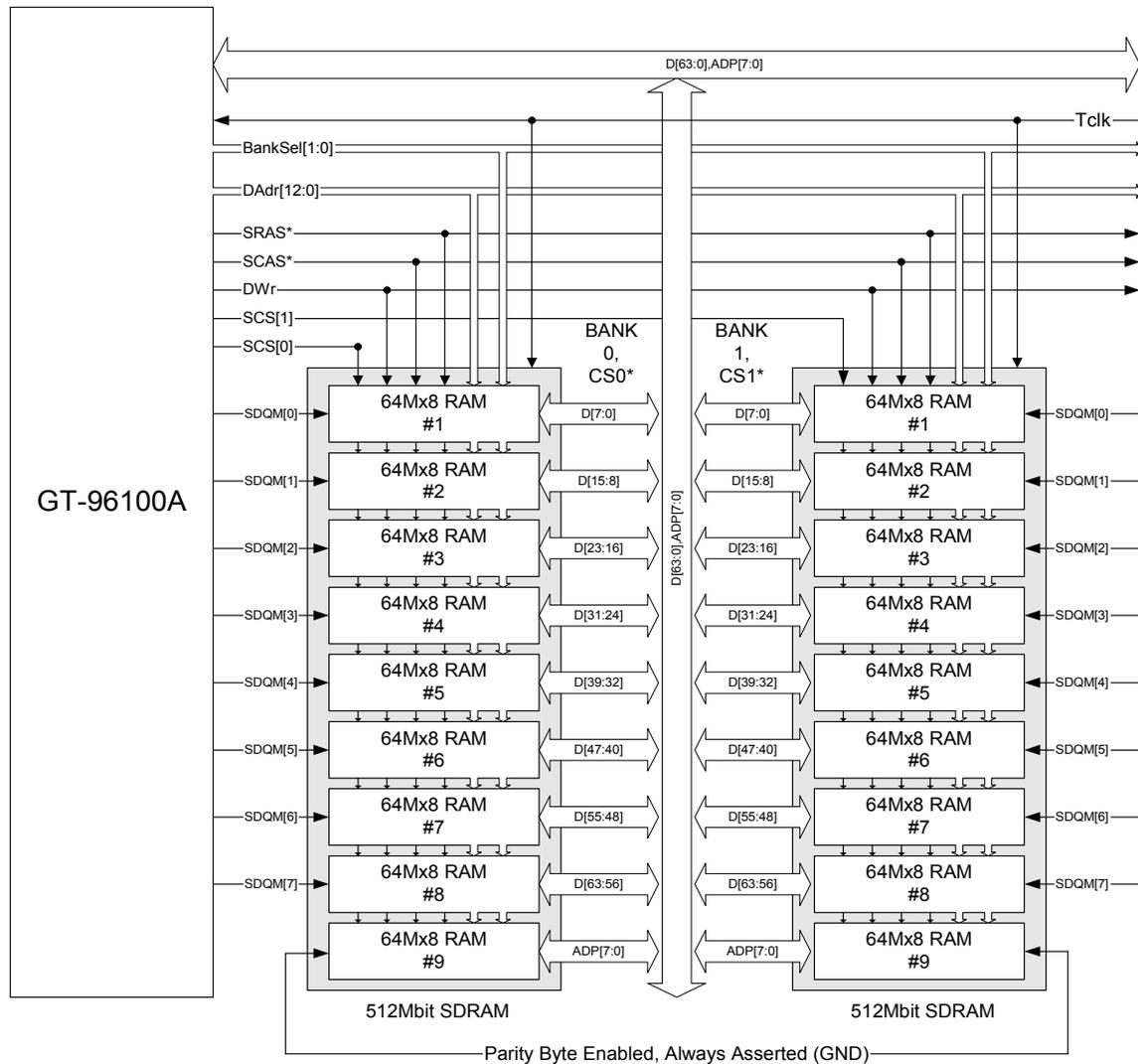
5.2.3 256 Mbit SDRAMs

For 256/512 MBit SDRAMs, DAdr[12:0] and BankSel[1:0] are outputs of the GT-96100A and must be connected directly to address bits 12-0 and Bank Select of the actual SDRAM.

During a SRAS cycle, a valid row address is placed on the DAdr[12:0] and BankSel lines. During the SCAS cycle, a valid column address is placed on DAdr[12-11,9:0] (12-bit). DAdr[10] is used as the auto-precharge select bit and is always written 0 during SCAS cycles. BankSel is held constant from the SRAS cycle.

With 256/512 MBit SDRAMs, the GT-96100A supports a maximum of 128M addresses, 15 address bits in SRAS and 12 address bits in SCAS. For a 64 bit wide SDRAM built from 128Mx4bit memories, 1Gbyte can be addressed by a single SDRAM device decoder.

Figure 16: 512 Mbit/64-bit SDRAM Connection to Memory Bus Using x8 Devices



5.3 Programmable SDRAM Parameters

The SDRAM controller of the GT-96100A device supports a wide range of SDRAMs with different access times and each bank can be programmed independently by the SDRAM Bank[3:0] Parameter registers (0x44c-0x458). These parameters include the number of clock cycles (based on TClk) between active SRAS* and SCAS*.

NOTE: To update the SCAS* Latency or the Burst Length, follow the procedure outlined in [Section 5.1.4.4 “Writing to the SDRAM’s Parameter Register”](#) on page 103 to update the SDRAM’s Mode Register.

Table 77 describes the programmable functions of the SDRAM parameters.

Table 77: Programmable SDRAM Parameters

Function	Description
SCAS* Latency	<p>SCAS* Latency is the number of TCIs from the assertion of SCAS* to the sampling of the first read data. This parameter can be programmed to be either 2 or 3 TCIs. Selecting this parameter depends on TCi frequency and the speed grade of the SDRAM.</p> <p>NOTE: Check your SDRAM data sheet for the most optimal setting.</p>
Flow-through	<p>Bit 2 specifies the number of times that the data is sampled by the GT-96100A on SDRAM reads when bypass is not enabled (bit 9 = 0). This option is included for future designs which will run at faster clock frequencies.</p> <p>NOTE: As of January 1999, Flow-through mode must always be enabled (bit 2 = 1). If ECC or registered SDRAM are used, Flow-through must be disabled (bit 2 = 0).</p>
SRAS* Precharge	<p>Bit 3 specifies the SRAS precharge time. This parameter specifies the number of TCIs following a precharge cycle that a new SRAS* transaction may generate.</p>
64-bit Interleaving	<p>Bit 5 specifies the number of banks that are supported for interleaving if the bank is set for 64/128/256 Mbit SDRAM. If the bank is NOT set for 64/128/256 Mbit SDRAM (bit 11 = 0), the setting of this bit is irrelevant.</p>
Bank Width	<p>Bit 6 specifies the data width of the particular bank.</p>
Bank Location	<p>Bit 7 specifies the location of the bank if the bank is set for 32-bit SDRAM. If the bank is not set for 32-bit SDRAM (bit 6 = 1), the setting of this bit is irrelevant.</p>
ECC Support	<p>Bit 8 enables or disables ECC on a 64-bit wide SDRAM bank.</p>
64-bit Bypass Mode for CPU Reads	<p>Bit 9 enables or disables 64-bit SDRAM Read bypass.</p> <p>NOTE: This option is only for SDRAM banks configured as 64-bit (the option is not available for devices).</p> <p>An optional bypass mode is available for CPU reads where a clock cycle of latency can be eliminated when the CPU executes read cycles from 64-bit SDRAM. The bypass mode requires additional bus switches to enable direct data flow to the CPU. Instead of passing response data from the SDRAM to the GT-96100A prior to presenting it to the CPU, data flows directly from the SDRAM to the CPU via bus switches. This reduces the latency from ValidOut* to ValidIn* from 9 TCi cycles to 8.</p> <p>NOTE: The bypass is used for partial reads as well. Reads from 64-bit SDRAM are no longer sampled by the GT-96100A prior to presenting the data to the CPU.</p> <p>64-bit bypass can only be enabled if the bank is set for 64-bit (bit 6 = 1).</p> <p>No writes are transferred via the bypass switches at any time.</p>
SRAS* to SCAS*	<p>Bit 10 specifies the number of TCIs that the GT-96100A inserts between the assertion of SRAS* with a valid row address to the assertion of SCAS* with a valid column address.</p>

Table 77: Programmable SDRAM Parameters (Continued)

Function	Description
16/64/128/256 MBit SDRAM Configuration	Bits [14,11] specify when the particular bank supports 16, 64/128, or 256 MBit SDRAMs. NOTE: The value of 10 is a reserved setting and must not be used.
Burst Length	Bit 13 specifies the data burst length supported for the particular SDRAM bank. The data can be either 32 or 64 bit, depending on the setting of bit 6.

5.4 SDRAM Performance

Depending on the setting of certain variables, SDRAM performance can vary on both the CPU and PCI interface.

5.4.1 CPU Access to SDRAM

SDRAM performance on the CPU interface is based on the latency between the CPU's assertion of ValidOut* to the GT-96100A's assertion of ValidIn* returning the first data on a burst read (cache line read).

Performance is different if the 64-bit Bypass feature is enabled. [Table 78](#) summarizes the latency between ValidOut* and ValidIn* on SDRAM reads. After the first data is read, the remaining data is returned with zero wait states. For example, if bypass is enabled and the CPU executes a cache line read from memory, data will be returned with 8-1-1-1 performance when bypass is enabled.

Table 78: CPU SDRAM Performance on Reads

SDRAM device	Number of TClks between ValidOut* to ValidIn*
Bypass Enabled ¹	8
Bypass Not Enabled	9

1. See [Table 77](#) for more information about the bypass feature.

On CPU writes to SDRAM, the data cycles will follow the address cycle with zero wait states. Further, the next data of a burst can also be written on the next clock cycle (zero wait states).

5.4.2 PCI Read Performance from SDRAM

The following sections outlines SDRAM memory performance. These figures depend on a number of variables including the PClk/TClk ratio, as well as the sync. mode that the device is configured for. The following numbers are based on the fastest SDRAM settings.

Performance is rated on three events:

Table 79: Events Determining PCI Read Performance from SDRAM

Event	Description
PCI Read request from SDRAM.	This event is based on the number of clocks between Frame* being asserted by a PCI master to the assertion of SRAS* by the SDRAM controller.
Data from SDRAM controller to PCI.	This event is based on the number of clocks between the first data placed on the AD bus until TRdy* is asserted on the PCI bus by the GT-96100A.
Latency till first data.	This event is based on the number of clocks between a PCI master asserting Frame* to the assertion of TRdy* by the GT-96100A.

There are five different sync. modes that the GT-96100A can be placed in depending on the PClk/TCIk ratio. These sync. modes are:

Table 80: GT-96100A Sync. Modes

Sync. Mode	Description
SYNC MODE 0, x00	No assumptions on TCik/PClk ratio.
SYNC MODE 1, 001	PClk frequency is greater than or equal to 1/2 TCik frequency.
SYNC MODE 2, 01x	PClk frequency is synchronized ¹ to TCik frequency and greater than or equal to 1/2 TCik frequency
SYNC MODE 5, 101	PClk frequency is greater than or equal to 1/3 TCik frequency but smaller than 1/2 TCik frequency.
SYNC MODE 6, 11x	PClk frequency is synchronized to TCik frequency and greater than or equal to 1/3 TCik frequency but smaller than 1/2 TCik frequency.

1. If TCik and PClk are synchronized, see [Section 32.1 “TCik/PClk Restrictions” on page 516](#).

Table 81: SDRAM Performance Summary PCI Read Accesses

TCIk/PCIk Frequency (MHz)	Sync. Mode	Event	# of TCIk	# of PCIk
100/33	0	A	9	3
		B	12	5
		C	26	10
	5	A	10	3
		B	12	5
		C	26	10
100/50	0	A	8	4
		B	8	5
		C	20	12
	1	A	8	4
		B	8	5
		C	20	12
100/66	0	A	5	4
		B	6	5
		C	16	12
	1	A	6	4
		B	6	5
		C	16	12
100/33	0	A	14	5
		B	12	4
		C	30	10
	5	A	14	5
		B	12	4
		C	30	10
	6 or 7	A	12	4
		B	8	3
		C	24	8
100/50	0	A	11	6
		B	9	5
		C	24	12
	1	A	11	6
		B	9	5
		C	24	12
	2	A	9	5
		B	5	3
		C	18	9

Table 81: SDRAM Performance Summary PCI Read Accesses (Continued)

TCIk/PCIk Frequency (MHz)	Sync. Mode	Event	# of TCIk	# of PCIk
100/66	0	A	9	6
		B	8	6
		C	21	14
	1	A	9	6
		B	8	6
		C	21	14

5.5 SDRAM Bank Interleaving

The GT-96100A supports two bank interleaving with 16 Mbit SDRAM and two or four bank interleaving with 64 Mbit SDRAMs. The SDRAM Address Control Register (0x47c) determines what address bits are used for SRAS and SCAS cycles. This allows flexibility for different software applications to select an address decoding scheme which may give data accesses a better probability of interleaving.

Interleaving provides higher system performance by hiding SRAS to SCAS cycles and precharge time of a pending transaction during the data cycles of a current transaction. This reduces the number of wait states before data can be read from or written to SDRAM, thus, increasing bandwidth. Interleaving occurs when two independent resources require access to SDRAM. These resources can be the CPU, PCI_0, PCI_1, or one of the IDMA controllers.

At the end of every SDRAM memory transaction, the GT-96100A will precharge the bank.

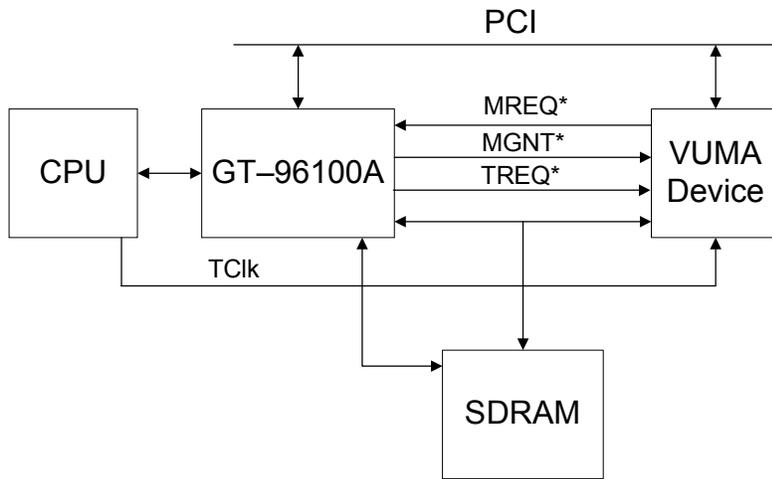
5.6 Unified Memory Architecture (UMA) Support

The GT-96100A supports UMA. This feature allows an external master device to share the same physical SDRAM memory is controlled by the GT-96100A device. This feature works according to the VESA Unified Memory Architecture (VUMA) specification¹.

A VUMA device refers to any type of controller which needs to share the same physical system memory and have direct access to it as shown in [Figure 17](#).

¹. More information about the VESA Unified Memory Architecture can be found at <http://www.vesa.org>

Figure 17: VUMA Device and The GT-96100A sharing SDRAM



5.6.1 UMA Hardware Support

UMA is enabled by a pin strapping option. If DAdr[7] is sampled LOW on RESET, DMAReq[0]* is programmed to function as MREQ*. The BypOE* pin will function as MGNT* as long as bit 9 is 0 for all of the SDRAM Bank Parameters registers (bypass disabled).

NOTE: The Bypass feature and UMA cannot be used simultaneously.

MREQ* is an input into the GT-96100A and must be an output for the VUMA device. This signal is used by the VUMA device to make a request of GT-96100A to access the shared SDRAM. MREQ* should be driven by the VUMA device on a rising edge of TCik. MREQ* is sampled on a rising edge of TCik by the GT-96100A.

MGNT* is an output of the GT-96100A and must be an input to the VUMA device. This signal is used by the GT-96100A to inform the VUMA device that it can access the shared SDRAM. MGNT* is driven by the GT-96100A on a rising edge of TCik. MGNT* must be sampled by the VUMA device on a rising edge of TCik.

Table 82: UMA AC Timing Parameters

Signals	Description	Min.	Max.	Unit	Loading
MREQ*	Setup.	3		ns	
MREQ*	Hold.	1		ns	
MGNT*	Output Delay From TCik rising.	2	10	ns	30 pF

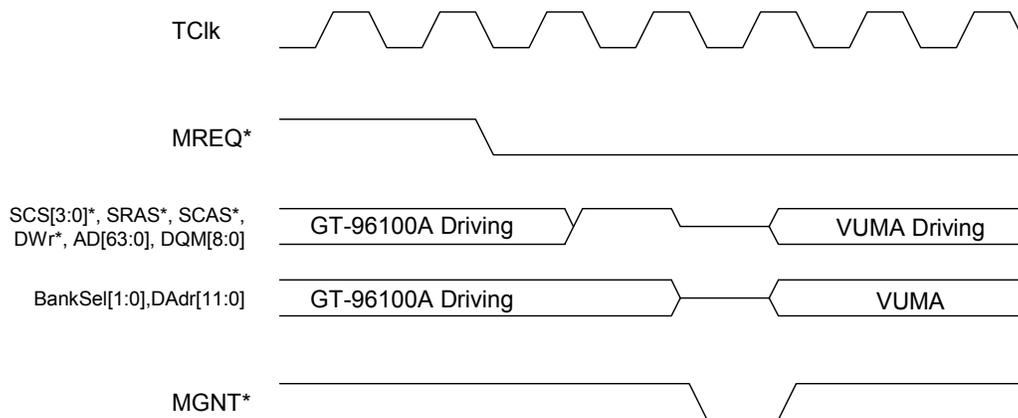
NOTE: The AC timing parameters are shown in [Section 32. “AC Timing” on page 513.](#)

5.6.2 SDRAM Pins

Once MGNT* is asserted by the GT-96100A and the VUMA device is granted access to SDRAM, the SCS[3:0]*, SRAS*, SCAS*, DWr*, AD[63:0], DQM[8:0], DAdr[11:0], and BankSel[1:0] are held in sustained tri-state until the GT-96100A regains access to SDRAM. During this period, the VUMA device must drive these signals to access SDRAM.

When the GT-96100A and the VUMA device hands the bus over to each other, they must drive all of the above signals HIGH for one TClk and then float the pins (except the SDRAM address lines). The SDRAM address lines do not need to be driven high before floating the bus. A sample waveform is shown in [Figure 18](#).

Figure 18: Handing the Bus Over



5.6.3 Address Decoding

The GT-96100A complies with the standard for CPU address to SDRAM Row and Column addressing. See [Section 5.1.5 “SDRAM Address Decode Register \(0x47c\)” on page 104](#) to see how the CPU address translation is performed. This register (0x47c) should be programmed with the binary value of 010 in order to properly use the UMA feature when using 16 Mbit/64-bit SDRAM.

NOTE: As of January 1999, there is no standard for UMA SDRAM addressing when using 16 Mbit/32-bit SDRAM or 64 Mbit SDRAMs (32 or 64 bit).

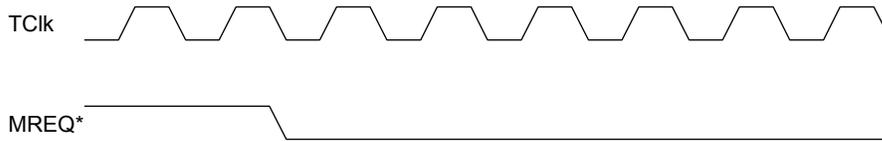
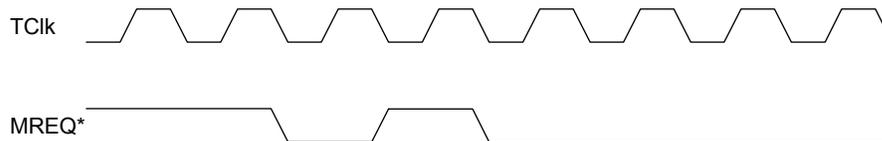
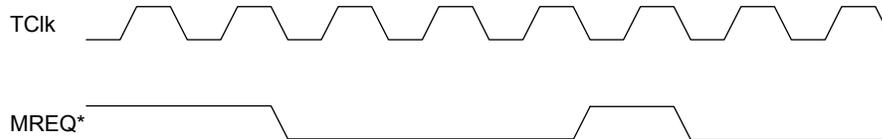
5.6.4 Arbitration

As shown in [Figure 17 on page 114](#), the VUMA device arbitrates with the GT-96100A for access to SDRAM through MREQ* and MGNT*, which should be synchronous to TCik.

The GT-96100A is always the default owner of the SDRAM and access to this memory is only allowed to the VUMA device upon demand.

The GT-96100A has the right to take the VUMA device off of the bus by de-asserting MGNT*.

The VUMA device may request access to SDRAM with either a low or high priority, and both of these priorities are conveyed to the core logic through the MREQ* signal.

Figure 19: MREQ* Requests from the VUMA Device
Low Priority Request

High Priority Request

Pending Low Priority converted to a High Priority


NOTE: Any other transitions asserted other than those shown in this figure will keep the state machine in the current state.

5.6.4.1 VUMA Device Access to SDRAM Rules

There are certain rules that must be followed when the VUMA device makes a request for access to the SDRAM.

1. Once MREQ* is asserted by the VUMA device for a low priority request, it must keep it asserted until the VUMA device is given access to SDRAM via MGNT*. The only reason to change the status of the MREQ* pin is to raise a high priority request or raise the priority of an already pending low priority request.
 - If MGNT* is sampled asserted, the VUMA device must not de-assert MREQ*. Instead, the VUMA device will have ownership of SDRAM and must continue asserting MREQ* until it has completed its transaction.
 - If MGNT* is sampled de-asserted, the VUMA device can de-assert MREQ* for one clock and assert it again regardless of the status of MGNT*. After re-assertion, the VUMA device must keep MREQ* asserted until the GT-96100A gives the VUMA device access to SDRAM via MGNT*.
2. The VUMA device may only assert the MREQ* for the purpose of accessing SDRAM and must stay asserted until MGNT* is sampled asserted except to raise the priority request. No speculative requests or request abortion is allowed.
3. Once the VUMA device samples MGNT* as asserted, it gains and retains access to SDRAM until MREQ* is de-asserted.
4. The GT-96100A always asserts MGNT* for one clock cycle only, and immediately requests back ownership of the bus.

5. The VUMA device retains ownership of SDRAM indefinitely. The standard calls for the VUMA device to keep ownership for no longer than 60 TCIs before it must release the bus. This is not a requirement for the GT-96100A and it will wait until the VUMA device releases the bus by de-asserting MREQ*.
6. When the VUMA device has ownership of the bus, it has full responsibility to execute refresh cycles on the SDRAM.
7. Once the VUMA device de-asserts MREQ* to transfer ownership back to the GT-96100A either on its own, or because of a preemption requires, MREQ* should be de-asserted for at least 2 TCIs before asserting it again to raise a request.

5.6.5 Latencies, Low and High Priority

If a VUMA device places a low priority request for access to SDRAM, there is no set time specified by the GT-96100A to assert MGNT*. Once there are no pending transactions to the memory controller, MGNT* is asserted.

If a VUMA device places a high priority request for an access to SDRAM, the GT-96100A has a maximum of 35 TCIs before it asserts MGNT*.

5.6.6 Total Request

The GT-96100A immediately requests back ownership of the bus after MGNT* assertion.

If bit 4 of SDRAM Bank2 Parameters register is set to 1, DMAReq[3]*/SCAS* pin functions as a total request pin. It indicates that there is a real internal request inside the GT-96100A that requires SDRAM bus ownership.

NOTE: This may cause some difficulty to implement a fair arbitration mechanism on the SDRAM bus.

5.6.7 Disable Refresh

In some applications, the GT-96100A will be most of the time OFF the SDRAM bus. The bus master has full responsibility to execute refresh cycles on the SDRAM.

In such applications, the user may want to disable the GT-96100A's refresh cycles (make memory controller arbitration cycle shorter).

Disable refresh cycles by setting bit 4 of SDRAM Bank1 Parameters register to 1.

5.6.8 Internal Register Reads with UMA Enabled

In order for the GT-96100A to return the data of an internal register read, the SDRAM and Memory Controller must have control over the AD bus. Therefore, the logic controller the input of MREQ* to the GT-96100A must de-assert its request of the memory.

The internal register read transaction is held off until the GT-96100A obtains mastership of the memory bus.

5.7 Device Controller

The device controller supports up to five group of devices. Various access parameters can be programmed on a per group basis as each group has its own parameters register (0x45c - 0x46c).

The supported memory space of each device can vary for each bank up to 256 Mbytes. The width of each device may be 8, 16, 32 or 64-bits. The maximum total device address space is 512 Mbytes for all five groups.

The five individual chip selects are typically broken up into four individual device groups plus one chip select for a boot device (non-volatile memory). Each device group can have unique programmable timing parameters to accommodate different device types (e.g. Flash, ROM, I/O Controllers). The devices share the local AD bus with the SDRAM. Unlike the SDRAM, the devices use the AD bus as a multiplexed address and data bus.

In the address phase, the device controller puts an address on the AD bus with a corresponding Chip Select asserted. ALE indicates the AD bus is output as address with a valid CS*. ALE is used to latch the address and the CS* in an external latch. ALE is HIGH by default, making the latch transparent.¹ ALE goes LOW a half clock cycle before CSTiming* is asserted for the particular read or write transaction. At the completion of the transaction, ALE goes HIGH again on the same rising TClk that CSTiming* is de-asserted.

CS* must then be qualified (OR-tied) with CSTiming*. A read or write cycle is indicated by DevRW*. The CSTiming* signal is valid for the programmable number of cycles of the specific CS* is active. TurnOff, AccToFirst and AccToNext can be set in registers 0x45c - 0x46c for each group's read timing parameters. ALEtoWr, WrActive, and WrHigh are set for each group's write timing parameters. There are certain restrictions to setting these timing parameters. See [Section 5.9 "Memory Controller Restrictions" on page 126](#) before configuring these bits.

NOTE: Some of these parameters can be extended by the Ready* pin. See [Section 5.7.10 "Ready* Support" on page 122](#).

5.7.1 TurnOff

TurnOff is the number of TClk cycles that the GT-96100A does not drive the memory bus after a read from a device. This prevents contention on the memory bus after a read cycle for a slow device.

This parameter is measured from the number of cycles between the de-assertion of DevOE* (an externally extracted signal which is the logical OR between CSTiming* and inverted DevRW*) to an new AD bus cycle.

5.7.2 AccToFirst

AccToFirst defines the number of cycles in a read access from the assertion of CS* (first rising TClk where CS* is asserted LOW) to the cycle that the first data is sampled by the GT-96100A.

NOTE: This parameter can be extended by the Ready* pin. See [Section 5.7.10 "Ready* Support" on page 122](#).

1. Note that this definition of ALE is slightly different than the GT-64010A/11/14/60. ALE on these devices is default LOW, and is only asserted HIGH for a half clock cycle to latch the address. This change in definition for ALE on the GT-96100A has no affect on system performance or architecture.

5.7.3 AccToNext

AccToNext defines the number of cycles in a read access from the cycle that the first data is latched to the cycle to the next data is latched (in burst accesses). This parameter can also be thought of as the delay between the rising edge of TClk which data is latched to the rising edge of TClk where the next data is latched in a burst cycle.

NOTE: This parameter can be extended by the Ready* pin. See [Section 5.7.10 “Ready* Support” on page 122](#).

5.7.4 ALEtoWr

There are eight byte write signals, Wr[7:0]*. ALEtoWr can also be thought of as the delay between the rising edge of TClk which drives ALE LOW to the assertion of Wr*, or for the first write pulse.

NOTE: The Wr[7:0]* signals are asserted and de-asserted off of the FALLING edge of TClk.

5.7.5 WrActive

WrActive is the number of TClks that Wr* are active (asserted). This parameter is measured from the first rising edge of TClk where Wr* is asserted LOW to the last rising edge of TClk where Wr* is LOW for that particular write pulse.

NOTE: This parameter can be extended by the Ready* pin. See [Section 5.7.10 “Ready* Support” on page 122](#). The Wr[7:0]* signals are asserted and de-asserted off of the FALLING edge of TClk.

5.7.6 WrHigh

WrHigh is the number of TClks that Wr* is inactive between burst writes. This parameter is measured from the first rising edge of TClk where Wr* is de-asserted HIGH to the last rising edge of TClk where Wr* is HIGH.

On the next rising edge of TClk, Wr* is asserted LOW for the next write pulse.

NOTE: The Wr[7:0]* signals are asserted and de-asserted off of the FALLING edge of TClk. The previous data remains on the AD bus during WrHigh.

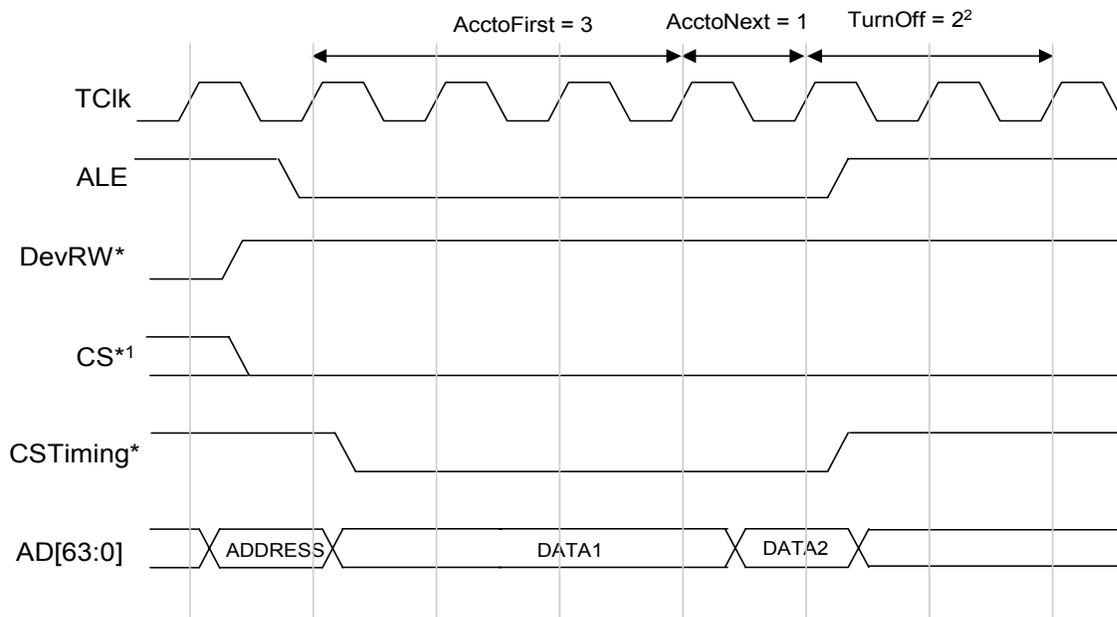
5.7.7 Device Bank Width and Location

Bit 21:20 of the Device Bank[3:0] Parameter registers (0x45c-0x46c), DevWidth, specifies whether the data width of the particular device bank is 8, 16, 32 (default except for BootCS*), or 64 bits. If the bank is set for 8-, 16-, or 32-bit operation, it can either reside on the even half (31:0) or odd half (63:32) by setting bit 23, DevLoc.

Selecting the even or the odd half allows for load balancing.

In case of a 64-bit device, DevLoc has no meaning.

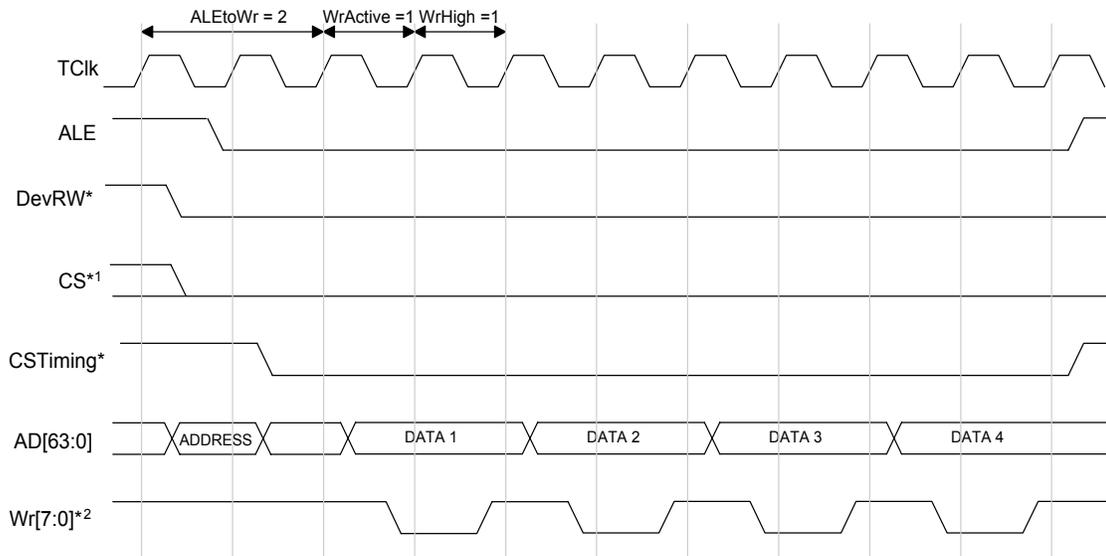
Figure 20: Waveform Showing Device Read Parameters



NOTES:

1. CS* is driven off the same rising TCik* as ALE. Throughout consecutive transactions to the same device, CS* remains asserted. This is why CS* must always be qualified with CSTiming.
2. The GT-96100A may start a new AD cycle after TurnOff.

Figure 21: Waveform Showing Device Write Parameters

**NOTES:**

1. CS* is driven off the same rising TClk* as ALE. Throughout consecutive transactions to the same device, CS* will remain asserted. This is why CS* must ALWAYS be qualified with CSTiming*.
2. Wr[7:0]* are asserted and deasserted from the falling edge of TClk.

5.7.8 Burst Writes

The device controller supports up to eight word burst accesses.

The burst address is supported by a 3-bit wide address bus (BA_{Adr}[2:0]) that is different from the latched address on the multiplexed AD bus.

NOTE: BA_{Adr}[2:0] are the same pins as the least significant SDRAM address lines, DA_{Adr}[2:0]. See [Section 33](#), “Pinout Table, 492 Pin BGA” on page 533 for more information.

5.7.9 Packing and Unpacking Data and Burst Support

The AD bus supports the packing of data into a 64-bit double word, in reads from devices that are 8-, 16-, or 32-bits wide. Devices that are 8- or 16-bits wide are supported by partial reads (up to 64-bits).

The controller supports CPU writes of one to eight bytes to 8-bit or 16-bit wide devices. Therefore, 8 and 16-bit devices **MUST NOT** be mapped to cacheable regions. The reason is that the R4xxx/R5000/R7000 have an eight or 16 word (32 or 64 bytes) cache line size. This would equate to a burst of 32/64 8-bit accesses or 16/32 16-bit accesses.

The GT-96100A supports cached accesses to 32- and 64-bit device spaces. It supports DMA/PCI writes of one to eight bytes to 8-bit or 16-bit wide devices.

5.7.10 Ready* Support

The Ready* pin is sampled on three occasions:

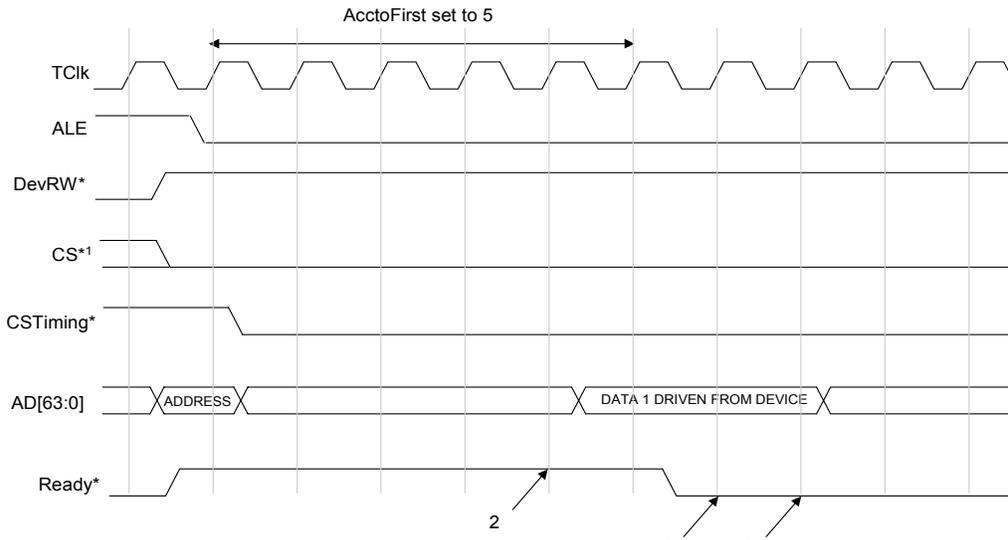
- One clock before the data is sampled to the GT-96100A.
- During both AccToFirst (see Figure 22) and AccToNext (see Figure 23) phases of read cycles.
- On the last rising edge of the WrActive (see below) phase during a write cycle.

During all other phases Ready* is not sampled by the GT-96100A.

If Ready* is not asserted during the WrActive, AccToFirst, or AccToNext phases. These phases are extended until Ready* is asserted again. The transaction may be indefinitely held off until Ready* is asserted.

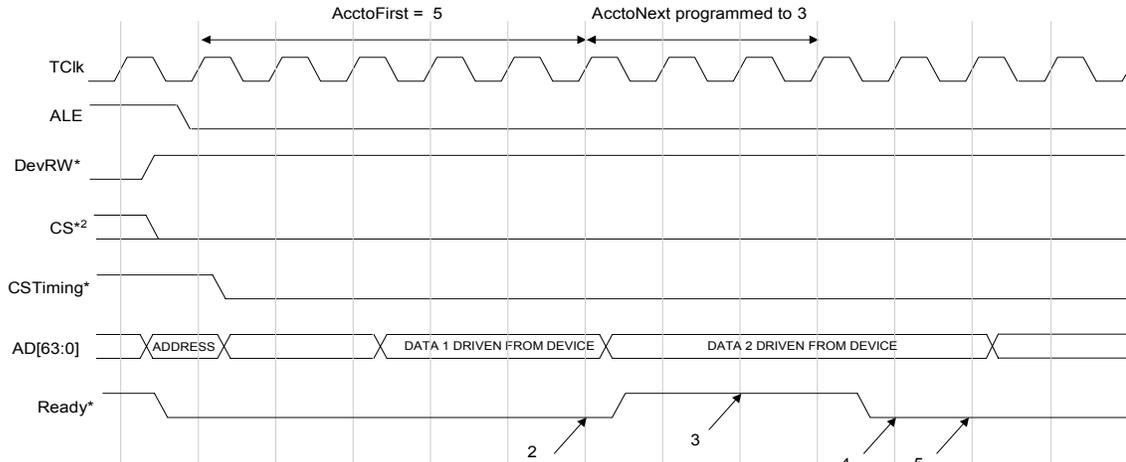
NOTE: There are no SDRAM refreshes as long as an access to a device is not completed. Use Ready* carefully on device accesses so it not interfere with the SDRAM refresh.

Figure 22: Ready* Extending AccToFirst on Read Cycle



1. CS* is driven on the same rising TClk* as ALE. Throughout consecutive transactions to the same device, CS* will remain asserted. This is why CS* must always be qualified with CSTiming*.
2. Ready* is sampled as deasserted one clock before data should be sampled according to AccToFirst. AD[63:0] is not sampled by GT-96100A on the following rising TClk.
3. Ready* is asserted on some following rising TClk.
4. AD[63:0] (Data1) is sampled on the following rising TClk of the rising TClk that Ready* was asserted. Effectively, AccToFirst is 7 in this example (instead of programmed 5).

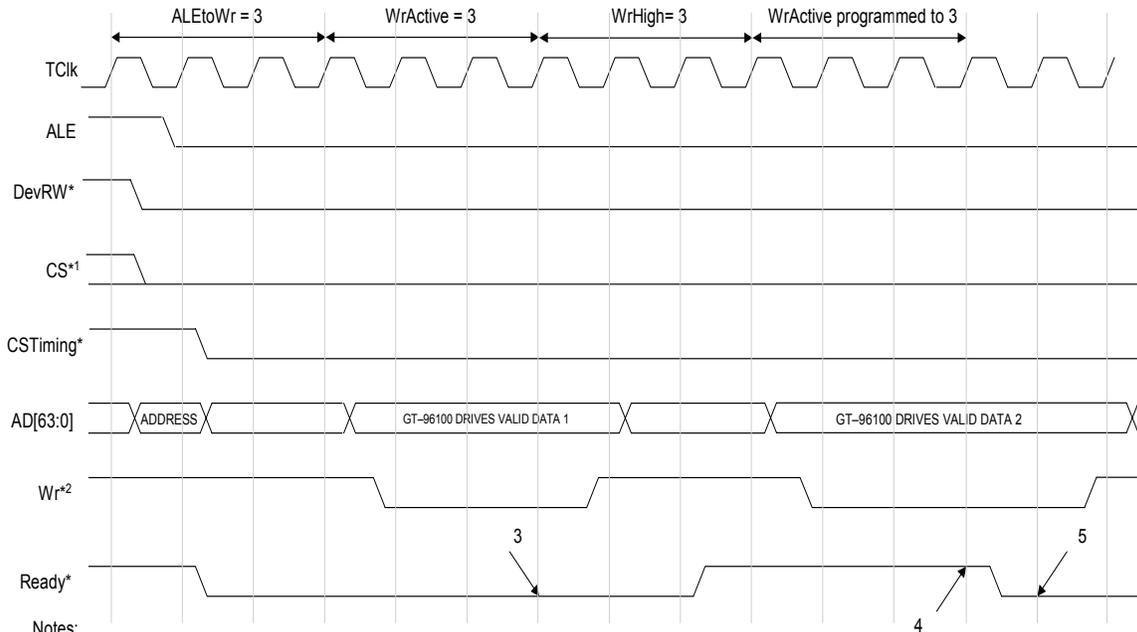
Figure 23: Ready* Extending AccToNext on Read Cycle



Notes:

1. CS* is driven off the same rising TClk* as ALE. Throughout consecutive transactions to the same device, CS* will remain asserted. This is why CS* must ALWAYS be qualified with CSTiming*.
2. Ready* is sampled as asserted one clock before data should be sampled according to AcctoFirst. DATA 1 is sampled on AD[63:0].
3. Ready* is sampled as de-asserted one clock before data should be sampled according to AcctoNext. DATA 2 is NOT sampled.
4. Ready* is asserted on some following rising TClk.
5. AD[63:0] (DATA 2) is sampled on the following rising TClk of the rising TClk that Ready* was asserted. Effectively, AccToNext is 5 (instead of the programmed 3).

Figure 24: Extending WrActive Parameter on Write Cycle



Notes:

1. CS* is driven off the same rising TCik* as ALE. Throughout consecutive transactions to the same device, CS* will remain asserted. This is why CS* must ALWAYS be qualified with CSTiming*.
2. Wr[7:0]* are asserted and de-asserted from the falling edge of TCik.
3. Ready* is asserted on last rising TCik of WrActive, therefore, the GT-96100A assumes the device is written to correctly and continues to the next write cycle.
4. Ready* is deasserted on the last rising TCik of WrActive, therefore, the GT-96100A does NOT continue to the next write cycle effectly extending the WrActive parameter. Wr* remains asserted.
5. Ready* is asserted on the next rising TCik, therefore, the GT-96100A assumes the device has been written to correctly and continues to the next cycle (end of write transaction). Effectively, WrActive is 4 (instead of the programmed 3). For clarification, if there was another word to burst, WrHigh would start counting from the rising TCik denoted by 5, not 4.

5.7.11 Parity Support for Devices

The GT-96100A has no dedicated logic to support device parity. If device parity checking is required, it can be implemented externally using '511 devices and some logic for interrupt generation.

Still, the GT-96100A generates EVEN parity on CPU SysADC lines during CPU read transaction from devices.

5.8 Programming the ADP lines for other Functions

If ECC is enabled for any SDRAM bank, the ADP lines function as pins used for reading and writing the ECC byte.

If ECC is not implemented in the system, the ADP lines are usable for other functions. These functions include duplicating the SDRAM control lines (used for load balancing) and duplicating the ALE signal (used for load balancing).

During RESET, certain pins are sampled (either HIGH or LOW) to select these pin functions, see [Section 22](#), “Reset Configuration” on page 452.

If ECC is not enabled, AND the ADP lines are not programmed to function as other pins on RESET, ADP[3:0] will function as EOT[3:0], see [Table 236](#), “DMA Channel Control Register (0x840 - 0x84c),” on page 221. Further, ADP[4] will be used as BankSel[1] and ADP[5] will be used as DAdr[11]. ADP[7:6] are unused.

[Table 83](#) summarizes the functionality of the ADP lines depending on system configuration.

Table 83: ADP[7:0] Pin Functionality

ECC IN SYSTEM	NO ECC IN SYSTEM		
Primary Pin (ECC Enabled for any SDRAM bank)	ECC Not Enabled for ANY SDRAM bank	DMAReq[1]* sampled HIGH on RESET	DMAReq[3]* sampled HIGH on RESET
ADP[0]	EOT[0]		
ADP[1]	EOT[1]		ALE
ADP[2]	EOT[2]		
ADP[3]	EOT[3]	DWr*	
ADP[4]	BankSel[1]		
ADP[5]	DAdr[11]		
ADP[6]		SCAS*	
ADP[7]		SRAS*	

5.9 Memory Controller Restrictions

1. Unless the boot device is 64-bits wide, the boot must be on the even half (bits 31:0 of AD[63:0] bus).
2. When working with an 8- or 16-bit configured bank, a read/write operation can not exceed 64-bits (eight bytes). Since PCI reads are always prefetchable, PCI access to a 8- or 16-bit wide device is not allowed (unless FRAME* is asserted for a single PClk cycle).
3. When an erroneous address is issued or a burst operation is performed to an 8- or 16-bit device, the GT-96100A forces an interrupt (unless masked). If a sequence of address misses occurs, there is no other interrupt prior to resetting the appropriate bit in the cause register and no new address is registered in the Address Decode Error register (0x470) prior to reading it.
4. When the CPU reads from an address which is decoded in the CPU Interface Unit as being a hit for CS[2:0]* or BootCS* and CS[3]* and decoded as a miss in the SDRAM/Device Interface Unit, the cycle completes only if Ready* is asserted (i.e., driven LOW).

Although being a result of improper and inconsistent programming of the address space defining registers, the following 2 workarounds exist:

- Ready* must be asserted (LOW) when CSTiming* is inactive (HIGH).
 - If the Ready* signal is not needed in the system, the Ready* pin must be driven active (LOW).
5. The minimum parameter for TurnOff, AccToFirst, AccToNext, WrActive, and WrHigh is 1 and for ALEToWr is 3.
 6. If address decode (register 0x47c) is set to 0, bursts of 64 bytes to 64-bit SDRAM are not supported. Address decode mode 0 (0x47c) should not be used when using 64, 128, or 256Mbit SDRAMs.
 7. PCI reads from 8- or 16-bit bank must not be prefetchable.
 8. Access to SDRAM during SDRAM initialization time after reset results in unpredictable behavior.
 9. When SDRAM CAS latency is 3, RAS precharge time (bit 3 of SDRAM parameter register) must be programmed to 1.
 10. When using 64/256 Mbit SDRAM, address decode 0 is not allowed.
 11. In order for memory controller to return data of an internal register read, it must have control over the AD bus.
 12. When using aggressive prefetch, the highest address that can be read from the PCI is the last DRAM address minus 0x8.
 13. [Table 84](#) describes the limitation of a 32 bit device in the GT-96100A.

Table 84: 32-bit Device Limitations

Terminology	Word: 32 bit Aligned: aligned to a word
Limitation	During a read/write cycle of an odd number of words to a 32-bit device from an aligned address, or a read/write cycle to a 32-bit device to an unaligned address, an extra read/write occurs to the complementary word of the double-word address accessed with byte enables not active. This may result in destructive reads and loss of performance while accessing the device.
Examples	<ol style="list-style-type: none"> 1. CPU writes one word to offset 0 of a 32-bit device. Result: a write to offset 0 with Wr^* asserted and a write to offset 4 with Wr^* not asserted. 2. CPU writes one word to offset 4 of a 32 bit device. Result: a write to offset 0 with Wr^* not asserted and a write to offset 4 with Wr^* asserted. 3. DMA writes five words to offset 0 of a 32 bit device. Result: Burst of six bits to the device, with the last Wr^* not asserted. <p>NOTE: If $Ready^*$ is used, the GT-96100A must sample it active an even number of times. i.e., once for address 0 and once for address 4 (see example 1 above).</p>
Workaround	<ol style="list-style-type: none"> 1. If the device is always accessed for single word transfers, connect the $AD[4:2]$, latched, to the device's least significant address pins instead of the $BAdr[2:0]$. This prevents destructive reads. 2. Access the device always with an even number of words at double word aligned addresses (e.g. 0x0, 0x8, 0x10...). This means that the DMA must have the $DatTransLimit$ field set to at least eight bytes. This will make sure there are no destructive reads and no loss of performance.

5.10 Registered SDRAM Interface Restrictions

1. All SDRAM DIMMs must be registered (no support for registered and non-registered DIMMs in one system).
2. ALL SDRAM DIMMs are 64-bit (no support for 32-bit registered DIMMs).
3. Bit 2 (Flow-Through) in all SDRAM parameters registers is set to 0.

5.11 Memory Interface Control Registers

Table 85: Memory Interface Register Map

Description	Offset	Page Number
<i>SDRAM and Device Address Decode</i>		
SCS[0]* Low Decode Address	0x400	page 130
SCS[0]* High Decode Address	0x404	page 130
SCS[1]* Low Decode Address	0x408	page 130
SCS[1]* High Decode Address	0x40c	page 130
SCS[2]* Low Decode Address	0x410	page 131
SCS[2]* High Decode Address	0x414	page 131
SCS[3]* Low Decode Address	0x418	page 131
SCS[3]* High Decode Address	0x41c	page 131
CS[0]* Low Decode Address	0x420	page 131
CS[0]* High Decode Address	0x424	page 132
CS[1]* Low Decode Address	0x428	page 132
CS[1]* High Decode Address	0x42c	page 132
CS[2]* Low Decode Address	0x430	page 132
CS[2]* High Decode Address	0x434	page 132
CS[3]* Low Decode Address	0x438	page 133
CS[3]* High Decode Address	0x43c	page 133
Boot CS* Low Decode Address	0x440	page 133
Boot CS* High Decode Address	0x444	page 133
Address Decode Error	0x470	page 133

Table 85: Memory Interface Register Map (Continued)

Description	Offset	Page Number
<i>SDRAM Configuration</i>		
SDRAM Configuration	0x448	page 134
SDRAM Operation Mode	0x474	page 135
SDRAM Burst Mode	0x478	page 135
SDRAM Address Decode	0x47c	page 136
<i>SDRAM Parameters</i>		
SDRAM Bank0 Parameters	0x44c	page 137
SDRAM Bank1 Parameters	0x450	page 138
SDRAM Bank2 Parameters	0x454	page 138
SDRAM Bank3 Parameters	0x458	page 139
<i>ECC</i>		
ECC Upper Data	0x480	page 139
ECC Lower Data	0x484	page 139
ECC from Memory	0x488	page 139
ECC Calculated	0x48c	page 139
ECC Error report	0x490	page 140
<i>Device Parameters</i>		
Device Bank0 Parameters	0x45c	page 140
Device Bank1 Parameters	0x460	page 141
Device Bank2 Parameters	0x464	page 141
Device Bank3 Parameters	0x468	page 141
Device Boot Bank Parameters	0x46c	page 142

5.11.1 SDRAM and Device Address Decode

Table 86: SCS[0]* Low Decode Address, Offset: 0x400

Bits	Field Name	Function	Initial Value
11:0	Low	SDRAM bank 0 is accessed when the decoded addresses are between Low and High.	0x00
31:12	Reserved		0x0

Table 87: SCS[0]* High Decode Address, Offset: 0x404

Bits	Field Name	Function	Initial Value
11:0	High	SDRAM bank 0 is accessed when the decoded addresses are between Low and High.	0x07
31:12	Reserved		0x0

Table 88: SCS[1]* Low Decode Address, Offset: 0x408

Bits	Field Name	Function	Initial Value
11:0	Low	SDRAM bank 1 is accessed when the decoded addresses are between Low and High.	0x08
31:12	Reserved		0x0

Table 89: SCS[1]* High Decode Address, Offset: 0x40c

Bits	Field Name	Function	Initial Value
11:0	High	SDRAM bank 1 is accessed when the decoded addresses are between Low and High.	0x0f
31:12	Reserved		0x0

Table 90: SCS[2]* Low Decode Address, Offset: 0x410

Bits	Field Name	Function	Initial Value
11:0	Low	SDRAM bank 2 is accessed when the decoded addresses are between Low and High.	0x10
31:12	Reserved		0x0

Table 91: SCS[2]* High Decode Address, Offset: 0x414

Bits	Field Name	Function	Initial Value
11:0	High	SDRAM bank 2 is accessed when the decoded addresses are between Low and High.	0x17
31:12	Reserved		0x0

Table 92: SCS[3]* Low Decode Address, Offset: 0x418

Bits	Field Name	Function	Initial Value
11:0	Low	SDRAM bank 3 is accessed when the decoded addresses are between Low and High.	0x18
31:12	Reserved		0x0

Table 93: SCS[3]* High Decode Address, Offset: 0x41c

Bits	Field Name	Function	Initial Value
11:0	High	SDRAM bank 3 is accessed when the decoded addresses are between Low and High.	0x1f
31:12	Reserved		0x0

Table 94: CS[0]* Low Decode Address, Offset: 0x420

Bits	Field Name	Function	Initial Value
11:0	Low	Device bank 0 is accessed when the decoded addresses are between Low and High.	0xc0
31:12	Reserved		0x0

Table 95: CS[0]* High Decode Address, Offset: 0x424

Bits	Field Name	Function	Initial Value
11:0	High	Device bank 0 is accessed when the decoded addresses are between Low and High.	0xc7
31:12	Reserved		0x0

Table 96: CS[1]* Low Decode Address, Offset: 0x428

Bits	Field Name	Function	Initial Value
11:0	Low	Device bank 1 is accessed when the decoded addresses are between Low and High.	0xc8
31:12	Reserved		0x0

Table 97: CS[1]* High Decode Address, Offset: 0x42c

Bits	Field Name	Function	Initial Value
11:0	High	Device bank 1 is accessed when the decoded addresses are between Low and High.	0xcf
31:12	Reserved		0x0

Table 98: CS[2]* Low Decode Address, Offset: 0x430

Bits	Field Name	Function	Initial Value
11:0	Low	Device bank 2 is accessed when the decoded addresses are between Low and High.	0xd0
31:12	Reserved		0x0

Table 99: CS[2]* High Decode Address, Offset: 0x434

Bits	Field Name	Function	Initial Value
11:0	High	Device bank 2 is accessed when the decoded addresses are between Low and High.	0xdf
31:12	Reserved		0x0

Table 100: CS[3]* Low Decode Address, Offset: 0x438

Bits	Field Name	Function	Initial Value
11:0	Low	Device bank 3 is accessed when the decoded addresses are between Low and High.	0xf0
31:12	Reserved		0x0

Table 101: CS[3]* High Decode Address, Offset: 0x43c

Bits	Field Name	Function	Initial Value
11:0	High	Device bank 3 is accessed when the decoded addresses are between Low and High.	0xfb
31:12	Reserved		0x0

Table 102: Boot CS* Low Decode Address, Offset: 0x440

Bits	Field Name	Function	Initial Value
11:0	Low	Boot bank is accessed when the decoded addresses are between Low and High.	0xfc
31:12	Reserved		0x0

Table 103: Boot CS* High Decode Address, Offset: 0x444

Bits	Field Name	Function	Initial Value
11:0	High	Boot bank is accessed when the decoded addresses are between Low and High.	0xff
31:12	Reserved		0x0

Table 104: Address Decode Error, Offset: 0x470

Bits	Field Name	Function	Initial Value
31:0	ErrAddr	The addresses of accesses to invalid address ranges (those not in the range programmed in the SDRAM or device decode registers) are captured in this register.	0xffffffff

5.11.2 SDRAM Configuration

Table 105: SDRAM Configuration, Offset: 0x448

Bits	Field Name	Function	Initial Value
13:0	RefIntCnt	Refresh Interval Count Value	0x0200
14	Interleave	Bank Interleaving Control 0 - Interleaving enabled 1 - Interleaving disabled	0x0
15	RMW	Enable Read Modify Write 0 - Disabled 1 - Enabled	0x0
16	StagRef	Staggered Refresh 0 - Staggered refresh 1 - Non-staggered refresh	0x0
17	DisECCEr	Disable Force ECC Error	0x0
18	IntOorT	Interrupt One or Two	0x0
19	DupCntl	Duplicate Control Pins 0 - Do not duplicate 1 - Duplicate control pins <ul style="list-style-type: none"> • DMAReq0* = SRAS* • DMAReq3* = SCAS* • BypsOE* = DWr* 	0x0
20	DupBA	Duplicate Bank Addresses 0 - Do not duplicate 1 - Duplicate bank addresses <ul style="list-style-type: none"> • DMAReq[2]* = DAdr[11] • DMAReq[1]* = BankSel[1] 	0x0
21	DupEOT0	Duplicate End of Transfer 0 0 - Do not duplicate 1 - Duplicate End of Transfer 0 <ul style="list-style-type: none"> • DMAReq[3]* = EOT0 	0x0
22	DupEOT1	Duplicate End of Transfer 1 0 - Do not duplicate 1 - Duplicate End of Transfer 1 <ul style="list-style-type: none"> • Ready* = EOT1 	0x0
23	RegSDRAM	Registered SDRAM Enable 0 - Disable 1 - Enable	0x0

Table 105: SDRAM Configuration, Offset: 0x448 (Continued)

Bits	Field Name	Function	Initial Value
24	DAdr12Sel	DAdr[12] Pin Select 0 - DAdr[12] driven on ADP[0] 1 - DAdr[12] driven on DMAReq[3]*	0x0
31:25	Reserved		0x0

Table 106: SDRAM Operation Mode, Offset: 0x474

Bits	Field Name	Function	Initial Value
2:0	SDRAMOp	Special SDRAM Mode Select 000 - Normal SDRAM mode 001 - NOP command 010 - All banks precharge command 011 - Mode register command enable 100 - CBR cycle enable 101,110, and 111 - Reserved	0x0
31:3	Reserved		0x0

Table 107: SDRAM Burst Mode, Offset: 0x478

Bits	Field Name	Function	Initial Value
1:0	Reserved	Must be 0x1.	0x1
2	Burst_Order	0 - Linear 1 - Sub-block	0x1
9:3	Reserved	Must be 0x1.	0x1
10	ArbitrationMode	This bit controls the internal arbitration scheme employed by the memory control unit: 0 - Normal arbitration mode (round robin: CPU -> PCI -> DMA ->...). 1 - CPU preferred arbitration mode (CPU -> PCI -> CPU -> DMA...). NOTE: Setting this bit is allowed only after the SDRAM was programmed.	0x0
11	Reserved	Must be 0.	0x0
31:12	Reserved	Read Only 0.	X

Table 108: SDRAM Address Decode, Offset: 0x47c

Bits	Field Name	Function	Initial Value
2:0	AddrDecode	SDRAM Address Decode See Section 5.1.5 “SDRAM Address Decode Register (0x47c)” on page 104 for more information.	0x2
31:3	Reserved		0x0

5.11.3 SDRAM Parameters

Table 109: SDRAM Bank0 Parameters, Offset: 0x44c

Bits	Field Name	Function	Initial Value
1:0	CASLat	CAS Latency Identical for all SDRAM banks. 1 - 2 cycles 2 - 3 cycles 3 and 0 - Reserved NOTE: This value must be the same as in the SRASToSCAS field.	0x1
2	FlowThrough	Flow-Through Enable 0 - One sample 1 - No Sample NOTE: Must be set to 0 if ECC or registered SDRAM is enabled.	0x1
3	SRASPrchg	SRAS Precharge Time 0 - 2 cycle 1 - 3 cycles	0x0
4	Reserved	Must be set to 0.	0x0
5	64bitInt	64-bit SDRAM Interleaving 0 - 2 way bank interleaving 1 - 4 way bank interleaving	0x0
6	BankWidth	Width of SDRAM bank 0 - 32 (36) bit wide SDRAM 1 - 64 (72) bit wide SDRAM	0x0
7	BankLoc	32-bit SDRAM Bank Location NOTE: Not applicable when using 64-bit SDRAMs. 0 - Even, AD[31:0] 1 - Odd, AD[63:32]	0x0
8	ECC	ECC support for the bank. 0 - No ECC support 1 - ECC supported	0x0
9	Bypass	Bypass enable to CPU 0 - No Bypass 1 - Bypass	0x0
10	SRASToSCAS	SRAS to SCAS delay 0 - 2 cycles 1 - 3 cycles NOTE: This value must be the same as in the CASLat field.	0x0

Table 109: SDRAM Bank0 Parameters, Offset: 0x44c (Continued)

Bits	Field Name	Function	Initial Value
11	SDRAMSize0	16 or 64/128/256 Mbit SDRAM 0 - 16 Mbit SDRAM 1 - 64/128/256 Mbit SDRAM	0x0
12	Reserved	Must be set to 0.	0x0
13	BrstLen	Burst Length 0 - Burst of 8 1 - Burst of 4	0x0
14	SDRAMSize1	256 Mbit SDRAM support 0 - 16 or 64/128 Mbit SDRAM 1 - 256Mbit SDRAM	0x0
31:15	Reserved		0x0

Table 110: SDRAM Bank1 Parameters, Offset: 0x450

Bits	Field Name	Function	Initial Value
3:0	Various	Fields function as in SDRAM Bank0.	0x5
4	Disable_Refresh	Disable refresh of ALL SDRAM banks. 0 - Do refresh 1 - Disable refresh	0x0
14:5	Various	Fields function as in SDRAM Bank0.	0x0
31:15	Reserved		0x0

Table 111: SDRAM Bank2 Parameters, Offset: 0x454

Bits	Field Name	Function	Initial Value
3:0	Various	Fields function as in SDRAM Bank0.	0x5
4	TREQ*_Enable	UMA Total Request pin enable. 0 - Disable 1 - Enable DMAReq[3]*/SCAS* pin functions as a total request pin.	0x0
14:5	Various	Fields function as in SDRAM Bank0.	0x0
31:15	Reserved		0x0

Table 112: SDRAM Bank3 Parameters, Offset: 0x458

Bits	Field Name	Function	Initial Value
3:0	Various	Fields function as in SDRAM Bank0.	0x5
4	Reserved	Must be 0.	0x0
14:5	Various	Fields function as in SDRAM Bank0.	0x0
31:15	Reserved		0x0

5.11.4 ECC

Table 113: ECC Upper Data, Offset: 0x480

Bits	Field Name	Function	Initial Value
31:0	ECCUpData	Bits[63:32] of the last data with an ECC error.	0x0

Table 114: ECC Lower Data, Offset: 0x484

Bits	Field Name	Function	Initial Value
31:0	ECCLoData	Bits[31:0] of the last data with an ECC error.	0x0

Table 115: ECC from Mem, Offset: 0x488

Bits	Field Name	Function	Initial Value
7:0	ECCMem	Eight bits of ECC code read from memory.	0x0
31:8	Reserved		0x0

Table 116: ECC Calculated, Offset: 0x48c

Bits	Field Name	Function	Initial Value
7:0	ECCCalc	Eight bits of ECC code calculated inside the GT-96100A.	0x0
31:8	Reserved		0x0

Table 117: ECC Error Report, Offset: 0x490

Bits	Field Name	Function	Initial Value
1:0	ECCErr	Number of ECC errors 00 - No errors 01 - One error detected and corrected 10 - Two or more errors detected 11 - Reserved	0x0
31:2	Reserved		0x0

5.11.5 Device Parameters

Table 118: Device Bank0 Parameters, Offset: 0x45c

Bits	Field Name	Function	Initial Value
2:0	TurnOff	The number of cycles between the de-assertion of DevOE* to a new AD bus cycle. NOTE: An externally extracted signal which is the logical OR between CSTiming* and inverted DevRW*.	0x7
6:3	AccToFirst	The number of cycles in a read access from the assertion of CS* to the cycle when the data is latched (by the external latches). Extend the number of cycles via the Ready* pin.	0xf
10:7	AccToNext	The number of cycles in a read access from the cycle that the first data is latched to the cycle that the next data is latched (in burst accesses). Extend the number of cycles via the Ready* pin.	0xf
13:11	ALEtoWr	The number of cycles from ALE de-asserted to the assertion of Wr*.	0x7
16:14	WrActive	The number of cycles Wr* signals are active. Extend the number of cycles via the Ready* pin.	0x7
19:17	WrHigh	The number of cycles between de-assertion and assertion of Wr* signals.	0x7

Table 118: Device Bank0 Parameters, Offset: 0x45c (Continued)

Bits	Field Name	Function	Initial Value
21:20	DevWidth	Device Width 00 - 8 bits 01 - 16 bits 10 - 32 bits 11 - 64 bits	0x2
22	DMAFlyBy	Forwarded to BootCS* during Flyby DMA	0x1
23	DevLoc	32/16/8 bit device location 0 - Even bank <ul style="list-style-type: none"> AD[31:0], AD[15:0], and AD[7:0] 1 - Odd bank <ul style="list-style-type: none"> AD[63:32], AD[47:32], and AD[39:32] 	0x0
24	Reserved	Read only.	0x0
25	Reserved	Must be 0.	0x0
29:26	DMAFlyBy	Forwarded to CS[3:0]* during FlyBy DMA.	0xe
31:30	Reserved	Must be 0.	0x0

Table 119: Device Bank1 Parameters, Offset: 0x460

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Device Bank0.	0x386ffff

Table 120: Device Bank2 Parameters, Offset: 0x464

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Device Bank0.	0x386ffff

Table 121: Device Bank3 Parameters, Offset: 0x468

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Device Bank0.	0x38?ffff

Table 122: Device Boot Bank Parameters, Offset: 0x46c

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Device Bank0. Bits bits [29:26] and bit 22 are reserved.	14?ffff

NOTE: In case of Bank3 or Boot Bank, bits [23:20] are shown as ‘?’ because bits 21:20 are sampled at reset via DAdr[4:3] to define the width of the boot device.

6. DATA INTEGRITY

GT-96100A supports:

- Parity generation and checking on CPU and PCI interfaces.
- ECC generation and checking on SDRAM interfaces.
- Errors propagation between these the CPU, PCI, and SDRAM interfaces.

6.1 SDRAM ECC

The GT-96100A implements Error Checking and Correction (ECC) on accesses to the SDRAM. It supports detection and correction of one data bit error, detection of two bits error, and detection of three or four bit errors, if residing in the same nibble.

6.1.1 ECC Calculation

Each of the 64 data bits and eight check bits has a unique 8-bit ECC check code, as shown in [Table 123](#). For example, data bit 12 has the check value of 01100001, and check bit 5 has the check value of 00100000.

Table 123: ECC Code Matrix

Check Bit	Data Bit	ECC Code Bits								Number of 1s in
		7	6	5	4	3	2	1	0	
	63	1	1	0	0	1	0	0	0	3
	62	1	1	0	0	0	1	0	0	3
	61	1	1	0	0	0	0	1	0	3
	60	1	1	0	0	0	0	0	1	3
	59	1	1	1	1	0	1	0	0	5
	58	1	0	0	0	1	1	1	1	5
4		0	0	0	1	0	0	0	0	1
3		0	0	0	0	1	0	0	0	1
	57	1	1	1	0	0	0	0	0	3
	56	1	0	1	1	0	0	0	0	3
	55	0	0	0	0	1	1	1	0	3
	54	0	0	0	0	1	0	1	1	3
	53	1	1	1	1	0	0	1	0	3
	52	0	0	0	1	1	1	1	1	5
5		0	0	1	0	0	0	0	0	5
2		0	0	0	0	0	1	0	0	1

Table 123: ECC Code Matrix (Continued)

Check Bit	Data Bit	ECC Code Bits								Number of 1s in
		7	6	5	4	3	2	1	0	
	51	1	0	0	0	0	1	1	0	1
	50	0	1	0	0	0	1	1	0	3
	49	0	0	1	0	0	1	1	0	3
	48	0	0	0	1	0	1	1	0	3
	47	0	0	1	1	1	0	0	0	3
	46	0	0	1	1	0	1	0	0	3
	45	0	0	1	1	0	0	1	0	3
	44	0	0	1	1	0	0	0	1	3
	43	1	0	1	0	1	0	0	0	3
	42	1	0	1	0	0	1	0	0	3
	41	1	0	1	0	0	0	1	0	3
	40	1	0	1	0	0	0	0	1	3
	39	1	0	0	1	1	0	0	0	3
	38	1	0	0	1	0	1	0	0	3
	37	1	0	0	1	0	0	1	0	3
	36	1	0	0	1	0	0	0	1	3
	35	0	1	0	1	1	0	0	0	3
	34	0	1	0	1	0	1	0	0	3
	33	0	1	0	1	0	0	1	0	3
	32	0	1	0	1	0	0	0	1	3
	31	1	0	0	0	1	0	1	0	3
	30	0	1	0	0	1	0	1	0	3
	29	0	0	1	0	1	0	1	0	3
	28	0	0	0	1	1	0	1	0	3
	27	1	0	0	0	1	0	0	1	3
	26	0	1	0	0	1	0	0	1	3
	25	0	0	1	0	1	0	0	1	3
	24	0	0	0	1	1	0	0	1	3
	23	1	0	0	0	0	1	0	1	3

Table 123: ECC Code Matrix (Continued)

Check Bit	Data Bit	ECC Code Bits								Number of 1s in
		7	6	5	4	3	2	1	0	
	22	0	1	0	0	0	1	0	1	3
	21	0	0	1	0	0	1	0	1	3
	20	0	0	0	1	0	1	0	1	3
	19	1	0	0	0	1	1	0	0	3
	18	0	1	0	0	1	1	0	0	3
	17	0	0	1	0	1	1	0	0	3
	16	0	0	0	1	1	1	0	0	3
	15	0	1	1	0	1	0	0	0	3
	14	0	1	1	0	0	1	0	0	3
	13	0	1	1	0	0	0	1	0	3
	12	0	1	1	0	0	0	0	1	3
	11	1	1	1	1	1	0	0	0	5
	10	0	1	0	0	1	1	1	1	5
7		1	0	0	0	0	0	0	0	1
0		0	0	0	0	0	0	0	1	1
	9	0	1	1	1	0	0	0	0	3
	8	1	1	0	1	0	0	0	0	3
	7	0	0	0	0	0	1	1	1	3
	6	0	0	0	0	1	1	0	1	3
	5	1	1	1	1	0	0	0	1	5
	4	0	0	1	0	1	1	1	1	5
6		0	1	0	0	0	0	0	0	1
1		0	0	0	0	0	0	1	0	1
	3	1	0	0	0	0	0	1	1	3
	2	0	1	0	0	0	0	1	1	3
	1	0	0	1	0	0	0	1	1	3
	0	0	0	0	1	0	0	1	1	3

The GT-96100A calculates ECC by taking the EVEN parity of ECC check codes of all data bits that are logic one. For example, if the 64 bit data is 0x45. The binary equivalent is 01000101. From [Table 123](#), the required check codes are 00001101 (bit[6]), 01000011 (bit[2]) and 00010011 (bit[0]). Bitwise XOR of this check codes (even parity) result in ECC value of 01011101.

On a write transaction to a 64-bit wide SDRAM, if ECC is enabled, the GT-96100A calculates the new ECC and writes it to the ECC bank along with the data that is written to the data bank. Since the ECC calculation is based on a 64-bit data width, if the write transaction is smaller than 64 bits, the GT-96100A runs a read modify write sequence.

For a read transaction from a 64-bit wide SDRAM, if ECC is enabled, the GT-96100A reads a 64-bit data and 8-bit ECC. It calculates ECC based on the 64-bit data and then compares it against the received ECC. The result of this comparison (bitwise XOR between received ECC and calculated ECC) is called syndrome. If the syndrome is 00000000, both the received data and ECC are correct. If the syndrome is any other value, it is assumed either the received data or the received ECC are in error.

If the syndrome contains a single 1, there is a single bit error in the ECC byte. For example, if the received data is 0x45, the calculated ECC is 01011101, as explained before. If the received ECC is 01010101, the resulting syndrome is 00001000. [Table 123](#) shows that this syndrome corresponds to check bit 3. GT-96100A will not report nor correct this type of error.

If the syndrome contains three or five 1's, it indicates that there is at least one data bit error. For example, if the received data is 0x45, the calculated ECC is 01011101, as explained before. If the received ECC is 00011110, the resulting syndrome is 01000011. This syndrome includes three 1's and it corresponds to data bit 2 as shown in [Table 123](#). GT-96100A corrects the data by inverting data bit 2 (the corrected data is 0x41).

If the result syndrome contains two 1's, it indicates that there is a double-bit error.

If the result syndrome contains four 1's, it indicates a 4-bit error located in four consecutive bits of a nibble.

If the result syndrome contains three or five 1's and does not corresponds to any data bit, it indicates a triple-bit error within a nibble.

These types of errors cannot be corrected. The GT-96100A reports an error but will not change the data.

NOTE: ECC is not supported in bypass mode (data must go through GT-96100A in order to be checked and corrected).

6.1.2 ECC Error Report

If the GT-96100A identifies an ECC data bit error, it sets MemErr bit in Interrupt Cause register (an interrupt is asserted if not masked).

Additionally, it stores error information in dedicated registers.

- The 64-bit data in ECC Upper Data and ECC Lower Data registers.
- The ECC byte read from memory in ECC from Memory register.
- The calculated ECC byte in ECC Calculated register.
- Address bits[31:2] of the erroneous data in ECC Address register. In bits[1:0] it reports whether it was a single data bit error or multiple data bits error.

6.2 PCI Parity Support

The GT-96100A implements all parity features required by PCI spec, including PAR, PERR*, and SERR* generation and checking (also PAR64 in case of 64-bit PCI configuration).

As an initiator, the GT-96100A generates even parity on PAR signals for address phase and data phase of write transaction. It samples PAR on data phase of read transactions. If the GT-96100A detects bad parity and PErrEn bit in Status and Command Configuration register is set, it asserts PERR*.

As a target, the GT-96100A generates even parity on PAR signal for data phase of a read transaction. It samples PAR on address phase and data phase of write transactions. If the GT-96100A detects bad parity and PErrEn bit in Status and Command Configuration register is set, it asserts PERR*.

The GT-96100A asserts SERR* if any of the following occur:

- Detects a bad address parity as a target.
- Detects a bad data parity on a write transaction as a master (detects PERR* asserted).
- Detects a bad data parity on a read transaction as a master.
- Detects ECC error on read from SDRAM or Device.
- Performs master abort.
- Detects target abort.

SERR* is asserted if SErrEn bit in Status and Command configuration register is set to 1 and if SERR* is not masked through SERR Mask register.

NOTE: For a description of the Serr0* and Serr1* registers, see [Table 418](#) and [Table 419](#).

6.3 Parity Support for Devices

There is no dedicated logic in the GT-96100A to support devices parity. If Devices parity checking is required, it can be implemented externally using '511 devices and some logic for interrupt generation.

Still, the GT-96100A generates EVEN parity on CPU SysADC lines during CPU read transaction from devices.

6.4 CPU Parity Support

CPU parity is supported through SysADC lines.

On write transactions, CPU drives even parity on SysADC. The GT-96100A samples the incoming data driven on SysAD and the parity driven on SysADC. In case of parity error, the GT-96100A latches:

- The address in CPU Error address register.
- The data in CPU Error Data register.
- The parity in CPU Error Parity register.

On read transactions, CPU drives even parity on SysADC, in parallel to the data it drives on SysAD.

CPU parity errors are reported through CPUOut interrupt bit in the interrupt cause register, and through SysCmd[5] in case of read transaction.

NOTES: CPU Error Address register is also used to latch the address in case of CPU address decoding error. More over, CPUOut interrupt bit is used both for CPU address decoding error as well as CPU parity error indication. In order for interrupt handler to distinguish between the two interrupts events, it needs to read CPU Error Data register and compare against its previous value. If register value changed, it implies it is a parity error.

The CPUOut interrupt bit is set in case of parity error only if SysADCValid bit in CPU Configuration register is set to 1.

6.5 Data Integrity Flow

6.5.1 CPU writes to SDRAM and PCI

The GT-96100A samples the SysADC (CPU parity) lines when the CPU performs a write transaction to PCI or SDRAM. Parity checking is performed by the GT-96100A.

If a parity error occurs, the GT-96100A latches the address, data, and parity lines, see [Section 6.6 “Register Information” on page 150](#). The GT-96100A also generates an interrupt (via Interrupt*) to the CPU indicating a parity error has been detected on the CPU parity lines.

In addition, the GT-96100A forces two ECC errors when writing the data to the SDRAM. This feature is configured through ForceEccErr bit in SDRAM Configuration register, see [Section 6.6 “Register Information” on page 150](#). This will make sure that when the data is read by another resource, the ECC bits will indicate erroneous data to the reading device.

6.5.2 CPU reads from SDRAM

The GT-96100A samples the ADP (SDRAM ECC) lines when the CPU performs a read transaction from the SDRAM. An ECC check is performed by the GT-96100A SDRAM interface.

In case there is a 2-bit ECC error, the GT-96100A generates an interrupt to the CPU and drives the Bus_Err bit (SysCmd[5]) to the CPU. The SysADC lines will NOT drive the wrong value, assuming the Bus_Err and the interrupt are sufficient indications for the CPU.

In case of a single-bit ECC error, the default does NOT interrupt the CPU and the error is corrected. The GT-96100A can be programmed to interrupt the CPU on single bit ECC errors, see [Section 6.6 “Register Information” on page 150](#). Regardless, the data is corrected and then returned to the CPU. Also, the address, data, ECC bits, and an indication for single or double ECC errors are latched in the GT-96100A ECC error report registers, see [Section 6.6 “Register Information” on page 150](#).

6.5.3 CPU reads from PCI

When the CPU reads data from the PCI, the PAR signal (and PAR64 in case of 64-bit PCI) is sampled. In case PAR does NOT match the data, an interrupt to the CPU is generated.

6.5.4 PCI writes to GT-96100A SDRAM

When a PCI master writes data to the GT-96100A's SDRAM, the PAR signal (PAR64 in case of 64-bit PCI) is sampled. In case PAR does NOT match the data, PERR* is asserted on the PCI bus and an interrupt to the CPU is generated. The ECC bits will not be intentionally damaged while being written to the SDRAM (GT-96100A will generate correct ECC to SDRAM based on the written data, ignoring the bad PAR indication).

6.5.5 PCI reads from the SDRAM

When a PCI master reads data from the GT-96100A's SDRAM, the GT-96100A samples the ADP (SDRAM ECC) lines. An ECC check is performed by the GT-96100A.

In case there is a 2-bit ECC error, the GT-96100A generates an interrupt to the CPU (and PCI) and defaults to driving PAR with the correct value matching the data. This feature is configured, see [Section 6.6 "Register Information" on page 150](#), and PAR can be chosen to be driven with a value NOT matching the data.

In case of a single ECC error, an interrupt to the CPU (and PCI) is generated and the data is corrected and returned to the PCI. PAR will drive a correct value, matching the data.

In both cases the address, data, ECC bits, and an indication for single or double ECC errors are latched in the GT-96100A ECC error report registers.

6.5.6 DMA cycles

There is no change in the implementation of data integrity during DMA transfers from the GT-64120 device. If a parity/ECC error is detected during a DMA transfer from the SDRAM or from the PCI, an interrupt is generated.

In case of a single-bit ECC error, the default does NOT interrupt the CPU and the error is corrected. The GT-96100A can be programmed to interrupt the CPU on single-bit ECC errors, see [Section 6.6 "Register Information" on page 150](#). Regardless, the data is corrected and then returned to the DMA Unit. Also, the address, data, ECC bits, and an indication for single or double ECC errors are latched in the GT-96100A ECC error report registers.

6.6 Register Information

Table 124 describes the registers used to implement parity and ECC in the GT-96100A.

Table 124: Registers for Implementing Parity and ECC

Register	Offset	Description
CPU Error Address	0x70	Holds the lower 32 bits of the address during a parity error on SysADC (CPU write).
	0x78	Holds the upper 5 bits of the CPU address during a parity error on SysADC (CPU write).
CPU Error Data	0x128	Holds the lower 32 bits of the data during a parity error on SysADC (CPU write).
	0x130	Holds the upper 32 bits of the data during a parity error on SysADC (CPU write).
CPU Error Parity	0x138	Holds the SysADC lines when parity error is detected (CPU write).
ECC Error Address	0x490	<ul style="list-style-type: none"> • Bits [31:2] holds the 30 MSB of the address to the SDRAM when an ECC error is detected (CPU/PCI/DMA read from SDRAM). • Bit [1], if active, indicates that two or more ECC errors are detected. • Bit [0] - If active, indicates that a single bit ECC error is detected.
ECC Error Data	0x480	Holds the upper 32 bits of the data when an error is detected (CPU/PCI/DMA read from SDRAM).
	0x484	Holds the lower 32 bits of the data when an error is detected (CPU/PCI/DMA read from SDRAM).
ECC from Memory	0x488	Holds the ECC read from memory when an error is detected.
ECC Calculation	0x48C	Holds the value calculated by the GT-96100A as correct ECC. This value may be helpful during debug.
Force bad PAR on PCI read bad ECC from SDRAM	0xc00, bit 26	0 - Par always drives matching value (Default). 1 - Par will drive wrong value if ECC error detected.
	0xc80, bit 26	
Force SDRAM ECC error on CPU writes with bad parity	0x448, bit 17	0 - SDRAM interface unit writes two ECC errors to the SDRAM (Default). 1 - ECC will always be written correctly to the SDRAM.
Interrupt for 1 or 2 ECC errors	0x448, bit 18	0 - Interrupt only if two ECC errors are detected (Default). 1 - Interrupt when one or two ECC errors are detected.

6.7 CPU Errors Report Registers

Table 125: CPU Error Address (Low), Offset: 0x070

Bits	Field Name	Function	Initial Value
31:0	llegLoAdd	This register captures bits 31:0 of an illegal 36-bit address, or an address of a CPU write transaction with bad parity driven on SysADC.	0x0

Table 126: CPU Error Address (High), Offset: 0x078

Bits	Field Name	Function	Initial Value
3:0	llegHiAdd	This register captures bits 35:32 of an illegal 36-bit address, or an address of a CPU write transaction with bad parity driven on SysADC.	0x0
31:4	Reserved		0x0

Table 127: CPU Error Data (Low), Offset: 0x128

Bits	Field Name	Function	Initial Value
31:0	DataErr	Holds the lower 32 bits of the data during a parity error on SysADC (CPU write).	0xffffffff

Table 128: CPU Error Data (High), Offset: 0x130

Bits	Field Name	Function	Initial Value
31:0	DataErr	Holds the upper 32 bits of the data during a parity error on SysADC (CPU write).	0xffffffff

Table 129: CPU Error Parity, Offset: 0x138

Bits	Field Name	Function	Initial Value
7:0	ParErr	Holds the SysADC lines when a parity error is detected (CPU write).	0xff
31:8	Reserved	Reserved.	0x0

7. PCI INTERFACES

The GT-96100A can be configured to have two 32-bit PCI buses, or one 64-bit PCI bus; all compliant with Revision 2.1 of the PCI Specification.

Both 3.3V and 5V operations are supported through the use of universal PCI buffers. Support for the I₂O specification is also included.

7.1 Reset Configuration

At reset, the GT-96100A can be configured to have one 64-bit PCI interface or two 32-bit PCI interfaces, see [Section 22. “Reset Configuration” on page 452](#). Each PCI interface can be either a master initiating a PCI bus operation or a target responding to a PCI bus operation.

NOTE: When the GT-96100A is configured with two 32-bit PCI interfaces, both interfaces are almost identical in behavior and structure. The only difference is that PCI interface 1 (PCI_1) does not support locked cycles and does not have a separate interrupt signal.

If no reference is made to a particular PCI interface, then the description in this section applies to both interfaces.

7.2 PCI Master Operation

When the CPU or the internal DMA machine initiates a bus cycle to a PCI device, the GT-96100A needs to decode the target address to determine which address space is being accessed.

If the GT-96100A is configured as one 64-bit PCI interface, then PCI_0 registers are used for address comparison and remapping.

If the GT-96100A is configured with two 32-bit PCI buses, the address registers of both interfaces are used for comparison. Based on the match results, either PCI_0 or PCI_1 becomes the master on the PCI bus and translates the cycle into the appropriate PCI bus cycle.

Supported master PCI cycles are:

- Memory Read
- Memory Write
- Memory Read Line
- Memory Read Multiple
- Memory Write and Invalidate
- I/O Read
- I/O Write
- Configuration Read
- Configuration Write
- Interrupt Acknowledge
- Special Cycle

Memory Write and Invalidate and Memory Read Line cycles are carried out when the transaction accessing PCI memory space requests a data transfer equal to the PCI cache line size. In case the transfer initiator is a DMA engine, the requested address must be cache line aligned. In case of write transaction, Memory Write and Invalidate Enable bit in the Configuration Command register must be set. When the PCI cache line size is set equal to 0, the GT-96100A never issues Memory Write and Invalidate or Memory Read Line cycles.

Memory Read Multiple is carried out when the transaction accessing PCI memory space requests a data transfer greater than the PCI cache line size.

As a master, the GT-96100A does not issue Dual Address cycles (DAC) or Lock cycles on the PCI.

The PCI posted write buffer in the GT-96100A permits the CPU to complete CPU-to-PCI memory writes even if the PCI bus is busy. The posted data is written to the target PCI device when the PCI bus becomes available.

7.2.1 PCI Master CPU Address Space Decode and Translation

Local masters access the PCI space through the PCI_0/1 Memory 0, PCI_0/1 Memory 1, and PCI_0/1 I/O decoders in CPU address space.

CPU accesses claimed by these decoders are translated into the appropriate PCI cycles by the appropriate PCI interface (PCI_0 only in case of 64-bit PCI interface). The address seen on the CPU bus is copied directly to the PCI bus (unless the CPU-to-PCI address remapping capability is enabled.) For example, if an access to 0x1200.0040 is programmed to be bridged as a memory read from PCI, the active PCI address for this cycle will be 0x1200.0040.

7.2.2 PCI Master CPU Byte Swapping

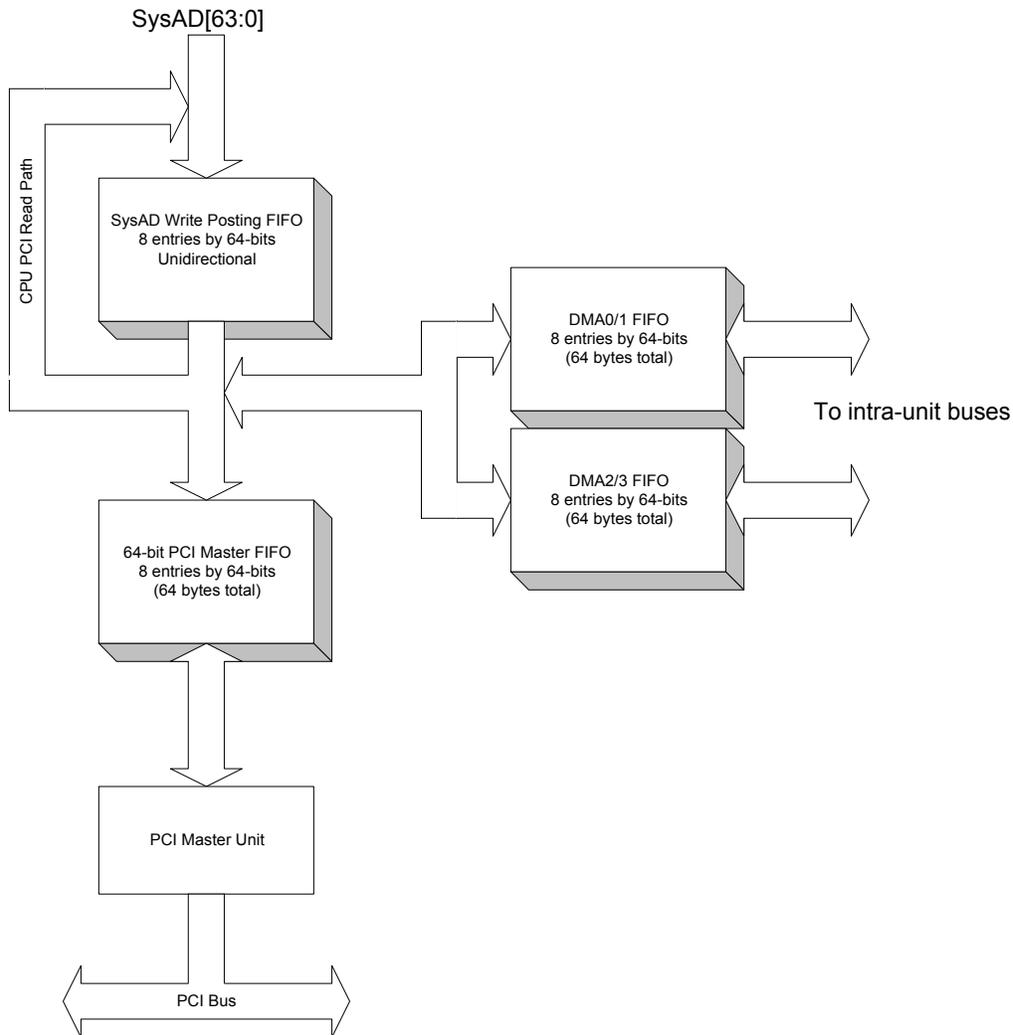
All accesses to PCI space by the CPU can have the data byte order swapped as the data moves through the GT-96100A. Byte swapping is turned on via the MByteSwap bit in the PCI Internal Command register (0xc00.)

When the GT-96100A is configured for 64-bit PCI mode, byte swapping occurs across all eight byte lanes when the ByteSwap bit is set for PCI_0.

7.2.3 PCI Master FIFOs

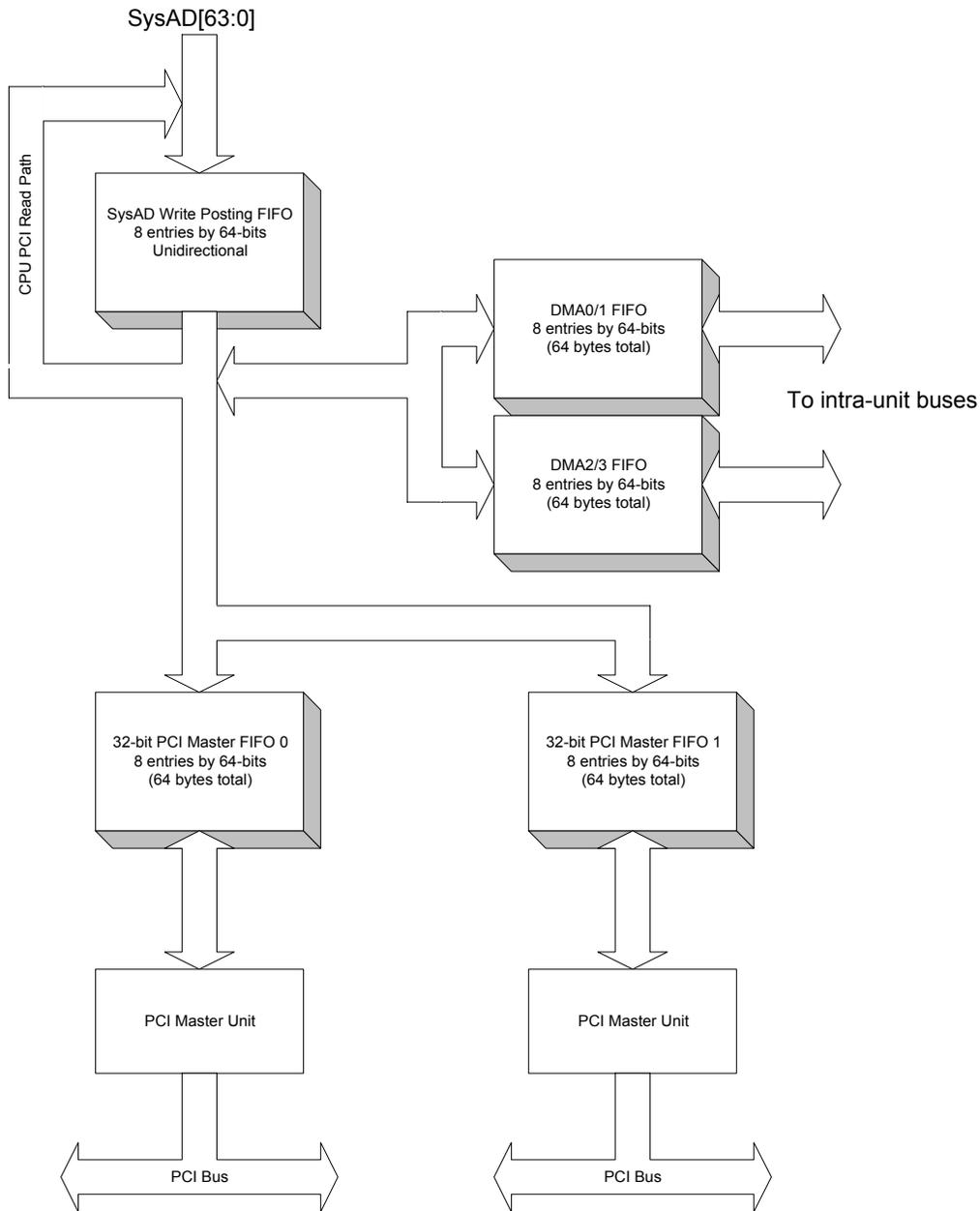
If the GT-96100A is configured to have one 64-bit PCI interface, then this interface includes a FIFO of eight entries, each 64-bits wide (64 bytes total), see [Figure 25](#).

Figure 25: PCI Master FIFOs in Single 64-bit Mode



If the GT-96100A is configured to have two 32-bit PCI interfaces, then each master interface includes its own FIFO of eight entries, each 64-bits wide, see [Figure 26](#). During writes to the PCI interface, the target PCI unit receives write data from the CPU interface or the DMA unit. When the PCI bus is granted, the PCI master's FIFO delivers the write data to the target on the PCI bus.

Figure 26: PCI Master FIFOs in Dual 32-bit Mode



Upon receiving the first 32- or 64-bit data from the CPU interface or DMA unit, the PCI master interface requests the PCI bus, if the GT-96100A is not already parked. Once granted, the appropriate write cycle is started on the PCI bus.

During reads, the PCI master interface FIFO receives read data from the PCI bus and delivers it to the CPU interface or the DMA unit. Upon receiving the first word or doubleword from the PCI target, the data is forwarded to the requesting unit (CPU interface or DMA unit). The GT-96100A supports sub-block ordering during CPU reads, therefore if the original read request address is not aligned to a cache line boundary, then the first 32-bit word (or 64-bit double-word in case of a 64-bit PCI interface) returned to the requesting unit is delayed until it is received from the PCI target, since reads across the PCI bus are linear.

The GT-96100A internal architecture allows zero wait-state data transfer over the PCI bus (IrDY* continuously asserted) during both master reads and writes.

7.2.4 PCI Master DMA

The GT-96100A's internal DMA engines act as PCI bus masters while transferring data to/from the PCI bus. The DMA engines only issue PCI memory space read and write cycles. The type of cycle issued follows the same rules as for the CPU.

The DMA engines transfers data between PCI devices using the on-chip DMA FIFOs for temporary storage.

NOTE: See [Section 9. “Independent DMA Controllers \(IDMA Controllers\)” on page 219](#) for a detailed description of the DMA engines.

7.2.5 PCI Master RETRY Counter

RETRYs detected by the PCI master interface are normally handled transparently from the point of view of the CPU or DMA engines.

In some rare circumstances, however, a target device may RETRY the GT-96100A excessively (or forever.) Use the Retry Counter to recover from this condition. Every time the number of RETRYs equals the value in the Retry Counter, the GT-96100A aborts the cycle and sends an interrupt to the CPU. If the cycle was a read, undefined data is returned and the ERROR bit is set within the data identifier.

Disable the Retry Counter by setting the Retry count to zero.

7.3 PCI Target Interface

The GT-96100A responds to the following PCI cycles as a target device:

- Memory Read
- Memory Write
- Memory Read Line
- Memory Read Multiple
- Memory Write and Invalidate
- I/O Read
- I/O Write
- Configuration Read
- Configuration Write
- Locked Read to PCI_0 only
- Locked Write to PCI_0 only

The GT-96100A will not act as a target for Interrupt Acknowledge, Special, and Dual Address cycles. These cycles are ignored.)

7.3.1 PCI Target FIFOs

The GT-96100A incorporates dual 64-byte posted write/read prefetch buffers, per PCI interface. These buffers allow full memory (AD) and PCI bus concurrency. The dual FIFOs can operate in a “ping-pong” fashion, each FIFO alternating between filling and draining, see [Figure 27](#) and [Figure 28](#).

When the GT-96100A is the target of PCI write cycles, data is first written to one of the FIFOs. When the first FIFO fills up (64 bytes), the data is written to the destination from the first FIFO while the second FIFO is filled. This “ping-pong” operation continues as long as data is received from the PCI bus. The GT-96100A de-asserts TRDY for 2 PCI clocks while switching target FIFOs.

Occasionally, the PCI target interface cannot drain the FIFOs (i.e. write to local memory) as fast as data is received. This occurs when access to memory is prevented (possibly by excessive CPU accesses) or when the target memory is particularly slow. In this case, the GT-96100A’s PCI target interface de-asserts TRDY until one of the FIFOs is empty again and might even issue a DISCONNECT to the PCI bus if reached timeout, see [Section 22. “Reset Configuration” on page 452](#).

The target FIFOs are also used to align data bursts that do not start on 64-byte boundaries for more efficient processing by the GT-96100A’s memory subsystem. When an incoming burst passes a 64-byte boundary, the target FIFOs switch and the remainder of the burst (now aligned to a 64-byte boundary) fills the new FIFO. TRDY de-asserts for two PCI clocks when the FIFO switch occurs.

Figure 27: PCI Target Interface “Ping-Pong” FIFOs

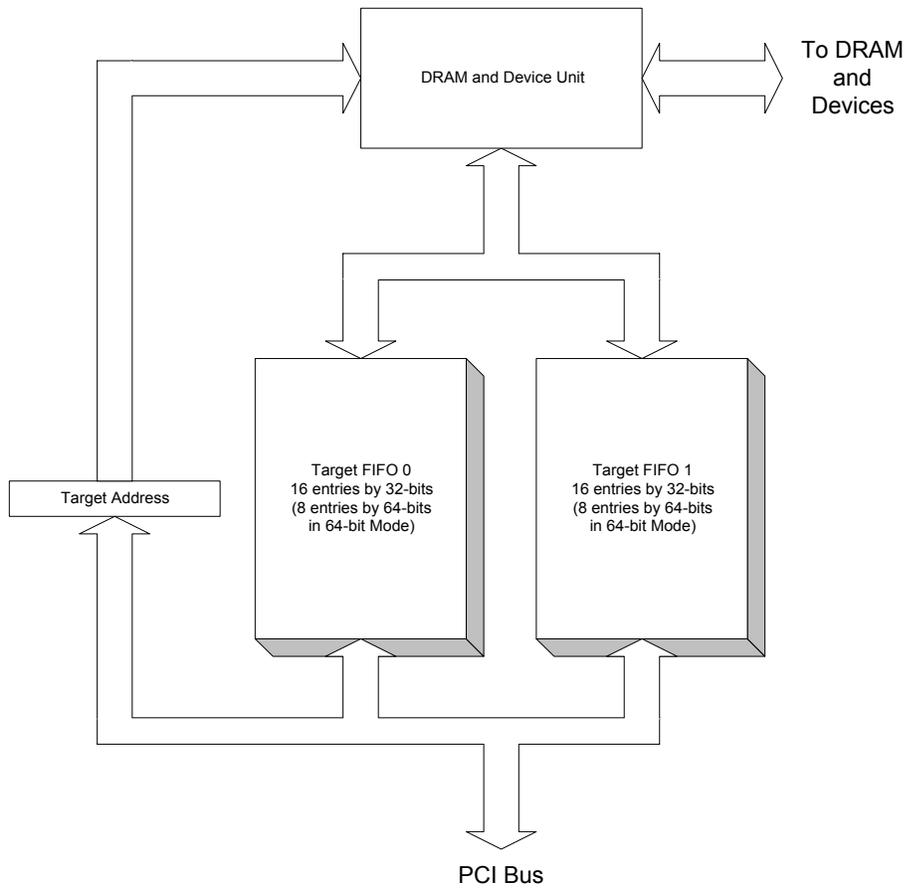
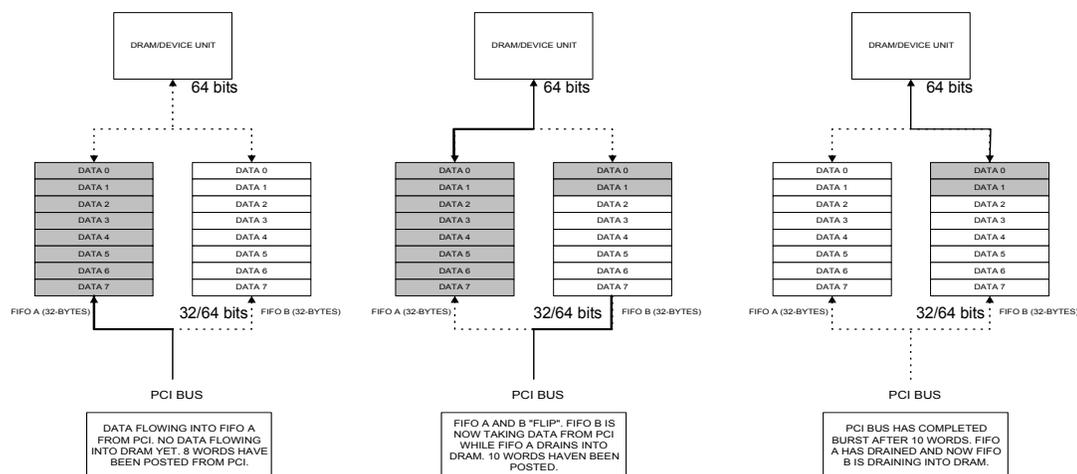


Figure 28: PCI Target Interface FIFOs Operational Example



NOTE: Graphic shows only 8 of the 16 entries in the FIFOs (in 32-bit mode)

7.3.2 Controlling Burst Length

The GT-96100A's memory interface is capable of maximum burst of eight data bursts to SDRAM or devices, regardless of the device width. This implies that if the device is 64-bits wide, it can burst up to 64 bytes in a single transaction. If the device is 32-bits wide it can burst up to 32 bytes per transaction and so on.

The PCI target is limited to requesting burst reads/writes from the memory interface that are not greater than the bursts supported by the controller. Each PCI target FIFO corresponds to a single memory interface transaction. The PCI target has a programmable FIFO depth. The MaxBurst register (offset 0xc40) defines the FIFO depth per each Base Address Register. If the FIFO depth is set to 16 (MaxBurst = 1) then the maximum burst is 64 bytes to/from the memory interface. If the FIFO depth is eight (MaxBurst = 0) then a maximum burst of 32 bytes is supported on the memory interface.

The system's software must program the MaxBurst register properly.

NOTE: In order to support PCI bursts to a 32-bit SDRAM or device, MaxBurst must be set to 0. This is also true for a 64-bit SDRAM programed to burst length of 4.

MaxBurst may be used for performance optimization in systems with 64-bit wide memory. In some cases, it is sometimes preferred to keep the bursts short to allow CPU or DMA to get more memory interface bandwidth. In this case, setting the MaxBurst value to 8 may be an appropriate strategy.

7.3.3 PCI Target Read Prefetching

The target FIFOs are also used for read prefetch. The Memory Read (MR), Memory Read Line (MRL), and Memory Read Multiple (MRM) cycles are executed as prefetchable read cycles.

The Device/DRAM Unit fills the target FIFOs as soon as it is determined that the PCI request will be longer than a single word (based on how long Frame* is asserted.) The “aggressiveness” of the prefetch is controlled by the type of read command (MR/MRL/MRM) and the state of the bits for each of the read command types in the Prefetch/Max_Burst register.

By default, the only “aggressive” prefetched read is the Memory Read Multiple. Both Memory Read and Memory Read Line commands are marked for aggressive prefetch via the Prefetch/Max_Burst register. With aggressive prefetch, as soon as at least one word is delivered from the FIFO to the PCI bus, the PCI slave requests from the Device/DRAM unit to prefetch into the second FIFO.

NOTE: When using aggressive prefetch, the upper address allowed to be read from the PCI is last DRAM address minus 0x8.

In a non-aggressive prefetch, read cycle data is prefetched into only one of the target FIFOs. Data is not fetched into the second FIFO until all the data from the first FIFO is delivered to the PCI bus. MR and MRL cycles are by default, non-aggressive prefetch.

NOTE: All three types of read commands will result in prefetch (multiple read cycles) on the Device/DRAM bus unless Frame* is only asserted for a single cycle.

Cycles to internal registers and Configuration cycles are non-postable nor prefetchable. These cycles are always single word.

NOTE: If a PCI master attempts to burst transaction to internal register, the GT-96100A disconnects after the first TRDY*.

7.3.4 PCI Target Address Space Decode and Byte Swapping

The GT-96100A decodes accesses on the PCI bus, for which it may be a target, by the values programmed into the Base Address registers (BARs).

There are two sets of BARs for each PCI interface: regular BARs (in PCI Function 0) and swap BARs (in PCI Function 1). Accesses decoded by the regular BARs are passed without modifying the data to or from the target memory device. Accesses decoded by the swap BARs are passed with to or from the target memory device after converting the endianness of the data (e.g. little-endian to big-endian).

The GT-96100A uses a two stage decode process for accesses through the PCI devices target interface. Once a PCI access is determined to be a “hit” based on the BAR comparison, the address is passed to the Device Unit for sub-decode. For example, PCI_0 Base Address Register 0 in Function 0 (PCI_0 BAR0) decodes non-byte swapped accesses to the SDRAM controlled by either SCS[0]* or SCS[1]*. The GT-96100A then uses the values programmed into the SCS[0]* Low and SCS[0]* High decode registers to determine if the access is to the SDRAM in SCS[0]* space.

NOTE: The second stage decoders are shared with the CPU, see [Figure 9 on page 79](#).

If the target address of a PCI write transaction “hits” based on the BAR decode, then misses in the Device Unit, transaction is completed properly on the PCI bus, but data is NOT written to memory. In case of read transaction, although there is no actual read from memory, transaction is completed properly on PCI bus, but the data driven

on the bus is undefined (unless timeout is reached first - RETRY termination). In both cases, MemOut bit in interrupt cause register is set (and interrupt is generated if not masked).

7.3.5 Enhancing Target Interface Performance

The GT-96100A includes special performance tuning features for the PCI target interface. The PCI_0/1 Timeout0 and Timeout1 registers allow the designer to force the GT-96100A to wait either longer than normal, or shorter than normal, before issuing a RETRY/DISCONNECT.

The Timeout0 value of PCI device0 or device1 sets the number of clocks the device will wait for the first data of an access before issuing a RETRY. The Timeout1 value sets the number of clocks the device will wait between subsequent data phases during an access before issuing a DISCONNECT. The PCI 2.1 specifications sets the maximum for both of these at 16 clocks (Timeout0) and 8 clocks (Timeout1) respectively. However, in many systems, especially those with long SDRAM latencies, it may be necessary to “bend” these restrictions.¹

NOTE: A value of 0x00 in Timeout0 or Timeout1 means "never timeout"

If excessive RETRY/DISCONNECT behavior occurs in the system, lengthen the Timeout0/1 values. This behavior may result from excessive memory activity due to the CPU or DMA engines and the PCI interface failing to access the SDRAM within 16 clocks.

The settings in the Prefetch/Max_Burst registers provide another area for performance optimization. Turning on aggressive prefetch for all read types results in a significant performance improvement, depending on the length and type of read commands.

NOTE: If a system uses many short reads from a PCI, aggressive prefetch on speculative reads that are not used wastes bandwidth on the Device/DRAM bus.

7.4 PCI Synchronization Barriers

The GT-96100A considers some cycles to be “synchronization barrier” cycles. In such cycles, the GT-96100A confirms that at the end of the cycle there remains no posted data within the chip. These cycles can be initiated either from the PCI slave side or the CPU side.

The slave “synchronization barrier” cycles are Lock Read (for PCI_0 only) and Configuration Read. If there is no posted data within the device, the cycle ends normally. If after a retry period there is still posted data, the cycle is retried. Until the original cycle ends, any other (different address/command) “synchronization barrier” cycles are retried.

Lock Read is a “synchronization barrier” cycle that lasts during the entire Lock period. For Example, when the slave of PCI_0 is locked, all Configuration Reads are retried. Also, all cycles addressed to internal registers are retried until Lock ends.

The CPU interface treats I/O Reads to PCI and Configuration Reads as “synchronization barrier” cycles as well. These reads are responded to once no posted data remains within the GT-96100A.

1. The PCI specification also states that a master may not enforce target rules. In other words, even if the GT-96100A takes longer than 16 clocks to return the first data, the master must wait.

The GT-96100A provides the CPU with a simpler way to perform synchronization with PCI buffers¹. The CPU issues a read command to “PCI_0/1 Sync Barrier Virtual” register to synchronize PCI_0/1 buffers. The response dummy read completes once no posted data remains within the addressed PCI interface.

7.5 PCI Master Configuration

The GT-96100A translates CPU read and write cycles into configuration cycles using PCI configuration mechanism #1 (per the PCI specification). Mechanism #1 defines a way to translate the CPU cycles into both PCI configuration cycles on the PCI bus and accesses to the GT-96100A’s internal configuration registers.

The GT-96100A includes two registers per PCI device: Configuration Address (at offset 0xcf8 for PCI0 and 0xcf0 for PCI1) and Configuration Data (at offset 0xcfc for PCI0 and 0xcf4 for PC1). The mechanism for accessing configuration registers is to write a value into the Configuration Address register that specifies:

- PCI bus number.
- The device on that bus.
- The function number within the device.
- The configuration register within that device/function being accessed.

A subsequent read or write to the Configuration Data register (at 0xcfc/0xcf4) then causes the GT-96100A to translate that Configuration Address value to the requested cycle on the PCI bus.

If the BusNum field in the Configuration Address register equals 0 but the DevNum field is other than 0, a Type0 access is performed that addresses a device attached to the local PCI bus. If the BusNum field in the Configuration Address register is other than 0, a Type1 access is performed that addresses a device attached to a remote PCI bus.

The GT-96100A performs address stepping for PCI configuration cycles. This allows for the use of the high-order PCI AD signals as IdSel signals through resistive coupling.² Table 130 shows DevNum to IdSel mapping.

Table 130: DevNum to IdSel Mapping

DevNum[15:11]	PAD_0[31:11]/PAD_1[31:11]
00001	0 0000 0000 0000 0000 0001
00010	0 0000 0000 0000 0000 0010
00011	0 0000 0000 0000 0000 0100
00100	0 0000 0000 0000 0000 1000
-	-
-	-
-	-
10101	1 0000 0000 0000 0000 0000
00000, 10110 - 11111	0 0000 0000 0000 0000 0000

1. This mechanism is not compatible with the GT-64010A and GT-64011 devices.

2. “Resistive Coupling” is a fancy way of saying “hook a resistor from ADx to IdSel” on a given device. Look at the Galileo-4PB backplane schematics for examples.

The CPU accesses the GT-96100A's internal configuration registers when the fields DevNum and BusNum in the Configuration Address register are equal to 0. The GT-96100A Configuration registers are also accessed from the PCI bus when the GT-96100A is a target responding to PCI configuration read and write cycles.

The CPU accesses the GT-96100A internal PCI_1 configuration registers through PCI_0 Configuration Address and Data registers (0xcf8,0xcfc). For example, in order to access the GT-96100A's PCI_1 Class Code and Rev Id register, CPU software should write 0x80000088 to PCI_0 Configuration Address register (0xcf8), and then read/write PCI_0 Configuration Data register. CPU will use PCI_1 Configuration Address register (0xcf0) and Configuration Data register (0xcf4) only in order to generate a configuration read/write transaction on PCI_1 bus.

The CPU interface unit cannot distinguish between an access to the GT-96100A's PCI configuration space and an access to an external PCI device configuration space. Both are accessed using an access to the GT-96100A's internal space (i.e. Configuration Data register). The software engineer must keep this in mind, especially when byte swapping is enabled on the PCI interface. In this case, internal configuration registers and configuration registers in external devices will appear to have a different neediness.

The configuration enable bit (ConfigEn) in the Configuration Address register must be set before the Configuration Data register is read or written. An attempt of CPU read a configuration register without this bit set, will result in undefined data returned on Sassed bus

7.5.1 Special Cycles and Interrupt Acknowledge

A Special cycle is generated whenever the Configuration Data register is written to and the Configuration Address register has been previously written with 0 for BusNum, 1f for DevNum, 7 for FunctNum and 0 for Reg-Num.

An Interrupt acknowledge cycle is generated whenever the Interrupt Acknowledge (0xc34) register is read.

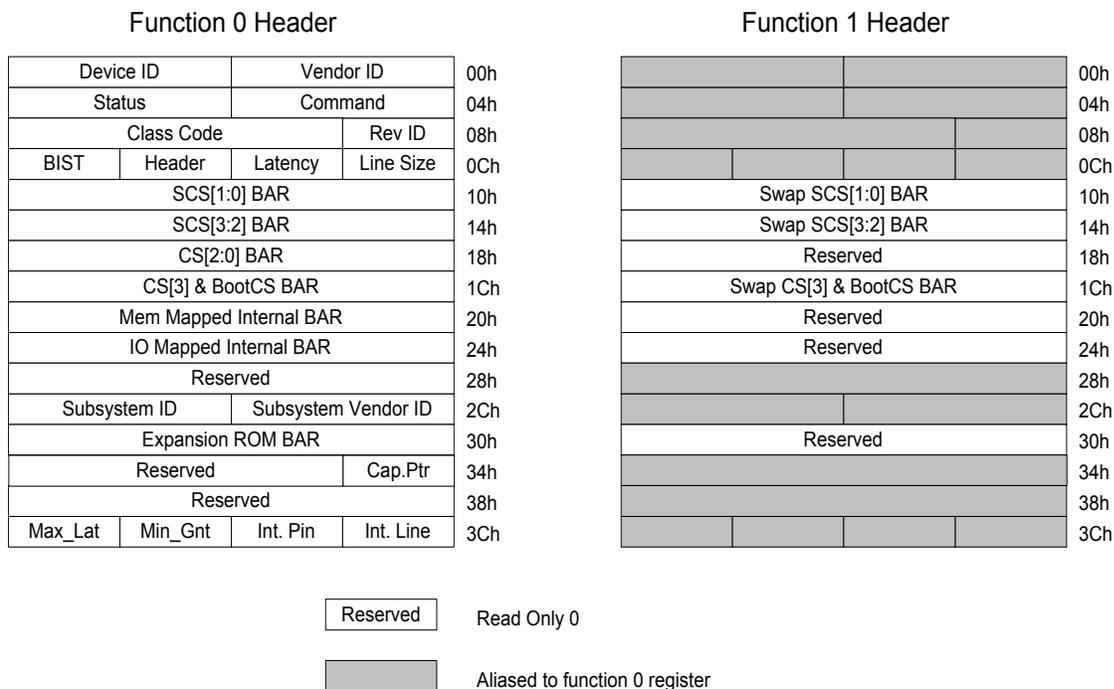
7.6 Target Configuration and Plug and Play

The GT-96100A includes all of the required plug and play PCI configuration registers. These registers, as well as the GT-96100A's internal registers are accessible from both the CPU and the PCI bus.

The GT-96100A acts as a two function device when being configured from the PCI bus. The base address registers available in Function 0 are used to decode accesses for which there is no byte swapping.

Function 1 is used to decode byte swapped addresses.

All other registers are shared between Function 0 and Function 1, as shown in [Figure 29](#).

Figure 29: PCI Configuration Header


7.6.1 Plug and Play Base Address Register Sizing

Systems adhering to the plug and play configuration standard determine the size of a base address register's decode range by first writing 0xFFFF.FFFF to the BAR and then reading back the value contained in the BAR. Any bits that were unchanged (i.e., read back a zero) indicate that they cannot be set and are therefore not part of the address comparison. With this information, the size of the decode region can be determined.¹

The GT-96100A responds to BAR sizing requests based on the values programmed into the Bank Size registers. These registers can be loaded automatically after RESET from the system ROM (see [Section 7.6.2 "PCI Autoload of Configuration Registers at RESET"](#) on page 164).

7.6.2 PCI Autoload of Configuration Registers at RESET

Thirteen PCI registers per PCI interface can be automatically loaded after Rst*. Autoload mode is enabled by asserting the DAdt[0] pin LOW on Rst*. Any PCI transactions targeted for the GT-96100A must be retried while the loading of the PCI configuration registers is in process.

¹. Refer to the PCI specification for more information on the BAR sizing process.

The PCI register values are loaded from the ROM controlled by BootCS*. These register values are shown in [Table 131](#) for PCI_0 and in [Table 132](#) for PCI_1. Although the configuration register information is typically stored in a Boot ROM device, a PLD device can also be programmed to store this information.

NOTE: The PLD should be programmed to support burst read cycles.

Table 131: PCI_0 Registers Loaded at RESET

Register	Offset	Boot Device Address
Command	0xc00	0x1ffff80
Timeout & Retry Counters	0xc04	0x1ffff84
RAS[1:0]* Bank Size	0xc08	0x1ffff88
RAS[3:2]* Bank Size	0xc0c	0x1ffff8c
CS[2:0]* Bank Size	0xc10	0x1ffff90
CS[3]* & Boot CS* Bank Size	0xc14	0x1ffff94
Conf_en	0xc3c	0x1ffff98
Prefetch/Max_burst	0xc40	0x1ffff9c
Device and Vendor ID	0x000	0x1ffffa0
Class Code and Revision ID	0x008	0x1ffffa4
Cache_line/Latency	0x00c	0x1ffffa8
Subsystem Device and Vendor ID	0x02c	0x1ffffac
Interrupt Pin and Line	0x03c	0x1ffffb0

Table 132: PCI_1 Registers Loaded at RESET

Register	Offset	Boot Device Address
Timeout & Retry Counters	0xc80	0x1ffffc0
Counter	0xc84	0x1ffffc4
RAS[1:0]* Bank Size	0xc88	0x1ffffc8
RAS[3:2]* Bank Size	0xc8c	0x1ffffcc
CS[2:0]* Bank Size	0xc90	0x1ffffd0
CS[3]* & Boot CS* Bank Size	0xc94	0x1ffffd4
Conf_en	0xcbc	0x1ffffd8
Prefetch/Max_burst	0xcc0	0x1ffffdc
Device and Vendor ID	0x080	0x1ffffe0
Class Code and Revision ID	0x088	0x1ffffe4
Cache_line/Latency	0x08c	0x1ffffe8

Table 132: PCI_1 Registers Loaded at RESET (Continued)

Register	Offset	Boot Device Address
Subsystem Device and Vendor ID	0x0ac	0x1ffffec
Interrupt Pin and Line	0x0bc	0x1fffff0

7.6.3 Expansion ROM Functionality

The GT-96100A implements the PCI Expansion ROM as required by PC boot devices. The PCI Expansion ROM functionality is enabled by strapping DAdr[5] low at RESET, see [Section 22. “Reset Configuration” on page 452](#).

If the PCI Expansion ROM is disabled, expansion ROM register (offset 0x30 of PCI configuration space) acts as reserved register and returns only 0 when read.

When the expansion ROM is enabled by the pin strapping option, the PCI Expansion ROM BAR appears in the GT-96100A’s PCI_0 configuration header. However, the Expansion ROM decoder shares functionality with the SCS[3]*/BootCS* resource. In normal operation (i.e., after the BIOS initialization is complete), the Expansion ROM is disabled via bit 0 in the Expansion ROM BAR. During BIOS boot, however, the BIOS “turns on” the Expansion ROM decoder by setting bit 0. When the Expansion ROM decoder is “on” the following happens in PCI_0:

- The PCI target acts as if the Timeout0 and Timeout1 MSBs are 1 (which means initial value of 0x8f and 0x87 respectively). This is done to allow for the long default access time from 8-bit boot ROMs.
- The Device unit will bypass it's address decoding for all transactions from PCI that are targeted to expansion ROM or SCS[3]*/BootCS*. All of these transactions assert CS[3] regardless of the actual address.
- The GT-96100A acts this way as long as bit[0] of expansion ROM register is set to 1. The BIOS must clear this bit when it is done executing/probing the Expansion ROM. If the BIOS does not clear bit[0] of expansion ROM BAR, program this bit to 0 from CPU side.

7.7 PCI Bus/Device Bus/CPU Clock Synchronization

The PCI interface is designed to run asynchronously with respect to the AD and CPU buses.

The synchronization delay between these two clock domains can be reduced by running the interfaces in synchronized mode. An example would be having the CPU/AD buses running at 66MHz and the PCI bus running at a 33MHz frequency that was derived from the 66MHz.

Latency through the GT-96100A is reduced to a minimum when synchronized mode is selected. The synchronization mode is set via the SyncMode bits in the PCI Internal Command registers (0xc00/0xc80). See [Section 2.4.2 “PCI Read Performance from SDRAM” on page 117](#) for a full description.

NOTE: Tclk cycle must be smaller than Pclk cycle by at least 1ns ($T_{clk} < T_{pclk} + 1ns$).

7.8 64-bit PCI Configuration

The GT-96100A is configured to work as a 64-bit PCI device if DAdr[2] and FRAME1*/REQ64* pins are sampled LOW on reset, see [Section 22. “Reset Configuration” on page 452](#).

NOTE: The hold time for REQ64*, in respect to RST* rise, is 0, as required by the PCI spec.

When the GT-96100A is configured to a 64-bit PCI, both master and target interfaces are configured to execute 64-bit transactions, whenever possible.

The PCI master interface always attempts to generate 64-bit transactions (asserts REQ64#) except for I/O or configuration transactions or when the required data is less than 64 bits. If the transaction target does not respond with ACK64#, the master completes the transaction as a 32-bit transaction.

The PCI target interface always responds with ACK64# to a 64-bit transaction, except for accesses to configuration space or internal registers.

NOTE: PClk1 must be tied to PClk0 on 64-bit PCI configuration.

7.9 Retry Enable

Some applications require that the local MIPS CPU program the PCI configuration registers in advance of other bus masters accessing them. In a PC add-in card application, for example, the MIPS CPU must set the device ID, BAR size requirements, etc. before the BIOS attempts to configure the card. The GT-96100A provides a mechanism by which the PCI target interface retries all transactions until this configuration is complete. This prevents race conditions between the local MIPS processor and the BIOS.

If DAdr[6] pin is sampled low on reset, the GT-96100A PCI target retries any transaction targeted to the GT-96100A's space. The GT-96100A stays in this retry mode until the Stop Retry bit in CPU configuration register (0x000) is set to 1.

7.10 Locked Cycles

The GT-96100A locks a cache line (fixed at 32 bytes) in the local memory address space when responding to Lock sequences on the PCI bus.

When a cache line is locked, any new PCI access to an address within the locked cache line (except accesses initiated by the LOCK owner) is terminated with RETRY. Also, every access from CPU or DMA to the locked cache line is on hold until the LOCK ends.

Although the GT-96100A does not support the Lock* pin on PCI_1, any access from PCI_1 to a locked cache line is not completed until the LOCK ends.

7.11 Hot-Swap Support

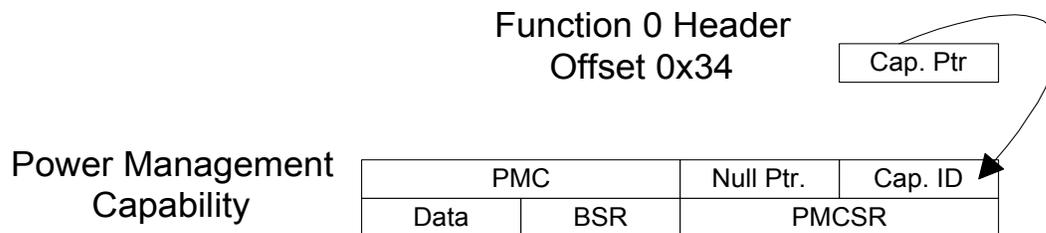
The GT-96100A is CompactPCI Hot Swap Capable. It supports the following hot swap device requirements:

- All PCI outputs float when RST# is asserted.
- All the GT-96100A's PCI state machines are kept in their idle state while RST# is asserted.
- The GT-96100A PCI interface does not leave its idle state until PCI bus is in an IDLE state. If reset is de-asserted in the middle of a PCI transaction, the GT-96100A PCI interface stays in its idle state until PCI bus is back in idle).
- The GT-96100A has no assumptions on clock behavior prior to its setup to the rising edge of RST#.
- The GT-96100A is tolerant of the 1V precharge voltage during insertion.
- The GT-96100A can be powered from Early VCC.

7.12 PCI Power Management Support

The GT-96100A is PCI Power Management compliant. It contains the required configuration registers as shown in [Figure 30](#).

Figure 30: Power Management Registers



The GT-96100A supports Power Management through reset configuration pins SDQM[1:0], each pin per each PCI interface.

If sampled HIGH, power management is enabled. Bit[4] of Configuration Status register is set, indicating the existence of a capability list. A capability list pointer register is implemented at offset 0x34, pointing to power management configuration registers implemented at offset 0x40 and 0x44.

If sampled LOW, capability list is not supported and a capability pointer as well as PMC registers are reserved.

Power Management registers are accessible from both CPU and PCI. Whenever PCI_0 or PCI_1 updates Power State bits (bits[1:0] of PMCSR register), bit[21] of interrupt cause register is set (bit[21] of high interrupt cause register in case of PCI_1 PMCSR configuration register) and an interrupt to CPU or PCI is generated, if not masked by interrupt mask registers. When Power Management is enabled, bit[21] in the interrupt cause register is used as a power management interrupt rather than a doorbell interrupt, see Interrupt section for details.

NOTE: The GT-96100A does not support its own power down. It only supports the software capability to power down the CPU or other on board devices.

7.13 PCI Interface Restrictions

7.13.1 General

1. TClk cycle must be smaller than PClk cycle by at least 1ns ($T_{clk} < T_{pclk} + 1ns$).

7.13.2 Master

1. Latency count, as specified in LatTimer (PCI Configuration Register 0x00c), must not be programmed to less than 6.

7.13.3 Slave

1. The set bits in the Bank Size registers must be sequential.
2. When the slave is locked, in order to prevent a deadlock, all transactions to internal registers (I/O or memory cycles) are not supported (retry will be issued).
3. Timeout0 value (PCI internal Register 0xc04) must not be programmed to less than 2.
4. All PCI reads result in prefetch read from the target device regardless of the BAR prefetch bit value, unless FRAME* is asserted for a single clock cycle.
5. If 64-bit SDRAM/device is used that supports bursts of eight 64-bit words, the MaxBurst can only be set to 1 (i.e. 64 bytes).

7.14 PCI Control and Configuration Registers

Table 133: PCI Control and Configuration Register Map

Description	Offset	Page Number
PCI Internal		
PCI_0 Command	0xc00	page 172
PCI_1 Command	0xc80	page 174
PCI_0 Time Out & Retry	0xc04	page 174
PCI_1 Time Out & Retry	0xc84	page 174
PCI_0 SCS[1:0]* Bank Size	0xc08	page 175
PCI_1 SCS[1:0]* Bank Size	0xc88	page 175
PCI_0 SCS[3:2]* Bank Size	0xc0c	page 175
PCI_1 SCS[3:2]* Bank Size	0xc8c	page 176
PCI_0 CS[2:0]* Bank Size	0xc10	page 176
PCI_1 CS[2:0]* Bank Size	0xc90	page 176
PCI_0 CS[3]* & Boot CS* Bank Size	0xc14	page 177

Table 133: PCI Control and Configuration Register Map (Continued)

Description	Offset	Page Number
<i>PCI Internal (Continued)</i>		
PCI_1 CS[3]* & Boot CS* Bank Size	0xc94	page 177
PCI_0 Internal Registers Space Size	0xc20	page 177
PCI_1 Internal Registers Space Size	0xca0	page 178
PCI_0 Base Address Registers' Enable	0xc3c	page 178
PCI_1 Base Address Registers' Enable	0xcbc	page 179
PCI_0 Prefetch/Max Burst Size	0xc40	page 179
PCI_1 Prefetch/Max Burst Size	0xcc0	page 179
PCI_0 SCS[1:0]* Base Address Remap	0xc48	page 180
PCI_1 SCS[1:0]* Base Address Remap	0xcc8	page 180
PCI_0 SCS[3:2]* Base Address Remap	0xc4c	page 181
PCI_1 SCS[3:2]* Base Address Remap	0xccc	page 181
PCI_0 CS[2:0]* Base Address Remap	0xc50	page 182
PCI_1 CS[2:0]* Base Address Remap	0xcd0	page 182
PCI_0 CS[3]* & Boot CS* Address Remap	0xc54	page 182
PCI_1 CS[3]* & Boot CS* Address Remap	0xcd4	page 182
PCI_0 Swapped SCS[1:0]* Base Address Remap	0xc58	page 180
PCI_1 Swapped SCS[1:0]* Base Address Remap	0xcd8	page 180
PCI_0 Swapped SCS[3:2]* Base Address Remap	0xc5c	page 181
PCI_1 Swapped SCS[3:2]* Base Address Remap	0xcdc	page 181
PCI_0 Swapped CS[3]* & BootCS* Base Address Remap	0xc64	page 183
PCI_1 Swapped CS[3]* & BootCS* Base Address Remap	0xce4	page 183
PCI_0 Configuration Address	0xcf8	page 183
PCI_1 Configuration Address	0xcf0	page 184
PCI_0 Configuration Data Virtual Register	0xcfc	page 184
PCI_1 Configuration Data Virtual Register	0xcf4	page 184
PCI_0 Interrupt Acknowledge Virtual Register	0xc34	page 184
PCI_1 Interrupt Acknowledge Virtual Register	0xc30	page 184

Table 133: PCI Control and Configuration Register Map (Continued)

Description	Offset	Page Number
PCI Configuration		
PCI_0 Device and Vendor ID	0x000	page 185
PCI_1 Device and Vendor ID	0x080	page 185
PCI_0 Status and Command	0x004	page 186
PCI_1 Status and Command	0x084	page 186
PCI_0 Class Code and Revision ID	0x008	page 188
PCI_1 Class Code and Revision ID	0x088	page 188
PCI_0 BIST, Header Type, Latency Timer, Cache Line	0x00c	page 188
PCI_1 BIST, Header Type, Latency Timer, Cache Line	0x08c	page 189
PCI_0 SCS[1:0]* Base Address	0x010	page 190
PCI_1 SCS[1:0]* Base Address	0x090	page 190
PCI_0 SCS[3:2]* Base Address	0x014	page 191
PCI_1 SCS[3:2]* Base Address	0x094	page 191
PCI_0 CS[2:0]* Base Address	0x018	page 191
PCI_1 CS[2:0]* Base Address	0x098	page 192
PCI_0 CS[3]* & Boot CS* Base Address	0x01c	page 192
PCI_1 CS[3]* & Boot CS* Base Address	0x09c	page 192
PCI_0 Internal Registers Memory Mapped Base Address	0x020	page 193
PCI_1 Internal Registers Memory Mapped Base Address	0x0a0	page 193
PCI_1 SCS[1:0]* Base Address	0x090	page 190
PCI_0 SCS[3:2]* Base Address	0x014	page 191
PCI_0 Internal Registers I/O Mapped Base Address	0x024	page 193
PCI_1 Internal Registers I/O Mapped Base Address	0x0a4	page 193
PCI_0 Subsystem ID and Subsystem Vendor ID	0x02c	page 194
PCI_1 Subsystem ID and Subsystem Vendor ID	0x0ac	page 194
Expansion ROM Base Address Register	0x030	page 194
PCI_0 Interrupt Pin and Line	0x03c	page 195
PCI_1 Interrupt Pin and Line	0x0bc	page 195

Table 133: PCI Control and Configuration Register Map (Continued)

Description	Offset	Page Number
PCI Configuration, Function 1		
PCI_0 Swapped SCS[1:0]* Base Address	0x110	page 198
PCI_1 Swapped SCS[1:0]* Base Address	0x190	page 198
PCI_0 Swapped SCS[3:2]* Base Address	0x114	page 198
PCI_1 Swapped SCS[3:2]* Base Address	0x194	page 199
PCI_0 Swapped CS[3]* & Boot CS* Base Address	0x11c	page 199
PCI_1 Swapped CS[3]* & Boot CS* Base Address	0x19c	page 200

7.14.1 PCI Internal Registers

Table 134: PCI_0 Command, Offset: 0xc00

Bits	Field Name	Function	Initial Value
0	MByteSwap	Master Byte Swap. When set to zero, the GT-96100A PCI master swaps the bytes of the incoming and outgoing PCI data (swap the 8 bytes of a long-word).	Set to the same value sampled at reset into bit[12] of the CPU Interface Configuration register.
3:1	SyncMode ¹	Indicates the ratio between TCik and PCik as follows: x00 - Default mode. When the PCik ranges from DC to 66MHz. This mode works in all cases. NOTE: Use following settings for higher performance. 001 - When PCik is higher than or equal to half the TCik frequency (e.g. when TCik is 100MHz, SyncMode can be set to 001 if the PCI frequency is higher than or equal to 50MHz). 01x - When the two clocks are synchronized and PCik is higher than or equal to half of TCLK frequency (e.g. TCik = 100MHz, PCik = 50MHz). 101 - When PCik is higher than or equal to a third of the TCik frequency but less than half of the TCLK frequency. 11x - When the two clocks are synchronized and PCik is higher than or equal to third of the TCLK frequency but less than half of the TCLK frequency.	0x0

Table 134: PCI_0 Command, Offset: 0xc00 (Continued)

Bits	Field Name	Function	Initial Value
7:4	Reserved	Read only.	0x0
9:8	Reserved	Must be 0.	0x0
10	MWordSwap*	<p>Master Word Swap</p> <p>When set to 1, the GT-96100A PCI master swaps the words of the incoming and outgoing PCI data (swaps the 2 words of a long-word).</p> <p>NOTE: Not supported when using a 64-bit PCI interface.</p>	0x0
11	SWordSwap*	<p>Slave Word Swap</p> <p>When set to 1, the GT-96100A PCI slave swaps the words of the incoming and outgoing PCI data (swap the 2 words of a long-word), if there is an address hit in one of SDRAM or Devices BARs.</p> <p>NOTE: Not supported when using a 64-bit PCI interface.</p>	0x0
12	SSBWordSwap*	<p>Slave Swap BAR Word Swap.</p> <p>When set to 1, the GT-96100A PCI slave swaps the words of the incoming and outgoing PCI data (swap the 2 words of a long-word), if address hit in one of SDRAM or Devices Swap BARs.</p> <p>NOTE: Not supported when using a 64-bit PCI interface.</p>	0x0
15:13	Reserved	Must be 0.	0x0
16	SByteSwap	<p>Slave Byte Swap.</p> <p>When set to zero, the GT-96100A PCI slave swaps the bytes of the incoming and outgoing PCI data (swap the 8 bytes of a long-word).</p>	Set to the same value as sampled at reset into bit[12] of the CPU Interface Configuration register.
31:17	Reserved	Must be 0.	0x0

1. Regardless of the selected syncmode, PClk frequency must be smaller than TClk frequency by at least 1MHz.

Table 135: PCI_1 Command, Offset: 0xc80

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for the PCI_0 Command.	0x0 ¹

1. For bit 16, set to the same value as sampled at reset into bit[12] of the CPU Interface Configuration register.

Table 136: PCI_0 Time Out & ReTry, Offset: 0xc04

Bits	Field Name	Function	Initial Value
7:0	Timeout0	Specifies in PCI clock units the number of clocks that the GT-96100A, as a slave, holds the PCI bus before the generation of retry termination. A value of 0x00 means "never timeout". Used for the first data transfer.	0x0f
15:8	Timeout1	Specifies in PCI clock units the number of clocks that the GT-96100A, as a slave, holds the PCI bus before the generation of disconnect termination. A value of 0x00 means "never timeout". Used for data transfers following the first data.	0x07
23:16	RetryCtr	Specifies the number of retries of the GT-96100A Master. The GT-96100A generates an interrupt when this timer expires. A value of 0x00 means "retry forever". The number in RetryCtr does not include the first access of the transaction.	0x0
31:24	Reserved		0x0

NOTE: See also [Section 7.6.3 "Expansion ROM Functionality"](#) on page 166

Table 137: PCI_1 Time Out & ReTry, Offset: 0xc84

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Time Out and ReTry.	0x070f

Table 138: PCI_0 SCS[1:0] Bank Size, Offset: 0xc08

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	BankSize	<p>Specifies the SCS[1:0]* address mapping in conjunction with the SCS[1:0]* Base Address register.</p> <p>0 - The corresponding bit in the address and in the base address must be equal in order to have a hit.</p> <p>1 -The corresponding bit in the address is a “don’t-care”.</p> <p>For example, bit 12 set to 1 indicates that the SCS[1:0]* size is 8KBytes. The set bits in the Bank Size must be sequential (e.g. 000...001, 000...011, 000...111 are correct values, whereas 000...010 and 000...100 are not).</p>	0x00fff

Table 139: PCI_1 SCS[1:0]* Bank Size, Offset: 0xc88

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 SCS[1:0]* Bank Size.	0x00fff000

Table 140: PCI_0 SCS[3:2]* Bank Size, Offset: 0xc0c

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	BankSize	<p>Specifies the SCS[3:2]* address mapping in conjunction with the SCS[3:2]* Base Address register.</p> <p>0 -The corresponding bit in the address and in the base address must be equal in order to have a hit.</p> <p>1 - The corresponding bit in the address is a “don’t-care”.</p> <p>For example, bit 12 set to 1 indicates that the SCS[3:2]* size is 8KBytes. The set bits in the Bank Size must be sequential (e.g. 000...001, 000...011, 000...111 are correct values, whereas 000...010 and 000...100 are not).</p>	0x00fff

Table 141: PCI_1 SCS[3:2]* Bank Size, Offset: 0xc8c

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 SCS[3:2]* Bank Size.	0x00fff000

Table 142: PCI_0 CS[2:0]* Bank Size, Offset: 0xc10

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only	0x0
31:12	BankSize	<p>Specifies the CS[2:0]* address mapping in conjunction with the CS[2:0]* Base Address register.</p> <p>0 - The corresponding bit in the address and in the base address must be equal in order to have a hit.</p> <p>1 - The corresponding bit in the address is a “don’t-care”.</p> <p>For example, bit 12 set to ‘1’ indicates that the CS[2:0]* size is 8KBytes. The set bits in the Bank Size must be sequential (e.g. 000...001, 000...011, 000...111 are correct values, whereas 000...010 and 000...100 are not).</p>	0x01fff

Table 143: PCI_1 CS[2:0]* Bank Size, Offset: 0xc90

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 CS[2:0]* Bank Size.	0x01fff000

Table 144: PCI_0 CS[3]* and Boot CS* Bank Size, Offset: 0xc14

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	BankSize	<p>Specifies the CS[3]* and Boot CS* address mapping in conjunction with the CS[3]* and Boot CS* Base Address register.</p> <p>0 - The corresponding bit in the address and in the base address must be equal in order to have a hit.</p> <p>1 - The corresponding bit in the address is a “don’t-care”.</p> <p>For example, bit 12 set to ‘1’ indicates that the CS[3]*/ Boot CS* size is 8KBytes. The set bits in the Bank Size must be sequential (e.g. 000...001, 000...011, 000...111 are correct values, whereas 000...010 and 000...100 are not).</p>	0x00fff

Table 145: PCI_1 CS[3]* and Boot CS* Bank Size, Offset: 0xc94

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 CS[3]* and Boot CS* Bank Size.	0x00fff000

Table 146: PCI_0 Internal Registers Space Size, Offset: 0xc20

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	BankSize	<p>Specifies the internal registers address mapping in conjunction with the Internal Registers Base Address register.</p> <p>0 - The corresponding bit in the address and in the base address must be equal in order to have a hit.</p> <p>1 - The corresponding bit in the address is a “don’t-care”.</p> <p>For example, bit 12 set to ‘1’ indicates that the internal registers space size is 8KBytes. The set bits in the BankSize must be sequential (e.g. 000...001, 000...011, 000...111 are correct values, whereas 000...010 and 000...100 are not).</p>	0x000000

Table 147: PCI_1 Internal Registers Space Size, Offset: 0xca0

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Internal Registers Space Size.	0x00000000

Table 148: PCI_0 Base Address Registers' Enable, Offset: 0xc3c

Bits	Field Name	Function	Initial Value
0	SwCS[3]_&_Boot_CSEn	Controls address matching with Swapped CS[3]* & Boot CS* base/size. 0 - Enable 1 - Disable	0x1
1	SwSCS[3:2]En	Controls address matching with Swapped SCS[3:2]* base/size. 0 - Enable 1 - Disable	0x1
2	SwSCS[1:0]En	Controls address matching with Swapped SCS[1:0]* base/size. 0 - Enable 1 - Disable	0x1
3	IntIOEn	Controls address matching with internal registers I/O mapped base/size. 0 - Enable 1 - Disable	0x1
4	IntMeMEn	Controls address matching with internal registers memory mapped base/size. 0 - Enable 1 - Disable	0x0
5	CS[3]*_&_Boot_CSEn	Controls address matching with CS[3]* & Boot CS* base/size. 0 - Enable 1 - Disable	0x0
6	CS[2:0]En	Controls address matching with CS[2:0]* base/size. 0 - Enable 1 - Disable	0x0
7	SCS[3:2]En	Controls address matching with SCS[3:2]* base/size. 0 - Enable 1 - Disable	0x0

Table 148: PCI_0 Base Address Registers' Enable, Offset: 0xc3c (Continued)

Bits	Field Name	Function	Initial Value
8	SCS[1:0]En	Controls address matching with SCS[1:0]* base/size. 0 - Enable 1 - Disable	0x0
31:9	Reserved		0x0

Table 149: PCI_1 Base Address Registers' Enable, Offset: 0xcbc**NOTE:** RESERVED if configured for only PCI_0)

Bits	Field Name	Function	Initial Value
31:0	Various	Same As for PCI_0 Base Address registers' enable.	0x0f

NOTE: The GT-96100A prevents disabling both memory mapped base/size address matching and I/O mapped base/size address matching at the same time (bits 3 and 4 cannot simultaneously be set to 1).

Table 150: PCI_0 Prefetch/Max Burst Size, Offset: 0xc40

Bits	Field Name	Function	Initial Value
0	Dram0MaxBrst	SCS[1:0]* Max Burst Length	0x0
1	Dram1MaxBrst	SCS[3:2]* Max Burst Length	0x0
2	Dev0MaxBrst	CS[2:0]* Max Burst Length	0x0
3	Dev1MaxBrst	CS[3]* & Boot CS* Max Burst Length	0x0
4	RdMemPref	Read Memory Prefetch	0x0
5	RdLnPref	Read Line Prefetch	0x0
6	RdMulPref	Read Multiple Prefetch	0x1
31:7	Reserved		0x0

Table 151: PCI_1 Prefetch/Max Burst Size, Offset: 0xcc0**NOTE:** RESERVED if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Prefetch/Max Burst Size.	0x040

Table 152: PCI_0 SCS[1:0]* Base Address Remap, Offset: 0xc48

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	DRAM0Remap	SCS[1:0]* memory address remap from PCI side. Reserved if this BAR is disabled through PCI_0 Base Address Registers' Enable.	0x0

Table 153: PCI_1 SCS[1:0]* Base Address Remap, Offset: 0xcc8
NOTE: RESERVED if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	DRAM0Remap	SCS[1:0]* memory address remap from PCI side Reserved if this BAR is disabled through PCI_1 Base Address Registers' Enable.	0x0

Table 154: PCI_0 Swapped SCS[1:0]* Base Address Remap, Offset: 0xc58

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	Swapped_DRAM0_Remap	Swapped SCS[1:0]* memory address remap from PCI side Reserved if this BAR is disabled through PCI_0 Base Address Registers' Enable.	0x0

Table 155: PCI_1 Swapped SCS[1:0]* Base Address Remap, Offset: 0xcd8
NOTE: RESERVED if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	Swapped_DRAM0Remap	Swapped SCS[1:0]* memory address remap from PCI side Reserved if this BAR is disabled through PCI_1 Base Address Registers' Enable.	0x0

Table 156: PCI_0 SCS[3:2]* Base Address Remap, Offset: 0xc4c

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	DRAM1Remap	SCS[3:2]* memory address remap from PCI side Reserved if this BAR is disabled through PCI_0 Base Address Registers' Enable.	0x01000

Table 157: PCI_1 SCS[3:2]* Base Address Remap, Offset: 0xcc**NOTE:** RESERVED if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	DRAM1Remap	SCS[3:2]* memory address remap from PCI side Reserved if this BAR is disabled through PCI_1 Base Address Registers' Enable.	0x01000

Table 158: PCI_0 Swapped SCS[3:2]* Base Address Remap, Offset: 0xc5c

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	Swapped_DRAM1Remap	Swapped SCS[3:2]* memory address remap from PCI side Reserved if this BAR is disabled through PCI_0 Base Address Registers' Enable.	0x01000

Table 159: PCI_1 Swapped SCS[3:2]* Base Address Remap, Offset: 0xcdc (RESERVED if configured for only PCI_0)

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	Swapped_DRAM1Remap	Swapped SCS[3:2]* memory address remap from PCI side Reserved if this BAR is disabled through PCI_1 Base Address Registers' Enable.	0x01000

Table 160: PCI_0 CS[2:0]* Base Address Remap, Offset: 0xc50

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	Dev0Remap	CS[2:0]* memory address remap from PCI side Reserved if this BAR is disabled through PCI_0 Base Address Registers' Enable.	0x1c000

Table 161: PCI_1 CS[2:0]* Base Address Remap, Offset: 0xcd0
NOTE: RESERVED if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	Dev0Remap	CS[2:0]* memory address remap from PCI side Reserved if this BAR is disabled through PCI_1 Base Address Registers' Enable.	0x1c000

Table 162: PCI_0 CS[3]* & BootCS* Base Address Remap, Offset: 0xc54

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	Dev1Remap	CS[3]* & BootCS* memory address remap from PCI side Reserved if this BAR is disabled through PCI_0 Base Address Registers' Enable.	0x1f000

Table 163: PCI_1 CS[3]* & BootCS* Base Address Remap, Offset: 0xcd4
NOTE: RESERVED if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	Dev1Remap	CS[3]* & BootCS* memory address remap from PCI side Reserved if this BAR is disabled through PCI_1 Base Address Registers' Enable.	0x1f000

Table 164: PCI_0 Swapped CS[3]* & BootCS* Base Address Remap, Offset: 0xc64

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	Swapped_ Dev1Remap	Swapped CS[3]* & BootCS* memory address remap from PCI side Reserved if this BAR is disabled through PCI_0 Base Address Registers' Enable.	0x1f000

Table 165: PCI_1 Swapped CS[3]* & BootCS* Base Address Remap, Offset: 0xce4**NOTE:** RESERVED if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	Swapped_ Dev1Remap	Swapped CS[3]* & BootCS* memory address remap from PCI side Reserved if this BAR is disabled through PCI_1 Base Address Registers' Enable.	0x1f000

Table 166: PCI_0 Configuration Address, Offset: 0xcf8

Bits	Field Name	Function	Initial Value
1:0	Reserved	Read only.	0x0
7:2	RegNum	Indicates the register number.	0x00
10:8	FunctNum	Indicates the function type.	0x0
15:11	DevNum	Indicates the device number.	0x00
23:16	BusNum	Indicates the bus number.	0x00
30:24	Reserved	Read only.	0x0
31	ConfigEn	When set, an access to the Configuration Data register is translated into a Configuration or Special cycle on the PCI bus.	0x0

Table 167: PCI_0 Configuration Data, Offset: 0xcfc

Bits	Field Name	Function	Initial Value
31:0	Config	The data is transferred to/from the PCI bus when the CPU accesses this register and the ConfigEn bit in the Configuration Address register is set. A CPU access to this register means the GT-96100A performs a Configuration or Special cycle on the PCI bus.	0x000

Table 168: PCI_1 Configuration Address, Offset: 0xcf0
NOTE: RESERVED if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Configuration Address.	0x000

Table 169: PCI_1 Configuration Data, Offset: 0xcf4
NOTE: RESERVED if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
31:0	Config	Same as for PCI_0 Configuration Data.	0x000

Table 170: PCI_0 Interrupt Acknowledge Virtual Register, Offset: 0xc34

Bits	Field Name	Function	Initial Value
31:0	IntAck_0	A CPU read access to this register forces an interrupt acknowledge cycle on PCI_0. Read only.	0x0

Table 171: PCI_1 Interrupt Acknowledge Virtual Register, Offset: 0xc30

Bits	Field Name	Function	Initial Value
31:0	IntAck_1	A CPU read access to this register forces an interrupt acknowledge cycle on PCI_1. Read only.	0x0

7.14.2 PCI Configuration Registers

All PCI Configuration registers are located at their standard offset when accessed from their corresponding PCI bus. For example, if a master on PCI_0 performs a PCI configuration cycle on PCI_0's Status and Command register, the register is located at 0x004. Likewise, if a master on PCI_1 performs a PCI configuration cycle on PCI_1's Status and Command register, the register is located at 0x004.

On the other hand, if a master on PCI_0 performs a PCI configuration cycle on PCI_1's Status and Command register, the register is located at 0x084. Likewise, if a master on PCI_1 performs a PCI configuration cycle on PCI_0's Status and Command register, the register is located at 0x084.

If the CPU masters PCI configuration cycles, PCI_0 configuration registers are located at their standard offsets and PCI_1 configuration registers are located at 0x80 + standard offset.

NOTE: All PCI_1 configuration registers are reserved if the GT-96100A is configured for only PCI_0 operation.

Table 172: PCI_0 Device and Vendor ID, Offset: 0x000 from PCI_0 or CPU; 0x080 from PCI_1

Bits	Field Name	Function	Initial Value
15:0	VenID	Provides the GT-96100A's manufacturer. Read only.	0x11ab
31:16	DevID	Provides the unique GT-96100A PCI device0 ID number. Read Only.	0x9653

Table 173: PCI_1 Device and Vendor ID, Offset: 0x080 from PCI_0 or CPU; 0x000 from PCI_1

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Device and VenID	0x965311ab

Table 174: PCI_0 Status and Command, Offset: 0x004 from PCI_0 or CPU; 0x084 from PCI_1

Bits	Field Name	Function	Initial Value
0	IOEn	Controls the GT-96100A's response to I/O accesses. 0 - Disable 1 - Enable	0x0
1	MEMEn	Controls the GT-96100A's response to memory accesses. 0 - Disable 1 - Enable	0x0
2	MasEn	Controls the GT-96100A's ability to act as a master on the PCI bus. 0 - Disable 1 - Enable	0x0
3	Reserved	Read only.	0x0
4	MemWrInvl	Controls the GT-96100A's ability to generate memory write and invalidate commands on the PCI bus. 0 - Disable 1 - Enable	0x0
5	Reserved	Read only.	0x0
6	PErrEn	Controls the GT-96100A's ability to respond to parity errors on the PCI by asserting the PErr* pin. 0 - Disable 1 - Enable	0x0
7	Reserved	Read only.	0x0
8	SErrEn	Controls the GT-96100A's ability to assert the SErr* pin. 0 - Disable 1 - Enable	0x0
19:9	Reserved	Read only.	0x0
20	CapList	Capability List Support	Sampled at Rst* via SDQM[0]* pin.
21	66MHzEn	66MHz Capable. The GT-96100A PCI interface is capable of running at 66MHz regardless of this bit value.	Sampled at RESET via the DAdr[9] pin.
22	Reserved	Read only.	0x0

Table 174: PCI_0 Status and Command, Offset: 0x004 from PCI_0 or CPU; 0x084 from PCI_1 (Continued)

Bits	Field Name	Function	Initial Value
23	TarFastBB	Read only. Indicates that the GT-96100A is capable of accepting fast back-to-back transactions on the PCI bus.	0x1
24	DataParDet	This bit is set by the GT-96100A when it detects a data parity error during master operation.	0x0
26:25	DevSelTim	These pins indicate that the GT-96100A's DevSel timing (medium), per the PCI standard.	0x1 Read only
27	Reserved	Read only.	0x0
28	TarAbort	This bit is set upon target abort.	0x0
29	MasAbort	This pin is set upon master abort.	0x0
30	SysErr	This pin is set upon system error.	0x0
31	DetParErr	This pin is set upon detection of a parity error in master or slave operations.	0x0

NOTE: For bits 24 and 28 to 31, the user cannot set the bit. The user can only clear the bit by writing 1 to it.

Table 175: PCI_1 Status and Command, Offset: 0x084 from PCI_0 or CPU; 0x004 from PCI_1

Bits	Field Name	Function	Initial Value
19:0	Various	Same as for PCI_0 status and command	
20	CapList	Capability List Support	Sampled at Rst* via SDQM[1]* pin.
31:21	CapList	Same as for PCI_0 status and command	

Table 176: PCI_0 Class Code and Revision ID, Offset: 0x008 from PCI_0 or CPU; 0x088 from PCI_1

Bits	Field Name	Function	Initial Value
7:0	RevID	Indicates the GT-96100A's PCI_0 revision number.	0x03
15:8	Reserved	Read only.	0x0
23:16	SubClass	Indicates the GT-96100A's subclass. 0x00 - Host bridge device 0x80 - Memory device	0x80
31:24	BaseClass	Indicates the GT-96100A's base class. 0x02 - Network controller 0x05 - Memory device	Depends on value sampled at reset on BankSel[0].

Table 177: PCI_1 Class Code and Revision ID, Offset: 0x088 from PCI_0 or CPU; 0x008 from PCI_1

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 class code and revision ID	Depends on value sampled at reset on BankSel[0].

Table 178: PCI_0 BIST, Header Type, Latency Timer, Cache Line, Offset: 0x00c from PCI_0 or CPU; 0x08c from PCI_1

Bits	Field Name	Function	Initial Value
7:0	CacheLine	Specifies the GT-96100A's cache line size.	0x00
15:8	LatTimer	Specifies, in units of PCI bus clocks, the value of the GT-96100A's latency timer.	0x00
23:16	HeadType	Specifies the layout of bytes 10h through 3fh.	0x00
31:24	BIST	Built In Self Test Reserved	0x00

**Table 179: PCI_1 BIST, Header Type, Latency Timer, Cache Line,
Offset: 0x08c from PCI_0 or CPU; 0x00c from PCI_1**

Bits	Field Name	Function	Initial Value
31:0	Various	As in PCI_0 BIST, Header Type, Latency Timer, Cache Line.	0x00

The BIST Field is reserved and is hardwired to 0.

Device and Vendor ID (0x000), Class Code and Revision ID (0x008), and Header Type (0x00e) fields are read only from the PCI bus. These fields can be modified and read via the CPU bus.

NOTE: For more information on these fields, refer to the PCI specification.

Access of PCI masters to SDRAM banks, Devices and internal space is achieved once there is a match between the address presented over the PCI bus and the space defined by the respective Base/Size register pair. The GT-96100A incorporates three Swapped Base Address registers for SCS[1:0]*, SCS[3:2]*, CS[3]*, and BootCS*. When the address matches a Swapped Base Address register x, the data transferred will undergo the opposite to what is indicated by the ByteSwap bit (bit[0] of 0xc00). e.g. using this mechanism, one could write data directly to SDRAM and read it byte-swapped without CPU processing.

NOTE: The address must not match its respective non-Swap Base Address register.

The Size registers cannot define a zero size space. In order to enable the system designer to use addresses which are within a certain space without having the GT-96100A respond to these addresses, a Base Address Enable register is incorporated. A disabled space will not trigger a device response if the address falls within the space defined by its Base/Size register pair.

Table 180: PCI_0 SCS[1:0]* Base Address, Offset: 0x010 from PCI_0 or CPU; 0x090 from PCI_1

Bits	Field Name	Function	Initial Value
0	MemSpace	Memory Space Indicator Read only.	0x0
2:1	Type	Type Read only.	0x0
3	Prefetch	Prefetch	0x1
11:4	Reserved	Read only.	0x0
31:12	Base	Defines the address assignment of SCS[1:0]*, see Table 138 . Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x00000

Table 181: PCI_1 SCS[1:0]* Base Address, Offset: 0x090 from PCI_0 or CPU; 0x010 from PCI_1

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 SCS[1:0]* Base Address	0x08

Table 182: PCI_0 SCS[3:2]* Base Address, Offset: 0x014 from PCI_0 or CPU; 0x094 from PCI_1

Bits	Field Name	Function	Initial Value
0	MemSpace	Memory Space Indicator Read only.	0x0
2:1	Type	Type Read only.	0x0
3	Prefetch	Prefetch	0x1
11:4	Reserved	Read only.	0x0
31:12	Base	Defines the address assignment of SCS[3:2]*, see Table 140 . Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x01000

Table 183: PCI_1 SCS[3:2]* Base Address, Offset: 0x094 from PCI_0 or CPU; 0x014 from PCI_1

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 SCS[3:2]* Base Address.	0x01000008

Table 184: PCI_0 CS[2:0]* Base Address, Offset: 0x018 from PCI_0 or CPU; 0x098 from PCI_1

Bits	Field Name	Function	Initial Value
0	MemSpace	Memory Space Indicator Read only.	0x0
2:1	Type	Type Read only.	0x0
3	Prefetch	Prefetch	0x0
11:4	Reserved	Read only.	0x0
31:12	Base	Defines the address assignment of CS[2:0]*, see Table 142 . Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x1c000

Table 185: PCI_1 CS[2:0]* Base Address, Offset: 0x098 from PCI_0 or CPU; 0x018 from PCI_1

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 CS[2:0]* Base Address	0x1c000000

Table 186: PCI_0 CS[3]* and Boot CS* Base Address, Offset: 0x01c from PCI_0 or CPU; 0x09c from PCI_1

Bits	Field Name	Function	Initial Value
0	MemSpace	Memory Space Indicator Read only.	0x0
2:1	Type	Type Read only.	0x0
3	Prefetch	Prefetch	0x0
11:4	Reserved	Read only.	0x0
31:12	Base	Defines the address assignment of CS[3]* and Boot CS*, see Table 144 . Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x1f000

Table 187: PCI_1 CS[3]* and Boot CS* Base Address, Offset: 0x09c from PCI_0 or CPU; 0x01c from PCI_1

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 CS[3]* and Boot CS* Base Address.	0x1f000000

**Table 188: PCI_0 Internal Registers Memory Mapped Base Address,
Offset: 0x020 from PCI_0 or CPU; 0x0a0 from PCI_1**

Bits	Field Name	Function	Initial Value
0	MemSpace	Memory Space Indicator Read only.	0x0
2:1	Type	Type Read only.	0x0
3	Prefetch	Prefetch Read only.	0x0
11:4	Reserved	Read only.	0x0
31:12	MemMapBase	Defines the address assignment of the GT-96100A's internal registers. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x14000

**Table 189: PCI_1 Internal Registers Memory Mapped Base Address,
Offset: 0x0a0 from PCI_0 or CPU; 0x020 from PCI_1**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Internal register's Memory Mapped Base Address.	0x14000000

**Table 190: PCI_0 Internal Registers I/O Mapped Base Address,
Offset: 0x024 from PCI_0 or CPU; 0x0a4 from PCI_1**

Bits	Field Name	Function	Initial Value
0	MemSpace	IO Space Indicator	0x1
11:1	Reserved	Read only.	0x0
31:12	IOMapBase	Defines the address assignment of the GT-96100A's internal registers. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x14000

Table 191: PCI_1 Internal Registers I/O Mapped Base Address, Offset: 0x0a4 from PCI_0 or CPU; 0x024 from PCI_1

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Internal register's I/O Mapped Base Address.	0x14000001

Table 192: PCI_0 Subsystem Device and Vendor ID, Offset: 0x02c from PCI_0 or CPU; 0x0ac from PCI_1

Bits	Field Name	Function	Initial Value
15:0	VenID	Provides the subsystem vendor Identification number.	0x0
31:16	DevID	Provides the subsystem device identification number.	0x0

Table 193: PCI_1 Subsystem Device and Vendor ID, Offset: 0x0ac from PCI_0 or CPU; 0x02c from PCI_1

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 subsystem device and vendor ID.	0x0

Table 194: Expansion ROM Base Address Register, Offset: 0x030 from PCI_0 or CPU; 0x0b0 from PCI_1

Bits	Field Name	Function	Initial Value
0	ERDecEn	Expansion ROM Decode Enable 0 - Disable 1 - Enable	0x0
11:1	Reserved		0x0
31:12	ERBase	Defines the address of the expansion ROM memory space region assigned to the GT-96100A. This is where the expansion ROM code appears in system memory when bit 0 of this register contains a value of 1 and bit 1 of this device's Command Register contains a value of 1. This register is reserved in case Expansion ROM was not enabled at Reset (DAdr[5] sampled 0).	0x1f000

**Table 195: PCI_0 Capability List Pointer Register,
Offset: 0x034 from PCI_0 or CPU, 0xb4 from PCI_1¹**

Bits	Field Name	Function	Initial Value
7:0	CapPtr	Capability List Pointer. Read only.	0x40
31:8	Reserved		0x0

1. Reserved if Power management is disabled

**Table 196: PCI_1 Capability List Pointer Register, Offset: 0x0b4¹ from
PCI_0 or CPU, 0x34 from PCI_1**

Bits	Field Name	Function	Initial Value
7:0	CapPtr	Capability List Pointer. Read only.	0x40
31:8	Reserved		0x0

**Table 197: PCI_0 Interrupt Pin and Line, Offset: 0x03c from
PCI_0 or CPU; 0x0bc from PCI_1**

Bits	Field Name	Function	Initial Value
7:0	IntLine	Provides interrupt line routing information.	0x0
15:8	IntPin	Indicates which interrupt pin is used by the GT-96100A. NOTE: The GT-96100A uses INTA.	0x1
31:16	Reserved	Read only.	0x0

**Table 198: PCI_1 Interrupt Pin and Line, Offset: 0x0bc from
PCI_0 or CPU; 0x03c from PCI_1**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 interrupt pin and line.	0x00000100

**Table 199: PCI_0 PMC Register, Offset: 0x040 from
PCI_0 or CPU, 0x0c0 from PCI_1¹**

Bits	Field Name	Function	Initial Value
7:0	CapID	Capability ID Read only from PCI.	0x1
15:8	NullPTR	Null Pointer Indicates that PMC is the last item in the capability list. Read Only from PCI	0x0
18:16	PMCVer	PCI Power Management Spec Revision Read only from PCI.	0x1
19	PMEClk	PME Clock Read only from PCI.	0x1
20	Reserved	Read only from PCI.	0x0
21	DSI	Device Specific Initialization Read only from PCI.	0x0
24:22	AuxCur	Auxulary Current Requirements Read only from PCI.	0x0
25	D1Sup	D1 Power Management State Support Read only from PCI	0x0
26	D2Sup	D2 Power Management State Support Read only from PCI.	0x0
31:27	PMESup	PME* Signal Support Read Only from PCI.	0x0

1. Reserved if Power Management is disabled

**Table 200: PCI_1 PMC Register, Offset: 0x0c0 from
PCI_0 or CPU, 0x040 from PCI_1¹**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 PMC register.	0x00090001

**Table 201: PCI_0 PMCSR Register, Offset: 0x044 from
PCI_0 or CPU, 0x0c4 from PCI_1¹**

Bits	Field Name	Function	Initial Value
1:0	PState	Power State	0x0
7:2	Reserved	Read only from PCI.	0x0
8	PME_EN	PME* Pin Assertion Enable Read only from PCI.	0x0
12:9	DSel	Data Select Read only from PCI.	0x0
14:13	DScale	Data Scale Read only from PCI.	0x0
15	PME_Stat	PME* Pin Status Read only from PCI.	0x0
21:16	Reserved	Read only from PCI.	0x0
22	B2_B3	B2/B3 Select Read only from PCI.	0x0
23	BPCC_En	Bus Power/Clock Control Enable Read only from PCI.	0x0
31:24	Data	State Data Read only from PCI.	0x0

**Table 202: PCI_1 PMCSR Register, Offset: 0x0c4 from
PCI_0 or CPU, 0x044 from PCI_1**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 PMCSR register.	0x0

7.14.3 Function 1 Configuration Registers

The GT-96100A acts as two function device. It's PCI slave interface responds to configuration transactions to function number 0 or 1. Most of function 1 configuration registers are aliased to function 0 registers or reserved, except of the 3 swap BARs. To access any of the Swap Base Address registers, a configuration access addressed to function 1 must be used with the appropriate offset. If an offset other than 0x010, 0x014, or 0x01c is accessed when specifying function 1, the transaction accesses the corresponding offset register in function 0. Configuration transactions to any other function number are ignored.

The GT-96100A acts as two function device regardless of multi-function bit (bit[7] in Header Type). However, this bit value after reset is 0. In a PC environment, in order for a BIOS to recognize the GT-96100A as a multi-function device (if swap BARs are required in the system), set this bit and enable the swap BARs (Base Address Enable register) before BIOS starts. This can be done by programming by CPU software or by using the GT-96100A's Auto-Load option.

Table 203: Function1 PCI_0 Swapped SCS[1:0]* Base Address, Offset: 0x110 from PCI_0 or CPU; 0x190 from PCI_1

Bits	Field Name	Function	Initial Value
0	MemSpace	Memory Space Indicator Read only.	0x0
2:1	Type	Type Read only.	0x0
3	Prefetch	Prefetch	0x1
11:4	Reserved	Read only.	0x0
31:12	Base	Defines the address assignment of Swapped SCS[1:0]*. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x0

Table 204: Function 1 PCI_1 Swapped SCS[1:0]* Base Address, Offset: 0x190 from PCI_0 or CPU; 0x110 from PCI_1

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Swapped SCS[1:0]* Base Address	0x08

Table 205: Function 1 PCI_0 Swapped SCS[3:2]* Base Address, Offset: 0x114 from PCI_0 or CPU; 0x194 from PCI_1

Bits	Field Name	Function	Initial Value
0	MemSpace	Memory Space Indicator Read only.	0x0

**Table 205: Function 1 PCI_0 Swapped SCS[3:2]* Base Address,
Offset: 0x114 from PCI_0 or CPU; 0x194 from PCI_1 (Continued)**

Bits	Field Name	Function	Initial Value
2:1	Type	Type Read only.	0x0
3	Prefetch	Prefetch	0x1
11:4	Reserved	Read only.	0x0
31:12	Base	Defines the address assignment of Swapped SCS[3:2]*. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x01000

**Table 206: Function 1 PCI_1 Swapped SCS[3:2]* Base Address,
Offset: 0x194 from PCI_0 or CPU; 0x114 from PCI_1**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Swapped SCS[3:2]* Base Address.	0x01000008

**Table 207: Function 1 PCI_0 Swapped CS[3]* & Boot CS* Base Address,
Offset: 0x11c from PCI_0 or CPU; 0x19c from PCI_1**

Bits	Field Name	Function	Initial Value
0	MemSpace	Memory Space Indicator Read only.	0x0
2:1	Type	Type Read only.	0x0
3	Prefetch	Prefetch	0x0
11:4	Reserved	Read only.	0x0
31:12	Base	Defines the address assignment of CS[3]* and Boot CS*, see Table 144 . Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.)	0x1f000

**Table 208: Function 1 PCI_1 Swapped CS[3]* & Boot CS* Base Address,
Offset: 0x19c from PCI_0 or CPU; 0x11c from PCI_1**

Bits	Field Name	Function	Initial Value
31:0	Various	As in PCI_0 Swapped CS[3]* and Boot CS* Base Address.	0x1f000000

8. INTELLIGENT I/O (I₂O) STANDARD SUPPORT

The GT-96100A includes hardware support for the Intelligent I/O (I₂O) Standard.

This support includes all of the registers required for implementing the I₂O Messaging Unit as defined in the I₂O Specification. This Messaging Unit (MU) is compatible with that found in Intel's i960Rx processors. However, the combination of MIPS processors and the GT-96100A delivers as much as 20 times the integer performance of the i960RP.

The I₂O hardware support found in the GT-96100A also provides designers of non-I₂O embedded systems with important benefits. For example, the circular queue support in the MU provides a simple, powerful mechanism for passing queued messages between intelligent agents on a PCI bus. Even the simple message and doorbell registers improve the efficiency of communication between agents on PCI.

NOTE: Even if you have no intention of using the entire I₂O “stack”, Galileo Technology recommends reading this entire section to learn about improving the application of the hardware to the design.

8.1 Overview

The best source for an overview description of I₂O support is in the I₂O specification documentation.

The I₂O specification defines a standard mechanism for passing messages between a host processor (a Pentium™, for example) and intelligent I/O processors (a networking card based on the GT-96100A and a MIPS processor, for example.) This same message passing mechanism is used to pass messages between peers in a system.

I₂O is defined to be bus independent but, in the real world, it runs over the PCI. The basic process is quite simple:

1. A master wishing to post a message to a device, fetches a pointer from the device from a defined register in the target devices PCI memory space (one of the I₂O registers).
2. The master assembles the message in the targets memory space and then posts the fetched pointer into another register in the target device. Posting the pointer generates an interrupt to the target's processor.

The I₂O specification documentation also defines a simpler mechanism for implementing message passing through doorbell and message registers. The GT-96100A also includes this support.

8.2 I₂O Registers

From the PCI side, the registers used to implement I₂O support resides in the first 128 bytes of the memory region defined by RAS[1:0] Base Address register in PCI function 0 of PCI interface 0¹. The I₂O registers are accessible from the CPU side through an offset from the CPU Internal Space Base register.

NOTE: Index registers are not supported in GT-96100A.

1. There is no I₂O support on the PCI_1 interface.

Table 209: I₂O PCI and CPU Offsets

I₂O Register	PCI Side¹	CPU Side²
Inbound Message Register 0	0x10	0x1c10
Inbound Message Register 1	0x14	0x1c14
Outbound Message Register 0	0x18	0x1c18
Outbound Message Register 1	0x1c	0x1c1c
Inbound Doorbell Register	0x20	0x1c20
Inbound Interrupt Cause Register	0x24	0x1c24
Inbound Interrupt Mask Register	0x28	0x1c28
Outbound Doorbell Register	0x2c	0x1c2c
Outbound Interrupt Cause Register	0x30	0x1c30
Outbound Interrupt Mask Register	0x34	0x1c34
Inbound Queue Port Virtual Register	0x40	0x1c40
Outbound Queue Port Virtual Register	0x44	0x1c44
Queue Control Register	0x50	0x1c50
Queue Base Address Register	0x54	0x1c54
Inbound Free Head Pointer Register	0x60	0x1c60
Inbound Free Tail Pointer Register	0x64	0x1c64
Inbound Post Head Pointer Register	0x68	0x1c68
Inbound Post Tail Pointer Register	0x6c	0x1c6c
Outbound Free Head Pointer Register	0x70	0x1c70
Outbound Free Tail Pointer Register	0x74	0x1c74
Outbound Post Head Pointer Register	0x78	0x1c78
Outbound Post Tail Pointer Register	0x7c	0x1c7c
RESERVED	0x80 to 4K	-
SDRAM Ras[1:0]	>4K	-

1. Offset from BAR0 in PCI Memory Space

2. Offset from CPU Internal Space Base register (location in MIPS CPU memory space)

8.3 Enabling I₂O Support

The I₂O registers are only visible from the PCI side when I₂O support is enabled via the pin strapping option shown in the RESET Configuration section.

When I₂O support is disabled, the locations from BAR0+0x0 to BAR0+0x7F appear as SDRAM.

8.4 Register Map Compatibility with the i960Rx Family

The GT-96100A's register map is compatible with the I₂O specification. However, some of the registers found in Intel's i960Rx processors are not implemented. This information is shown in [Table 210](#).

Table 210: Register Differences Between the GT-96100A and i960Rx

Register Name	GT-96100A	i960Rx	Comment
APIC Register Select	Not implemented	Implemented	No APIC support required for the GT-96100A, not a part of the I ₂ O spec.
APIC Window Select	Not implemented	Implemented	No APIC support required for the GT-96100A, not a part of the I ₂ O spec.
Index Registers	Not implemented	Implemented	Index registers are not used for I ₂ O message passing so this is not a compatibility issue. Index registers are implemented as normal SDRAM in the GT-96100A.

8.5 Message Registers

The GT-96100A uses the message registers to send and receive short messages over the PCI bus, without transferring data into local memory. When written, message registers may cause an interrupt to be generated either to the MIPS CPU or to the PCI bus. There are two types of message registers:

- Inbound messages are sent by an external PCI bus agent and received by the GT-96100A
- Outbound messages are sent by the GT-96100A's local CPU and received by an external PCI agent

The interrupt status for inbound messages is recorded in the Inbound Interrupt Cause register.

The interrupt status for outbound messages is recorded in the Outbound Interrupt Cause register.

8.5.1 Inbound Messages

There are two Inbound Message registers (IMRs).

When an IMR is written from the PCI side, a maskable interrupt request is generated in the Inbound Interrupt Status register (IISR). If this request is unmasked, an interrupt request is issued to the MIPS CPU. The interrupt is cleared when the CPU writes a value of 1 to the Inbound Message Interrupt bit in the IISR. The interrupt may be masked through the mask bits in the Inbound Interrupt Mask register.

8.5.2 Outbound Messages

There are two Outbound Message registers (OMRs).

When an OMR is written from the CPU side, a maskable interrupt request is generated in the Outbound Interrupt Status register (OISR). If this request is unmasked, an interrupt request is issued to the PCI_0 unit. The interrupt is cleared when an external PCI agent writes a value of 1 to the Outbound Message Interrupt bit in the OISR. The interrupt may be masked through the mask bits in the Outbound Interrupt Mask register.

8.6 Doorbell Registers

The GT-96100A uses the doorbell registers to request interrupts on the PCI and CPU buses. There are two types of doorbell registers:

- Inbound doorbells are set by an external PCI bus agent to request interrupt service from the MIPS CPU
- Outbound doorbells are set by the GT-96100A's local CPU and to request an interrupt on PCI

8.6.1 Outbound Doorbells

The local MIPS processor generates an interrupt request to the PCI bus by setting bits in the Outbound Doorbell register (ODR). The interrupt may be masked in the OIMR register, however masking the interrupt does not prevent the corresponding bit from being set in the ODR.

External PCI agents clear the interrupt by setting bits in the ODR to 1 through a write.

8.6.2 Inbound Doorbells

The PCI bus can generate an interrupt request to the local MIPS processor by setting bits in the Inbound Doorbell register (IDR). The interrupt may be masked in the IIMR register, however masking the interrupt does not prevent the corresponding bit from being set in the IDR.

The CPU clears the interrupt by setting bits in the IDR (writing a 1).

8.7 Circular Queues

The circular queues form the heart of the I₂O message passing mechanism, and are also the most powerful part of the MU built into the GT-96100A. There are four circular queues in the MU: two inbound and two outbound.

8.7.1 Inbound Message Queues

There are two inbound message queues:

- The Inbound Posts queue is for messages from other PCI agents for the MIPS CPU to process
- The Inbound Free queue is for messages from MIPS CPU to PCI agent in response to an incoming message.

The two inbound queues allow external PCI agents to post inbound messages for the local MIPS CPU in one queue and receive free messages (no longer in use) returning from the MIPS CPU. The process is as follows:

1. An external PCI agent posts an inbound message.
2. The MIPS CPU receives and processes the message.
3. When the processing is complete, the MIPS CPU places the message back into the inbound free queue so that it may be reused.

8.7.2 Outbound message queues

There are two outbound message queues:

- The Outbound Post queue is for messages from the MIPS CPU to other PCI agents to process.
- The Outbound Free queue is for messages are from PCI agent to the MIPS CPU in response to an outgoing message.

The two outbound queues allow the MIPS CPU to post outbound messages for external PCI agents in one queue and receive free messages (no longer in use) returning from external PCI agents. The process is as follows:

1. The MIPS CPU posts an outbound message.
2. The external PCI agent receives and processes the message.
3. When the processing is complete, the external PCI agent places the message back into the outbound free queue so that it may be reused.

8.7.3 Memory for Circular Queues

Data storage for the circular queues must be allocated in local SDRAM.

The base address for the queues is set in the Queue Base Address register (QBAR). Each queue entry is a 32-bit data value. The circular queue sizes range from 4K entries (16Kbytes) to 64K entries (256Kbytes) yielding a total local memory allotment of 64Kbytes to 1Mbyte. All four queues must be the same size and be contiguous in the memory space. Queue size is set in the Queue Control register.

The starting address of each queue is based on the QBAR address and the size of the queues as shown in the table below.

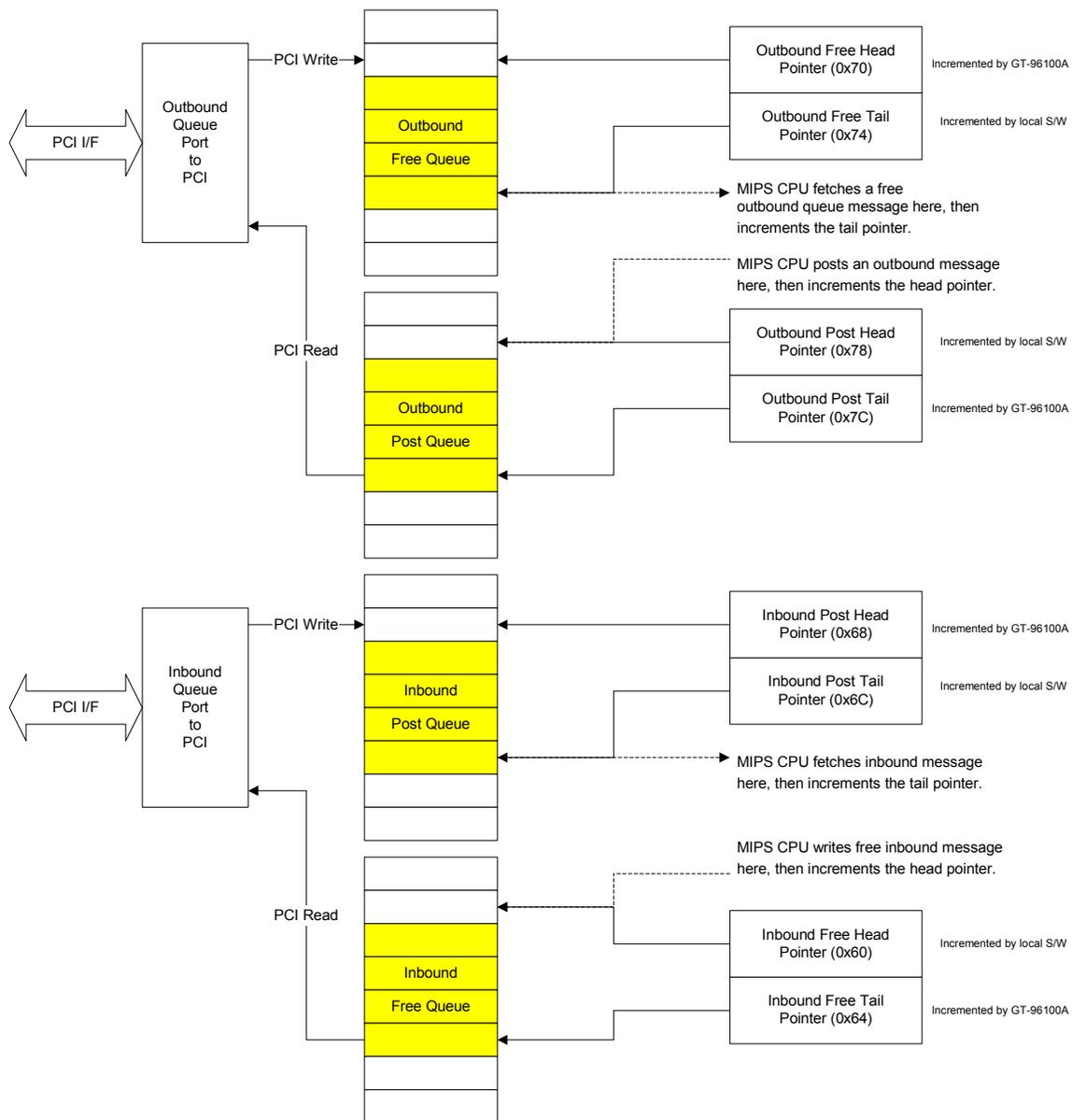
Table 211: Circular Queue Starting Addresses

Queue	Starting Address
Inbound Free	QBAR
Inbound Post	QBAR + Queue Size
Outbound Post	QBAR + 2*Queue Size
Outbound Free	QBAR + 3*Queue Size

Each queue has a head pointer and a tail pointer kept in the GT-96100A's internal registers. These pointers are offsets from the QBAR. Writes to a queue occur at the head of the queue; reads occur from the tail. The head and tail pointers are incremented by either the CPU software or messaging unit hardware. The pointers wrap around to the first address of a queue when they reach queue size.

PCI read/write from queue is always single-word. An attempt to burst from an I₂O queue will result in disconnect after 1st data transfer.

Figure 31: I₂O Circular Queue Operation



8.7.4 Inbound/Outbound Queue Port Register Function

The circular queues are accessed by external PCI agents through the Inbound and Outbound Queue Port virtual registers in the I₂O/PCI address space, decoded by BAR0.

NOTE: The Inbound and Outbound Queue Port virtual registers are not read/write physical registers within the GT-96100A. These virtual registers are reading and writing pointers into the circular queues (located in SDRAM) that are controlled by the GT-96100A. Refer to [Figure 31](#).

8.7.4.1 Inbound Queue Port Reads and Writes

When Inbound Queue Port (IQP) is written from PCI, the written data is placed on the Inbound Post Queue. The IQP is posting the message to the local CPU. An interrupt is generated to the MIPS CPU when the Inbound Post Queue is written to alert the CPU that a message needs processing. When this register is read from the PCI side, it is returning a free message from the tail of Inbound Free Queue.

8.7.4.2 The Outbound Queue Port

The Outbound Queue Port (OQP) returns data from the tail of the Outbound Post Queue when read from the PCI side. The OQP is returning the next message requiring service by the external PCI agent. When this register is written from PCI, the data for the write is placed on the Outbound Free Queue. This, in turn, returns a free message for reuse by the local MIPS CPU.

8.7.5 Inbound Post Queue

The Inbound Post Queue holds posted messages from external PCI agents to the MIPS CPU. The MIPS CPU fetches the next message process from the queue tail. External agents post new messages to the queue head. The tail pointer is maintained in software by the MIPS CPU. The head pointer is maintained automatically by the GT-96100A upon posting of a new inbound message.

PCI writes to the IQP are passed to local memory location at QBAR + Inbound Post Head Pointer. After this write completes, the GT-96100A increments the Inbound Post Head Pointer by 4 bytes (1 word); it now points to the next available slot for a new inbound message. An interrupt is also sent to the MIPS CPU to indicate the presence of a new message pointer.

From the time the PCI write ends till the data is actually written to DRAM, any new write to Inbound port will result in RETRY. If queue is full, a new PCI write to the queue will result in RETRY.

Inbound messages are fetched by the MIPS CPU by reading the contents of the address pointed to by the Inbound Post Tail Pointer. It is the CPU's responsibility to increment the tail pointer to point to the next unread message.

8.7.6 Inbound Free Queue

The Inbound Free Queue holds available inbound free messages for external PCI agents to use. The MIPS CPU places free messages at the queue head; external agents fetch free messages from the queue tail. The head pointer is maintained in software by the MIPS CPU. The tail pointer is maintained automatically by the GT-96100A upon a PCI agent fetching a new inbound free message (except when there is an error, see below.)

PCI reads from the Inbound Queue Port return data to the local memory location at QBAR + Inbound Free Tail Pointer according to the following conditions:

- If the Inbound Free Queue is not empty (as indicated by Head Pointer not equal to Tail Pointer), then the data pointed to by QBAR + Inbound Free Tail Pointer is returned.
- If the queue is empty (Head Pointer equals Tail Pointer), the value 0xFFFF.FFFF is returned. This indicates there are no Inbound Message slots available (an error condition.)

The MIPS processor places free message buffers in the Inbound Free Queue by writing the pointer to the location pointed to by the head pointer. It is the processor's responsibility to then increment the head pointer.

8.7.7 Outbound Post Queue

The Outbound Post Queue holds outbound posted messages from the MIPS CPU to external PCI agents. The MIPS CPU places outbound messages at the queue head; external agents fetch the posted messages from the queue tail. The Outbound Post Tail Pointer is automatically incremented by the GT-96100A. The head pointer must be incremented by the local MIPS CPU.

PCI reads from the Outbound Queue Port return the data pointed to by QBAR + Outbound Post Tail Pointer (the next posted message in the Outbound Queue.) The following conditions apply:

- If the Outbound Post Queue is not empty (the head and tail pointers are not equal), the data is returned as usual and the GT-96100A increments the Outbound Post Tail Pointer
- If the Outbound Post Queue is empty (the head and tail pointers are equal), the value 0xFFFF.FFFF is returned

As long as the Outbound Post Head and Tail pointers are not equal, a PCI interrupt is requested. This is done to indicate the need to have the external PCI agent read the Outbound Post Queue. When the head and tail pointers are equal, no PCI interrupt is generated since no service is required on the part of the external PCI agent (or PCI system host in the case of a PC server.) In either case, the interrupt can be masked in the OIMR register.

The MIPS CPU places outbound messages in the Outbound Post Queue by writing to the local memory location pointed to by the Outbound Post Head Pointer. After writing this pointer, it is the CPU's responsibility to increment the head pointer.

8.7.8 Outbound Free Queue

The Outbound Free Queue holds available outbound message buffers for the local MIPS processor to use. External PCI agents place free message at the queue head; the MIPS CPU fetches free message pointers from the queue tail. The tail pointer is maintained in software by the MIPS CPU. The head pointer is maintained automatically by the GT-96100A upon a PCI agent posting a new ("returned") outbound free message.

PCI writes to the Outbound Queue Port result in the data being written to the local memory location at QBAR + Outbound Free Head Pointer. After the write completes, the GT-96100A increments the head pointer.

From the time the PCI write ends till the data is actually written to DRAM, any new write to Outbound port will result in RETRY. If the head pointer and tail pointer become equal (an indication that the queue is full), an interrupt is sent to the MIPS CPU. If queue is full, a new PCI write to the queue will result in RETRY.

The MIPS processor obtains free outbound message buffers from the Outbound Free Queue by reading data from the location pointed to by the tail pointer. It is the processor's responsibility to then increment the tail pointer.

Table 212: I₂O Circular Queue Functional Summary

Queue Name	PCI Port	Generate PCI Interrupt?	Generate CPU Interrupt?	Head Pointer maintained by...	Tail Pointer maintained by...
Inbound Post	Inbound Queue Port	No	Yes, when queue is written	GT-96100A	MIPS CPU
Inbound Free		No	No	MIPS CPU	GT-96100A
Outbound Post	Outbound Queue Port	Yes, when queue is not empty	No	MIPS CPU	GT-96100A
Outbound Free		No	Yes, when queue is full	GT-96100A	MIPS CPU

8.8 I₂O Support Registers

If I₂O is enabled (i.e., DAdr[8] was sampled '0' at reset) its related registers are accessible from the CPU side and the PCI_0 side. To access registers from the PCI_0 side, the address must be in the first 4KBytes of PCI_0 SCS[1:0]* Base register space. To access from CPU side, the address must be in the 4KBytes of CPU Internal Space Base register space.

PCI_1 has no access to I₂O registers. If PCI_1 address hits first 4KBytes of PCI_1 SCS[1:0]* BAR space, it accesses SDRAM.

If accessed from the PCI_0 side, address offset is with respect to the PCI_0 SCS[1:0]* Base Address register contents. If accessed from the CPU side, address offset is with respect to the CPU Internal Space Base register + 0x1c00.

NOTE: I₂O registers can be accessed from the CPU and PCI_0 sides (unless stated otherwise). If accessed from the PCI_0 side, address offset is with respect to the PCI_0 SCS[1:0]* Base Address register contents. If accessed from CPU side, the address offset is with respect to the CPU Internal Space Base Register + 0x1c00.

Table 213: I₂O Support Register Map

Description	Offset	Page Number
Inbound Message Register 0	0x10	page 211
Inbound Message Register 1	0x14	page 211
Outbound Message Register 0	0x18	page 211
Outbound Message Register 1	0x1c	page 211
Inbound Doorbell Register	0x20	page 212
Inbound Interrupt Cause Register	0x24	page 212
Inbound Interrupt Mask Register	0x28	page 213
Outbound Doorbell Register	0x2c	page 213
Outbound Interrupt Cause Register	0x30	page 214
Outbound Interrupt Mask Register	0x34	page 214
Inbound Queue Port Virtual Register	0x40	page 215
Outbound Queue Port Virtual Register	0x44	page 215
Queue Control Register	0x50	page 215
Queue Base Address Register	0x54	page 216
Inbound Free Head Pointer Register	0x60	page 216
Inbound Free Tail Pointer Register	0x64	page 216
Inbound Post Head Pointer Register	0x68	page 216
Inbound Post Tail Pointer Register	0x6c	page 217
Outbound Free Head Pointer Register	0x70	page 217
Outbound Free Tail Pointer Register	0x74	page 217
Outbound Post Head Pointer Register	0x78	page 218
Outbound Post Tail Pointer Register	0x7c	page 218

Table 214: Inbound Message Register 0, Offset: 0x10

Bits	Field Name	Function	Initial Value
31:0	InMsg0	Inbound Message Register_0 Read only from the CPU. When written, a bit is set in the Inbound Interrupt Cause register and an interrupt is generated to the CPU (if unmasked). First register of two intended for messages from PCI to CPU.	0x0

Table 215: Inbound Message Register 1, Offset: 0x14

Bits	Field Name	Function	Initial Value
31:0	InMsg1	Inbound Message Register_1 Read only from the CPU. When written, a bit is set in the Inbound Interrupt Cause Register and an interrupt is generated to the CPU (if unmasked). Second register of two intended for messages from PCI to CPU.	0x0

Table 216: Outbound Message Register 0, Offset: 0x18

Bits	Field Name	Function	Initial Value
31:0	OutMsg0	Outbound Message Register_0 Read only from the PCI. When written, a bit is set in the Outbound Interrupt Cause register and an interrupt is generated to the PCI (if unmasked). First register of two intended for messages from CPU to PCI.	0x0

Table 217: Outbound Message Register 1, Offset: 0x1c

Bits	Field Name	Function	Initial Value
31:0	OutMsg1	Outbound Message Register_1 Read only from the PCI. When written, a bit is set in the Outbound Interrupt Cause register and an interrupt is generated to the PCI (if unmasked). Second register of two intended for messages from CPU to PCI.	0x0

Table 218: Inbound Doorbell Register, Offset: 0x20

Bits	Field Name	Function	Initial Value
31:0	InDoor	Inbound Doorbell Register If not masked by Inbound Interrupt Mask register, setting a bit in this register to 1 by the PCI causes a CPU interrupt. Writing 1 to the bit by the CPU clears the bit (and de-assert the interrupt).	0x0

Table 219: Inbound Interrupt Cause Register, Offset: 0x24

Bits	Field Name	Function	Initial Value
0	InMsg0Int	Inbound Message_0 Interrupt This bit is set when Inbound Message 0 register is written. The CPU writes it with 1 to clear it. ¹	0x0
1	InMsg1Int	Inbound Message_1 Interrupt This bit is set when Inbound Message_1 register is written. CPU writes it with 1 to clear it. ²	0x0
2	InDoorInt	Inbound Doorbell Interrupt This bit is set when at least one bit of Inbound Doorbell register is set. This bit is read only.	0x0
3	Reserved		0x0
4	InPQInt	Inbound Post Queue Interrupt This bit is set when the Inbound Post Queue gets written. The CPU writes it with 1 to clear it.	0x0
5	OutFQOvr	Outbound Free Queue Overflow Interrupt This bit is set when the Outbound Free Queue is full. The CPU writes it with 1 to clear it.	0x0
31:6	Reserved		0x0

1. Unlike Intel's i960RP, bits [8:6] and bit 3 are reserved since the GT-96100A does not support Index, APIC, or NMI mechanisms.

2. An interrupt to CPU is generated if any of the bits 0,1,2,4, or 5 is set to 1 given that its corresponding entry in the Inbound Interrupt Mask Register is NOT set.

Table 220: Inbound Interrupt Mask Register, Offset: 0x28

Bits	Field Name	Function	Initial Value
0	InMsg0IntMsk	Inbound Message_0 Interrupt Mask	0x0
1	InMsg1IntMsk	Inbound Message_1 Interrupt Mask	0x0
2	InDoorIntMsk	Inbound Doorbell Interrupt Mask	0x0
3	Reserved		0x0
4	InPQIntMsk	Inbound Post Queue Interrupt Mask	0x0
5	OutFQOvrMsk	Outbound Free Queue Overflow Interrupt Mask When set, no interrupt to CPU is generated for set OutFQOvr bit in Inbound Interrupt Cause Register.	0x0
31:6	Reserved		0x0

Table 221: Outbound Doorbell Register, Offset: 0x2c

Bits	Field Name	Function	Initial Value
31:0	OutDoor	Outbound Doorbell Register Setting a bit in this register to 1 by the CPU causes a PCI interrupt if not masked by the Outbound Interrupt Mask register. ¹ Writing 1 to the bit by the PCI clears the bit (and de-assert the interrupt).	0x0

1. Unlike Intel's i960RP, there are no reserved bits for PCI interrupts INTA#, INTB#, INTC#, INTD#.

Table 222: Outbound Interrupt Cause Register, Offset: 0x30

Bits	Field Name	Function	Initial Value
0	OutMsg0Int	Outbound Message_0 Interrupt This bit is set when Outbound Message_0 register is written. The PCI writes it with 1 to clear it. For the CPU, this bit is read only.	0x0
1	OutMsg1Int	Outbound Message_1 Interrupt This bit is set when Outbound Message_1 register is written. PCI writes it with 1 to clear it. For the CPU, this bit is Read Only. ¹	0x0
2	OutDoorInt	Outbound Doorbell Interrupt: This bit is set when at least one bit of Outbound Doorbell register is set. This bit is read only.	0x0
3	OutPQInt	Outbound Post Queue Interrupt This bit is set as long as Outbound Post Queue is not empty. This bit is read only.	0x0
31:4	Reserved		0x0

1. An interrupt to PCI is generated if any of the bits 0,1,2, or 3 is set to 1 given that its corresponding entry in the Outbound Interrupt Mask Register is NOT set.

Table 223: Outbound Interrupt Mask Register, Offset: 0x34

Bits	Field Name	Function	Initial Value
0	OutMsg0IntMsk	Outbound Message_0 Interrupt Mask.	0x0
1	OutMsg1IntMsk	Outbound Message_1 Interrupt Mask.	0x0
2	OutDoorIntMsk	Outbound Doorbell Interrupt Mask	0x0
3	OutPQIntMsk	Outbound Post Queue Interrupt Mask: When set, no interrupt to PCI is generated for set OutPQInt bit in the Outbound Interrupt Cause register.	0x0
31:4	Reserved		0x0

Table 224: Inbound Queue Port Virtual Register, Offset: 0x40

Bits	Field Name	Function	Initial Value
31:0	InQPVReg	Inbound Queue Port Virtual Register A PCI write to this port results in a write to the Inbound Post Queue. A read from this port results in a read from the Inbound Free Queue. Reserved from the CPU side.	0x0

Table 225: Outbound Queue Port Virtual Register, Offset: 0x44

Bits	Field Name	Function	Initial Value
31:0	OutQPVReg	Outbound Queue Port Virtual Register A PCI write to this port results in a write to the Outbound Free Queue. A read from this port results in a read from the Outbound Post Queue. Reserved from the CPU side.	0x0

Table 226: Queue Control Register, Offset: 0x50

Bits	Field Name	Function	Initial Value
0	CirQEn	Circular Queue Enable If 0, a PCI write to this queue is ignored; upon a PCI read from queue 0xffffffff is returned. Reserved from the PCI side.	0x0
5:1	CirQSize	Circular Queue Size 00001 - 16 Kbytes 00010 - 32 Kbytes 00100 - 64 Kbytes 01000 - 128 Kbytes 10000 - 256 Kbytes Reserved from PCI side.	0x1
31:6	Reserved		0x0

Table 227: Queue Base Address Register, Offset: 0x54

Bits	Field Name	Function	Initial Value
19:0	Reserved		0x0
31:20	QBAR	Queue Base Address Register Reserved from the PCI side.	0x0

Table 228: Inbound Free Head Pointer Register, Offset: 0x60

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	InFHPtr	Inbound Free Head Pointer ¹ Reserved from the PCI side.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

1. This register is maintained by CPU software. It is reserved for PCI accesses.

Table 229: Inbound Free Tail Pointer Register, Offset: 0x64

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	InFTPtr	Inbound Free Tail Pointer ¹ Reserved from the PCI side.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

1. This register is incremented by the GT-96100A after a PCI read from Inbound port. It is reserved for PCI accesses.

Table 230: Inbound Post Head Pointer Register, Offset: 0x68

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	InPHPtr	Inbound Post Head Pointer ¹ Reserved from the PCI side.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

1. This register is incremented by the GT-96100A after a PCI write to Inbound port. It is reserved for PCI accesses.

Table 231: Inbound Post Tail Pointer Register, Offset: 0x6c

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	InPTPtr	Inbound Post Tail Pointer ¹ Reserved from the PCI side.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

1. This register is maintained by the CPU software. It is reserved for PCI accesses.

Table 232: Outbound Free Head Pointer Register, Offset: 0x70

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	OutFHPtr	Outbound Free Head Pointer ¹ Reserved from the PCI side.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

1. This register is incremented by the GT-96100A after PCI write to Outbound port. It is reserved for PCI accesses.

Table 233: Outbound Free Tail Pointer Register, Offset: 0x74

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	OutFTPtr	Outbound Free Tail Pointer ¹ Reserved from the PCI side.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

1. This register is maintained by CPU software. It is reserved for PCI accesses.

Table 234: Outbound Post Head Pointer Register, Offset: 0x78

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	OutPHPtr	Outbound Post Head Pointer ¹ Reserved from the PCI side.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

1. This register is maintained by the CPU software. It is reserved for PCI accesses.

Table 235: Outbound Post Tail Pointer Register, Offset: 0x7c

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	OutPTPtr	Outbound Post Tail Pointer ¹ Reserved from the PCI side.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

1. This register is incremented by the GT-96100A after a PCI read from the Outbound port. It is reserved for PCI accesses.

9. INDEPENDENT DMA CONTROLLERS (IDMA CONTROLLERS)

The GT-96100A has four Independent DMA controllers. The IDMA controllers are used to optimize system performance by moving large amounts of data without significant CPU intervention.

Rather than having the CPU read data from one source and write it to another, use the IDMA controllers to directly transfer data independent of the CPU. This allows the CPU to continue executing other instructions simultaneous to the movement of data.

It is possible for each IDMA controller to move data between peripherals on the SDRAM/Device Controller bus, between devices on the PCI buses, or between peripherals on the SDRAM/Device Controller bus and devices on the PCI buses.

Each IDMA transfer uses one of two internal 64-byte FIFOs for moving data. Data is transferred from the source device into an internal FIFO, and from the internal FIFO to the destination device.

The IDMA controller can be programmed to move up to 64KBytes of data per transaction. The burst length of each transfer of each IDMA can be set from 1 to 64 bytes. Accesses can be non-aligned both in the source and the destination.

The GT-96100A has two “72-byte” FIFOs available for the utilization by the DMA engines. Although the maximum DMA burst size is 64 bytes, the extra eight bytes in the FIFO are required for non-double word aligned transfers. Two FIFOs allow for concurrency between two DMA transactions. This means one DMA channel can be reading data from the SDRAM into the first FIFO while another channel is writing data to a PCI target from the second FIFO.

The DMA channels support chained mode of operation. The descriptors are stored in memory, and the DMA engine moves the data until the Null Pointer is reached.

Fly-By DMA transfers are also supported. This type of DMA transfers greatly increase memory bandwidth. Fly-By transfers are permitted to and from a device or to and from SDRAM.

The DMA can be initiated by the CPU writing a register, an external request via a DMAReq* pin, or from a timer/counter. In cases where the transfer needs to be externally terminated, an End of Transfer (EOT[3:0]) pin must be asserted (driven low) for the corresponding DMA channel.

9.1 DMA Channel Registers

Each DMA Channel record consists of the following registers. These registers can be written by the CPU, PCI, or IDMA controller in the process of fetching a new record from memory.

9.1.1 Byte Count Register

The Byte Count Register consists of four registers at offsets 0x800 - 0x80c.

This register is programmed with a 16-bit number containing the number of data bytes this channel must DMA. The maximum number of bytes the DMA controller can be configured to transfer is 64K-1. This register decrements at the end of every burst of transmitted data from source to destination.

When the byte count register is 0, or the End of Transfer pin is asserted, the DMA transaction is finished or terminated.

9.1.2 Source Address Register

The Source Address Register is at offset 0x810 - 0x81c.

This register is programmed with a 32-bit address. This is the source address for data and can be from the SDRAM/Device controller or from PCI.

This register either increments, decrements, or holds the same value according to bits [3:2] of the Channel Control register, see [Table 236](#) .

9.1.3 Destination Address Register

The Destination Address register is at offset 0x820 - 0x82c.

This register is programmed with a 32-bit address. This is the destination address for data. It can be programmed to the SDRAM/Device or to PCI.

This register either increments, decrements, or holds the same value according to bits [5:4] of the Channel Control register, see [Table 236](#) .

9.1.4 Pointer to the Next Record Register

The Pointer to the Next Record Register is at offset 0x830 - 0x83c.

The register contains a 32-bit address where the values for the next DMA Channel record can be loaded for chained operation. This can be from the SDRAM/Device controller or from PCI. The byte count, source address, and destination address must be located at sequential addresses.

NOTE: The next record pointer must be 16 bytes aligned. This means bits [3:0] must be set to 0.

A value of 0 (NULL) for this register indicates that this is the last record in the chain. This register is only used when the channel is configured for Chained Mode, see [Table 236](#) .

9.1.5 Channel Registers

The Channel Register is at offset 0x840 - 0x84c.

This register controls the DMA operation modes. A detailed description of this register's bits is in [Table 9.2](#) .

9.2 DMA Channel Control Register (0x840 - 0x84c)

Each DMA channel has its own unique control register where certain DMA modes can be programmed. [Table 236](#) provides the bit descriptions for each field in the control register and describes the functionality of the DMA Control Register in certain modes. See [Table 264](#) for further information.

Table 236: DMA Channel Control Register (0x840 - 0x84c)

Function	Description
FlyByEn bit	FlyByEn determines whether or not a DMA transfer uses an internal DMA FIFO to host the data from the source, prior to transferring it to the destination. The SDRAM address must always be programmed in the Source Address register when performing a fly-by DMA whether the DMA source or destination is the SDRAM.
R/W bit	This bit is meaningful only in Fly-By mode, FlyByEn bit set to 1. R/W indicates whether the DMA transaction with the SDRAM is a read or write.
SrcDir, bits[3:2]	The SrcDir field contains information about how the source address for the DMA transfer is handled.
DestDir, bits[5:4]	The DestDir field contains information about how the destination address for the DMA transfer is handled.
DatTransLim, bits[8:6]	The DatTransLim field contains the burst limit of each data transfer. The burst limit can vary from one to 64 bytes in module-2 steps (i.e. 1, 2, 4, 8, ..., 64).
ChainMod, bit 9	ChainMod determines whether this channel is set in chained mode or not. In chained mode, the channel record's parameters for the current transaction (Byte Count, Source, Destination, and Next Record Pointer) must be initialized in SDRAM/Device memory space or PCI devices. The address of the first record must be initialized by writing it to the channel's Next Record Pointer register. In non-chained mode the Byte Count, Source, and Destination Registers must be initialized prior to enabling the channel.
IntMode, bit 10	IntMode controls when this channel asserts the DMAComp (DMA Complete) Interrupt. If chained mode is disabled, the setting of IntMode is irrelevant and DMAComp Interrupt will be asserted every time the Byte Count reaches 0.
TransMod, bit 11	TransMod indicates whether the channel is set to operate in demand mode or block mode.
ChanEn, bit 12	ChanEn enables or disables the DMA channel. The DMA channel is enabled or disabled via ChanEn if the channel is in Demand or Block Mode.
FetNexRec, bit 13	FetNexRec is a field which is significant only when chained mode is enabled for the channel.

Table 236: DMA Channel Control Register (0x840 - 0x84c) (Continued)

Function	Description
DMAActSt, bit 14 (Read Only)	<p>DMAActSt is a read only field that can be polled to see the DMA activity status of the channel.</p> <p>In non-chain mode, this bit is de-asserted when Byte_Count reaches zero. In chain-mode, this bit is de-asserted when the pointer to next record is NULL and Byte_Count reaches zero.</p> <p>This bit is reset if the CPU sets chanEn to 0 during DMA transfer.</p>
SDA, Source/Destination Alignment, bit 15	<p>The SDA bit determines whether address alignment is done for source or destination.</p> <p>When a device such as a FIFO is the destination of a DMA, it is recommended to use Destination Alignment to avoid destructive writes. Likewise, if a device such as a FIFO is the source of a DMA, it is recommended to use Source Alignment to avoid destructive reads.</p> <p>If both the DMA Source and Destination addresses are aligned, the meaning of this bit is irrelevant.</p>
Mask DMA Requests, MDREQ, bit 16	<p>Some slower devices require extra time in order to de-assert a DMA request signal. This bit can be used to provide this extra time.</p>
Close Descriptor Enable, CDE, bit 17	<p>A DMA transfer may be halted (by an EOT signal or FetchNextRec is asserted) with some data remaining in the buffer pointed at by the current descriptor. This bit allows writing the remaining byte count in bits 31:16 of the Byte_Count field of the descriptor (located in memory).</p> <p>By writing this field, ownership of the descriptor is returned to the CPU. The CPU then calculates the total number of bytes transferred by the DMA channel by subtracting the remaining byte count from the original Byte_Count.</p>
End Of Transfer Enable, EOTE, bit 18	<p>This bit provides devices which have access to a DMA engine to stop a DMA transfer prior to its completion. In chain mode, this causes fetching a new descriptor from memory (if pointer to next record is not equal to NULL) and executing the next DMA. If the DMA channel is in non-chain mode, then the current DMA transfer is stopped without further action.</p>
End Of Transfer Interrupt Enable, EOTIE, bit 19	<p>EOTIE enables or disables interrupts due to End Of Transfer (EOT) signal activation.</p>
Abort DMA, ABR, bit 20	<p>It is possible that the CPU may need to abort a DMA transfer and reprogram the DMA. This bit flushes internal indications which would normally not get flushed by a mere channel disable.</p> <p>The ABR bit must be used together with En/Dis if CDE and/or EOT are enabled and the CPU requires to abort a transfer for reprogramming.</p>
SLP (bits [22:21]), DLP (bits [24:23]), RLP (bits [26:25])	<p>SLP, DLP, and RLP bits are used to redefine address space of source, destination, or record address location. These enable overriding the local address space with PCIMem0 or PCIMem1 address space.</p>

Table 237: Location of Source Address, SLP

SLP ([22:21])	Function
00	Source address is in local address space.
01	Source address is in PCI_0 memory space.
10	Source address is in PCI_1 memory space.
11	Reserved.

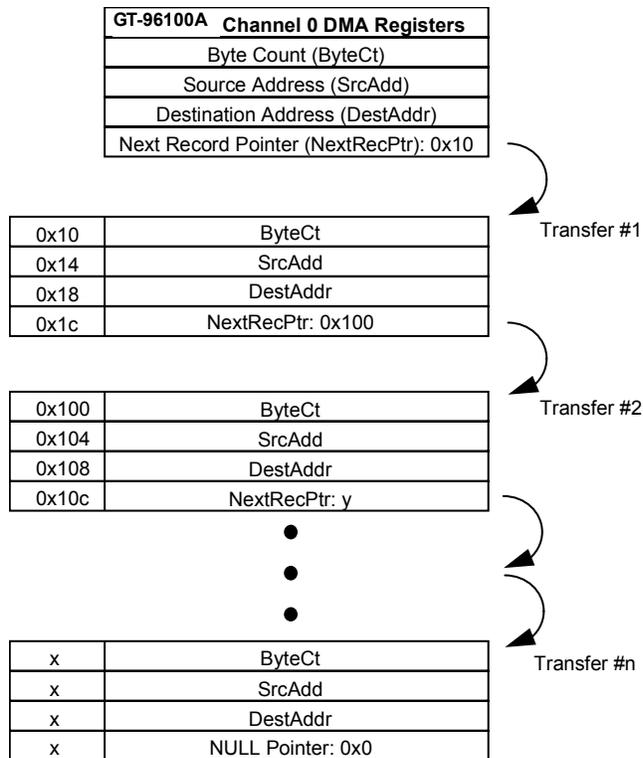
Table 238: Location of Destination Address, DLP

DLP ([24:23])	Function
00	Destination address is in local address space.
01	Destination address is in PCI_0 memory space.
10	Destination address is in PCI_1 memory space.
11	Reserved.

Table 239: Location of Record Address, RLP

RLP ([26:25])	Function
00	Record address is in local address space.
01	Record address is in PCI_0 memory space.
10	Record address is in PCI_1 memory space.
11	Reserved.

Figure 32: Chained Mode DMA



9.3 Restarting a Disabled Channel

In Non-Chained mode, ChanEn must be set to 1.

In Chained mode, the software must find out if the first fetch took place. If it did, only ChanEn is set to 1. If it did not, the FetNexRec is also be set to 1.

9.4 Reprogramming an Active Channel

To reprogram an active channel, the channel must first be disabled by setting ChanEn to 0.

If CDE and/or EOTE are set, then ABR must also be set. Then it must be assured that the channel is no longer active (for example by polling the DMAActSt of the channel).

New DMA parameters must be programmed prior to re-enabling the channel via setting ChanEn to 1.

9.5 Arbitration

The DMA controller has a programmable arbitration scheme between its four channels. The channels are grouped into two groups:

- One group includes channel 0 and 1
- The other group includes channels 2 and 3.

The channels in each group are programmed to have priority so that a selected channel has the higher priority or the same priority in round robin.

The priority between the two groups is programmed in a similar way so that a selected group has a higher priority or to have the same priority in round robin.

The priority scheme has additional flexibility with the programmable Priority Option. With the Priority Option, the DMA bandwidth allocation is divided in a fairer way.

The DMA arbiter control register can be reprogrammed any time regardless of the channels' status (active or not active).

9.6 Current Descriptor Pointer Registers

Each DMA channel has a current descriptor pointer register (CDPTR) associated with it. They are located at offsets 0x870-0x7c.

These descriptor pointers are read/write registers, however, the CPU should not write them. When the NPTR (Next pointer) is written by the CPU, then the CDPTR reloads itself with the same value written to NPTR.

After processing a descriptor, the DMA channel updates the current descriptor using CDPTR, saves NPTR into CDPTR, and fetches a new descriptor. This register is used for closing the current descriptor before fetching the next descriptor.

9.7 Design Information

The following sections contain more detailed information about the GT-96100A's IDMA controllers. The following definitions are used throughout this section:

Table 240: IDMA Controller Design Information Terms and Definitions

Term	Definition
Device	A device located on the memory bus mapped to one of the GT-96100A's Chip Selects (including BootCS).
PCI Agent	Any device located on the PCI bus.
SDRAM	SDRAM memory located on the memory bus.

9.7.1 DMA in Demand Mode

Demand mode is especially designed for transferring data between Memory (Device, SDRAM, PCI agent) and a Device. This is because the DMAAck* is asserted only when the GT-96100A is accessing a Device.

In this mode, the transfer initiator (usually a Device) asserts DMAReq* to signal the GT-96100A that a new DMA transfer should begin. As an acknowledgment response, the GT-96100A asserts the DMAAck* to signal that the asserted DMAReq* is currently being processed.

In each DMA transfer, the DMA attempts to read the amount of DatTranLim bytes from the source address and writes it to destination address. In the source direction, at the beginning and end of the DMA, there may be less than DatTranLim bytes if the address is not aligned or the remaining Byte Count to be transferred is smaller than the DatTranLim. In the destination direction, the DMA writes all data that was read from the source to the destination. This may happen in two DMA accesses if the destination address is not aligned.

The channel stays active until the Byte Count reaches the terminal count or until the CPU disables the channel.

NOTE: If the DMAReq* is always asserted, then this is equivalent to transfer data in Block Mode.

9.7.1.1 Asserting and Deasserting DMAReq*

The DMAReq* must be asserted as long as the transfer initiator has at least DatTranLim of bytes to provide (in case that it is the source) or as long as it has space to absorb at least DatTranLim of bytes (in case that it is the destination).

The DMAReq* should be deasserted by the source when the transfer initiator sees that it does not have at least DatTranLim of bytes to provide (i.e. FIFO empty). DMAReq* should also be deasserted by the destination when it does not have enough space to absorb at least DatTranLim of bytes (i.e. FIFO full) AND DMAAck* is asserted LOW.

9.7.1.2 Asserting DMAAck*

The DMAAck* is asserted only when the GT-96100A accesses a device. It is asserted when ALE is asserted.

9.7.1.3 DMAReq* Sampling

The DMAReq* is sampled all the time but it influences arbitration only in the DMA arbitration cycle or when all channels are idling. The DMA arbitration cycle is the cycle in which the destination unit inside the GT-96100A acknowledges the last written data from the DMA unit.

9.7.1.4 Transferring data examples/recommendations

Table 241: Source and Data Transfer Examples

Source	Destination	Description
SDRAM/PCI	SDRAM/PCI	In this case it is better to use BlockMode cause Memory is typically ready all the time and DMAAck* is NOT asserted. If you choose to work in Demand mode, DMAAck* should be externally generated (i.e., polling accesses of the GT-96100A to certain addresses).
Device	SDRAM or PCI	The Device transfer initiator asserts a DMAReq* when it has at least DatTranLim number of bytes to give. It must de-assert the DMAReq* when no more data is ready and DMAAck* is asserted. Even if another master drives the DMAReq*, the DMAAck* can signal to that master that the GT-96100A is currently accessing the device for read.
SDRAM or PCI	Device	<p>The Device transfer initiator asserts a DMAReq* when it has enough room for DatTranLim number of bytes to be written to it and DMAAck* is asserted. Even if another master drives the DMAReq*, the DMAAck* can signal the master that the GT-96100A is currently accessing the device for write.</p> <p>NOTE: In this situation, the GT-96100A asserts the DMAAck* when its accessing the device for write. This is problematic if DatTranLimit is smaller than or equals eight bytes. The DMAAck* is seen outside late, and due to this the de-assertion of the DMAReq*, is NOT seen in the DMA arbitration cycle. Although the device does not want to be accessed, a new transfer may begin.</p> <ul style="list-style-type: none"> • If DatTranLimit is bigger then eight bytes there is no problem. In this case, it is recommended to have a device that can accepts bursts. • A solution when using one channel only and DatTranLim is less or equal to eight bytes is to set the MDREQ bit.

9.7.1.5 DMA in Block Mode

In block mode, no hand shake signals are used to initiate DMA transfers. The DMA unit completes the transfer once the CPU has programmed the DMA and enabled it.

9.7.2 Non-Chain Mode

In non-chain mode, the CPU or PCI master initiates the DMA channel parameters (Source, Destination Byte Count and command Registers). The DMA starts to transfer data after the enable bit in the Command register is set to 1. The DMA remains in an active state until the Byte Count reaches a terminal count or until the channel is disabled.

9.7.3 Chain Mode

In chain mode, the DMA channel parameters (Source, Destination Byte Count and Pointer to Next Record) are read from records located in Memory, Device, or PCI. The DMA channel stays in the active state until Pointer to Next Record is NULL and the Byte Count reaches the terminal count.

In this mode, an interrupt can be asserted every time the byte count reaches the terminal count or when BOTH the Byte Count reaches the terminal count and the Pointer to Next Record is NULL.

9.7.4 Dynamic DMA chaining

Dynamic chaining is when DMA records are added to a chain that an IDMA controller is actively working on. The main issue is to synchronize between when the GT-96100A reads the last chain record (the NULL pointer) to the time the CPU changes the current last DMA record. Following is an algorithm which provides this synchronization mechanism.

1. Prepare the new record.
2. Change the last record's Pointer to Next Record to point to the new record.
3. Read the DMA control register.

```
If the DMAActSt bit is 0 (NOT active) {
    Update the Pointer to Next Record in the GT-96100A
    assert the FetNexRec bit
}
else {
    read the Pointer to Next Record the GT-96100A.
    If it's equal to NULL {
        Mark (by using a flag) that in the next DMA chain complete interrupt
        you'll need to {
            Update the NRP register in the GT-64010
            Write the Fetch Next Record
        }
    }
}
```

9.7.5 Fly-by DMA

Fly-by is a way to move data directly from the source of the data to its destination. While the source drives the data onto the data bus, the destination immediately latches it into its buffers/memory. The data does not pass through the GT-96100A. This saves at least half of the AD bus bandwidth. Fly-by cycles are requested by the DMA channel and controlled by the memory unit.

9.7.5.1 FlyBy in the GT-96100A

During fly-by, the GT-96100A supplies the full information to the SDRAM involved in the transaction. This includes all control signals (SRAS*, SCAS*, SWr*, SCS* and SDQM) and the address lines (DAdr, BA0, BA1). For the Device, it supplies the control signals (CSTiming*, DmaAck* and DevRdWr*) that support the same waveforms as if the device were not working in fly-by mode. This means that the DmaAck* and DevRdWr* are

latched using ALE. The external logic will have to form the address to the device (if needed) and correct write signals to the device if the device is the destination.

The fly-by cycle is totally compliant with the SDRAM waveforms and the device has to keep up with its speed.

NOTE: The “device parameter register” is ignored during flyby transaction.

9.7.5.2 How to program the GT-96100A for Fly-By

The DMA control register has two bits for Fly-by indications:

Table 242: Fly-By Bits

Bit	Description
FlyByEn	DMA accesses are Fly-by, i.e., the data does not enter the internal FIFO.
FlyByDir	Determines if the device is the source or the destination. This bit affects the DevRdWr* towards the device. NOTE: The SDRAM address must be written to the source register of the DMA channel, regardless of whether the SDRAM is the source or the destination.

9.7.5.3 Design Considerations

1. Device (FIFOs, FPGA) must be fast enough to maintain read/write at SDRAM burst speed.
2. For RAS to CAS setting of 3 DMAReq* must be deasserted within 3 TClk cycles following CSTiming* assertion.
3. For RAS to CAS setting of 2 DMAReq* must be deasserted within 2 TClk cycles following CSTiming* assertion.

9.7.5.4 Determining CS during Fly-By

Bits [29:26,22] in the DeviceX (Bank0, Bank1, Bank2, Bank3 and Boot) parameter registers are used to determine which CS (Chip Select) are activated during FlyBy.

- DMA channel 0 uses bits [29:26,22] of Bank0 parameter register.
- DMA channel 1 uses bits [29:26,22] of Bank1.
- DMA channel 2 uses bits [29:26,22] of Bank2
- DMA channel 3 uses bits [29:26,22] of Bank3.

The interpretation of bits [29:26,22] is as follows:

- Bit 22 Low - BootCS* are the active CS.
- Bit 26 Low - CS0* are the active CS.
- Bit 27 Low - CS1* are the active CS.
- Bit 28 Low - CS2* are the active CS.
- Bit 29 Low - CS3* are the active CS.

NOTE: As a consequence of the nature of this mechanism, only one bit within bits [29:26,22] should be Low.

By default, bit 26 is low. This means CS0 is the active CS unless otherwise programmed.

9.8 Initiating a DMA from a Timer/Counter

Each channel can be programmed to have the DMAReq* sourced from the external DMAReq* pin or from the associated timer/counter. For example, DMA channel 0 can only be enabled by Timer/Counter 0, DMA channel 1 can only be enabled by Timer/Counter 1, etc. If bit 28 in the DMA command register is set to 1, then when the timer/counter reaches the terminal count, an internal DMAReq* is set and a new DMA transfer is initiated. When this bit is set to 1, DMAReq* is ignored. When set to 0, DMAs are initiated by asserting DMAReq*. Initiating DMA from timer/counter is enabled only in demand mode.

9.9 DMA Restrictions

1. In order to reprogram a channel after it has been enabled, it must first be checked that the DMAActSt bit is set to NOT ACTIVE (see [Table 236](#)). If working in CDE mode or EOTE mode then ABR bit must be set to 1 along with En/Dis bit.
2. When Source or Destination address is decremented, both addresses must be double-word-aligned (that is, A2, A1 and A0 should be all zero), and Byte Count must be a multiple of eight (this applies for burst limits greater than eight bytes).
3. Burst reads of more than two double-words from PCI devices have all Byte Enables (BEs) active. This implies that DMA read from PCI I/O space must be aligned or with a burst limit no bigger than eight bytes, in order to avoid PCI spec violation (PCI spec defines correlation between two LSB address bits and byte enables on I/O transaction).
4. When using the address hold option in the source direction (see [Table 236](#)), and SDA bit is set to 1, the source and destination addresses must be double-word aligned.
5. When using the address hold option in the destination direction, both source and destination addresses must be double-word aligned.
6. Records addresses (NPTR) must be a multiple of 16 bytes. In chained mode, if the descriptors are stored in a device, the device must be 32 or 64 bit. If the descriptors are stored in SDRAM or in PCI memory, there are no restrictions on the width of the resource.

NOTE: All descriptors must reside on word-aligned addresses.

7. No support for destination alignment (SDA has no affect) when burst limit is 1, 2, or 4 bytes.
8. When DMA accesses an unmapped address (see [Section 3.4 “Address Space Decoding Errors” on page 63](#)), it results in unpredictable behavior and the DMA channel might need to be stopped by clearing the activate bit of the channel control register.
9. If the DMA state machine has a pending read of the next descriptor AND the descriptor is located in the PCI space, any PCI accesses to the DMA registers are stopped.
10. If the PCI master accessing the GT-96100A DMA registers and the DMA descriptors resides on the far side of a PCI-to-PCI bridge, a lock-up may occur because the PCI requires that all writes must occur before any reads can take place across a PCI-to-PCI bridge.

9.9.1 Fly-by Mode DMA Restrictions

1. The device and SDRAM must be the same width, 32 or 64 bit.
2. In Fly-By transfers, Byte_Count must be a multiple of eight and source address must be word aligned.
3. The SDRAM CAS latency must be programmed to 2.
4. SRASPrchg in all SDRAM parameter registers must pre-programmed to 1(e.g. 3 cycles).

9.10 DMA Control Registers

Table 243: DMA Control Register Map

Description	Offset	Page Number
<i>DMA Record</i>		
Channel 0 DMA Byte Count	0x800	page 232
Channel 1 DMA Byte Count	0x804	page 232
Channel 2 DMA Byte Count	0x808	page 232
Channel 3 DMA Byte Count	0x80c	page 233
Channel 0 DMA Source Address	0x810	page 233
Channel 1 DMA Source Address	0x814	page 233
Channel 2 DMA Source Address	0x818	page 233
Channel 3 DMA Source Address	0x81c	page 233
Channel 0 DMA Destination Address	0x820	page 233
Channel 1 DMA Destination Address	0x824	page 234
Channel 2 DMA Destination Address	0x828	page 234
Channel 3 DMA Destination Address	0x82c	page 234
Channel 0 Next Record Pointer	0x830	page 234
Channel 1 Next Record Pointer	0x834	page 234
Channel 2 Next Record Pointer	0x838	page 235
Channel 3 Next Record Pointer	0x83c	page 235
Channel 0 Current Descriptor Pointer	0x870	page 235
Channel 1 Current Descriptor Pointer	0x874	page 235
Channel 2 Current Descriptor Pointer	0x878	page 235
Channel 3 Current Descriptor Pointer	0x87c	page 236
Channel 0 Control	0x840	page 236
Channel 1 Control	0x844	page 239

Table 243: DMA Control Register Map (Continued)

Description	Offset	Page Number
<i>DMA Record</i>		
Channel 2 Control	0x848	page 239
Channel 3 Control	0x84c	page 239
<i>DMA Arbiter</i>		
Arbiter Control	0x860	page 240

9.10.1 DMA Record

Table 244: Channel 0 DMA Byte Count, Offset: 0x800

Bits	Field Name	Function	Initial Value
15:0	ByteCt	The number of bytes that are left in DMA transfers.	0x0
31:16	ByteRemain	If CDE is set to 1 and the DMA engine owns the descriptor (i.e. DMA is currently in progress), the remaining bytes to transfer will be written. If the CPU owns the descriptor, these bytes can be written to any value.	0x0

Table 245: Channel 1 DMA Byte Count, Offset: 0x804

Bits	Field Name	Function	Initial Value
31:0	Various	Functions as in Channel 0 DMA Byte Count.	0x0

Table 246: Channel 2 DMA Byte Count, Offset: 0x808

Bits	Field Name	Function	Initial Value
31:0	Various	Functions as in Channel 0 DMA Byte Count.	0x0

Table 247: Channel 3 DMA Byte Count, Offset: 0x80c

Bits	Field Name	Function	Initial Value
31:0	Various	Functions as in Channel 0 DMA Byte Count.	0x0

Table 248: Channel 0 DMA Source Address, Offset: 0x810

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address from which the IDMA controller reads the data.	0x0

Table 249: Channel 1 DMA Source Address, Offset: 0x814

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address from which the IDMA controller reads the data.	0x0

Table 250: Channel 2 DMA Source Address, Offset: 0x818

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address from which the IDMA controller reads the data.	0x0

Table 251: Channel 3 DMA Source Address, Offset: 0x81c

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address from which the IDMA controller reads the data.	0x0

Table 252: Channel 0 DMA Destination Address, Offset: 0x820

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address to which the IDMA controller writes the data.	0x0

Table 253: Channel 1 DMA Destination Address, Offset: 0x824

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address to which the IDMA controller writes the data.	0x0

Table 254: Channel 2 DMA Destination Address, Offset: 0x828

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address to which the IDMA controller writes the data.	0x0

Table 255: Channel 3 DMA Destination Address, Offset: 0x82c

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address to which the IDMA controller writes the data.	0x0

Table 256: Channel 0 Next Record Pointer, Offset: 0x830

Bits	Field Name	Function	Initial Value
31:0	NextRecPtr	The address for the next record of DMA. A value of 0 means a NULL pointer (end of the chained list). NOTE: The next record pointer must be 16 bytes aligned. This means bits [3:0] must be set to 0.	0x0

Table 257: Channel 1 Next Record Pointer, Offset: 0x834

Bits	Field Name	Function	Initial Value
31:0	NextRecPtr	The address for the next record of DMA. A value of 0 means a NULL pointer (end of the chained list). NOTE: The next record pointer must be 16 bytes aligned. This means bits [3:0] must be set to 0.	0x0

Table 258: Channel 2 Next Record Pointer, Offset: 0x838

Bits	Field Name	Function	Initial Value
31:0	NextRecPtr	The address for the next record of DMA. A value of 0 means a NULL pointer (end of the chained list). NOTE: The next record pointer must be 16 bytes aligned. This means bits [3:0] must be set to 0.	0x0

Table 259: Channel 3 Next Record Pointer, Offset: 0x83c

Bits	Field Name	Function	Initial Value
31:0	NextRecPtr	The address for the next record of DMA. A value of 0 means a NULL pointer (end of the chained list). NOTE: The next record pointer must be 16 bytes aligned. This means bits [3:0] must be set to 0.	0x0

Table 260: Current Descriptor Pointer 0, Offset: 0x870

Bits	Field Name	Function	Initial Value
31:0	CDPTR0	Channel 0 Current Address Descriptor Pointer	0x0

Table 261: Current Descriptor Pointer 1, Offset: 0x874

Bits	Field Name	Function	Initial Value
31:0	CDPTR1	Channel 1 Current Address Descriptor Pointer	0x0

Table 262: Current Descriptor Pointer 2, Offset: 0x878

Bits	Field Name	Function	Initial Value
31:0	CDPTR2	Channel 2 Current Address Descriptor Pointer	0x0

Table 263: Current Descriptor Pointer 3, Offset: 0x87c

Bits	Field Name	Function	Initial Value
31:0	CDPTR3	Channel 3 Current Address Descriptor Pointer	0x0

9.10.2 DMA Channel Control

Table 264: Channel 0 Control, Offset: 0x840

Bits	Field Name	Function	Initial Value
0	FlyByEn	Data Internal/External to DMA FIFO 0 - Internal Data is read from source address into DMA FIFO and written to destination address 1 - External (Fly By) Data is transferred to/from devices on SDRAM bus from/to SDRAM	0x0
1	RdWrFly	SDRAM Read/Write Meaningful only in Fly-by mode. 0 - Read from SDRAM 1 - Write to SDRAM	0x0
3:2	SrcDir	Source Direction. 00 - Increment source address 01 - Decrement source address 10 - Hold in the same value 11 - Reserved	0x0
5:4	DestDir	Destination Direction. 00 - Increment destination address 01 - Decrement destination address 10 - Hold in the same value 11 - Reserved	0x0
8:6	DatTransLim	Data Transfer Limit in each DMA access. 101 - 1 Byte 110 - 2 Bytes 010 - 4 Bytes 000 - 8 Bytes 001 - 16 Bytes 011 - 32 Bytes 111 - 64 bytes	0x0

Table 264: Channel 0 Control, Offset: 0x840 (Continued)

Bits	Field Name	Function	Initial Value
9	ChainMod	<p>Chained Mode</p> <p>0 - Chained mode</p> <p>When a DMA access is terminated, the parameters of the next DMA access comes from a record in memory that is pointed to by the NextRecPtr register.</p> <p>1 - Non-Chained mode</p> <p>Only uses the values programmed by the CPU (or PCI) directly into the ByteCt, SrcAdd, and DestAdd registers.</p>	0x0
10	IntMode	<p>Interrupt Mode</p> <p>0 - Interrupt asserted every time the DMA byte count reaches terminal count.</p> <p>1 - Interrupt every NULL pointer (in Chained mode).</p>	0x0
11	TransMod	<p>Transfer Mode</p> <p>0 - Demand</p> <p>1 - Block</p>	0x0
12	ChanEn	<p>Channel Enable</p> <p>0 - Disable</p> <p>1 - Enable</p>	0x0
13	FetNexRec	<p>Fetch Next Record</p> <p>1 - Forces a fetch of the next record, even if the current DMA has not ended.</p> <p>This bit is reset after fetch is completed and is only meaningful in Chained mode.</p>	0x0
14	DMAActSt	<p>DMA Activity Status</p> <p>Read only.</p> <p>Assertion of this bit is caused by asserting ChanEn (bit 12).</p> <p>0 - Channel is not active.</p> <p>1 - Channel is active.</p>	0x0
15	SDA	<p>Source Destination Alignment</p> <p>0 - Alignment is towards the source address.</p> <p>1 - Alignment is towards the destination address</p> <p>NOTE: No support for destination alignment when burst limit is 1, 2, or 4 bytes.</p>	0x0

Table 264: Channel 0 Control, Offset: 0x840 (Continued)

Bits	Field Name	Function	Initial Value
16	MDREQ	Mask DMA Requests 0 - Don't mask 1 - Mask DMA requests for 3 cycles starting DMA arbitration cycle.	0x0
17	CDE	Close Descriptor Enable If enabled, DMA writes remaining byte count to bits [31:16] of ByteCount field in the descriptor. 0 - Disable 1 - Enable	0x0
18	EOTE	End Of Transfer Enable If enabled, DMA transfer can be stopped in the middle of transfer using the EOT signal. If DMA channel is working in chain mode, this will cause fetching a new descriptor, otherwise the DMA transfer is stopped. 0 - Disable 1 - Enable	0x0
19	EOTIE	End Of Transfer Interrupt Enable If enabled and EOT pin is asserted, DMA generates an interrupt. 0 - Disable 1 - Enable	0x0
20	ABR	Abort DMA Transfer This bit must be set if the CPU wants to stop and re-program DMA, and CDE (bit 17) and/or EOTE (bit 19) is set to 1. When the CPU issues this command, it must also set ChanEn (bit 12) to 0. 0 - No influence on channel behavior. 1 - Abort DMA.	0x0
22:21	SLP	Override Source Address 00 - No override. Use local address space for source 01 - Source address is in PCI_0 memory space. 10 - Source address is in PCI_1 memory space. 11 - Reserved	0x0

Table 264: Channel 0 Control, Offset: 0x840 (Continued)

Bits	Field Name	Function	Initial Value
24:23	DLP	Override Destination Address 00 - No override. Use local address space for destination. 01 - Destination address is in PC_0 memory space. 10 - Destination address is in PCI_1 memory space. 11 - Reserved	0x0
26:25	RLP	Override Record Address 00 - No override. Use local address space for record. 01 - Record address is in PCI_0 memory space. 10 - Record address is in PCI_1 memory space. 11 - Reserved	0x0
27	Reserved		0x0
28	DMAReqSrc	DMA Request Source 0 - Request taken from DMAReq* pin. 1 - Request taken from timer/counter.	0x0
31:29	Reserved		0x0

Table 265: Channel 1 Control, Offset: 0x844

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Channel 0 Control.	0x0

Table 266: Channel 2 Control, Offset: 0x848

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Channel 0 Control.	0x0

Table 267: Channel 3 Control, Offset: 0x84c

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Channel 0 Control.	0x0

9.10.3 DMA Arbiter

Table 268: Arbiter Control, Offset: 0x860

Bits	Field Name	Function	Initial Value
1:0	PrioChan1/0	Priority between Channel 0 and Channel 1. 00 - Round robin 01 - Priority to channel 1 over channel 0 10 - Priority to channel 0 over channel 1 11 - Reserved	0x0
3:2	PrioChan3/2	Priority between Channel 2 and Channel 3. 00 - Round robin 01 - Priority to channel 3 over 2 10 - Priority to channel 2 over 3 11 - Reserved	0x0
5:4	PrioGrps	Priority between the group of channels 0/1 and the group of channels 2/3. 00 - Round robin 01 - Priority to channels 2/3 over 0/1 10 - Priority to channels 0/1 over 2/3 11 - Reserved	0x0
6	PrioOpt	Priority Option Enabled/Disable 0 - High priority device relinquishes the bus for a requesting device for one DMA transaction after it was serviced. 1 - High priority device is granted as long as it requests the bus.	0x0
31:7	Reserved		0x0

10. PCI ARBITER

The GT-96100A integrates two PCI arbiter functions. One is dedicated for PCI_0 and the other for PCI_1. The PCI_0 arbiter handles up to six external agents and one internal agent (PCI_0 master). The PCI_1 arbiter supports a total of four external agents in addition to the internal PCI_1 master¹.

The PCI arbiters implement a priority based weighted Round Robin (RR) arbitration mechanism. Each agent is assigned a programmable priority tag (either high or low), and the arbitration is done according to these priorities. A simple Round Robin arbitration is performed within each priority level, while a weighted function is implemented for arbitrating between the high priority and the low priority groups.

10.1 Interface

Figure 33: PCI Arbiter's Interface Diagram

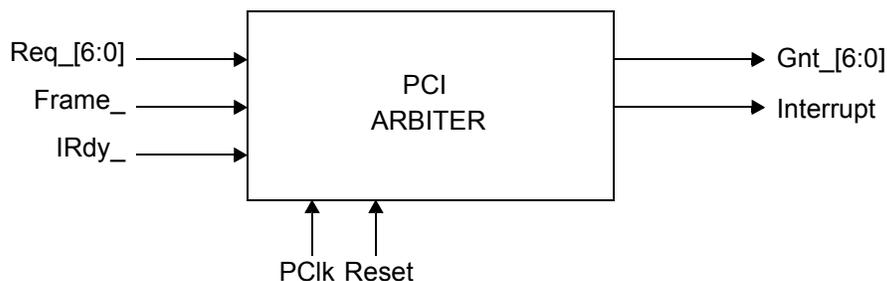


Table 269: PCI Arbiter's Interface

Signal	Description
PClk	PCI Clock
Reset	Master Reset
Interrupt	Interrupt NOTE: This signal is asserted when the arbiter recognizes a "Broken" Master (i.e. a master which does not respond to a grant signal assertion).
Gnt_[6:0]	GRANT# output signals NOTE: Gnt_[0] of PCI_0 arbiter is internally connected PCI_0 master. The same holds for PCI_1.
Req_[6:0]	Input REQ# signals. NOTE: Req_[0] of PCI_0 arbiter is internally connected PCI_0 master. The same holds for PCI_1.

1. The internal design of the PCI arbiter logic handles a total of 7 request/grant pairs for each of the arbiters. But, due to package limitations, the GT-96100A is limited to a total of nine external request/grant pairs. These are divided between PCI_0 arbiter and PCI_1 arbiter. Thus, the user cannot connect six external agents to PCI_0 arbiter and four external agents to PCI_1 arbiter at the same time (it's either 6 to PCI_0 and 3 to PCI_1, or 5 to PCI_0 and 4 to PCI_1).

Table 269: PCI Arbiter's Interface (Continued)

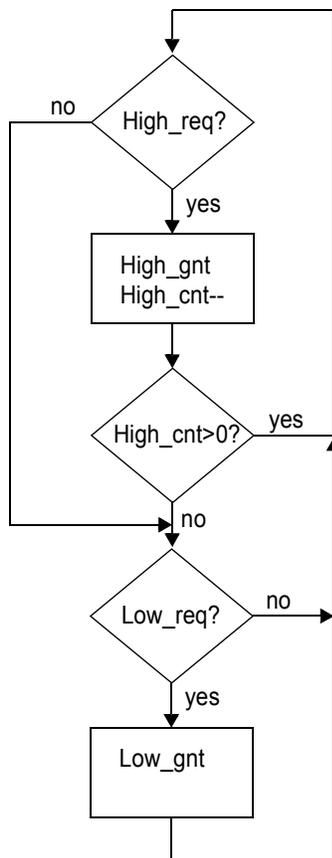
Signal	Description
Frame_	PCI FRAME# signal.
IRdy_	PCI IRDY# signal.

10.2 Arbitration Scheme

As noted above, each of the PCI arbiter's requests has a user programmable priority level associated with it. Two levels of priority are supported: High and Low. PCI Bandwidth allocation per priority is programmable as well, enabling the user to optimize the PCI to application needs.

Arbitration flow is shown in [Figure 34](#).

Figure 34: PCI Arbitration Flow



In relation to [Figure 34](#), the following points should be noted:

- The two request signals (High_req, Low_req) are generated by “ANDing” each of the request lines with its respective priority attribute, and ORing the results. For example:

$$\text{High_req} = (\text{req_}[0] \text{ AND } (\text{req_prio}[0]==\text{high})) \text{ OR} \dots$$

$$(\text{req_}[1] \text{ AND } (\text{req_prio}[1]==\text{high})) \text{ OR} \dots$$
- There is a counter associated with the priority scheme - High_cnt. The counter is used to assign different weights to each priority level. This is a count down counter that decrements each time a high priority request (High_req) is granted. When High_cnt expires, a slot is opened for low priority requests, and the counter is set to its preset value.
- Each time a low priority request (Low_req) is granted, High_cnt counter is preset.

10.3 Arbitration Parking

The PCI arbiter is designed to perform a default parking on the last agent granted. In order to overcome problems that happen with some PCI devices that do not handle parking properly, there is an option to disable parking on a per PCI master basis. This is done via the PCI_Arbiter_Configuration register PD[6:0] bits.

NOTE: In addition to disabling parking to avoid issues with some problematic devices, the user must also disable parking on any unused request/grant pair. This is required to avoid possible parking on non-existent PCI masters. For example, if only 3 external agents are connected to PCI_0 arbiter (using REQ0/GNT0, REQ1/GNT1, REQ2/GNT2 pins), then PD[6:4] should be set to 1.

10.4 PCI Arbiter Configuration Register

Table 270: PCI_0 Arbiter Configuration Register, Offset: 0x101AE0

Bits	Field Name	Function	Initial Value
0	GCen	Gap Cycle enable When this bit is set to 1, the PCI arbiter forces grant to be asserted only after the PCI bus is idle. This guarantees two turn around cycles.	0
1	BDen	Broken Detection enable Setting this bit to 1 enables the detection of broken master. A master is said to be broken if it fails to respond to grant assertion within a window specified in BV (see BV description above).	0

Table 270: PCI_0 Arbiter Configuration Register, Offset: 0x101AE0 (Continued)

Bits	Field Name	Function	Initial Value
2	PAen	<p>Priority Arbitration Enable</p> <p>When this bit is set to 1, weighted round robin arbitration is performed between high priority and low priority groups.</p> <p>When this bit is reset, no round robin arbitration takes place and low priority requests are granted only when no high priority request is pending.</p> <p>NOTE: If HPPV is set to zero value and PAen is 1, priority scheme is reversed. This means that high priority requests are granted only if no low priority request is pending.</p> <p>NOTE: Gap Cycle must be enabled in the PCI Arbiter when working with priority Arbitration.</p>	0
6:3	BV	<p>Broken Value</p> <p>This value sets the maximum number of cycles that the arbiter waits for a PCI master to respond to its grant assertion. If a PCI master fails to assert FRAME* within this time, the PCI arbiter aborts the transaction and performs a new arbitration cycle. In addition, a maskable interrupt is generated.</p> <p>NOTE: The PCI arbiter waits for the current transaction to end before starting to count the wait-for-broken cycles.</p>	0
13:7	P[6:0]	<p>Priority</p> <p>These bits assign priority levels to the requests connected to the PCI arbiter. When a PM bit is set to 1, priority of the associated request is high.</p> <p>The mapping between P bits and the request/grant pairs is similar to the one shown for PD bits above.</p>	0
20:14	PD[6:0]	<p>Parking Disable</p> <p>These bits can be used to disable parking on any of the PCI masters. When a PD is set to 1, parking on the associated PCI master is disabled.</p> <p>The mapping between the PD bits and the request/grant pairs is as follows:</p> <ul style="list-style-type: none"> • PD[0] - internal PCI master unit • PD[1] - external REQ0/GNT0 • PD[2] - external REQ1/GNT1 • PD[3] - external REQ2/GNT2 • PD[4] - external REQ3/GNT3 • PD[5] - external REQ4/GNT4 • PD[5] - external REQ5/GNT5 <p>NOTE: The arbiter parks on the last master granted unless disabled through the PD bit. Also, if PD bits are all 1, the PCI arbiter parks on the internal PCI master.</p>	0
28:21	HPPV	<p>High Priority Preset Value</p> <p>This is the preset value of the high priority counter (High_cnt). This counter decrements each time a high priority request is granted. When the counter reaches zero, it reloads with this preset value. The counter also reloads when a low priority request is granted.</p>	0

Table 270: PCI_0 Arbiter Configuration Register, Offset: 0x101AE0 (Continued)

Bits	Field Name	Function	Initial Value
29	AR6en	<p>Arbiter Request 6 enable</p> <p>Setting this bit enables the 6th request/grant pair connection to PCI_0 arbiter.</p> <p>0 - pins PB[4], PB[5] are connected to PCI_1 arbiter and function as PARB1_Gnt4, PARB1_Req4.</p> <p>1 - pins PB[4], PB[5] are connected to PCI_0 arbiter and function as PARB0_Gnt6, PARB0_Req6.</p>	0
30	GNT/CLK	<p>Grant or OTSCLK</p> <p>This bit controls the function of pin PB6.</p> <p>0 - this pin to function as OTSCLK1.</p> <p>1 - PB6 functions as PABR0_GNT4.</p>	0
31	EN	<p>Enable</p> <p>Setting this bit to 1 enables operation of the arbiter. When the arbiter is enabled, the internal PCI_0 master is connected to it.</p> <p>The arbiter configuration must be set before the arbiter is enabled. This means two writes must be made to this register. The first write sets the arbiter configuration and must be done with the enable bit set to 0. With the second write, set the enable bit to 1 to enable operation of the arbiter.</p> <p>NOTE: When the arbiter is enabled, PCI REQ* and GNT* pins change their function: REQ* becomes Grant (PCI arbiter output) and GNT* becomes Req (PCI arbiter input). For example, enabling PCI_0 arbiter causes (req0*,gnt0*) pair to function as (PARB0_GNT1, PARB0_REQ1) respectively.</p> <p>BV must be set to a value greater than zero, or else there will always be a break detection, as the arbiter waits 0 cycles for FRAME* assertion.</p>	0

Table 271: PCI_1 Arbiter Configuration Register, Offset: 0x101AE4

Bits	Field Name	Function	Initial Value
31:0	Various	Similar to PCI_0 arbiter configuration register (except for bit 30:29 which are reserved).	0

11. COMMUNICATION INTERFACE UNIT (CIU)

The GT-96100A's IDMA, Ethernet ports, and SDMA engines have one common interface with the other units within the GT-96100A. These other units include the CPU interface unit (Punit), the memory control unit (Dunit), and the PCI unit (Lunit). The common interface supports a single master, meaning that only one agent can gain control and access the memory (through the Dunit) or the PCI (through the Lunit) at a time.

In order to support this shared interface, the GT-96100A provides an arbitration mechanism, which arbitrates between requests generated by the IDMA, SDMA, and Ethernet. The arbitration logic supports multiple priority levels as well as different weights assigned to each priority level. The priority is programmable per request and the weights are programmable per priority.

This arbitration mechanism also handles MASTER type requests. MASTER requests originate within the communication unit and are targeted at other units in the GT-96100A. For example, a request generated by one of the Ethernet units and is targeted at the Dunit (for SDRAM read/write access) is a MASTER request. Similarly, a request generated by the IDMA and is targeted at the PCI unit is also a MASTER request. The transactions associated with MASTER requests are referred to as MASTER transactions.

In addition to handling MASTER type requests and MASTER transactions, the arbiter also handles SLAVE transactions, which originate from other units within the GT-96100A (e.g. Punit, Lunit) and are targeted at the communication unit. SLAVE requests are used by the other units to access internal registers within the communication unit. Arbitration between SLAVE type requests is fixed and gives the highest priority to the Punit (i.e. requests from CPU).

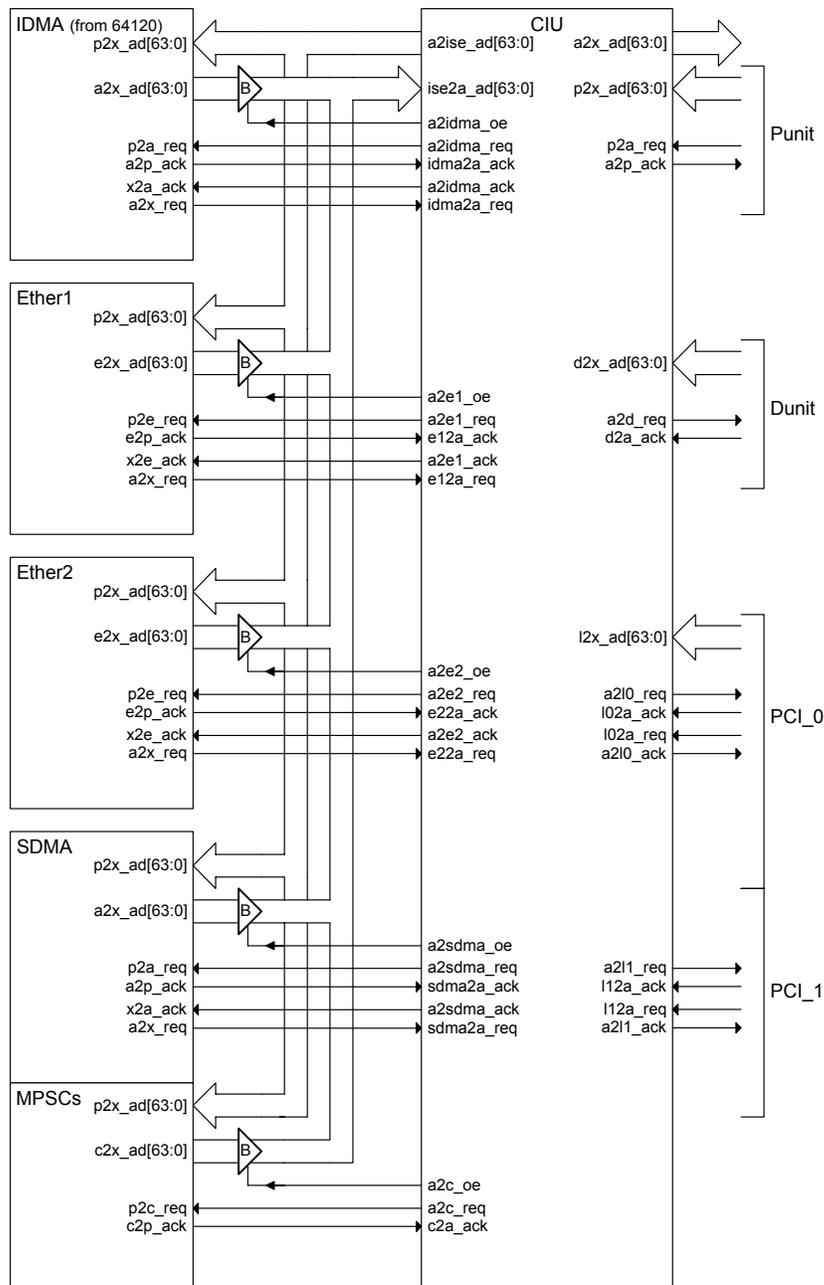
The MASTER/SLAVE arbiter and the associated logic that handle the MASTER/SLAVE transactions are collectively called the Communication Interface Unit (CIU).

This section provides details about the arbitration scheme as well as high level design aspects of the CIU.

11.1 CIU Connectivity

Figure 35 shows a block diagram of the connectivity between the CIU and the communication unit agents.

Figure 35: CIU Connection Diagram



11.2 Address Decoding and PCI Override (MASTER)

The GT-96100A's CIU supports two types of MASTER transactions - normal and PCI override.

For normal transactions, the CIU performs address decoding and access either the SDRAM/DEVICE or PCI_0/PCI_1 based on the decoding result. See [Section 3. "Address Space Decoding" on page 56](#) for a description of the address decoding scheme.

The PCI override feature allows the user to direct all transactions from a specific communication agent to the PCI_0 direction, overriding address decoding. The override feature is programmable for the IDMA, Ethernet, SDMA units. For information on activating this feature, see:

- IDMA: [Table 236](#).
- Ethernet: [Table 296](#).
- SDMA: [Table 316](#).

11.2.1 Address Decoding Errors

The CIU is capable of handling transactions, which result in address decoding errors. If an address decoding error occurs, the CIU performs a dummy transaction with the initiating unit, discards the data, and sets the DMAout bit in the interrupt Main_Cause register, see [Table 391](#).

11.3 Arbitration Scheme

The CIU arbiter is designed to handle one transaction at a time. This could be either a MASTER or SLAVE transaction. Priority at the MASTER/SLAVE level is fixed and gives higher priority to SLAVE type transactions. The priorities are listed below (from high to low):

- Punit - slave request from CPU (highest priority).
- L0unit - slave request from PCI_0.
- L1unit - slave request from PCI_1.
- MASTER requests (lowest priority).

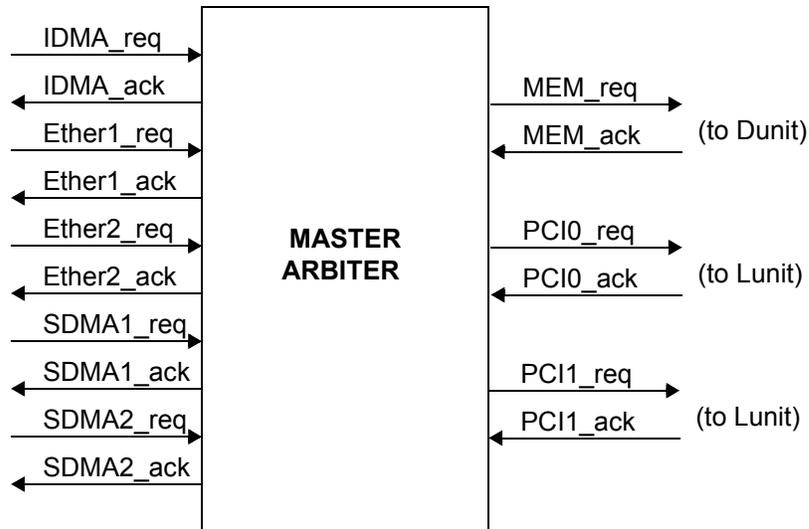
Arbitration within the MASTER level is very flexible and is detailed in the following sections.

11.3.1 Master Arbitration

The arbiter supports MASTER requests from the following sources within the communication unit:

- IDMA,
- Ethernet 1,
- Ethernet 2,
- SDMA 1,
- SDMA 2.

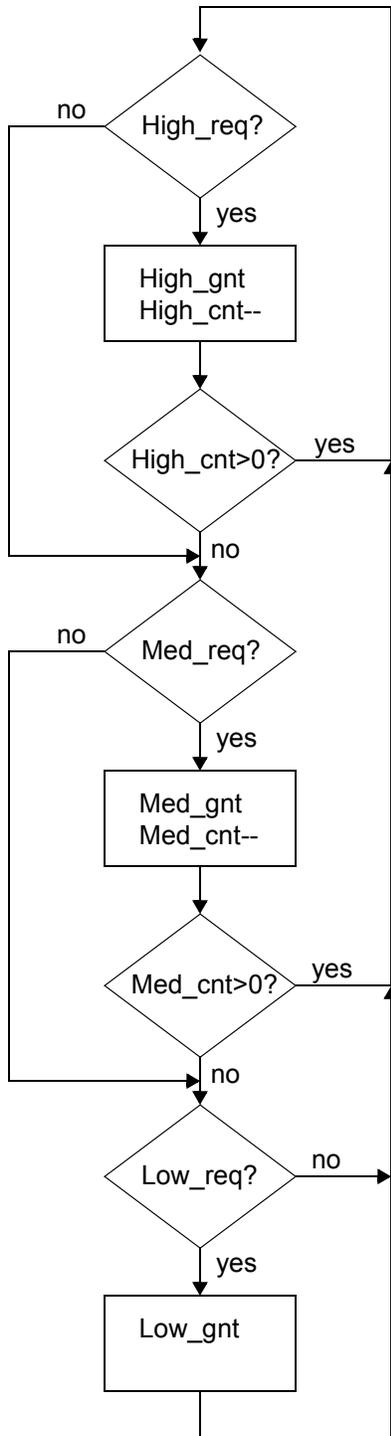
[Figure 36](#) depicts the arbiters' logical connectivity, for MASTER requests.

Figure 36: Arbiter Connectivity

Each of the requests shown in [Figure 36](#) has a user programmable priority level associated with it. The three levels of priority are High, Medium, Low. In addition, bandwidth allocation per priority is programmable as well, providing the user with the capability to optimize memory access to application needs.

Arbitration flow is shown in [Figure 37](#).

Figure 37: MASTER Arbitration Flow



In relation to [Figure 37](#), note the following:

- The three request signals (High_req, Med_req, Low_req) are generated by ANDing each of the request lines shown in [Figure 36](#) with its respective priority attribute, and ORing the results. For example -
- High_req = (IDMA_req AND (IDMA_prio=high)) OR (SDMA_req1 AND (SDMA_prio1=high)) OR...
- There are two counters associated with the priority scheme - High_cnt and Med_cnt. These counters are used to assign different weights to each priority level. The counters are countdown based on the request being granted. Each time a high priority request (High_req) is granted, the high priority counter (High_cnt) is decremented. When High_cnt reaches zero, a slot is opened for lower priority requests, and the counter is set to its preset value. The same holds for Med_cnt.
- Each time a lower priority request is granted, the higher level counters are preset. When a medium priority request (Med_req) is granted, High_cnt counter is preset. When a low priority request is granted, both High_cnt and Med_cnt are preset.

11.4 CIU Arbiter Configuration Register

Table 272: CIU Arbiter Configuration Register, Offset: 0x101AC0

Bits	Field Name	Function	Initial Value
1:0	IPL	IDMA Priority Level These bits assign the priority to the IDMA request according to the following: 11 - High priority 10 - Medium priority 01 - Low priority 00 - Disabled (Request is masked.)	0
3:2	EPL1	Ethernet port Priority Level 1	0
5:4	EPL2	Ethernet port Priority Level 2	0
7:6	SPL1	SDMA Priority Level 1 These bits assign the priority to the SDMA request 1. See the IPL bit description in this table for details on priority assignment.	0
9:8	SPL2	SDMA Priority Level 2 These bits assign the priority to the SDMA request 2. See the IPL bit description in this table for details on priority assignment.	0
15:10		Reserved	0

Table 272: CIU Arbiter Configuration Register, Offset: 0x101AC0 (Continued)

Bits	Field Name	Function	Initial Value
18:16	MPPV	Medium Priority Preset Value This is the preset value of the medium priority counter (Med_cnt). When the counter reaches zero, it reloads with this preset value. NOTE: If the medium priority counter is enabled (MPCE bit set to 1), MPPV value must not be set to zero. A value of zero in MPPV is allowed only when the counter is disabled (i.e. when MPCE = 0).	0
19	MPCE	Medium Priority Counter Enable When set, enables operation of the medium priority counter (Med_cnt). When reset, the medium priority counter is disabled and low priority requests are only granted only when no medium priority request is pending.	0
22:20	HPPV	High Priority Preset Value This is the preset value of the high priority counter (High_cnt). When the counter reaches zero, it reloads with this preset value. NOTE: If the high priority counter is enabled (HPCE bit set to 1), HPPV value must not be set to zero. A zero value in HPPV is allowed only when the counter is disabled (i.e. when HPCE = 0).	0
23	HPCE	High Priority Counter Enable When set, enables operation of the high priority counter (High_cnt). When reset, the high priority counter is disabled and lower priority requests are only granted when no high priority request is pending.	0
24	E0R	Ethernet Unit 0 Software Reset 0 - resets the Ethernet_1 unit	0
25	E1R	Ethernet Unit 1 Software Reset 0 - resets the Ethernet_1 unit w	0
30:26	Reserved	Reserved.	0
31	BLED	Big Little Endian for Descriptors 0 - Big Endian 1 - Little Endian This bit controls the endianness of all descriptors handled by the Ethernet DMA and the Serial DMA (SDMAs) engines. It does not affect the data portion - endianness of the data is controlled via the various DMA configuration registers.	1

12. 10/100MB ETHERNET UNIT

12.1 Functional Overview

The 10/100Mb Ethernet unit handles all functionality associated with moving packet data between local memory or PCI and the Ethernet ports. The unit in the GT-96100A is designed to support two independent 10/100Mb Ethernet ports.

Each 10/100 Mbit port is fully compliant with the IEEE 802.3 and 802.3u standards and integrates the MAC function and a dual speed MII interface. The port's speed (10 or 100Mb/s) as well as the duplex mode (half or full duplex) is auto-negotiated through the PHY and does not require user intervention. The port also features 802.3x flow-control mode for full-duplex and backpressure mode for half duplex.

Integrated address filtering logic provides support for up to 8K MAC addresses. The address table resides in memory and a proprietary hash function is used for address table management. The address table functionality supports Multicast as well as Unicast address entries.

An important feature related to the address recognition is IGMP packet trapping mode. In this mode layer 3 hardware analysis is performed in order to check if a packet being received is an IGMP packet. Each packet identified as IGMP is queued in the high priority queue of the port from which it was received. The IGMP analysis is performed on the fly, so it does not impact bandwidth capability.

Another important feature related to priority queues is the Type of Service queuing algorithm. This algorithm is based on the decoding in layer 3 of the DSCP field from the IP header. If enabled, the decoded field indexes a 64 entry IPT Table in the Ethernet register space. The 2-bit output of this table sets the priority queue of the received packet.

The Ethernet unit integrates powerful DMA engines, which automatically manage data movement between buffer memory and the ports, and guarantee wire-speed operation on all ports (even when all ports are in 100Mb full-duplex mode). There are two DMA engines per port - one dedicated for receive and the other for transmit.

The DMA logic handles multiple priority queues per port, providing support for priority sensitive data in both directions. There are four receive priority queues and two transmit priority queues per port. Priority information for received packets is either extracted from the packet tag (if the packet is VLAN tagged) or from the destination address entry in the address table (if the packet is not tagged) of the IP parameters as described above. The priority function is MAX (ip_parameters, vlan_tag, address_entry).

12.2 Port Features

The 10/100Mb Ethernet port provides the following features:

- IEEE 802.3 compliant MAC layer function.
- IEEE 802.3u compliant MII interface.
- 10/100Mb operation - half and full duplex.
- Flow control features:
 - IEEE 802.3x flow-control for full-duplex operation mode.
 - Backpressure for half duplex operation mode.
- Internal and external loopback modes.
- Transmit functions:
 - Short frame (less than 64 bytes) zero padding.
 - Long frames transmission (limited only by external memory size).
 - Programmable values for IPG and Blinder timers.
 - CRC generation (programmable per packet).
 - Automatic frame retransmission upon collision (with programmable retransmit limit).
 - Backoff algorithm execution.
 - Error report.
- Receive functions:
 - 1/2k or 8k address filtering capability.
 - Address filtering modes:
 - Perfect filtering.
 - Reverse filtering.
 - Promiscuous mode.
 - Broadcast reject mode.
 - IGMP packet trapping (layer 3 analysis in hardware).
 - Automatic discard of errored frames, short (less than 64 bytes) or collided.
 - Reception of long frames (programmable up to 64Kbytes).
 - CRC checking.
 - Pass bad frames mode.
 - Error report.

12.3 Operational Description

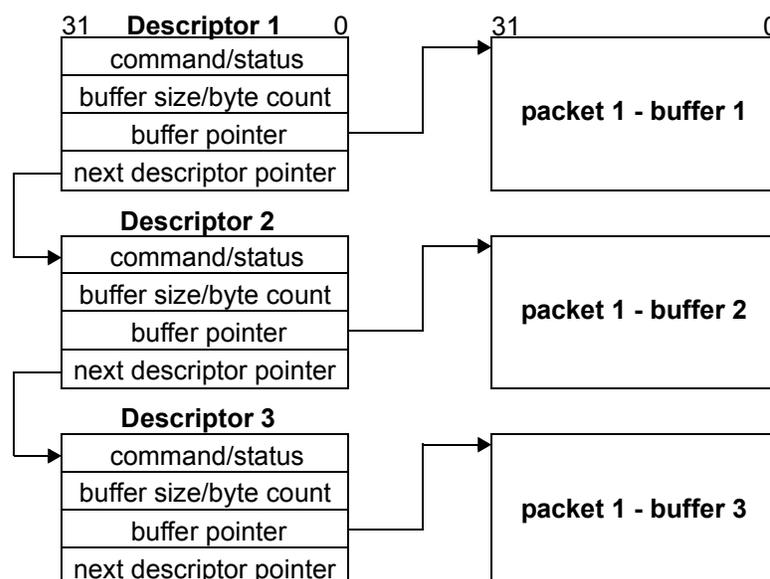
12.3.1 General Overview

The Ethernet unit provides multiple Ethernet ports functionality, with each port capable of running at either 10 or 100Mb/s (half or full-duplex) independently of the other port. Each port interfaces a MII PHY on its serial side and manages packet data transfer between memory and MII. The data is stored in memory buffers, with any single packet spanning multiple buffers if necessary. Upon completion of packet transmission or reception, a status report, which includes error indications, is written by the Ethernet unit to the first or last descriptor associated with this packet.

The buffers are allocated by the CPU and are managed through chained descriptor lists. Each descriptor points to a single memory buffer and contains all the relevant information relating to that buffer (i.e. buffer size, buffer pointer, etc.) and a pointer to the next descriptor. Data is read from buffer or written to the buffer according to information contained in the descriptor. Whenever a new buffer is needed (end of buffer or end of packet), a new descriptor is automatically fetched and the data movement operation is continued using the new buffer.

Figure 38 shows an example of memory arrangement for a single packet using three buffers.

Figure 38: Ethernet Descriptors and Buffers



The following sections provide detailed information about the operation and user interface of the Ethernet unit and its logic subsections.

12.3.2 Transmit Operation

In order to initialize a transmit operation, the CPU must do the following:

1. Prepare a chained list of descriptors and packet buffers.

NOTE: The TxDMA supports two priority transmit queues - high and low. If the user wants to take advantage of this capability, a separate list of descriptors and buffers must be prepared for each of the priority queues.

2. Write the pointer to the first descriptor to the DMA's current descriptor registers (TxCDP) associated with the priority queue to be started. If both priority queues are needed, initialize TxCDP for each queue.
3. Initialize and enable the Ethernet port by writing to the port's configuration and command registers.
4. Initialize and enable the DMA by writing to the DMA's configuration and command registers.

After completing these steps, the DMA starts and performs arbitration between the transmit queues according to the value programmed in Port_Configuration_Extend<PRIOtx> (see [Table 289](#) for more details). The DMA then fetches the first descriptor from the specific queue it decided to serve, and starts transferring data from memory buffer to the TX-FIFO. When either 384 bytes of packet data are in the FIFO or when the entire packet is in the FIFO (for packets shorter than 384 bytes), the port initiates transmission of the packet across the MII. While data is read from the FIFO, new data is written into the FIFO by the DMA.

For packets that span more than one buffer in memory, the DMA will fetch new descriptors and buffers as necessary.

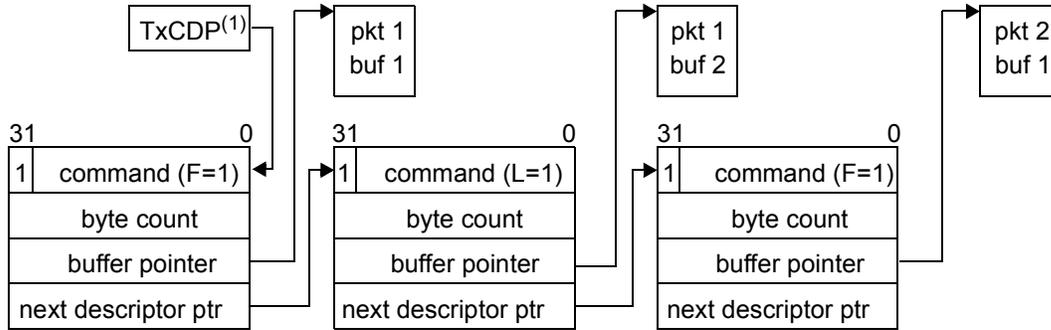
When transmission is completed, status is written to the first longword of the last descriptor. The Next Descriptor's address, which belongs to the next packet in the queue, is written to the current descriptor pointer register.

This process (starting with DMA arbitration) is repeated as long as there are packets pending in the transmit queues.

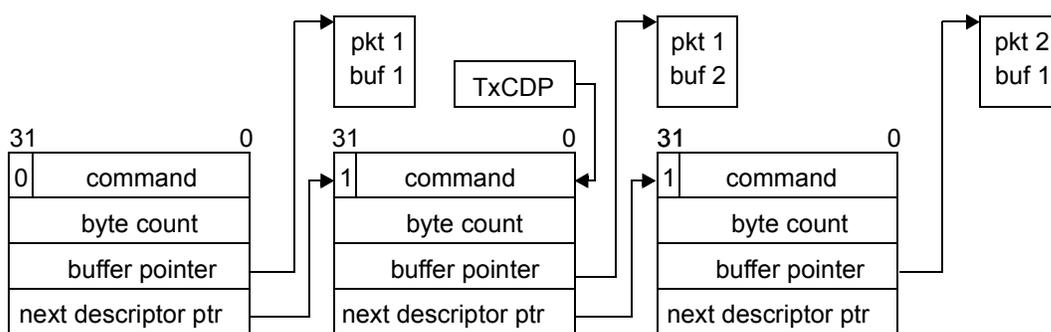
[Figure 39](#) shows how the TX descriptors are managed when a two buffers packet is transmitted.

Figure 39: Ethernet Packet Transmission Example

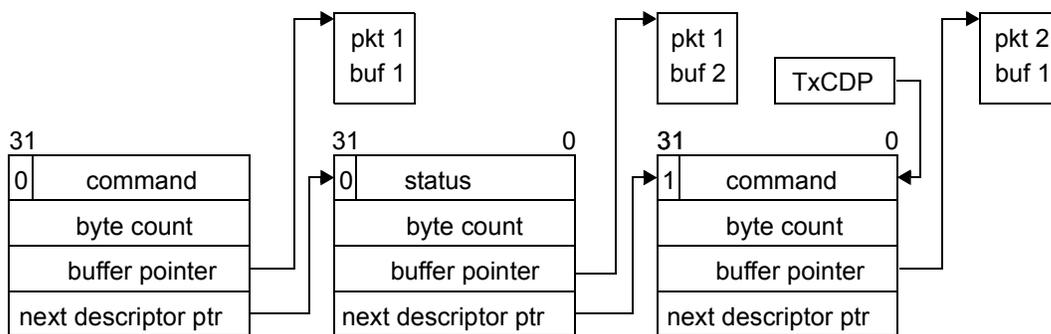
1. Packet 1 - transmitting 1st buffer



2. Packet 1 - transmitting 2nd buffer



3. Packet 2 - transmitting 1st buffer



1. TxCDP = Transmit Current Descriptor Pointer

Ownership of any descriptor other than the last is returned to CPU upon completion of data transfer from the buffer pointed by that descriptor. The Last descriptor, however, is returned to CPU ownership only after the actual transmission of the packet is completed. While changing the ownership bit of the last descriptor, the DMA also writes status information, which indicates any errors that might have happened during transmission of this packet.

12.3.2.1 Retransmission (Collision)

Full collision support is integrated into the Ethernet port for half duplex operation mode.

In half duplex operation mode, a collision event is indicated each time receive and transmit are active simultaneously. When that happens, active transmission is stopped, jam pattern is transmitted and collision count for the packet increments. The packet is retransmitted after a waiting period, which conforms to the binary Backoff algorithm specified in the IEEE 802.3 standard. Retransmit process continues for multiple collision events as long as a specified limit is not reached. This retransmit limit, which sets the maximum number of transmit retries for a single packet, is defined by the IEEE 802.3 as 16. However, the user can program a different value (see [Table 296](#) for more details). The event of a single packet colliding 16 times is known as **EXCESSIVE COLLISION**.

As long as a packet is being retransmitted, its last descriptor is kept under port ownership. When a successful transmission takes place (i.e. no collision), a status word containing collision information is written to the last descriptor and ownership is returned to CPU.

If a retransmit limit is reached with no successful transmission, a status word with error indication is written to the packet's last descriptor, and the transmit process continues with the next packet.

It is important to note that collision is considered legal only if it happens before transmitting the 65th byte of a packet. Any collision event that happens outside the first 64 byte window is known as **LATE COLLISION**, and is considered a fatal network error. Late collision is reported to the CPU through packet status, and no retransmission is done.

NOTE: A collision occurring during the transmission of the transmit packet's last two bytes are not detected.

12.3.2.2 Zero Padding (for short packets)

Zero Padding is a term used to denote the operation of adding zero bytes to a packet. This feature is used for CPU off-loading.

The Ethernet port offers a per packet padding request bit in the TX descriptor. This causes the port logic to enlarge packets shorter than 64 bytes by appending zero bytes. When this feature is used, only packets equal or larger than 64 bytes are transmitted as is. Packets smaller than 64 bytes are zero padded and transmitted as 64 byte packets.

12.3.2.3 CRC Generation

Ethernet CRC denotes four bytes of Frame-Check-Sequence appended to each packet.

CRC logic is integrated into the port and can be used to automatically generate and append CRC to a transmitted packet. One bit in the TX descriptor is used for specifying if CRC generation is required for a specific packet.

Table 273: Ethernet TX Descriptor - Command/Status word

Bits	Name	Description
31	O	Ownership bit When set to '1', the buffer is "owned" by the device. When set to '0', the buffer is owned by the CPU. Buffers owned by the CPU are not processed by the DMA.
30	AM	Auto Mode When set, the DMA does not clear the Ownership bit at the end of buffer processing.
29:24		Reserved.
23	EI	Enable Interrupt The device generates a maskable TxBuffer interrupt upon closing the descriptor. NOTE: In order to limit the number of interrupts and prevent an interrupt per buffer situation, the user should set this bit only in descriptors associated with LAST buffers. If this is done, TxBuffer interrupt will be set only when transmission of a frame is completed.
22	GC	Generate CRC When set, CRC is generated and appended to this packet. NOTE: Valid only if L (bit 16) is set.
21:19		Reserved.
18	P	Padding When this bit is set, zero bytes are appended to the packet if the packet is smaller than 60 bytes. Use this feature to prevent transmission of fragments. NOTE: Valid only if L (bit 16) is set.
17	F	First Indicates first buffer of a packet.
16	L	Last Indicates last buffer of a packet.
15	ES	Error Summary ES = LC or UR or RL Set by the device to indicate an error event that occurred during packet the packet. NOTE: Valid only if L (bit 16) is set.
14		Reserved.
13:10	RC[3:0]	Retransmit Count. Indicates actual number of retransmits for this packet. RC is valid only if L (bit 16) is set.
9	COL	Collision When set, indicates that at least one collision event occurred during transmission of the packet. NOTE: Valid only if L (bit 16) is set.

Table 273: Ethernet TX Descriptor - Command/Status word (Continued)

Bits	Name	Description
8	RL	Retransmit Limit (Excessive Collision) error Indicates that retransmit count reached the limit specified in the DMA configuration register, see Table 296 . NOTE: Valid only if L (bit 16) is set.
7		Reserved.
6	UR	Under-Run error Indicates that part of the packet's data was not available while transmission was in progress, probably due to memory access delays. NOTE: Valid only if L (bit 16) is set.
5	LC	Late Collision error Collision occurred outside the collision window (i.e. more than 512 bits were transmitted before collision assertion). NOTE: Valid only if L (bit 16) is set.
4:0		Reserved

Table 274: Ethernet TX Descriptor - Byte Count

Bits	Name	Description
31:16	Byte Count	Number of bytes to be transmitted from associated buffer. This is the payload size in bytes.
15:0		Reserved.

Table 275: Ethernet TX Descriptor - Buffer Pointer

Bits	Name	Description
31:0	Buffer Pointer	32-bit pointer to the beginning of the buffer associated with this descriptor. NOTE: The alignment restrictions for buffers that have Byte-Count smaller than 8 bytes (see Figure 41 on page 259).

Table 276: Ethernet TX Descriptor - Next Descriptor Pointer

Bits	Name	Description
31:0	Next Descriptor Pointer	32-bit pointer that points to the beginning of next descriptor. Bits [3:0] must be set to 0. DMA operation is stopped when a NULL (all zero) value in the Next Descriptor Pointer field is encountered.

12.3.2.5 TX DMA Pointer Registers

The TX DMA employs a single 32-bit pointer register per queue: TxCDP.

- TxCDP - TX DMA Current Descriptor Pointer.

TxCDP is a 32-bit register used to point to the current descriptor of a transmit packet. The CPU must initialize this register before enabling DMA operation. The value used for initialization should be the address of the first descriptor to use.

12.3.2.6 TX DMA Notes

Transmit DMA process is packet oriented. The transmit DMA does not close the last descriptor of a packet, until the packet has been fully transmitted. When closing the last descriptor, the DMA writes packet transmission status to the Command/Status word and resets the ownership bit. A TxBuffer maskable interrupt is generated if the EI bit in the last descriptor is set.

Transmit DMA stops processing a TX queue whenever a descriptor with a NULL value in the Next Descriptor Pointer field is reached or when a CPU owned descriptor is fetched. When that happens, a Tx_End maskable interrupt is generated. In order to restart the queue, the CPU should issue a Start_Tx command by writing '1' to the Start_Tx bit in the DMA command register.¹

The transmit DMA does not expect a NULL Next Descriptor Pointer or a CPU owned descriptor in the middle of a packet. When that happens, the DMA aborts transmission and stops queue processing. A TX_Resource_Error maskable interrupt is generated. In order to restart the queue, the CPU should issue a Start_Tx command.

A transmit underrun occurs when the DMA can not access the memory fast enough and packet data is not transferred to the FIFO before the FIFO gets empty. In this case, the DMA aborts transmission and closes the last descriptor with a UR bit set in the status word. Also, a Tx_Underrun maskable interrupt is generated. Transmit process continues with the next packet.

In order to stop DMA operation before the DMA reaches the end of descriptor chain, the CPU should issue a STOP command by writing '1' to the Stop_Tx bit in the DMA command register. The DMA stops queue processing as soon as the current packet transmission is completed and its last descriptor returned to CPU ownership. In addition, a Tx_End maskable interrupt is generated. In order to restart this queue, the CPU should issue a Start_Tx command.

NOTE: Most of the terms used to denote either DMA commands (Start_Tx and Stop_Tx) or interrupts (TxBuffer, Tx_End, TX_Resource_Error) actually reflect multiple terms (one per queue). For example, the GT-96100A provides two Start_Tx commands. There is a separate Start_Tx_High command, associated with the high priority queue, and a Start_Tx_low command that is related to the low priority queue. The same applies to the other commands and interrupts listed above.

1. When the DMA stops due to NULL descriptor pointer, the CPU has to write TxCDP before issuing a Start_Tx command. Otherwise, TxCDP remains NULL and the DMA can not restart queue processing.

12.3.3 Receive Operation

In order to initialize a receive operation, the CPU must do the following:

1. Prepare a chained list of descriptors and packet buffers.

NOTE: The RxDMA supports four priority queues. If the user wants to take advantage of this capability, a separate list of descriptors and buffers should be prepared for each of the priority queues.

2. Write the pointer to the first descriptor to the DMA's first and current descriptor registers (RxFDP, RxCDP) associated with the priority queue to be started. If multiple priority queues are needed, the user has to initialize TxFDP and TxCDP for each queue.
3. Initialize and enable the Ethernet port by writing to the port's configuration and command registers.
4. Initialize and enable the DMA channel by writing to the DMA's configuration and command registers.

After completing these steps, the port starts waiting for a receive frame to arrive at the MII interface. When this occurs, receive data is packed and transferred to the RxFiFO. At the same time, address filtering test is done in order to decide if the packet is destined to this port. If the packet passes address filtering check, a decision is made regarding the destination queue to which this packet should be transferred. When this is done, actual data transfer to memory takes place.

NOTE: Packets which fail address filtering are dropped and not transferred to memory.

For packets that span more than one buffer in memory, the DMA will fetch new descriptors as necessary. However, the first descriptor pointer will not be changed until packet reception is done.

When reception is completed, status is written to the first longword of the first descriptor, and the Next Descriptor's address is written to both first and current descriptor pointer registers. This process is repeated for each received packet.

NOTES: The RxCDP and RxFDP point to the same descriptor whenever the DMA is ready for receiving a new packet. RxFDP is not modified during packet reception and points to the first descriptor. Only after the packet had been fully received and status information was written to the first LW of the first descriptor, will the ownership bit be reset (i.e. descriptor returned to CPU ownership).

Ownership of any descriptor other than the first is returned to CPU upon completion of data transfer to the buffer pointed by that descriptor. This means that the first descriptor of a packet is the last descriptor to return to CPU ownership (per packet).

12.3.3.1 RX DMA Descriptors

Figure 42 shows the format of RX DMA descriptors.

The following set of restrictions apply to RX descriptors:

- Descriptor length is 4LW and it must be 4LW aligned (i.e. Descriptor_Address[3:0]=0000).
- Descriptors reside anywhere in the CPU address space except NULL address, which is used to indicate end of descriptor chain.
- RX buffers associated with RX descriptors are limited to 64K bytes and must be 64-bit aligned. Minimum size for RX buffers is 8 bytes.

Table 277: Ethernet RX Descriptor - Command/Status word (Continued)

Bits	Name	Description
11	FT	<p>Frame Type</p> <ul style="list-style-type: none"> • 1 - 802.3 • 0 - Ethernet <p>Set to '1' when the Type/Length field in the received packet has a value not bigger than 1500 (decimal).</p> <p>NOTE: Valid only if F (bit 17) is set.</p>
12	M	<p>Missed Frame</p> <ul style="list-style-type: none"> • 0 - Match • 1 - Miss <p>Set to indicate that this packet's destination address is not found in the address table. This bit may be set if HDM or PM are set in the Port Configuration register (see Table 288).</p> <p>Also, set to receive broadcast packets regardless of the HDM or PM settings in the Port Configuration register.</p> <p>NOTE: This bit is valid only if F (bit 17) is set.</p>
13	HE	<p>Hash Table Expired</p> <p>Set to indicate that hash process was not completed in time. This means there is no definite answer as to whether this packet's address is in the hash table or not.</p> <p>Also, set when there is no room in the table for this address.</p> <p>NOTE: Valid only if F (bit 17) is set.</p>
14	IGMP	<p>Set to indicate that this packet has been identified as an IGMP packet.</p> <p>NOTE: Valid only if F (bit 17) is set.</p>
15	ES	<p>Error Summary</p> <p>ES = CE or COL or LC or OR or MFL or SF</p> <p>NOTE: Valid only if F (bit 17) is set.</p>
16	L	<p>Last</p> <p>Indicates last buffer of a packet.</p>
17	F	<p>First</p> <p>Indicates first buffer of a packet.</p>
22:18		Reserved.
23	EI	<p>Enable Interrupt</p> <p>The device generates a maskable interrupt upon closing the descriptor.</p> <p>NOTE: In order to limit the number of interrupts and prevent an interrupt per buffer situation, the user should set the EI bits in all the Rx descriptors and set RIFB bit in the DMA Configuration register (see Table 296). The RxBuffer interrupt is set only on frame (rather than buffer) boundaries.</p>
29:24		Reserved.
30	AM	<p>Auto Mode</p> <p>When set, the DMA does not clear the Ownership bit at the end of buffer processing.</p>

Table 277: Ethernet RX Descriptor - Command/Status word (Continued)

Bits	Name	Description
31	O	Ownership bit. When set to '1', the buffer is "owned" by the device. When set to '0', the buffer is owned by CPU.

Table 278: Ethernet RX Descriptor - Buffer Size / Byte Count

Bits	Name	Description
15:0	Byte Count	When the descriptor is closed this field is written by the device with a value indicating number of bytes actually written by the DMA into the buffer.
31:16	Buffer Size	Buffer Size in Bytes When number of bytes written to this buffer is equal to Buffer Size value, the DMA closes the descriptor and moves to the next descriptor. NOTE: Bits [18:16] must be set to 0.

Table 279: Ethernet RX Descriptor - Buffer Pointer

Bits	Name	Description
31:0	Buffer Pointer	32-bit Pointer to The Beginning of the Buffer Associated with The Descriptor RX buffers have to be 64-bit aligned, so bits [2:0] must be set to 0.

Table 280: Ethernet RX Descriptor - Next Descriptor Pointer

Bits	Name	Description
31:0	Next Descriptor Pointer	32-bit Next Descriptor Pointer to the Beginning of Next Descriptor Bits [3:0] must be set to 0. DMA operation is stopped when a NULL value in the Next Descriptor Pointer field is encountered.

12.3.3.2 RX DMA Pointer Registers

The RX DMA employs two 32-bit pointer registers per queue: Rx FDP and Rx CDP.

- Rx FDP - RX DMA First Descriptor Pointer.

Rx FDP is a 32-bit register used to point to the first descriptor of a receive packet. The CPU must initialize this register before enabling DMA operation. The value used for initialization should be the address of the first descriptor to use.

- Rx CDP - RX DMA Current Descriptor Pointer.

Rx CDP is a 32-bit register used to point to the current descriptor of a receive packet. The CPU must initialize this register before enabling DMA operation. The value used for initialization should be the same as the value used for initializing Rx FDP (i.e. address of first descriptor to use).

12.3.3.3 Type of Service Queueing

The Type of Service queuing algorithm is based on the decoding of the DSCP field from the IP header. The DSCP field is located in the 6-MSB bits of the second byte in the IP header (See [Figure 43](#)). This field indexes the 64 IPT Table entries, which reside in the GT-96100A Ethernet register space. The 2-bit priority output of this table is referred to in the algorithm as **tos_priority**.

The **tos_priority** is valid only if the **tos2prio** enable bit 21 in the Ethernet Port Configuration Extend register, referred to in the algorithm as **tos2prio_en**, is set.

If a VLAN tag exists in the packet, the VLAN priority tag is decoded from the 3-MSB bits of the 2nd word in the VLAN tag. This field is the index to the 8 entries in the VPT Table, which reside in the GT-96100A Ethernet register space. The 2-bit priority output of this table is referred to in the algorithm as **vlan_priority**.

The GT-96100A can decode BPDU and IGMP protocol packets. These packets are referred to in the algorithm as **frame_bpdu** and **frame_igmp** respectively. Protocol detection is controlled by the SPAN and IGMP bits in the Ethernet Port Configuration Extend register, referred to in the algorithm as **bpdu_capture** and **igmp_capture** respectively.

BPDU and IGMP protocol packets are sent to the highest value queue unless protocol detection is turned off.

The PRIORx Override bit in the Ethernet Port Configuration Extend register, referred to in the algorithm as **override_priority**, takes precedence over **tos_priority** or **vlan_priority**. If this bit is set, ALL packets (except **frame_bpdu** and **frame_igmp**) are sent to the **default_priority** queue. The algorithm notation for the PRIORx 2-bit field in the Ethernet Port Configuration Extend register is **default_priority**.

The packet type is checked after checking the source address, VLAN tag (if it exists), and LLC-SNAP (if it exists). The packet type is compared to 0x8100, referred to in the algorithm as **vlan_type**, or to 0x800, referred to in the algorithm as **ip_type**. If **vlan_type** or **ip_type** with VALID **tos_priority**, or both, are found on the packet, the packet is referred to in the algorithm as **frame_tagged**.

Broadcast packets, which are referred to in the algorithm as **frame_broadcast** and are not marked as **frame_tagged** are also sent to the **default_priority** queue.

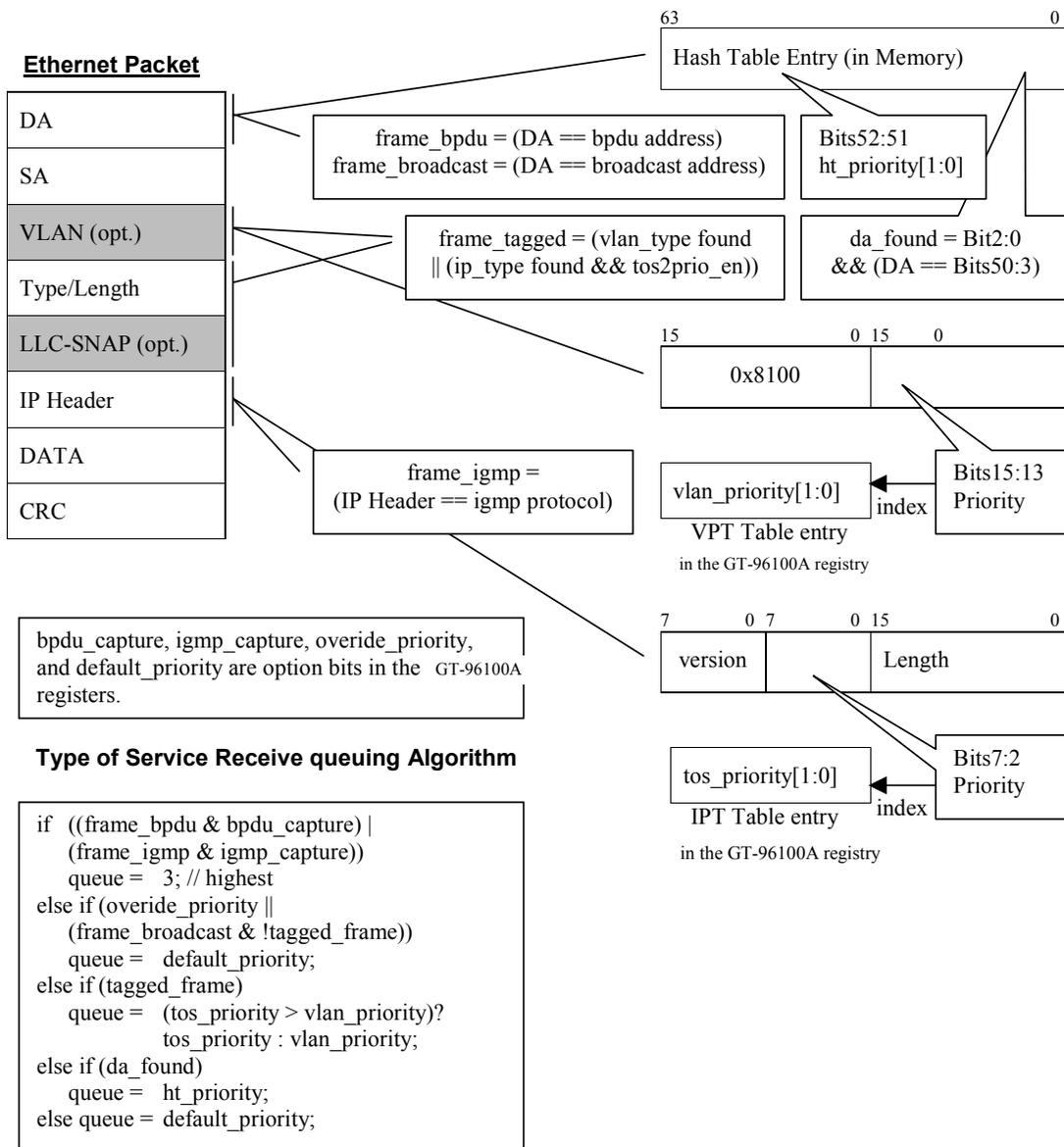
If the packet is marked as **frame_tagged**, the GT-96100A sends the packet to the **tos_priority** queue or **vlan_priority** queue. If both **tos_priority** and **vlan_priority** are extracted from the packet, the GT-96100A sends the packet to the higher value queue.

If both **tos_priority** and **vlan_priority** are missing from the packet, the GT-96100A uses the priority value found in the matched Hash Table entry. The Hash Table entry match, referred to in the algorithm as **da_found**, occurs when the destination address matches the entry's address and the entry is valid.

The 2-bit priority value, referred to in the algorithm as **ht_priority**, is located on bits 52:51 of the Hash Table entry. The address to be compared is located on bits 50:3 of the Hash Table entry. The validity of the entry is stated in bits 2:0.

When the Hash Table entry does not return a priority value, the packet is sent to the **default_priority** queue.

Figure 43: Type of Service Queuing Algorithm



12.3.3.4 RX DMA notes

The Receive DMA process is packet oriented. The DMA does not close the first descriptor of a packet, until the last descriptor of the packet is closed. When closing the first descriptor, the DMA writes status to the Command/Status word and resets the ownership bit. A RxBuffer maskable interrupt is generated if the EI bit in the first descriptor is set.

The receive DMA never expects a NULL next descriptor pointer or a CPU owned descriptor during normal operation. It is assumed that whenever the receive DMA needs a buffer, a buffer is ready for it. If this is not the case, the RxDMA engine stops serving the current priority queue and a Rx_resource_error maskable interrupt is generated. To resume operation of the stopped queue, the following must be performed:

1. Read the RxCDP associated with the stopped queue.
2. If RxCDP is not NULL, it means that the error is due to a CPU owned descriptor. In this case, flip the ownership bit of the descriptor pointed by RxCDP.
3. If RxCDP is NULL, it means that the error is due to a NULL descriptor pointer. In this case, re-initialize the queue by writing a valid pointer to both RxCDP and RxFDP.

Stopping RX DMA operation is possible using the RX_ABORT command (see [Table 297](#)).

12.3.4 Ethernet Address Recognition

The following chapter describes the Hash algorithm and Hash table data structure. The CPU must build this table for the GT-96100A before enabling the Ethernet port.

12.3.4.1 Hash Table Structure

The GT-96100A Hash table is a data structure prepared by the CPU and resides in the system DRAM. Its location is identified by a 32 bit pointer stored in the GT-96100A EHTP internal register (addresses 0x84828 and 0x88828). The Hash table must be octet-byte aligned. The lowest three bits of the EHTP register are hard wired to '0'.

There are two possible sizes for the Hash table. Table size is selected by the HS bit in the Ethernet Configuration Register (PCR, address 0x84800 and 0x88800).

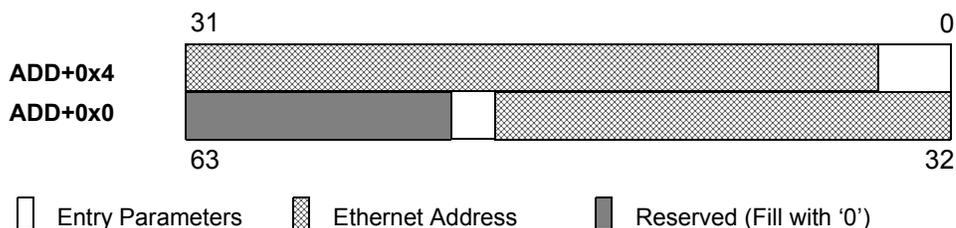
- 8K address table. 256KByte of DRAM required (4 x 64KByte banks)
- 1/2K address table. 16KByte of DRAM required (4 x 4KByte banks)

A multiple of 4 banks are used in order to reduce the number of addresses that are mapped to the same table entry.

NOTE: The user must initialize the Hash table before enabling the Ethernet Controller.

Each Address entry is a two word data field (64 bits) as shown below:

Figure 44: Ethernet Hash Table Entry



The following table describes the Hash table entry fields.

Table 281: Hash Table Entry Fields

Bit	Command	Usage
0	Valid	Indicates Valid Entry
1	Skip	Skip empty entry in a chain
2	Receive/Discard (RD)	0 - Discard packet upon match 1 - Receive packet upon match
6:3	Ethernet Address[3:0]	Mapped to Ethernet MAC address[43:40].
11:7	Ethernet Address[7:4]	Mapped to Ethernet MAC address[47:44].
14:11	Ethernet Address[11:8]	Mapped to Ethernet MAC address[35:32].
18:15	Ethernet Address[15:12]	Mapped to Ethernet MAC address[39:36].
22:19	Ethernet Address[19:16]	Mapped to Ethernet MAC address[27:24].
26:23	Ethernet Address[23:20]	Mapped to Ethernet MAC address[31:28].
30:27	Ethernet Address[27:24]	Mapped to Ethernet MAC address[19:16].
34:31	Ethernet Address[31:28]	Mapped to Ethernet MAC address[23:20].
38:35	Ethernet Address[35:32]	Mapped to Ethernet MAC address[11:8].
42:39	Ethernet Address[39:36]	Mapped to Ethernet MAC address[15:12].
46:43	Ethernet Address[43:40]	Mapped to Ethernet MAC address[3:0].
50:47	Ethernet Address[47:44]	Mapped to Ethernet MAC address[7:4].
52:51	Priority	The priority queue of a packet sent to this Ethernet address, in case there are no other priority signals (i.e. ToS or VLAN).
63:53	Reserved	Fill With '0'

12.3.4.2 Hash Modes

There are two Hash functions in the GT-96100A; Hash Mode 1 and Hash Mode 0.

12.3.4.3 Hash Mode 0

In Hash mode 0, the Hash entry address is calculated in the following manner:

$\text{hashResult}[14:0] = \text{hashFunc0}(\text{ethernetADD}[47:0])$

- hashResult is the 15 bits Hash entry address.
- ethernetADD is a 48 bit number, which is derived from the Ethernet MAC address, by nibble swapping in every byte; i.e. MAC address of 0x123456789abc translates to ethernetADD of 0x21436587a9cb.
- inverse every nibble; i.e. ethernetADD of 0x21436587a9cb translates to 0x482c6a1e59d

hashFunc0 calculates the hashResult in the following manner:

- $\text{hashResult}[14:9] = \text{ethernetADD}[7:2]$
- $\text{hashResult}[8:0] = \text{ethernetADD}[14:8,1,0] \text{ XOR } \text{ethernetADD}[23:15] \text{ XOR } \text{ethernetADD}[32:24]$

12.3.4.4 Hash Mode 1

In Hash mode 1, the Hash entry address is calculated in the following manner:

$\text{hashResult}[14:0] = \text{hashFunc1}(\text{ethernetADD}[47:0])$

- hashResult is the 15 bits Hash entry address.
- ethernetADD is a 48 bit number, which is derived from the Ethernet MAC address, by nibble swapping in every byte (i.e. MAC address of 0x123456789abc translates to ethernetADD of 0x21436587a9cb).
- inverse every nibble; i.e. ethernetADD of 0x21436587a9cb translates to 0x482c6a1e59d

hashFunc1 calculates the hashResult in the following manner:

- $\text{hashResult}[14:9] = \text{ethernetADD}[0:5]$
- $\text{hashResult}[8:0] = \text{ethernetADD}[6:14] \text{ XOR } \text{ethernetADD}[15:23] \text{ XOR } \text{ethernetADD}[24:32]$

12.3.4.5 Hash Entry

For each Ethernet address, the Hash table entry address is the lower 13 bits of the hashResult for the 8KByte address table, or the lower 9 bits for the 0.5KByte address table. The entry is an offset from the address base and is octet-byte aligned. The address entry is therefore:

- 8K Address Table: $\text{tblEntryAdd} = \text{EHTP} + \{\text{hashResult}[14:0],000\}$
- 1/2K Address Table: $\text{tblEntryAdd} = \text{EHTP} + \{\text{hashResult}[10:0],000\}$

12.3.4.6 Hash Table Numbers

12.3.4.7 Table Filling

When preparing the Hash table data structure, the CPU must first (typically at boot time) initialize the Hash table memory to '0'.

The table filling algorithm is described below. The hopNumber should be selected and initialized before entering this routine. The Hash table hopNumber (Number of Hops) is 12. After 12 tries to identify an address, the GT-96100A passes the address to the CPU and sets the HE (Hash Expired) bit in the descriptor status field. Therefore, the hopNumber is the number of times the CPU will attempt to write a newly learned Ethernet address into the Hash table.

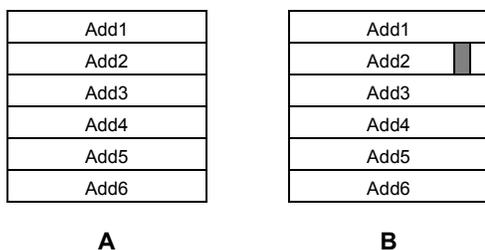
- Calculate tblEntryAdd according to mode of operation (Hash Mode 1 or Hash Mode 0).
- Check that tblEntry is empty (Valid Bit is "0").
- If the tblEntry is empty, Write the hashEntry (Valid, Skip and RD bits and Ethernet Address).
- If the tblEntry is occupied (i.e. Valid bit is 1 and Skip bit is 0), move to tblEntry+1.
- If less than hopNumber tries, Repeat to Step c.

If after hopNumber failed tries, the CPU has been unable to located a free table entry. The CPU can then:

- Defragment the table.
- Create a new Hash table using the alternate Hash Mode, which may redistribute the addresses more evenly in the table.

In cases where more than one address is mapped to the same table entry, an address chain is created. In this case, when the CPU needs to erase an address that is part of an address chain, it cannot clear its Valid bit since this would cut the chain. Instead, the CPU should set the Skip bit to '1'. This is shown in [Table 45](#).

Figure 45: Address Chain



In case A where Add1-6 has the same Hash function, and thus start with the same tblEntry, the CPU allocates them in the table by increasing tblEntry by one entry each time. Add1 is the first address to be written into the table and Add6 is the last.

When the CPU is required to remove Add2 from the table, it cannot clear its valid bit since that would break the chain from Add1 to Add3. Instead, it sets Add2's Skip bit to '1' (denoted as ). It is also recommended that the CPU defragments the table from time to time.

12.3.4.8 Address Recognition Process

The following terms are used when referring to the address recognition process.

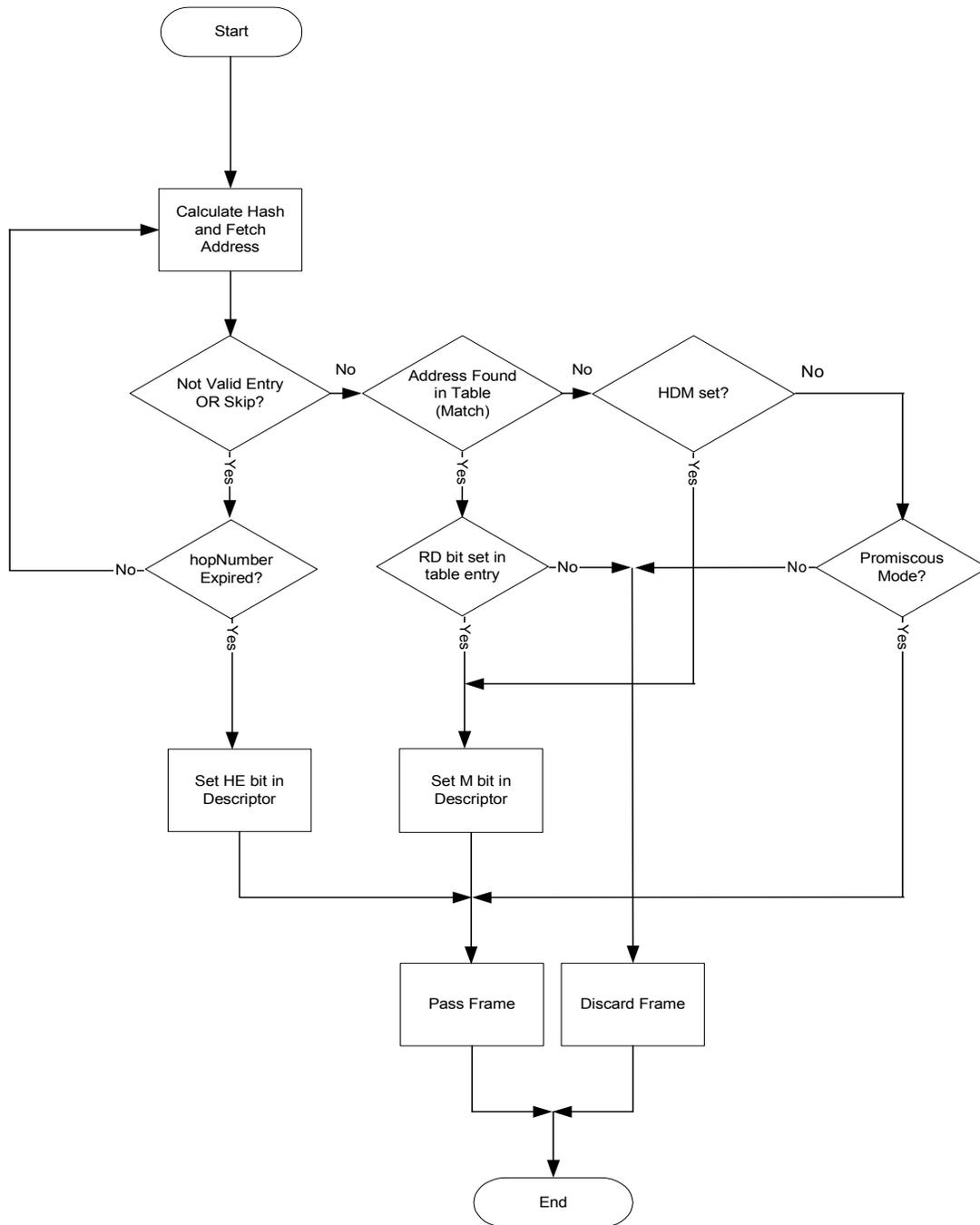
- **Match** - Address is found in the table
- **Miss** - Address is not found in the table
- **Hit** - Address is in the table and RD bit is 1 (receive), or address is not in table and HDM (Hash Default Mode) is 1 (receive).
- **Occupied Entry** - A valid Hash table entry that is occupied by another address, or an entry that has its Skip bit set,
- **Promiscuous Mode** - When enabled, all packets are passed to the CPU. The GT-96100A still executes the Hash process reporting to the CPU, regardless whether the address is in the Hash table or not.

The GT-96100A address recognition process is described below, and is illustrated by [Figure 46 on page 274](#).

The process starts with the GT-96100A fetching the address from the calculated table entry.

- If Occupied Entry is encountered, the GT-96100A proceeds to the next Hash table entry.
- After hopNumber failed tries, the GT-96100A passes the packet to the CPU and marks it by setting the HE bit in the descriptor. The same process is used in case the Discard Window is over, or the frame ends before the GT-96100A accomplishes the Hash process (which happens in rare situations when the GT-96100A cannot gain enough access to the DRAM).
- When the GT-96100A finds the address in the table there is a Match.
- When the GT-96100A encounters an empty entry, there is a Miss, meaning that the address is not in the table.
- In case of a Match, and if the RD bit is set, then there is a Hit. The GT-96100A marks the packet by setting the M bit of the receive descriptor.
- In case of a Miss, and if the HDM bit is set, then there is also a Hit. The GT-96100A marks the packet by setting the M bit of the receive descriptor.
- If there was no Hit, then the packet should be discarded. However, packets will pass if Promiscuous Mode is enabled.

Figure 46: Address Filtering Process



The GT-96100A uses the HE (Hash Expired) and M (Match) bits in the descriptor for reporting the packet filtering status. [Table 282](#) describes the various reports and summarizes their meaning.

Table 282: Packet Filtering Status

HE	M	Condition
0	0	Hash Table No Hit The address was not found in the Hash table, but Promiscuous Mode is enabled
0	1	Hash Table Hit Either by an address found in the Hash table and RD bit set OR by an address that was not found in the Hash table, in case that HDM bit is set
1	0	Hash Table Expired The hopNumber expired before the address was found in the Hash table
1	1	UnUsed

12.4 Ethernet Port

12.4.1 Network Interface

The Ethernet port interfaces directly to a MII (Media Independent Interface) PHY compliant with the IEEE standard (please refer to IEEE 802.3u Fast Ethernet standard for detailed interface and timing information). The MII port has the following characteristics:

- Capable of supporting both 10 Mbps and 100 Mbps data rates in half or full duplex modes.
- Data and delimiters are synchronous to clock references.
- Provides independent 4-bit wide transmit and receive paths.
- Uses TTL signal levels.
- Provides a simple management interface (common to all ports).
- Capable of driving a limited length of shielded cable.

The port incorporates all the required digital circuitry to interface with a 100BaseTX, 100BaseT4, and 100BaseFX MII PHYs.

12.4.1.1 10/100 MII/RMII Compatible Interface

The port's MAC (Media Access Control) logic supports connection to a 10Mbps or 100Mbps network.

The MII interface consists of a separate nibble-wide stream for both transmit and receive data. Data transfers are clocked by the 25 MHz transmit and receive clocks in 100 Mbps operation, or by 2.5 MHz transmit and receive clocks in 10 Mbps operation. The clock inputs are driven by the PHY, which controls the clock rate according to the network connection speed.

The RMII interface consists of a separate 2bit-wide stream for both transmit and receive data. Data transfers are clocked by the 50 MHz clock in both 100 Mbps and 10 Mbps operation. The clock input is driven by an external source.

12.4.1.2 Media Access Control (MAC)

The MAC logic performs all of the functions of the 802.3 protocol such as frame formatting, frame stripping, collision handling, deferral to link traffic, etc. It also ensures that any outgoing packet complies with the 802.3 specification in terms of preamble structure - 56 preamble bits are transmitted before Start of Frame Delimiter (SFD).

The MAC operates in half duplex or full duplex modes. In half duplex mode, the MAC's transmit logic checks that there is no competitor for the network media before transmission.

In addition to waiting for idle before transmitting, the port handles collisions in a predetermined way. If two nodes attempt to transmit at the same time, the signals collide and the data on the line is garbled. The port listens while it is transmitting, and can detect a collision. If a collision is detected, 'JAM' pattern is transmitted and retransmission is delayed for a random time period determined by the Backoff algorithm. In full-duplex mode, the port transmits unconditionally.

12.4.1.3 Auto-Negotiation for Duplex Mode

The port's duplex operation mode (either half or full duplex) can be auto-negotiated or set by the CPU.

In order to enable auto-negotiation for duplex, the CPU must set the Port_Configuration_Extend<DPLXen> bit. When auto-negotiation for duplex is enabled, the port decodes the duplex mode from the values of the PHY's Auto-Negotiation Advertisement register and Auto-Negotiation Link Partner Ability register at the end of the Auto-Negotiation process. Once the duplex mode is resolved, Port_Status<Duplex> bit is updated accordingly.

In order to resolve the duplex mode, the following operations are continuously performed:

1. Read the PHY's Auto-Negotiation Complete status as reported by the PHY bit 1.5 (Register 1, bit 5). If this bit is '0' switch to Half-Duplex mode and continue to read PHY register bit 1.5. Continue to step 2 when PHY bit 1.5 is '1', indicating that Auto-Negotiation is complete.

NOTE: Steps 2 through 6 are performed once for every transition of PHY bit 1.5 from '0' to '1'. Once PHY bit 1.5 remains '1' and PHY registers 4 and 5 have already been read, the port will continue to read PHY register 1, and monitor PHY bit 1.5. However, if after Rst* deassertion, the PHY bit 1.5 is already read as '1', steps 2 to 6 are performed at least once in order to update the port's duplex mode.

PHY bit 1.2 (Link Status) is read and latched during this same register read operation, regardless of the Auto-Negotiation status.

2. Read the Auto-Negotiation Advertisement register, PHY register 4. Continue to step 3.
3. Read the Auto-Negotiation Link Partner Ability register, PHY register 5. Continue to step 4.
4. Resolve the highest common ability of the two link partners in the following manner (according to the 802.3u Priority Resolution clause 28B.3):

if (bit 4.8 AND bit 5.8) == '1' then ability is 100BASE-TX Full Duplex

else if (bit 4.9 AND bit 5.9) == '1' then ability is 100BASE-T4 Half Duplex

else if (bit 4.7 AND bit 5.7) == '1' then ability is 100BASE-TX Half Duplex

else if (bit 4.6 AND bit 5.6) == '1' then ability is 10BASE-T Full Duplex

else ability is 10BASE-T Half Duplex;

Continue to step 5.

5. Resolve the duplex mode of the two link partners in the following manner:

```
if ((ability == "100BASE-TX Full Duplex") or (ability == "10BASE-T Full Duplex")) then
    duplex mode = FULL DUPLEX
else
    duplex mode = HALF DUPLEX;
```

Continue to step 6.

6. Update the Port_Status register by writing the correct duplex mode bit. Continue with step 1.

12.4.1.4 Auto-Negotiation for Flow Control

Flow control mode (either enabled or disabled) can be auto-negotiated or set by the CPU. In order to enable auto-negotiation for flow-control, the CPU should set Port_Configuration_Extend<FCTLen> bit.

If Port_Configuration_Extend<FCTLen>=1, then auto-negotiation is initiated in the following cases:

- After RESET.
- After link fail (phy register 1 bit 2).

NOTE: The user may force the port to implement Flow-control by disabling auto-negotiation for flow-control and programming Port_Configuration_Extend<FCTL>=1.

Auto-negotiation for flow-control is done in two stages:

1. Setting Phy advertise word to support Flow Control.
This is done by writing Phy register 4 in order to set advertise bit 10 (phy-reg4 bit 10 - Enable FC). The flow of such a cycle is:
 - Read Phy register 1. If link_status=1 and was 0 in the last cycle - continue.
 - Read Phy register 4.
 - Write Phy register 4 with bit 10 set.
2. Reading Phy Flow-Control status and determine result.
This is done by constantly reading PHY's register 4 and register 5 in order to determine if Flow-control is supported or not. Only if both link partners support FC (registers 4.10 and 5.10 are both SET), Port_Status<FCTL> is set to '1', and the port will send PAUSE packets when instructed to do so by the CPU. Otherwise, Port_Status<FCTL> is set to '0', indicating that the support for 802.3x flow-control is disabled.

12.4.1.5 Backoff Algorithm Options

The port implements the truncated exponential Backoff algorithm defined by the 802.3 standard. Aggressiveness of the Backoff algorithm used is controlled by Serial Parameters Register<Limit4> bit.

Limit4 function controls the number of consecutive packet collisions that will occur before the collision counter is reset.

When Limit4 feature is disabled, the port resets its collision counter after 16 consecutive retransmit trials and restarts the Backoff algorithm. Retransmission is done using the data already stored in the FIFO.

When Limit4 feature is enabled, the port will reset its collision counter and restart the Backoff algorithm after 4 consecutive transmit trials. This makes the port more aggressive in getting hold of the media following a collision. This may result better overall throughput in standardized tests.

12.4.1.6 Data Blinder

The data blinder field (DataBlind in the Serial_Parameters register) sets the period of time during which the port does not sense the wire before transmission (inhibit time). The default value is 32 bit times.

12.4.1.7 Inter Packet Gap (IPG)

IPG is the minimum idle time between transmission of any two successive packets from the same port. The default (from the standard) is 9.6 μ s for 10Mbps Ethernet and 960nsec for 100-Mbps Fast Ethernet. Note that the IPG can be made smaller or larger than standard definition by programming the Serial_Parameters register.

12.4.1.8 10/100 Mbps MII Transmission

When the port has a frame ready for transmission, it samples link activity indicators. If the CrS signal is inactive (no activity on the link), and the Inter-packet gap (IPG) timer had expired, frame transmission begins. The data is transmitted via pins TxD[3:0] of the transmitting port, clocked on the rising edge of TxClk. The signal TxEn is asserted at this same time. In the case of collision, the PHY asserts the CoL signal causing the port to stop transmitting the frame and append a jam pattern to the transmitted bit stream. At the end of a collided transmission, the port will back off and attempt to retransmit once the Backoff counter expires. Per the IEEE 802.3 specification, the clock to output delay must be a minimum of 0ns and a maximum of 25ns as shown in [Figure 48](#).

12.4.1.9 10/100 Mbps RMII Transmission

The port starts transmission when it has a frame ready, and Inter-packet gap (IPG) timer has expired.

If in half_duplex mode, it also samples CRS_DV indicator for no activity. The data is transmitted via pins TXD[1:0] of the transmitting port, clocked on the rising edge of REF_CLK and the signal TX_EN is asserted.

In half_duplex mode, in the case of collision (TX_EN asserted with CRS_DV), the port stops transmitting the frame and appends a jam pattern to the transmitted bit stream. At the end of a collided transmission, the port backs off and attempts to retransmit once the Backoff counter expires. As the REF_CLK frequency is 10 times the data rate in 10 Mbps, the value on TXD[1:0] shall be valid so that it may be sampled every 10th cycle. For the RMII, transmission of each octet shall be done a di-bit at a time as per the order described in the [Figure 47](#).

12.4.1.10 10/100 Mbps RMII Reception

Frame reception starts with the assertion of CRS_DV by the PHY. The port begins sampling incoming data on pins RxD[1:0] on the rising edge of REF_CLK. Reception ends when CRS_DV is deasserted by the PHY. The last di-bit sampled by the port is the data present on RxD[1:0] on the last REF_CLK rising edge in which CRS_DV is still asserted. CRS_DV is continuously asserted during reception. If an error is detected while CRS_DV is asserted, the decoded data is replaced in the receiving stream with "01" until the end of carrier activity. By replacing the data in the remainder of the frame, the CRC check is guaranteed to reject the packet as an error. When no reception takes place, CRS_DV should remain de-asserted. As the REF_CLK frequency is 10 times the data rate in 10 Mbps, the value of each octet shall be valid so that it may be sampled every 10th cycle. For the RMII, reception of each octet shall be done a di-bit at a time as per the order described in [Figure 47](#).

The RMII transmission and reception of each octet is described in [Figure 47](#).

Figure 47: RMII Di-Bit Stream

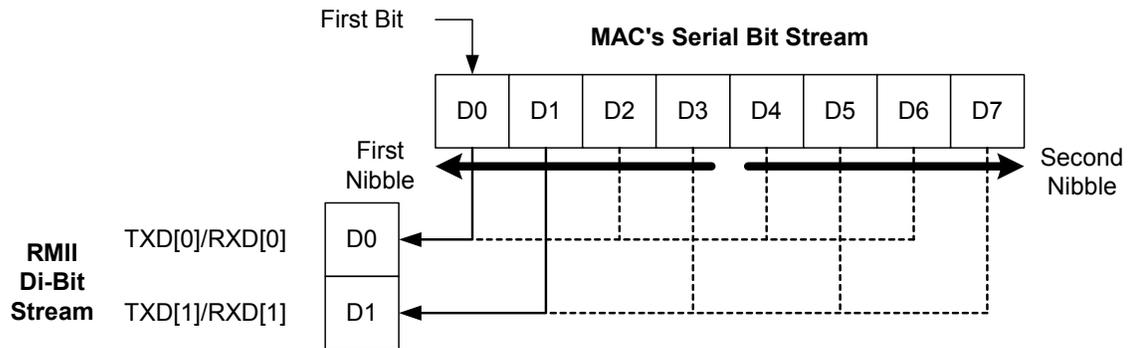
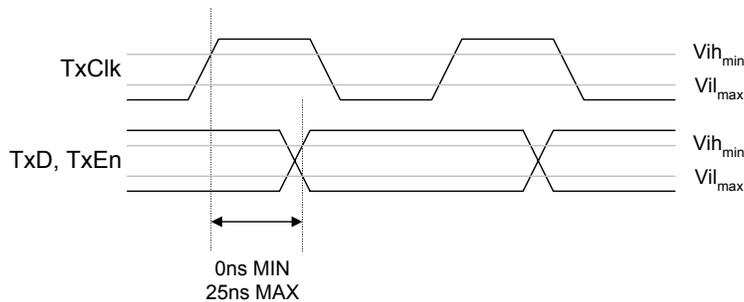


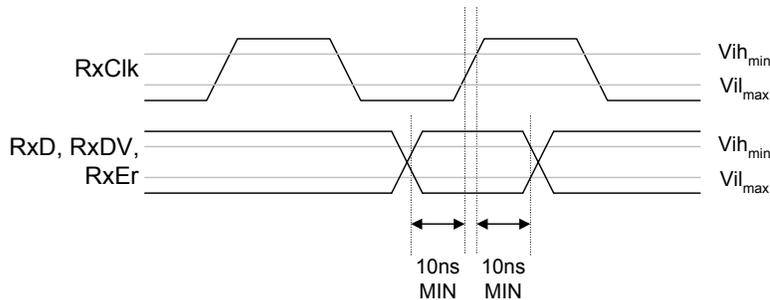
Figure 48: MII Transmit Signal Timing



12.4.1.11 10/100 Mbps MII Reception

Frame reception starts with the assertion of CrS (while the port is not transmitting) by the PHY.

Once RxDV is asserted, the port begins sampling incoming data on pins RxD[3:0] on the rising edge of RxClk. Reception ends when RxDV is deasserted by the PHY. The last nibble sampled by the port is the nibble present on RxD[3:0] on the last RxClk rising edge in which RxDV is still asserted. During reception RxDV is continuously asserted. If, while RxDV is asserted, RxEr is asserted, it designates current packet as corrupted. When no reception takes place, RxDV should remain deasserted. The input setup time should be a minimum of 10ns and the input hold time must be a minimum of 10ns and shown in [Figure 49](#).

Figure 49: MII Receive Signal Timing


12.4.1.12 10/100 Mbps Full-Duplex Operation

When operating in Full-duplex mode the port can transmit and receive frames simultaneously.

In full-duplex mode, the CrS signal is associated with received frames only and has no effect on transmitted frames. The Col signal is ignored while in Full-duplex mode. Transmission starts when TxEn goes active. Transmission starts regardless of the state of CrS. Reception starts when the CrS signal is asserted indicating traffic on the receive port of the PHY.

12.4.1.13 Back Pressure

The port implements a back pressure algorithm, which is only for use when the port is operating in half duplex mode. It is enabled through Port_Command<FJ> bit.

While in backpressure mode, the port transmits a JAM pattern for a programmable period of time (JAM_LENGTH). The IPG between two consecutive JAM patterns (or between the last transmit and the first JAM) is also a programmable value (JAM_IPG). The values are set in Serial_Parameters register.

12.4.1.14 Flow Control

IEEE 802.3x flow control is enabled while in full-duplex mode. Activating this mode is done by setting the Port_Configuration_Extend<FCTL> bit or by enabling auto-negotiation for Flow-Control, see [Section 12.4.1.4 “Auto-Negotiation for Flow Control”](#) on page 277.

The port supports 802.3x flow-control (PAUSE packets, in the standard term), if it is operating in full-duplex and if Port_Configuration_Extend<FCTL>=1.

When the port receives a PAUSE packet, it does not transmit a new packet for a period of time specified in this PAUSE packet.

A received packet is recognized as flow control PAUSE, if it was received without errors and is either of the following:

- DA = 01-80-C2-00-00-01 and type=88-08 and MAC_Control_Opcode=01
- DA = (The port address) and type=88-08 and MAC_Control_Opcode=01. The 48-bit port address is in the registers Source_Address_Low, Source_Address_High. This address is also used as source address for PAUSE packets that the port generates (to DA=01-80-C2-00-00-01)

PAUSE packets are sent by the port when instructed to do so by the CPU. This is done by setting Port_Command<FJ> bit.

12.4.2 MII Serial Management Interface (SMI)

The Ethernet unit has an integrated MII Serial Management Interface (SMI) logic for controlling MII compliant PHYs. This interface consists of two signals: serial data (MDIO); and, clock (MDC).

These signals enable control and status parameters to be passed between the PHYs and the port logic (or CPU). Multiple PHY devices can be controlled using this simple 2-pin interface.

Typically, the SMI unit continuously queries the PHY devices for their link status, without the need for CPU intervention. The PHY addresses for the link query operation are programmable per port in the PHY_Address register.

A CPU can write/read to/from all PHY addresses/registers by writing and reading to/from the SMI control register. The SMI allows the CPU to directly control a MII compatible PHY device via the SMI control register. This enables the driver software to program the PHY into specific operation mode such as Full Duplex, Loopback, Power Down, 10/100 speed selection as well as control of the PHY device's Auto-Negotiation function, if it exists. The CPU writes commands to the SMI register and the SMI unit performs the actual data transfer via MDIO, which is a bi-directional data pin. These serial data transfers are clocked by the MDC clock output.

12.4.2.1 MII Management Frame Structure

The GT-96100A's SMI cycles support the MII management frame structure.

Frames transmitted on the MII management interface have a structure that is shown in [Table 283](#) and the order of bit transmission is from left to right.

Table 283: MII Management Frame Format

	PRE	ST	OP	PhyAd	RegAd	TA	Data	IDLE
READ	1...1	01	10	AAAAA	RRRRR	Z0	D...D(16)	Z
WRITE	1...1	01	01	AAAAA	RRRRR	10	D...D(16)	Z

The format of the bit transmission's parts is as follows:

Table 284: Bit Transmission Parts

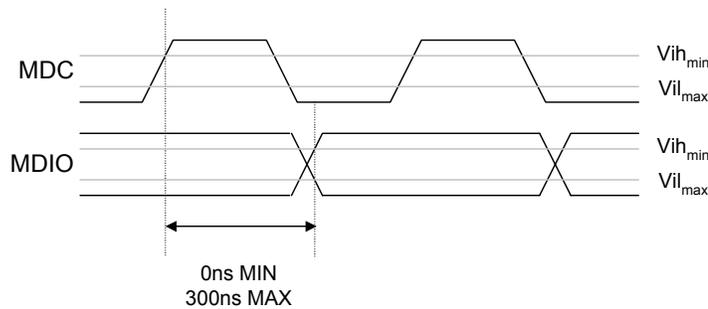
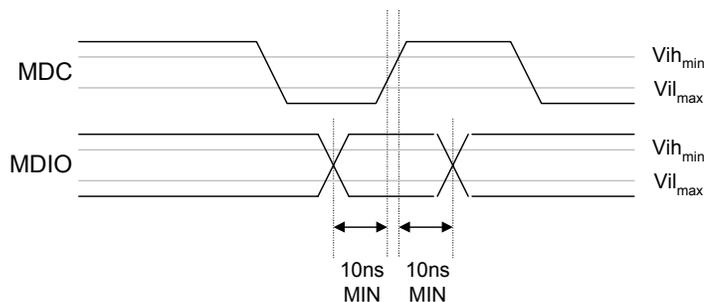
Part	Description
PRE (Preamble)	At the beginning of each transaction, the port sends a sequence of 32 contiguous logic one bits on MDIO with 32 corresponding cycles on MDC to provide the PHY with a pattern that it can use to establish synchronization.
ST (Start of Frame)	A Start of Frame pattern of 01.
OP (Operation Code)	10 - Read; 01 - Write.
PhyAd (PHY Address)	A 5 bit address of the PHY device (32 possible addresses). The first PHY address bit transmitted by the port is the MSB of the address.
RegAd (Register Address)	A 5 bit address of the PHY register (32 possible registers in each PHY). The first register address bit transmitted by the port is the MSB of the address. The port always queries the PHY device for status of the link by reading register 1 bit 2.

Table 284: Bit Transmission Parts

Part	Description
TA (Turn Around)	The turnaround time is a 2 bit time spacing between the Register Address field and the Data field of the SMI frame to avoid contention during a read transaction. During a Read transaction the PHY should not drive MDIO in the first bit time and drive '0' in the second bit time. During a write transaction, the port drives a '10' pattern to fill the TA time.
Data (Data)	The data field is 16 bits long. The PHY drives the data field during Read transactions. The port drives the data field during write transactions. The first data bit transmitted and received shall be bit 15 of the PHY register being accessed.
IDLE (Idle)	The IDLE condition on MDIO is a high impedance state. The MDIO driver is disabled and the PHY should pull-up the MDIO line to a logic one.

12.4.3 SMI Timing Requirements

When the MDIO signal is driven by the PHY, it is sampled synchronously with respect to the rising edge of MDC. Per IEEE 802.3 specification, the MDC to output delay must be a minimum of 0ns and a maximum of 300ns as shown in Figure 10. Further, when the MDIO signal is driven by the port, it has a minimum of 10ns setup time and minimum of 10ns hold time as shown in Figure 50 and Figure 51.

Figure 50: MDIO Output Delay

Figure 51: MDIO Setup and Hold Time


12.4.3.1 Link Detection and Link Detection Bypass (ForceLinkPass)

Typically, the port continuously queries the PHY devices for their link status without CPU intervention.

The PHY addresses used for the link query are determined by the PHY_Address register and are programmable for each port. The port alternately reads register 1 from the PHYs and updates the internal link bits according to the value of bit 2 of register 1. In the case of “link down” (i.e. bit 2 is ‘0’), that port will enter link test fail state.

In this state, all of the port’s logic is reset. The port exits from link test fail state only when the “link is up” (i.e. bit 2 of register 1 is read from the port’s PHY as ‘1’).

There is an option to disable the link detection mechanism by forcing the link state of a specific port. This is done by setting Port_Configuration_Extend<FLP> bit.

12.5 Internal Control Registers

Table 285: Ethernet Unit Register Map

Description	Offset	Page Numbers
Ethernet PHY Address Register (EPAR)	0x080800	page 285
Ethernet SMI Register (ESMIR)	0x080810	page 286
Ethernet0		
Ethernet0 Port Configuration Register (E0PCR)	0x084800	page 286
Ethernet0 Port Configuration Extend Register (E0PCXR)	0x084808	page 288
Ethernet0 Port Command Register (E0PCMR)	0x084810	page 291
Ethernet0 Port Status Register (E0PSR)	0x084818	page 291
Ethernet0 Serial Parameters Register (E0SPR)	0x084820	page 292
Ethernet0 Hash Table Pointer Register (E0HTPR)	0x084828	page 293
Ethernet0 Flow Control Source Address Low (E0FCSAL)	0x084830	page 293
Ethernet0 Flow Control Source Address High (E0FCSAH)	0x084838	page 294
Ethernet0 SDMA Configuration Register (E0SDCR)	0x084840	page 294
Ethernet0 SDMA Command Register (E0SDCMR)	0x084848	page 295
Ethernet0 Interrupt Cause Register (E1ICR)	0x084850	page 296
Ethernet0 Interrupt Mask Register (E0IMR)	0x084858	page 299
Ethernet0 IP Differentiated Services CodePoint to Priority0 low (E0DSCP2P0L)	0x84860	page 299
Ethernet0 IP Differentiated Services CodePoint to Priority0 high (E0DSCP2P0H)	0x84864	page 299

Table 285: Ethernet Unit Register Map (Continued)

Description	Offset	Page Numbers
<i>Ethernet0 (Continued)</i>		
Ethernet0 IP Differentiated Services CodePoint to Priority1 low (E0DSCP2P1L)	0x84868	page 299
Ethernet0 IP Differentiated Services CodePoint to Priority1 high (E0DSCP2P1H)	0x8486c	page 299
Ethernet0 VLAN Priority Tag to Priority (E0VPT2P)	0x84870	page 299
Ethernet0 First Rx Descriptor Pointer 0 (E0FRDP0)	0x084880	page 263
Ethernet0 First Rx Descriptor Pointer 1 (E0FRDP1)	0x084884	
Ethernet0 First Rx Descriptor Pointer 2 (E0FRDP2)	0x084888	
Ethernet0 First Rx Descriptor Pointer 3 (E0FRDP3)	0x08488C	
Ethernet0 Current Rx Descriptor Pointer 0 (E0CRDP0)	0x0848A0	
Ethernet0 Current Rx Descriptor Pointer 1 (E0CRDP1)	0x0848A4	
Ethernet0 Current Rx Descriptor Pointer 2 (E0CRDP2)	0x0848A8	
Ethernet0 Current Rx Descriptor Pointer 3 (E0CRDP3)	0x0848AC	
Ethernet0 Current Tx Descriptor Pointer 0 (E0CTDP0)	0x0848E0	page 255
Ethernet0 Current Tx Descriptor Pointer 1 (E0CTDP1)	0x0848E4	
Ethernet0 MIB Counters	0x085800 - 0x0858FF	page 302
<i>Ethernet1</i>		
Ethernet1 Port Configuration Register (E1PCR)	0x088800	page 286
Ethernet1 Port Configuration Extend Register (E1PCXR)	0x088808	page 288
Ethernet1 Port Command Register (E1PCMR)	0x088810	page 291
Ethernet1 Port Status Register (E1PSR)	0x088818	page 291
Ethernet1 Serial Parameters Register (E1SPR)	0x088820	page 292
Ethernet1 Hash Table Pointer Register (E1HTPR)	0x088828	page 293
Ethernet1 Flow Control Source Address Low (E1FCSAL)	0x088830	page 293
Ethernet1 Flow Control Source Address High (E1FCSAH)	0x088838	page 294
Ethernet1 SDMA Configuration Register (E1SDCR)	0x088840	page 294
Ethernet1 SDMA Command Register (E1SDCMR)	0x088848	page 295
Ethernet1 Interrupt Cause Register (E1ICR)	0x088850	page 296
Ethernet1 Interrupt Mask Register (E1IMR)	0x088858	page 299

Table 285: Ethernet Unit Register Map (Continued)

Description	Offset	Page Numbers
<i>Ethernet1 (Continued)</i>		
Ethernet IP Differentiated Services CodePoint to Priority0 low (E0DSCP2P0L)	0x88860	page 299
Ethernet IP Differentiated Services CodePoint to Priority0 high (E0DSCP2P0H)	0x88864	page 299
Ethernet IP Differentiated Services CodePoint to Priority0 low (E0DSCP2P1L)	0x88868	page 299
Ethernet IP Differentiated Services CodePoint to Priority0 high (E0DSCP2P1H)	0x8886c	page 299
Ethernet1 VLAN Priority Tag to Priority (E0VPT2P)	0x88870	page 299
Ethernet1 First Rx Descriptor Pointer 0 (E1FRDP0)	0x088880	page 264
Ethernet1 First Rx Descriptor Pointer 1 (E1FRDP1)	0x088884	page 266
Ethernet1 First Rx Descriptor Pointer 2 (E1FRDP2)	0x088888	page 266
Ethernet1 First Rx Descriptor Pointer 3 (E1FRDP3)	0x08888C	page 266
Ethernet1 Current Rx Descriptor Pointer 0 (E1CRDP0)	0x0888A0	
Ethernet1 Current Rx Descriptor Pointer 1 (E1CRDP1)	0x0888A4	
Ethernet1 Current Rx Descriptor Pointer 2 (E1CRDP2)	0x0888A8	
Ethernet1 Current Rx Descriptor Pointer 3 (E1CRDP3)	0x0888AC	
Ethernet1 Current Tx Descriptor Pointer 0 (E1CTDP0)	0x0888E0	page 260
Ethernet1 Current Tx Descriptor Pointer 1 (E1CTDP1)	0x0888E4	page 261
Ethernet1 MIB Counters	0x089800 - 0x0898FF	page 302

Table 286: PHY Address Register, Offset: 0x080800

Bits	Field Name	Function	Initial Value
4:0	PhyAD0	PHY device address for port 0.	00100
9:5	PhyAD1	PHY device address for port 1.	00101

Table 287: SMI Register (SMIR), Offset: 0x080810

Bits	Field Name	Function	Initial Value
15:0	Data	<ul style="list-style-type: none"> • SMI READ operation Two transactions are required: (1) write to the SMI register with OpCode = 1, PhyAd, RegAd with the Data being any value; (2) read from the SMI register. When reading back the SMI register, the Data is the addressed Phy register contents if the Read-Valid bit (#27) is 1. The Data remains undefined as long as ReadValid is 0. • SMI WRITE operation One transaction is required. Write to the SMI register with OpCode = 0, PhyAd, RegAd with the Data to be written to the addressed Phy register. 	0
20:16	PhyAd	PHY Device Address	0
25:21	RegAd	PHY Device Register Address	0
26	OpCode	0 - Write 1 - Read	1
27	ReadValid	1 - indicates that the Read operation has been completed for the addressed RegAd register, and the data is valid on the Data field.	0
28	Busy	1 - indicates that an operation is in progress and that CPU must not write to the SMI register at this time.	0
31:29	N/A	These bits must be written as 0 for any write to the SMI register.	0

**Table 288: Port Configuration Register (PCR),
Offset: 0x084800 for Ethernet_0; 0x088800 for Ethernet_1**

Bits	Field Name	Function	Initial Value
0	PM	Promiscuous mode 0 - Normal mode. Frames are received only if destination address is found in hash table. 1 - Promiscuous mode. Frames are received regardless of their destination address. Errored frames are discarded unless PBF is set.	0

**Table 288: Port Configuration Register (PCR),
Offset: 0x084800 for Ethernet_0; 0x088800 for Ethernet_1 (Continued)**

Bits	Field Name	Function	Initial Value
1	RBM	Reject Broadcast Mode 0 - Receive Broadcast address. 1 - Reject Frames with broadcast address. Overridden by promiscuous mode.	0
2	PBF	Pass Bad Frames 0 - Normal mode. 1 - Pass bad Frames. The Ethernet receiver will pass to CPU errored frames, which are normally rejected, like fragments and collided packets. Frames are passed only if they pass address filtering successfully.	0
6:3	Reserved	Reserved.	0
7	EN	Enable 0 - Disabled. 1 - Enable. Ethernet port is ready to transmit/receive.	0
9:8	LPBK	Loop Back Mode 00 - Normal mode. 01 - Internal Loopback mode. TX data is looped back to the RX lines. No transition is seen on the interface pins. 10 - External Loopback mode. TX data is looped back to the RX lines and also transmitted out to the MII interface pins. 11 - Reserved.	0
10	FC	Force Collision 0 - Normal mode. 1 - Force Collision on any TX frame. For RXM test (in Loop-back mode).	0
11		Reserved.	0
12	HS	Hash Size 0 - 8K address filtering (256KB of memory space required). 1 - 1/2K address filtering (16KB of memory space required).	0
13	HM	Hash Mode 0 - Hash Function 0. 1 - Hash Function 1.	0
14	HDM	Hash Default Mode 0 - Discard addresses not found in address table. 1 - Pass addresses not found in address table.	0

**Table 288: Port Configuration Register (PCR),
Offset: 0x084800 for Ethernet_0; 0x088800 for Ethernet_1 (Continued)**

Bits	Field Name	Function	Initial Value
15	HD	Duplex Mode 0 - Half Duplex. 1 - Full Duplex. NOTE: Valid only when Auto-Negotiation for duplex mode is disabled.	0
27:16	Reserved		
30:28		0x6 ISL Enabled 0x0 ISL Disabled NOTE: When ISL is enabled, bit 31 must set to 0.	
31	ACCS	Accelerate Slot Time 0 - Normal mode. 1 - Reserved.	0

**Table 289: Port Configuration Extend Register (PCXR),
Offset: 0x084808 for Ethernet_0; 0x088808 for Ethernet_1**

Bits	Field Name	Function	Initial Value
0	IGMP	IGMP packets capture enable. 0 - IGMP packets are treated as normal Multicast packets. 1 - IGMP packets on IPv4/IPv6 over Ethernet/802.3 are trapped and sent to high priority RX queue.	0
1	SPAN	Spanning Tree packets capture enable. 0 - BPDU (Bridge Protocol Data Unit) packets are treated as normal Multicast packets. 1 - BPDU packets are trapped and sent to high priority RX queue.	0
2	PAR	Partition enable. When more than 61 collisions occur while transmitting, the port enters Partition mode. It waits for the first good packet from the wire, and then goes back to Normal mode. Under Partition mode it continues transmitting, but it does not receive. 0 - Normal mode. 1 - Partition mode.	0

**Table 289: Port Configuration Extend Register (PCXR),
Offset: 0x084808 for Ethernet_0; 0x088808 for Ethernet_1 (Continued)**

Bits	Field Name	Function	Initial Value
5:3	PRIOrx	Priority weight in the round-robin between high and low priority TX queues: 000 - 1 pkt transmitted from HIGH, 1 pkt from LOW. 001 - 2 pkt transmitted from HIGH, 1 pkt from LOW. 010 - 4 pkt transmitted from HIGH, 1 pkt from LOW. 011 - 6 pkt transmitted from HIGH, 1 pkt from LOW. 100 - 8 pkt transmitted from HIGH, 1 pkt from LOW. 101 - 10 pkt transmitted from HIGH, 1 pkt from LOW. 110 - 12 pkt transmitted from HIGH, 1 pkt from LOW. 111 - All pkt transmitted from HIGH, 0 pkt from LOW. LOW will be served only if HIGH is empty. If HIGH queue is emptied before finishing the count, the count will be reset until next first HIGH comes in.	0
7:6	PRIOrx	Default priority for packets received on this port: 00 - Lowest priority. 11 - Highest priority.	0
8	PRIOrx_Override	Override priority for packets received on this port: 0 - Do not override. 1 - Override with <PRIOrx> field.	0
9	DPLXen	Enable auto-negotiation for duplex mode: 0 - Enable. 1 - Disable.	0
10	FCTLen	Enable auto-negotiation for 802.3x flow-control: 0 - Enable. NOTE: When enabled, 1 is written (through SMI access) to PHY's register 4 bit 10 to advertise flow-control capability. 1 - Disable. NOTE: Only enable flow control after the PHY address is set by the CPU. When changing the PHY address the flow control auto-negotiation must be disabled.	1
11	FLP	Force Link Pass 0 - Force Link Pass. 1 - Do NOT Force Link pass.	1
12	FCTL	Flow-Control Mode 0 - Enable IEEE 802.3x flow-control. 1 - Disable IEEE 802.3x flow-control. NOTE: Valid only when auto negotiation for flow control is disabled.	0
13	Reserved	Reserved.	0

**Table 289: Port Configuration Extend Register (PCXR),
Offset: 0x084808 for Ethernet_0; 0x088808 for Ethernet_1 (Continued)**

Bits	Field Name	Function	Initial Value
15:14	MFL	Max Frame Length. Maximum packet allowed for reception (including CRC): 00 - 1518 bytes 01 - 1536 bytes 10 - 2048 bytes 11 - 64K bytes	0
16	MIBclrMode	MIB counters clear Mode. Setting this bit causes the counters to reset when the CPU performs a counter read operation. In order to reset all MIB counters, the CPU should set this bit and read all the counters.	0
17	MIBctrMode	Reserved.	0
18	Speed	Port Speed 0 - 10Mbit/Sec. 1 - 100Mbit/sec. NOTE: Valid only if SpeedEn bit is set.	0
19	SpeedEn	Enable Auto-negotiation for Speed 0 - Enable. 1 - Disable.	0
20	RMIIen	RMII enable 0-Port MII1 functions as MII port 1 1-Port MII1 functions as RMII port 0 and RMII port 1	0
21	DSCPen	DSCP enable 0-IP DSCP field decoding is disabled 1-IP DSCP field decoding is enabled	0
27:22	Reserved	Reserved.	0
31:28	Test	These bits must be set to 0 for proper operation of the Ethernet port.	0

Table 290: Port Command Register (PCMR), Offset: 0x084810 for Ethernet_0; 0x088810 for Ethernet_1

Bits	Field Name	Function	Initial Value
14:0	Reserved	Reserved.	0
15	FJ	Force Jam / Flow Control. When in halfduplex, the CPU uses this bit to force collisions on the Ethernet segment. When the CPU recognizes that it is going to run out of receive buffers, it can force the transmitter to send jam frames, forcing collisions on the wire. The CPU must clear the FJ bit when more resources are available in order to allow transmission on the Ethernet segment. When in full-duplex and flow-control is enabled, this bit causes the port's transmitter to send flow-control PAUSE packets. The CPU should reset this bit when more resources are available.	0
31:16	Reserved	Reserved.	0

Table 291: Port Status Register (PSR), Offset: 0x084818 for Ethernet_0; 0x088818 for Ethernet_1

Bits	Field Name	Function	Initial Value
0	Speed	Indicates port speed. 0 - 10Mbit/s. 1 - 100Mbit/s. This bit is read-only.	0
1	Duplex	Indicates port duplex mode. 0 - Half duplex. 1 - Full duplex. This bit is read-only.	0
2	Fctl	Indicates Flow-control mode. 0 - Flow-control mode enabled. 1 - Flow-control mode disabled. This bit is read-only.	0
3	Link	Indicates link status. 0 - Link is down. 1 - Link is up. This bit is read-only.	0

Table 291: Port Status Register (PSR), Offset: 0x084818 for Ethernet_0; 0x088818 for Ethernet_1 (Continued)

Bits	Field Name	Function	Initial Value
4	Pause	Indicates that the port is in flow-control disabled state. This bit is set when an IEEE 802.3x flow-control PAUSE (XOFF) packet is received (assuming that flow-control is enabled and the port is in full-duplex mode). Reset when XON is received, or when the XOFF timer has expired. This bit is read-only.	0
5	TxLow	Tx Low priority status. Indicates the status of the low priority transmit queue: 0 - Stopped. 1 - Running. This bit is read-only.	0
6	TxHigh	Tx High priority status. Indicates the status of the high priority transmit queue: 0 - Stopped. 1 - Running. This bit is read-only.	0
7	TXinProg	TX in Progress. Indicates that the port's transmitter is in an active transmission state. This bit is read-only.	0
31:8	Reserved	Reserved.	0

Table 292: Serial Parameters Register (SPR), Offset: 0x084820 for Ethernet_0; 0x088820 for Ethernet_1

Bits	Field Name	Function	Initial Value
1:0	JAM_LENGTH	Two bits to determine the JAM Length (in Backpressure) as follows: 00 = 12K bit-times. 01 = 24K bit-times. 10 = 32K bit-times. 11 = 48K bit-times.	11 (48K bit time)
6:2	JAM_IPG	Five bits to determine the JAM IPG. The step is four bit-times. The JAM IPG varies between 4 bit time to 124.	01000 (32 bit time)
11:7	IPG_JAM_TO_DATA	Five bits to determine the IPG JAM to DATA. The step is four bit-times. The value may vary between 4 bit time to 124.	10000 (64 bit time)

Table 292: Serial Parameters Register (SPR), Offset: 0x084820 for Ethernet_0; 0x088820 for Ethernet_1 (Continued)

Bits	Field Name	Function	Initial Value
16:12	IPG_DATA	Inter-Packet Gap (IPG): The step is 4 bit-times. The value may vary between 12 bit time to 124. NOTE: These bits may be changed only when all Ethernet port is disabled.	11000 (96 bit time)
21:17	Data_Blind	Data Blinder The number of nibbles from the beginning of the IPG, in which the IPG counter is restarted when detecting a carrier activity. Following this value, the port enters the Data Blinder zone and does not reset the IPG counter. This ensures fair access to the medium. Value should be written in hexadecimal format. The default is 0x19 (64 bit time -2/3 of the default IPG). The step is 4bit (nibble) time. Valid range is 0xc to 0x1f. NOTE: These bits may be changed only when the Ethernet port is disabled.	11001 (64 bit time)
22	Limit4	The number of consecutive packet collisions that will occur before the collision counter is reset. 0- The port resets its collision counter after 16 consecutive retransmit trials and restarts the Backoff algorithm. 1- The port will reset its collision counter and restart the Backoff algorithm after 4 consecutive transmit trials.	0
31:23	Reserved	Reserved	0

Table 293: Hash Table Pointer Register (HTPR), Offset: 0x084828 for Ethernet_0; 0x088828 for Ethernet_1

Bits	Field Name	Function	Initial Value
31:0	HTP	32-bit pointer to the address table. Bits [2:0] must be set to zero.	0x0

Table 294: Flow Control Source Address Low (FCSAL), Offset: 0x084830 for Ethernet_0; 0x088830 for Ethernet_1

Bits	Field Name	Function	Initial Value
15:0	SA[15:0]	Source Address The least significant bits of the source address for the port. This address is used for Flow Control.	0

**Table 295: Flow Control Source Address High (FCSAH),
Offset: 0x084838 for Ethernet_0; 0x088838 for Ethernet_1**

Bits	Field Name	Function	Initial Value
31:0	SA[47:16]	Source Address The most significant bits of the source address for the port. This address is used for Flow Control.	0

**Table 296: SDMA Configuration Register (SDCR), Offset: 0x084840 for
Ethernet_0; 0x088840 for Ethernet_1**

Bits	Field Name	Function	Initial Value
1:0	Reserved	Reserved.	0
5:2	RC	Retransmit Count Sets the maximum number of retransmits per packet. After executing retransmit for RC times, the TX SDMA closes the descriptor with a Retransmit Limit error indication and processes the next packet. When RC is set to 0, number of retransmits is unlimited. In this case, retransmit process is terminated only if CPU issues an Abort command.	1111
6	BLMR	Big/Little endian Receive Mode The DMA supports big or little endian configurations per channel. The BLMR bit only affects data transfer to memory. 0 - Big endian. 1 - Little endian.	1
7	BLMT	Big/Little endian Transmit Mode The DMA supports big or little endian configurations per channel. The BLMT bit only affects data transfer from memory. 0 - Big endian. 1 - Little endian.	1
8	POVR	PCI Override When set, causes the SDMA to direct all its accesses in PCI_0 direction and overrides normal address decoding process.	0
9	RIFB	Receive Interrupt on Frame Boundaries When set, the SDMA Rx generates interrupts only on frame boundaries (i.e. after writing the frame status to the descriptor).	0
11:10	Reserved	Reserved.	0

Table 296: SDMA Configuration Register (SDCR), Offset: 0x084840 for Ethernet_0; 0x088840 for Ethernet_1 (Continued)

Bits	Field Name	Function	Initial Value
13:12	BSZ	Burst Size Sets the maximum burst size for SDMA transactions: 00 - Burst is limited to 1 64bit words. 01 - Burst is limited to 2 64bit words. 10 - Burst is limited to 4 64bit words. 11 - Burst is limited to 8 64bit words.	0
31:14	Reserved	Reserved.	0

Table 297: SDMA Command Register (SDCMR), Offset: 0x084848 for Ethernet_0; 0x088848 for Ethernet_1

Bits	Field Name	Function	Initial Value
6:0	Reserved	Reserved.	0
7	ERD	Enable RX DMA. Set to '1' by the CPU to cause the SDMA to start a receive process. Cleared when the CPU issues an Abort Receive command.	0
14:8	Reserved	Reserved.	0
15	AR	Abort Receive Set to '1' by the CPU to abort a receive SDMA operation. When the AR bit is set, the SDMA aborts its current operation and moves to IDLE. No descriptor is closed. AR bit is cleared upon entering IDLE. After setting AR bit, the CPU should poll it in order to verify that the abort sequence is completed.	0
16	STDH	Stop TX High Set to '1' by the CPU to stop the transmission process from the high priority queue at the end of the current frame. An interrupt is generated when the stop command has been executed. Writing '1' to STDH resets TXDH bit. Writing '0' to this bit has no effect.	0
17	STDL	Stop TX Low Set to '1' by the CPU in order to stop the transmission process from the low priority queue at the end of the current frame. An interrupt is generated when the stop command has been executed. Writing '1' to STDL resets TXDL bit. Writing '0' to this bit has no effect.	0

Table 297: SDMA Command Register (SDCMR), Offset: 0x084848 for Ethernet_0; 0x088848 for Ethernet_1 (Continued)

Bits	Field Name	Function	Initial Value
22:18	Reserved	Reserved.	0
23	TXDH	Start Tx High Set to '1' by the CPU in order to cause the SDMA to fetch the first descriptor and start a transmit process from the high priority Tx queue. Writing '1' to TXDH resets STDH bit. Writing '0' to this bit has no effect.	0
24	TXDL	Start Tx Low Set to '1' by the CPU to cause the SDMA to fetch the first descriptor and start a transmit process from the low priority Tx queue. Writing '1' to TXDL resets STDL bit. Writing '0' to this bit has no effect.	0
30:25	Reserved	Reserved.	0
31	AT	Abort Transmit Set to '1' by the CPU to abort a transmit DMA operation. When the AT bit is set, the SDMA aborts its current operation and moves to IDLE. No descriptor is closed. Cleared upon entering IDLE. After setting AT bit, the CPU must poll it in order to verify that the abort sequence is completed.	0

Table 298: Interrupt Cause Register (ICR), Offset: 0x084850 for Ethernet_0; 0x088850 for Ethernet_1

Bits	Field Name	Function	Initial Value
0	RxBuffer	Rx Buffer Return Indicates a Rx buffer returned to CPU ownership or that the port finished reception of a Rx frame in either priority queues. NOTE: In order to get a Rx Buffer return per priority queue, use bit 19:16. This bit is set upon closing any Rx descriptor which has its EI bit set. In order to limit the interrupts to frame (rather than buffer) boundaries, the user should set SDCR<RIFB> bit. When RIFB is set, an interrupt will be generated only upon closing the first descriptor of a received packet if this descriptor has it EI bit set.	0
1	Reserved	Reserved.	0

Table 298: Interrupt Cause Register (ICR), Offset: 0x084850 for Ethernet_0; 0x088850 for Ethernet_1 (Continued)

Bits	Field Name	Function	Initial Value
2	TxBufferHigh	Tx Buffer for High priority Queue Indicates a Tx buffer returned to CPU ownership or that the port finished transmission of a Tx frame. NOTE: This bit is set upon closing any Tx descriptor which has its EI bit set. In order to limit the interrupts to frame (rather than buffer) boundaries, the user should set EI only in the last descriptor.	0
3	TxBufferLow	Tx Buffer for Low Priority Queue Indicates a Tx buffer returned to CPU ownership or that the port finished transmission of a Tx frame. NOTE: This bit is set upon closing any Tx descriptor which has its EI bit set. In order to limit the interrupts to frame (rather than buffer) boundaries, the user should set EI only in the last descriptor.	0
5:4	Reserved	Reserved.	0
6	TxEndHigh	Tx End for High Priority Queue Indicates that the Tx DMA stopped processing the high priority queue after stop command, or that it reached the end of the high priority descriptor chain.	0
7	TxEndLow	Tx End for Low Priority Queue Indicates that the Tx DMA stopped processing the low priority queue after stop command, or that it reached the end of the low priority descriptor chain.	0
8	RxError	Rx Resource Error Indicates a Rx resource error event in either priority queues. NOTE: In order to get a Rx Resource Error Indication per priority queue, use bit 23:20. event	0
9	Reserved	Reserved.	0
10	TxErrorHigh	Tx Resource Error for High Priority Queue Indicates a Tx resource error event during packet transmission from the high priority queue.	0
11	TxErrorLow	Tx Resource Error for Low Priority Queue Indicates a Tx resource error event during packet transmission from the low priority queue.	0
12	RxOVR	Rx Overrun Indicates an overrun event that occurred during reception of a packet.	0
13	TxUdr	Tx Underrun Indicates an underrun event that occurred during transmission of packet from either queue.	0

Table 298: Interrupt Cause Register (ICR), Offset: 0x084850 for Ethernet_0; 0x088850 for Ethernet_1 (Continued)

Bits	Field Name	Function	Initial Value
16	RxBuffer-Queue[0]	Rx Buffer Return in Priority Queue[0], Indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in a receive priority queue[0]	0
17	RxBuffer-Queue[1]	Rx Buffer Return in Priority Queue[1], Indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in a receive priority queue[1]	0
18	RxBuffer-Queue[2]	Rx Buffer Return in Priority Queue[2], Indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in a receive priority queue[2]	0
19	RxBuffer-Queue[3]	Rx Buffer Return in Priority Queue[3], Indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in a receive priority queue[3]	0
20	RxError-Queue[0]	Rx Resource Error in Priority Queue[0], Indicates a Rx resource error event in receive priority queue[0]	0
21	RxError-Queue[1]	Rx Resource Error in Priority Queue[1], Indicates a Rx resource error event in receive priority queue[1]	0
22	RxError-Queue[2]	Rx Resource Error in Priority Queue[2], Indicates a Rx resource error event in receive priority queue[2]	0
23	RxError-Queue[3]	Rx Resource Error in Priority Queue[3], Indicates a Rx resource error event in receive priority queue[3]	0
27:24	Reserved	Reserved.	0
28	MIIPhySTC	MII PHY Status Change Indicates a status change reported by the PHY connected to this port. Set when the MII management interface block identifies a change in PHY's register 1.	0
29	SMDone	SMI Command Done Indicates the SMI completed a MII management command (either read or write) that was initiated by the CPU writing to the SMI register.	0
30	Reserved	Reserved.	0
31	EtherIntSum	Ethernet Interrupt Summary This bit is a logical OR of the (unmasked) bits [30:4] in the Interrupt Cause register.	0

Table 299: Interrupt Mask Register (IMR), Offset: 0x084858 for Ethernet_0; 0x088858 for Ethernet_1

Bits	Field Name	Function	Initial Value
31:0	Various	Mask bits for Interrupt Cause register.	0x0

Table 300: IP Differentiated Services CodePoint to Priority0 low (DSCP2P0L)

Bits	Field Name	Function	Initial Value
31:0	Priority0 low	The LSB priority bit for DSCP[31:0] entries.	0x0

Table 301: IP Differentiated Services CodePoint to Priority0 high (DSCP2P0H)

Bits	Field Name	Function	Initial Value
31:0	Priority0 high	These bits are the LSB priority bit for DSCP[63:32] entries.	0x0

Table 302: IP Differentiated Services CodePoint to Priority1 low (DSCP2P1L)

Bits	Field Name	Function	Initial Value
31:0	Priority1 low	These bits are the MSB priority bit for DSCP[31:0] entries.	0x0

Table 303: IP Differentiated Services CodePoint to Priority1 high (DSCP2P1H)

Bits	Field Name	Function	Initial Value
31:0	Priority1 high	These bits are the MSB priority bit for DSCP[63:32] entries.	0x0

Table 304: VLAN Priority Tag to Priority (VPT2P)

Bits	Field Name	Function	Initial Value
Initial Value 7:0	Priority0	These bits are the LSB priority bit for VLAN Priority[7:0] entries.	0xcc
15:8	Priority1	These bits are the MSB priority bit for VLAN Priority[7:0] entries.	0xf0
31:16		Reserved.	0x0

12.5.1 Defining a priority queue to the IP DSCP or VLAN Entry

To define a priority queue to the IP DSCP or VLAN entry, the entry's priority 0 and priority 1 bits must be defined.

Table 305 and Table 306 describe the writing of IP DSCP and VLAN entries respectively for a few set examples. Table 307 describes three example cases for mixed priority queueing.

Table 305: Writing IP DSCP Priority Example

IP DSCP Value	Priority MSB bit	Priority LSB bit
0	DSCP2P1L[0]	DSCP2P0L[0]
16	DSCP2P1L[16]	DSCP2P0L[16]
31	DSCP2P1L[31]	DSCP2P0L[31]
32	DSCP2P1H[0]	DSCP2P0H[0]
48	DSCP2P1H[16]	DSCP2P0H[16]
63	DSCP2P1H[31]	DSCP2P0H[31]

Table 306: Writing VLAN Priority Example

VLAN Priority Value	Priority MSB bit	Priority LSB bit
0	VPT2P[8]	VPT2P[0]
4	VPT2P[12]	VPT2P[4]
7	VPT2P[15]	VPT2P[7]

Table 307: Writing IP DSCP and VLAN Priority Example

Case	IPDSCP	All Others	VLAN Tag Packets
A	0x0 and 0x3f (63) directed to priority queue 3	directed to priority queue 0	ignored
B	<0x20 (32) directed to priority queue 2	directed to priority queue 1	directed to priority queue 3
C	0x1f (31) and 0x20 (32) directed to priority queue 3 0x0 and 0x3f (63) directed to priority queue 0 <0x1f (31) directed to priority queue 1	directed to priority queue 2	>3 and IP DSCP ≠ 0x1f or 0x20 directed to priority queue 2 other tags are ignored

Table 308: Writing IP DSCP and VLAN Priority Register mapping Example

Register	Case A	Case B	Case C
DSCPC2P0L	0x00000001	0x00000000	0xFFFFFFFF
DSCP2P0H	0x80000000	0xFFFFFFFF	0x00000001
DSCP2P1L	0x00000001	0xFFFFFFFF	0x80000000
DSCP2P1H	0x80000000	0x00000000	0x7FFFFFFF
VPT2P	0x00000001	0x0000FFFF	0x0000F000

12.6 Ethernet MIB Counters

The Ethernet unit includes a set of counters that are used to count events occurring on the segment to which the port is connected to. All counters are 32 bit wide.

The CPU must read all the MIB counters during initialization in order to reset the counters to '0'. The counters will only be reset to '0' if MIBclrMode bit in the Port_Configuration_Extend register is set to '0' (default). If MIBclrMode bit is '1', reading the MIB counters will have no effect on their value.

NOTE: [Table 309](#) lists definitions of terms used in the counter descriptions.

Table 309: Terms Used in MIB Counters Descriptions

Term	Definition
Packet Data Section	All data bytes in the packet following the SFD until the end of the packet.
Packet Data Length	The number of data bytes in the packet data section.
Data Octet	A single byte from the packet data section.
Nibble	4 bits (half byte) of a data octet.
Misaligned Packet	A packet with an odd number of nibbles.

Table 309: Terms Used in MIB Counters Descriptions (Continued)

Term	Definition
Received Good Packet	A received packet which is well formed.
Received Bad Packet	A received packet which has an error such as bad CRC, Rx Error Event, Invalid size (too short or too long).
Transmitted Packet	Any transmitted packet (not including collision fragments).
Collision Event	Any collision event that is indicated by assertion of MII_COL signal within the collision window interval.
Late Collision Event	Any collision event that is indicated by assertion of MII_COL signal outside the collision window interval.
Rx Error Event	An error event that is indicated by assertion of MII_RX_ERR signal.
Dropped Packet	A received packet which is dropped by the port due to lack of resources (e.g. no Rx buffers available).
MIBctrMode	MIBctrMode bit in the Port Configuration Extend register.
MaxFrameSize	1518, 1536, 2 K or 64Kbytes depending on the setting in the Port Configuration Extend register.

Table 310: Ethernet MIB Counters, Offset: 0x085800–0x0858FF for Ethernet_0; 0x089800–0x0898FF for Ethernet_1

Address for Port 0	Address for Port 1	Counter Name	Function	Initial Value
0x085800	0x089800	Bytes Received	This counter increments once for every data octet of good packets (Unicast + Multicast + Broadcast) received by the port.	-
0x085804	0x089804	Bytes Sent	This counter increments once for every data octet of transmitted packets sent by the port.	-
0x085808	0x089808	Frames Received	This counter increments once for every good packet (Unicast + Multicast + Broadcast) received by the port.	-
0x08580C	0x08980C	Frames Sent	This counter increments once for every transmitted packet sent by the port.	-

Table 310: Ethernet MIB Counters, Offset: 0x085800–0x0858FF for Ethernet_0; 0x089800–0x0898FF for Ethernet_1 (Continued)

Address for Port 0	Address for Port 1	Counter Name	Function	Initial Value
0x085810	0x089810	Total Bytes Received	This counter increments once for every data octet of all received packets. This includes data octets of BAD packets, which might be automatically rejected by the port (e.g fragments). This counter reflects all the data octets received from the line. NOTE: A nibble is NOT counted as a whole byte.	-
0x085814	0x089814	Total Frames Received	This counter increments once for every received packet. This includes BAD packets. This counter reflects all packets received from the line.	-
0x085818	0x089818	Broadcast Frames Received	This counter increments once for every good broadcast packet received.	-
0x08581C	0x08981C	Multicast Frames Received	This counter increments once for every good Multicast packet received. This counter does not count Broadcast packets.	-
0x085820	0x089820	CRC Error	This counter increments once for every received packet which meets all the following conditions (i.e. logical AND of the following conditions): <ul style="list-style-type: none"> • Packet data length is between 64 and MaxFrameSize bytes inclusive (i.e. valid packet data length per IEEE std). • Packet has invalid CRC. • Collision Event has not been detected. • Late Collision Event has not been detected. • Rx Error Event has not been detected. 	-

Table 310: Ethernet MIB Counters, Offset: 0x085800–0x0858FF for Ethernet_0; 0x089800–0x0898FF for Ethernet_1 (Continued)

Address for Port 0	Address for Port 1	Counter Name	Function	Initial Value
0x085824	0x089824	Oversize Frames	<p>This counter increments once for every received packet which meets all the following conditions (i.e. logical AND of the following conditions):</p> <ul style="list-style-type: none"> • Packet data length is greater than MaxFrameSize. • Packet has valid CRC. • Rx Error Event has not been detected. 	-
0x085828	0x089828	Fragments	<p>This counter increments once for every received packet which meets all the following conditions (i.e. logical AND of the following conditions):</p> <ul style="list-style-type: none"> • Packet data length is less than 64 bytes -OR- packet without SFD and is less than 64 bytes in length. • Collision Event has not been detected. • Late Collision Event has not been detected. • Rx Error Event has not been detected. • Packet has INVALID CRC. 	-
0x08582c	0x08982c	Jabber	<p>This counter increments once for every received packet which meets all the following conditions (i.e. logical AND of the following conditions):</p> <ul style="list-style-type: none"> • Packet data length is greater than MaxFrameSize. • Packet has invalid CRC. • Rx Error Event has not been detected. 	-
0x085830	0x089830	Collision	<p>This counter increments once for every received packet which meets both of the following conditions (i.e. logical AND of the following conditions):</p> <ul style="list-style-type: none"> • Collision Event has been detected. • Rx Error Event has not been detected. 	-

Table 310: Ethernet MIB Counters, Offset: 0x085800–0x0858FF for Ethernet_0; 0x089800–0x0898FF for Ethernet_1 (Continued)

Address for Port 0	Address for Port 1	Counter Name	Function	Initial Value
0x085834	0x089834	Late Collision	This counter increments once for every received packet which meets both of the following conditions (i.e. logical AND of the following conditions): <ul style="list-style-type: none"> Late Collision Event has been detected. Rx Error Event has not been detected. 	-
0x085838	0x089838	Frames 64 Bytes	This counter increments once for every received and transmitted packet with size of 64 bytes. This counter does not count BAD received packets.	-
0x08583c	0x08983c	Frames 65-127 Bytes	This counter increments once for every received and transmitted packet with size of 65 to 127 bytes. This counter does not count BAD received packets.	-
0x085840	0x089840	Frames 128-255 Bytes	This counter increments once for every received and transmitted packet with size of 128 to 255 bytes. This counter does not count BAD received packets.	-
0x085844	0x089844	Frames 256-511 Bytes	This counter increments once for every received and transmitted packet with size of 256-511 bytes. This counter does not count BAD received packets.	-
0x085848	0x089848	Frames 512-1023 Bytes	This counter increments once for every received and transmitted packet with size of 512-1023 bytes. This counter does not count BAD received packets.	-
0x08584c	0x08984c	Frames 1024-MaxFrameSize Bytes	This counter increments once for every received and transmitted packet with size of 1024 to MaxFrameSize bytes. This counter does not count BAD received packets.	-
0x085850	0x089850	Rx Error	This counter increments once for every received packet in which the Rx Error Event has been detected. When a Rx Error event occurs, the following counters do not increment: CRC Error, Oversize Frames, Fragments, Jabbers, Collision and Late Collision.	-
0x085854	0x089854	Dropped Frames	Reserved.	-

Table 310: Ethernet MIB Counters, Offset: 0x085800–0x0858FF for Ethernet_0; 0x089800–0x0898FF for Ethernet_1 (Continued)

Address for Port 0	Address for Port 1	Counter Name	Function	Initial Value
0x085858	0x089858	Out Multicast Frames	The number of Multicast frames sent by the port. This counter does not count Broadcast packets.	-
0x08585C	0x08985C	Out Broadcast Frames	The number of Broadcast frames sent by the port.	-
		Out Unicast Frames	Calculated from: <ul style="list-style-type: none"> • "Frames Sent" • "Out Multicast Frames" • "Out Broadcast Frames" 	
0x085860	0x089860	Undersize Frames	This counter increments once for every received packet which meets all the following conditions (i.e. logical AND of the following conditions): <ul style="list-style-type: none"> • Packet data length is less than 64 bytes. • Collision Event has not been detected. • Late Collision Event has not been detected. • Rx Error Event has not been detected. • Packet has valid CRC. 	-

13. SERIAL DMA (SDMA)

13.1 Overview

There are 16 SDMA channels on the GT-96100A that are dedicated for moving data between the serial communications channels (MPSCs) and memory buffers. Each SDMA channel consists of a DMA engine for receiving and one for transmitting.

The 16 SDMA channels are logically divided in two identical groups. Each group consists of 8 SDMA channels - one SDMA channel per MPSC. In both groups SDMA channel 0 is allocated to data transfer from/to MPSC0, SDMA1 is tied to MPSC1 and so on. Each MPSC can be programmed to use a specific SDMA channel from one of the groups, making it possible to split the serial data flow into two logical streams. These streams can be assigned a different priority tag at the CIU level.

Each SDMA channel has two dedicated FIFOs for data buffering (for a total of 32 FIFOs). All FIFOs are 256 bytes deep.

For receive operations, the MPSC moves received data into the dedicated FIFO of the corresponding SDMA. Then, using descriptors set up by the user, the SDMA moves the data into memory buffers. For transmit operations, the SDMA uses descriptors set up by the user to move data out of buffers into the dedicated FIFO. The MPSC moves the data down to the serial communications link.

The SDMA channel descriptors use a chained data structure. They work without CPU interference after appropriate initialization. SDMA channels can be programmed to generate interrupts on buffer or frame boundaries.

When enabled, the receive SDMAs run freely and expect to find a valid descriptor, when one is required. When a receive SDMA channel accesses an invalid descriptor, the receive SDMA process halts with a resource error status indication.

When enabled, the transmit SDMAs run freely until the end of the descriptor chain is reached. When a transmit SDMA accesses an invalid descriptor and the last descriptor was not marked as an end of frame descriptor, the transmit SDMA process halts with resource error status indication.

The SDMAs in each group arbitrate for accessing the descriptors and buffers. A standard round-robin scheme is used for arbitration within the group. The arbitration between the groups is done at the CIU level, which supports a programmable, weighted round-robin algorithm.

SDMA buffers and descriptor reside either in SDRAM space or in PCI space. Address decoding is automatic and does not require user intervention. However, the user may choose to override the address decoding and force one (or more) of the SDMAs to direct all its accesses to the PCI.

13.2 SDMA Descriptors

All SDMA data transfers are done via a chained link of descriptors. The following rules must be followed when using the GT-96100A SDMA descriptors:

- Descriptor length is 4LW and it must be 4LW aligned (i.e. Descriptor_Address[3:0]=0000).
- Descriptors may reside anywhere in CPU address space except NULL address, which is used to indicate end of descriptor chain.
- In normal mode (HDLC and Transparent) RX buffers associated with RX descriptors must be 64-bit aligned. Minimum size for RX buffers is 8 bytes. In low latency, or byte, mode (BISYNC, UART, and Transparent) RX buffers have no alignment restrictions.
- Tx buffers associated with TX descriptors can start in any byte location.
- SDMA RX and TX buffers are limited to 64Kbytes.

Figure 52: SDMA Descriptor Format

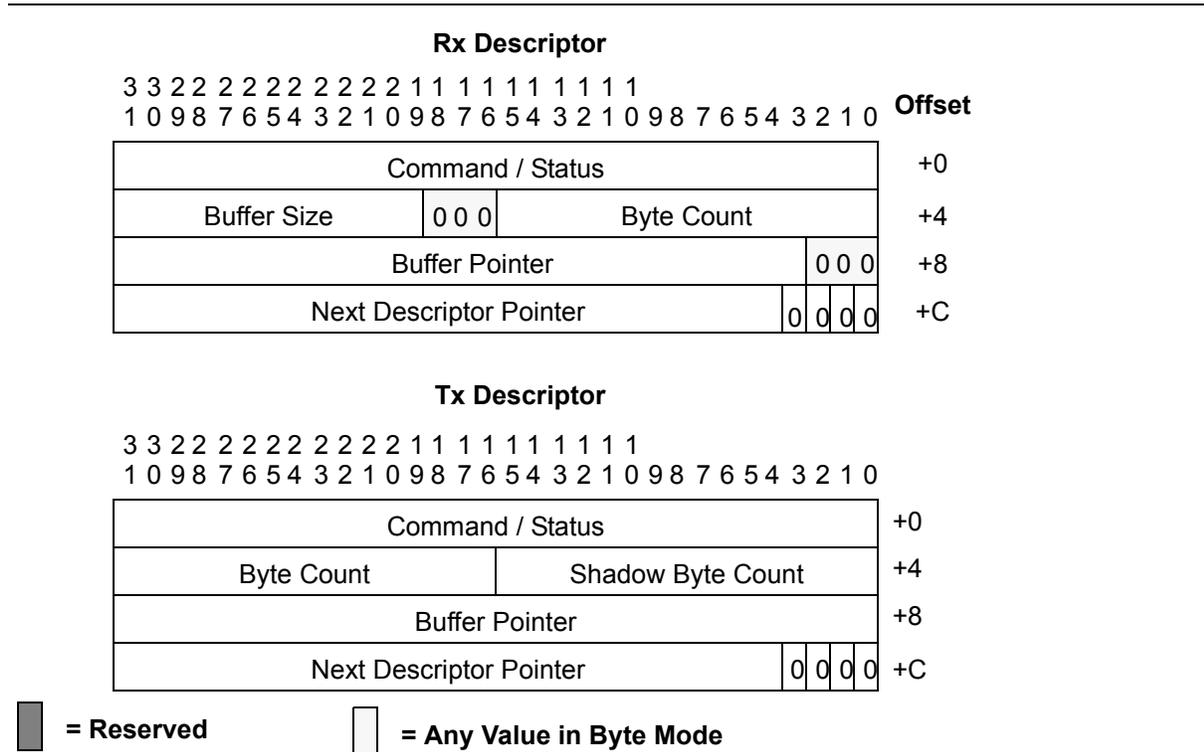


Table 311 through Table 315 provide detailed information about the descriptor fields.

Table 311: SDMA Descriptor - Command/Status word

Bits	Field Name	Function
31:0	Command/Status	Contains commands bits that instruct the SDMA how to process a buffer and status bits that the SDMA updates upon closing a descriptor. The CPU uses the status bits to evaluate the buffer status. Except for bits 31, 30, 23, 17, and 16, the definition of the bits vary depending on which mode is being used. See: <ul style="list-style-type: none"> • Section 14.5.2 “SDMAx Command/Status Field for HDLC Mode” on page 338. • Section 14.6.3 “SDMAx Command/Status Field for BISYNC Mode” on page 350. • Section 14.7.2 “SDMAx Command/Status Field for UART Mode” on page 362. • Section 14.8.1 “SDMAx Command/Status Field for Transparent Mode” on page 374.
31	O	Owner Bit When set to '1', the buffer can be processed by the GT-96100A. When set to '0', the buffer can be processed by the CPU. An SDMA process will halt when a descriptor with owner bit set to '0' is fetched.
30	AM	Auto Mode When set, the SDMA won't clear the Owner bit of the descriptor at the end of buffer processing.
29:24	Reserved	Determined by the mode selected.
23	EI	Enable Interrupt The GT-96100A generates a maskable interrupt when closing descriptor with EI bit set. NOTE: If the RIFB bit is set in the SDMA configuration register, a Rx interrupt is generated only if this is the last descriptor associated with a received frame. In this case, EI bit setting is masked for intermediate descriptors.
22:18		Determined by the mode selected.
17	F	First Bit Indicates first buffer of a frame.
16	L	Last Bit Indicates last buffer of a frame.
15:0		Determined by the mode selected.

Table 312: SDMA Descriptor - Buffer Size, Byte Count (Rx Descriptor)

Bits	Field Name	Function
31:16	Buffer Size	<p>The buffer size field is valid only in receive descriptors and is reserved in transmit descriptors. The field is written by the CPU and read by the GT-96100A. When the buffer byte counter of a SDMA receive channel reaches the buffer size value, the SDMA will close the buffer descriptor and will move to the next buffer.</p> <p>Buffer Size must be a multiple of 8 when the MPSC is programmed to work in normal mode (HDLC and Transparent). Buffer Size can be arbitrary when working in low bandwidth mode (BISYNC, UART, and Transparent).</p>
15:0	Byte Count	<p>The number of bytes that were actually written by the SDMA into the buffer. This number is never greater than Buffer Size. The CPU must initialize the Byte Count field with 0x0000.</p>

Table 313: SDMA Descriptor - Byte Count, Shadow Byte Count (Tx Descriptor)

Bits	Field Name	Function
31:16	Byte Count	<p>Byte count is the number of bytes to be transmitted.</p> <p>Zero byte counters are not supported with retransmission. Do not use zero byte buffers with LAP-D protocol.</p>
15:0	Shadow Byte Count	<p>The CPU must initialize this field with a value identical to the Byte Count field. The GT-96100A subtracts the number of bytes actually transmitted from this parameter.</p> <p>Usually the GT-96100A writes '0' in this field when closing a descriptor. However, when the transmit SDMA halts due to a transmit error, this number can be used to determine the number of bytes that were fetched into the GT-96100A.</p> <p>Setting both the Byte Count and Shadow Byte Count to '0' will cause the SDMA to close the descriptor and move to the next descriptor, if both or neither of the F and L bits are set. Setting Byte Count and Buffer Size to '0' in transmit descriptors with one of the F or L bits set will lead to unpredictable behavior.</p>

Table 314: SDMA Descriptor - Buffer Pointer

Bits	Field Name	Function
31:0	Buffer Pointer	<p>32-bit pointer to the beginning of the buffer associated with the descriptor. The buffer can reside anywhere in memory or PCI address space.</p>

Table 316: SDMA Configuration Register (SDCx), Offset: 0x000900

Bits	Field Name	Function	Initial Value
0	RFT	<p>Receive FIFO Threshold</p> <ul style="list-style-type: none"> • 0 - 8 bytes • 1 - Half FIFO (128 bytes) <p>NOTES:When working with an 8-bit data path, the threshold is always one byte regardless of the RFT value. It is recommended that RFT bit be set to '0' in this case.</p> <p>When RFT is set to '0', the SDMA will not burst. It will transfer one word (64 bits) on each transfer.</p>	0
1	SFM	<p>Single Frame Mode</p> <ul style="list-style-type: none"> • 0 - Multi frame mode. The GT-96100A will read as many frames as needed into the FIFO in order to keep the transmit FIFO full. The FIFO can handle more than one frame at a time. • 1 - Single frame mode. The first descriptor will not be fetched before the current frame's last descriptor is closed. <p>NOTES:The SFM bit must be set to '1' for HDLC Collision mode, BISYNC and UART protocols.</p> <p>When the SFM bit is set to '0', CTS Lost cannot be reported in the correct descriptor/frame. In LAN HDLC mode SFM must be set for proper operation.</p>	0
5:2	RC	<p>Retransmit Count</p> <p>In collision modes (LAP-D), after executing a backoff procedure RC times, the Tx SDMA will close the buffer with a Retransmit Limit (RL) error, a maskable interrupt will be generated, and the SDMA will go to OFF state. A new Transmit Demand command should be issued in order to start a new transmission process.</p> <p>When RC field is 0000, the GT-96100A will try to retransmit forever. The CPU needs to issue an abort command in order to stop the retransmit process.</p>	1111
6	BLMR	<p>Rx Big Little/Endian Receive Mode</p> <p>The GT-96100A supports big or little endian configuration per channel for maximum system flexibility. The BLMR bit only affects data movements.</p> <ul style="list-style-type: none"> • 0 - Big endian convention. • 1 - Little endian convention 	1

Table 317: SDMA Command Register (SDCMx), Offset: 0x000908

Bits	Field Name	Function	Initial Value
6:0	Reserved	Reserved.	0
7	ERD	<p>Enable Rx DMA</p> <p>When set to '1', the Rx SDMA will fetch the 1st descriptor and will be ready for a receive frame. The GT-96100A clears ERD when the GT-96100A receive SDMA has a resource error or when the CPU issues an abort command.</p>	0
14:8	Reserved	Reserved.	0
15	AR	<p>Abort Receive</p> <p>The CPU sets the AR bit when it needs to abort a receive SDMA channel operation. When the AR bit is set, the SDMA aborts its operation and goes to IDLE state. No descriptor is closed. The GT-96100A clears both the AR and ERD bits when entering IDLE state. The CPU must poll bit 15. When it is '0', the GT-96100A has completed the abort sequence. After an abort the CPU should write the 1st descriptor address and then set ERD bit to '1'.</p>	0
16	STD	<p>Stop Tx</p> <p>The SDMA stops transmission at the end of frame (i.e. at the end of buffer with L bit set to '1'). After transmitting the last buffer, the transmit SDMA goes to IDLE state. The GT-96100A clears the TXD bit when entering IDLE state. After the SDMA stops, the CPU must write the first descriptor address and then set the TXD bit to '1'. The GT-96100A signals the CPU with interrupt when the stop procedure is accomplished.</p>	0
22:17	Reserved	Reserved.	0
23	TXD	<p>Tx Demand</p> <p>When this bit is set to '1', the Tx DMA will fetch the first descriptor and will start the transmission process. The GT-96100A clears TXD when it successfully ends an SDMA transmit process. It also clears TXD when a resource error occurs, when the transmit process is halted due to channel error (i.e. CTS# lost), or when the CPU issues an abort command.</p>	0
30:24	Reserved	Reserved	0

Table 317: SDMA Command Register (SDCMx), Offset: 0x000908 (Continued)

Bits	Field Name	Function	Initial Value
31	AT	<p>Abort Transmit</p> <p>The CPU sets the AT bit to '1' when it needs to abort a transmit SDMA channel operation. When the AT bit is set, the SDMA aborts its operation and goes to IDLE state. No descriptor is closed. The GT-96100A clears both the AT and TXD bits when entering IDLE state.</p> <p>The CPU must poll bit 31. When it is '0', the GT-96100A has completed the abort sequence. After an abort, the CPU must write the first descriptor address and then set TXD bit to '1'.</p>	0

13.5 SDMA Group Configuration Register

Use this register to assign a specific SDMA channel from one of the SDMA groups to handle the data stream associated with the corresponding MPSC.

NOTE: MPSC's receive and transmit data flows does not have to be assigned to the same SDMA group. For example, there is no problem with assigning MPSC0 transmit to SDMA channel 0 of group 0, while MPSC0 receive flow is handled by SDMA channel 0 of group 1. Moreover, for certain asymmetric protocols, like ADSL, the bandwidth requirements for Rx and Tx are different, and splitting the data streams between the SDMA groups may be critical for controlling bandwidth allocation.

Table 318: SDMA Group Register (SGC), Offset: 0x101AF0

Bits	Field Name	Function	Initial Value
7:0	RxSG[7:0]	<p>Rx Group</p> <p>These bits are used to assign one of the SDMA groups to the RX data of each MPSC. RxSG[0] controls MPSC0, RxSG[1] controls MPSC1 and so on.</p> <ul style="list-style-type: none"> 0 - Receive data from the MPSC is handled by SDMA group 0. 1 - Receive data from the MPSC is handled by SDMA group 1. 	0
15:8	TxSG[7:0]	<p>Tx SDMA Group</p> <p>These bits are used to assign one of the SDMA groups to the TX data of each MPSC. TxSG[0] controls MPSC0, TxSG[1] controls MPSC1 and so on.</p> <ul style="list-style-type: none"> 0 - Transmit data to the MPSC is handled by SDMA group 0. 1 - Transmit data to the MPSC is handled by SDMA group 1. 	0
31:16	Reserved	Reserved.	0

13.6 SDMA Descriptor Pointer Registers

Each SDMA channel has three 32-bit registers that reside in a special descriptor’s Dual Port memory located in the internal address space of the GT-96100A.

Figure 55: SDMA Descriptor Pointer Registers

3 3 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0

SDMAx Current Receive Descriptor Pointer (SCRDPx)
SDMAx Current Transmit Descriptor Pointer (SCTDPx)
SDMAx First Transmit Descriptor Pointer (SFTDPx)

13.6.1 SDMA Current Receive Descriptor Pointer (SCRDP)

SCRDPx points to the current receive descriptor in memory. The CPU must write this register with the first descriptor address before enabling the SDMA receive channel. When a SDMA receive channel is enabled it will fetch the first descriptor pointed to by SCRDPx as part of its SDMA starting procedure.

13.6.2 SDMA Current Transmit Descriptor Pointer (SCTDP)

SCTDPx points to the current transmit descriptor in memory. The CPU must write this register with the first descriptor address before enabling the SDMA transmit channel. When a SDMA transmit channel is enabled it will fetch the first descriptor pointed to by SCRDPx as part of its SDMA starting procedure.

13.6.3 SDMA First Transmit Descriptor Pointer (SFTDP)

SFTDPx points to the first descriptor in a transmit frame. The CPU must write this register with the first descriptor address before enabling the SDMA transmit channel. The SDMA transmit controller uses the SFTDP when it needs to restart a transmission after collision (HDLC mode only). The GT-96100A updates the content of SFTDP each time it fetches a descriptor with the F (first) bit set to ‘1’.

NOTE: The CPU must write the same value to both SCTDP and SFTDP before enabling the corresponding SDMA transmit channel.

13.7 Transmit SDMA

13.7.1 Transmit SDMA Definitions

- **SOF** (Start Of Frame descriptor): Descriptor with F (First) bit set to ‘1’.
- **EOF** (End Of Frame descriptor): Descriptor with L (Last) bit set to ‘1’.

F and L bits are set by the CPU before releasing a descriptor to the GT-96100A for transmission.

A frame starts with a SOF descriptor and ends with a EOF descriptor. A frame can consist of one buffer or split over many buffers. If a frame is stored in one buffer, the associated descriptor will have both the F and L bits set to '1'. In a non-frame oriented protocol (e.g. BISYNC or UART), it is recommended that both F and L bits be set to '1' for each buffer.

13.7.2 Transmit SDMA Flow

The following steps are executed during a normal transmit SDMA process:

1. Before enabling a SDMA Tx channel the CPU must prepare a valid descriptor with the owner bit set to '1'.
2. The CPU must then write the first descriptor address to both SCTDP and SFTDP registers.
3. The CPU issues a Transmit Demand command. The SDMA controller will then fetch the first descriptor and will start the SDMA process.
4. When buffer transmission is completed, the SDMA will close the buffer descriptor by setting the correct transmit status and writing '0' in the Owner Bit, returning the buffer to the CPU.

13.7.3 Retransmit in HDLC (LAP-D) mode

When working in collision mode (see MPSC section), the GT-96100A retransmits if collision occurs before the SDMA fetches the 3rd descriptor. If the frame consists of more than two buffers, the user must assure that there is enough data in the first two buffers to compensate for this behavior. The GT-96100A can buffer up to 256 bytes in its internal Tx FIFO. This should be considered when preparing a LAP-D transmit frame.

13.7.4 Transmit SDMA Notes

The transmit SDMA process is *frame oriented*.

The Transmit SDMA does not clear the frame's first descriptor ownership bit until the last descriptor associated with this frame is closed. The transmit SDMA then writes '0' to the first descriptor Owner bit and generate an interrupt if the EI bit of the first descriptor is set.

The transmit SDMA stops the DMA process whenever it reaches a descriptor with NULL (0x00000000) value in the NextDescriptorPointer (NDP) field or when it fetches a descriptor with Owner Bit set to '0'.

When the transmit SDMA controller encounters a NULL NDP value or a Not-Owned descriptor with its First field bit set, after the last descriptor of a frame, the transmit idles. The TxD bit is cleared and the transmit SDMA controller return to IDLE state.

When the transmit SDMA controller encounter a NULL NDP value, or a Not-Owned descriptor with its First field bit set, in the middle of a frame or a Not-Owned descriptor with its First field bit reset, after the last descriptor of a frame, the transmit aborts. The TxD bit is cleared, a Tx RESOURCE ERROR maskable interrupt is generated and the transmit SDMA controller return to IDLE state.

When the transmit SDMA controller encounters a Not-Owned descriptor with its First field bit reset, in the middle of a frame, the transmit stops. The TxD bit is not cleared and the transmit SDMA controller waits for an Abort transmit command. In such cases, the SDMA controller clears the TxD bit before returning to IDLE state.

In normal operation, the transmit SDMA never expects to find a NULL NextDescriptorPointer or Not-Owned descriptor in the middle of a frame. When this occurs, the transmit SDMA controller aborts, the TxD bit is cleared and a Tx RESOURCE ERROR maskable interrupt is generated.

NOTE: In collision mode, if a collision occurs exactly one clock cycle after a resource error, the GT-96100A ignores the resource error and retransmit the frame.

When the CPU needs to interfere with the transmit process without corrupting the ongoing transmit process, it can issue a STOP command by writing ‘1’ to the STD bit in the SDMA command register. The transmit SDMA controller stops after completing the transmission of the active frame.

When issuing an STD command TXD is reset to ‘0’ upon entering IDLE state. The CPU can then issue a new Transmit Demand command to restart the SDMA process.

13.8 Receive SDMA

13.8.1 Receive SDMA Definitions

Table 319: SDMA Definitions

Term	Definition
SOF	Start Of Frame descriptor Descriptor with F (First) bit set to ‘1’.
EOF	End Of Frame descriptor Descriptor with L (Last) bit set to ‘1’.

F and L bits are set by the CPU before releasing a descriptor to the GT-96100A.

A frame starts with an SOF descriptor and ends with an EOF descriptor. A frame can be contained in one buffer or split over many buffers. If a frame is stored in one buffer, the associated descriptor will have both F and L bits set to ‘1’.

13.8.2 Receive SDMA Flow

The following steps are executed during a normal transmit SDMA process:

1. Before enabling a SDMA Rx channel the CPU must prepare a valid descriptor with the owner bit set to ‘1’.
2. The CPU must then write the descriptor address to the SCRDP register before enabling the receive SDMA channel.
3. The CPU writes ‘1’ to the ERD bit in the SDCM register, enabling the receive SDMA channel.
4. Normally the receive SDMA controller will then run continuously, processing received data from the MPSC.

NOTES:The receive SDMA controller never expects to encounter a descriptor with owner bit set to ‘0’ or a NULL value (0x00000000) in the NDP field. If this occurs, the receive SDMA aborts and a maskable Rx RESOURCE ERROR interrupt is generated.
Use the receive abort command for the CPU to stop the receive SDMA. It is the CPU’s responsibility to properly restart the descriptor chain.

13.9 SDMA Interrupt and Mask register (SDI and SDM)

Each SDMA channel has two maskable interrupt sources. One is for Resource Error events and the other one is for descriptor closed events.

13.9.1 Resource Error Interrupt

When a receive SDMA encounters a NULL descriptor pointer or a not owned descriptor, a Resource Error interrupt is generated. A Resource Error interrupt is generated whenever a transmit SDMA encounters a NULL descriptor pointer or a not-owned descriptor in a middle of a frame.

NOTE: When the GT-96100A encounters a descriptor with Owner bit set to 0, it still expects to find that all the other fields of the descriptor are legitimate. A descriptor with Owner bit set to 0, with non-legitimate fields (such as Start Of Frame descriptor with F (First) bit not set to '1') can lead to unpredictable behavior.

13.9.2 Descriptor/Frame Closed Interrupt

When a SDMA channel closes a descriptor with the EI (Enable Interrupt) bit set to '1', a Descriptor Closed interrupt is generated.

NOTES: In case the RIFB bit is set in the SDMA configuration register, an interrupt is generated by the Rx channel only on receive frame boundaries.

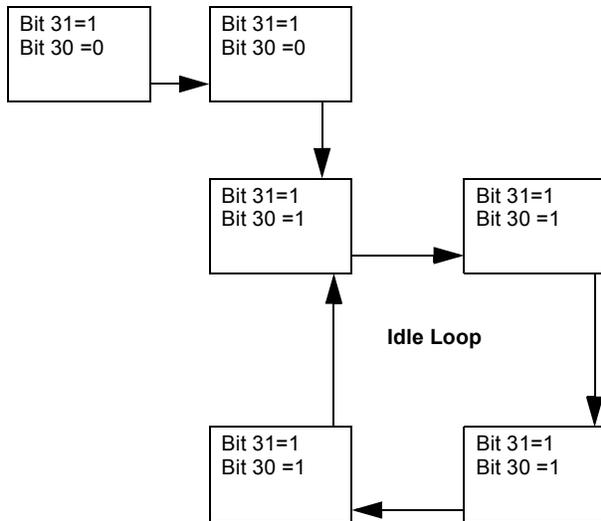
The correct operation of the frame level interrupt requires all Rx descriptors to have their EI bit set.

13.10 SDMA in Auto Mode

The CPU can set bit 30 in the command/status field of transmit or receive descriptors directing the GT-96100A to work in Auto Mode.

When working with an Auto Mode descriptor, the GT-96100A SDMA works as usual except that it does not clear the Ownership bit when closing the descriptor. The CPU can use this for example to cause the GT-96100A to transmit endlessly (until CPU intervention).

Figure 56: Using Auto Mode to Create Idle Loop



13.11 SDMA Registers

Table 320: SDMA Group 0 Register Map

Description	Offset	Page Number
SDMA Group Configuration Register	0x101AF0	page 315
Channel0		
Channel0 Configuration Register (S0DC0)	0x000900	page 312
Channel0 Command Register (S0DCM0)	0x000908	page 314
Channel0 Rx Descriptor	0x008900 - 0x00890F	Not to be accessed during normal operation.
Channel0 Current Rx Descriptor Pointer (S0CRDP0)	0x008910	page 316
Channel0 Tx Descriptor	0x00C900 - 0x00C90F	Not to be accessed during normal operation.
Channel0 Current Tx Descriptor Pointer (S0CTDP0)	0x00C910	page 316
Channel0 First Tx Descriptor Pointer (S0FTDP0)	0x00C914	page 316

Table 320: SDMA Group 0 Register Map (Continued)

Description	Offset	Page Number
Channel1		
Channel1 Configuration Register (S0DC1)	0x010900	For a description of the Channel1 registers, see the descriptions for the Channel 0 registers.
Channel1 Command Register (S0DCM1)	0x010908	
Channel1 Rx Descriptor	0x018900 - 0x01890F	
Channel1 Current Rx Descriptor Pointer (S0CRDP1)	0x018910	
Channel1 Tx Descriptor	0x01C900 - 0x01C90F	
Channel1 Current Tx Descriptor Pointer (S0CTDP1)	0x01C910	
Channel1 First Tx Descriptor Pointer (S0FTDP1)	0x01C914	
Channel2		
Channel2 Configuration Register (S0DC2)	0x020900	For a description of the Channel2 registers, see the descriptions for the Channel 0 registers.
Channel2 Command Register (S0DCM2)	0x020908	
Channel2 Rx Descriptor	0x028900 - 0x02890F	
Channel2 Current Rx Descriptor Pointer (S0CRDP2)	0x028910	
Channel2 Tx Descriptor	0x02C900 - 0x02C90F	
Channel2 Current Tx Descriptor Pointer (S0CTDP2)	0x02C910	
Channel2 First Tx Descriptor Pointer (S0FTDP2)	0x02C914	
Channel3		
Channel3 Configuration Register (S0DC3)	0x030900	For a description of the Channel3 registers, see the descriptions for the Channel 0 registers.
Channel3 Command Register (S0DCM3)	0x030908	
Channel3 Rx Descriptor	0x038900 - 0x03890F	
Channel3 Current Rx Descriptor Pointer (S0CRDP3)	0x038910	
Channel3 Tx Descriptor	0x03C900 - 0x03C90F	
Channel3 Current Tx Descriptor Pointer (S0CTDP3)	0x03C910	
Channel3 First Tx Descriptor Pointer (S0FTDP3)	0x03C914	

Table 320: SDMA Group 0 Register Map (Continued)

Description	Offset	Page Number
<i>Channel4</i>		
Channel4 Configuration Register (S0DC4)	0x040900	For a description of the Channel4 registers, see the descriptions for the Channel 0 registers.
Channel4 Command Register (S0DCM4)	0x040908	
Channel4 Rx Descriptor	0x048900 - 0x04890F	
Channel4 Current Rx Descriptor Pointer (S0CRDP4)	0x048910	
Channel4 Tx Descriptor	0x04C900 - 0x04C90F	
Channel4 Current Tx Descriptor Pointer (S0CTDP4)	0x04C910	
Channel4 First Tx Descriptor Pointer (S0FTDP4)	0x04C914	
<i>Channel5</i>		
Channel5 Configuration Register (S0DC5)	0x050900	For a description of the Channel5 registers, see the descriptions for the Channel 0 registers.
Channel5 Command Register (S0DCM5)	0x050908	
Channel5 Rx Descriptor	0x058900 - 0x05890F	
Channel5 Current Rx Descriptor Pointer (S0CRDP5)	0x058910	
Channel5 Tx Descriptor	0x05C900 - 0x05C90F	
Channel5 Current Tx Descriptor Pointer (S0CTDP5)	0x05C910	
Channel5 First Tx Descriptor Pointer (S0FTDP5)	0x05C914	
<i>Channel6</i>		
Channel6 Configuration Register (S0DC6)	0x060900	For a description of the Channel6 registers, see the descriptions for the Channel 0 registers.
Channel6 Command Register (S0DCM6)	0x060908	
Channel6 Rx Descriptor	0x068900 - 0x06890F	
Channel6 Current Rx Descriptor Pointer (S0CRDP6)	0x068910	
Channel6 Tx Descriptor	0x06C900 - 0x06C90F	
Channel6 Current Tx Descriptor Pointer (S0CTDP6)	0x06C910	
Channel6 First Tx Descriptor Pointer (S0FTDP6)	0x06C914	

Table 320: SDMA Group 0 Register Map (Continued)

Description	Offset	Page Number
Channel7		
Channel7 Configuration Register (S0DC7)	0x070900	For a description of the Channel7 registers, see the descriptions for the Channel 0 registers.
Channel7 Command Register (S0DCM7)	0x070908	
Channel7 Rx Descriptor	0x078900 - 0x07890F	
Channel7 Current Rx Descriptor Pointer (S0CRDP7)	0x078910	
Channel7 Tx Descriptor	0x07C900 - 0x07C90F	
Channel7 Current Tx Descriptor Pointer (S0CTDP7)	0x07C910	
Channel7 First Tx Descriptor Pointer (S0FTDP7)	0x07C914	

Table 321: SDMA Group 1 Register Map

Description	Offset	Page Number
Channel0		
Channel0 Configuration Register (S1DC0)	0x100900	page 312
Channel0 Command Register (S1DCM0)	0x100908	page 314
Channel0 Rx Descriptor	0x108900 - 0x10890F	Not to be accessed during normal operation.
Channel0 Current Rx Descriptor Pointer (S1CRDP0)	0x108910	page 316
Channel0 Tx Descriptor	0x10C900 - 0x10C90F	Not to be accessed during normal operation.
Channel0 Current Tx Descriptor Pointer (S1CTDP0)	0x10C910	page 316
Channel0 First Tx Descriptor Pointer (S1FTDP0)	0x10C914	page 316
Channel1		
Channel1 Configuration Register (S1DC1)	0x110900	For a description of the Channel1 registers, see the descriptions for the Channel 0 registers.
Channel1 Command Register (S1DCM1)	0x110908	
Channel1 Rx Descriptor	0x118900 - 0x11890F	
Channel1 Current Rx Descriptor Pointer (S1CRDP1)	0x118910	
Channel1 Tx Descriptor	0x11C900 - 0x11C90F	
Channel1 Current Tx Descriptor Pointer (S1CTDP1)	0x11C910	
Channel1 First Tx Descriptor Pointer (S1FTDP1)	0x11C914	

Table 321: SDMA Group 1 Register Map (Continued)

Description	Offset	Page Number
Channel2		
Channel2 Configuration Register (S1DC2)	0x120900	For a description of the Channel2 registers, see the descriptions for the Channel 0 registers.
Channel2 Command Register (S1DCM2)	0x120908	
Channel2 Rx Descriptor	0x128900 - 0x12890F	
Channel2 Current Rx Descriptor Pointer (S1CRDP2)	0x128910	
Channel2 Tx Descriptor	0x12C900 - 0x12C90F	
Channel2 Current Tx Descriptor Pointer (S1CTDP2)	0x12C910	
Channel2 First Tx Descriptor Pointer (S1FTDP2)	0x12C914	
Channel3		
Channel3 Configuration Register (S1DC3)	0x130900	For a description of the Channel3 registers, see the descriptions for the Channel 0 registers.
Channel3 Command Register (S1DCM3)	0x130908	
Channel3 Rx Descriptor	0x138900 - 0x13890F	
Channel3 Current Rx Descriptor Pointer (S1CRDP3)	0x138910	
Channel3 Tx Descriptor	0x13C900 - 0x13C90F	
Channel3 Current Tx Descriptor Pointer (S1CTDP3)	0x13C910	
Channel3 First Tx Descriptor Pointer (S1FTDP3)	0x13C914	
Channel4		
Channel4 Configuration Register (S1DC4)	0x140900	For a description of the Channel4 registers, see the descriptions for the Channel 0 registers.
Channel4 Command Register (S1DCM4)	0x140908	
Channel4 Rx Descriptor	0x148900 - 0x14890F	
Channel4 Current Rx Descriptor Pointer (S1CRDP4)	0x148910	
Channel4 Tx Descriptor	0x14C900 - 0x14C90F	
Channel4 Current Tx Descriptor Pointer (S1CTDP4)	0x14C910	
Channel4 First Tx Descriptor Pointer (S1FTDP4)	0x14C914	

Table 321: SDMA Group 1 Register Map (Continued)

Description	Offset	Page Number
Channel5		
Channel5 Configuration Register (S1DC5)	0x150900	For a description of the Channel5 registers, see the descriptions for the Channel 0 registers.
Channel5 Command Register (S1DCM5)	0x150908	
Channel5 Rx Descriptor	0x158900 - 0x15890F	
Channel5 Current Rx Descriptor Pointer (S1CRDP5)	0x158910	
Channel5 Tx Descriptor	0x15C900 - 0x15C90F	
Channel5 Current Tx Descriptor Pointer (S1CTDP5)	0x15C910	
Channel5 First Tx Descriptor Pointer (S1FTDP5)	0x15C914	
Channel6		
Channel6 Configuration Register (S1DC6)	0x160900	For a description of the Channel6 registers, see the descriptions for the Channel 0 registers.
Channel6 Command Register (S1DCM6)	0x160908	
Channel6 Rx Descriptor	0x168900 - 0x16890F	
Channel6 Current Rx Descriptor Pointer (S1CRDP6)	0x168910	
Channel6 Tx Descriptor	0x16C900 - 0x16C90F	
Channel6 Current Tx Descriptor Pointer (S1CTDP6)	0x16C910	
Channel6 First Tx Descriptor Pointer (S1FTDP6)	0x16C914	
Channel7		
Channel7 Configuration Register (S1DC7)	0x170900	For a description of the Channel7 registers, see the descriptions for the Channel 0 registers.
Channel7 Command Register (S1DCM7)	0x170908	
Channel7 Rx Descriptor	0x178900 - 0x17890F	
Channel7 Current Rx Descriptor Pointer (S1CRDP7)	0x178910	
Channel7 Tx Descriptor	0x17C900 - 0x17C90F	
Channel7 Current Tx Descriptor Pointer (S1CTDP7)	0x17C910	
Channel7 First Tx Descriptor Pointer (S1FTDP7)	0x17C914	

14. MULTI PROTOCOL SERIAL CONTROLLER (MPSC)

The GT-96100A includes eight MPSCs that support:

- Bit oriented protocols (e.g. HDLC)
- Byte oriented protocols (e.g. BISYNC)
- Transparent protocols
- The UART (Start/Stop) mode.

All eight MPSCs can operate simultaneously. Four MPSCs can operate up to a guaranteed bit rate of 55Mbps while the remaining MPSCs can work up to guaranteed bit rate of 2Mbps (in NRZ/NRZI).

All eight MPSCs can be routed out via serial interface ports which implement interfaces like EIA-232 and V.34.

Alternatively, the MPSCs can be connected to the GT-96100A's FlexTDMs. A FlexTDM can also be used to connect the various MPSCs to a PCM highway bus or any other time slot assigner bus. See [Section 15. "FlexTDM Units \(FTDM\)" on page 379](#) for a description of the FlexTDM.

14.1 DPLL

Each MPSC has a dedicated transmit and receive digital phase lock loop (DPLL).

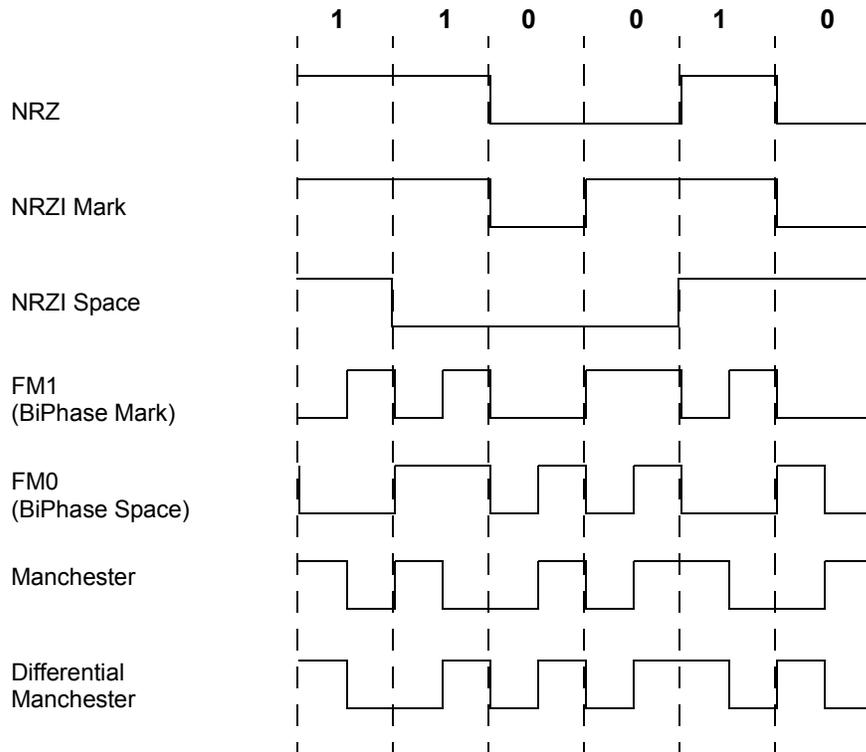
The transmit DPLL encodes the transmit bit stream to the selected code and monitors the transmit clock for glitches. If a clock glitch is detected and the Glitch Detect Enable (GDE) bit in the Main Configuration register (MMCR) is set to '1', a maskable interrupt is generated.

The receive DPLL decodes the incoming bit stream according to the selected mode. If a code violation is detected (for example, no transition in Manchester code) the DE (Decoding Error) in the receive descriptor is set. The receive DPLL also performs clock recovery from the incoming bit stream and monitors the receive clock for glitches. If a clock glitch is detected and the Glitch Detect Enable (GDE) bit in the Main Configuration register (MMCR) is set to '1', a maskable interrupt is generated.

14.1.1 Data Encoding/Decoding

Figure 57 shows the data encoding and decoding schemes The GT-96100A DPLL supports.

Figure 57: MPSC DPLL Encoding/Decoding Schemes



14.1.2 DPLL Clock Source

Each received DPLL uses the MPSC receive clock input and each transmit DPLL uses the MPSC transmit clock input as its source clock.

NOTE: The GT-96100A DPLLs can accept a clock source of up to 83MHz. This allows the GT-96100A to have a bit rate of up to 5MHz using a 16X clock rate scheme.

14.2.1 MPSCx Main Configuration Register Low (MMCRLx)

Table 322: MPSCx Main Configuration Register Low (MMCRLx), Offset: 0x000A00, 0x008A00, 0x010A00, 0x018A00, 0x020A00, 0x028A00, 0x030A00, 0x038A00 (where x is the port number 0 to 7)

Bits	Field Name	Function	Initial Value
2:0	MODE	Mode 000 -HDLC (default) 001 -Reserved 010 -Reserved 011 -Reserved 100 -UART 101 -BISYNC 110 -Reserved 111 -Reserved	0
3	TTX	Transparent Transmitter 0 - Normal Mode. (default) 1 - Transparent Mode. (Transparent Mode overrides the program mode in MODE bits.)	0
4	TRX	Transparent Receiver 0 - Normal Mode (default) 1 - Transparent Mode. (Transparent Mode overrides the program mode in MODE bits.)	0
5		Reserved.	0
6	ET	Enable Transmit 0 - Disabled. The Tx channel is in Low Power Mode. 1 - Enable. The Tx controller is ready for data. When the SDMA has data to transmit it loads the data to the Tx controller, that will transmit the data in the selected protocol.	0
7	ER	Enable Receive 0 - Disabled. The Rx channel is in Low Power Mode. 1 - Enable. The Rx controller is ready to receive data.	0
9:8	LPBK	Loop Back (for diagnostic) mode 00 -Normal Operation, no loopback (Default) 01 -Loopback 10 -Echo 11 -Loop Back + Echo In loopback mode, which is only for diagnostic purposes, the transmitted data on TxD is fed into RxD. In this mode, the same clock source should be used for both Rx and TX. Echo mode re-transmits received data on RxD (with one clock delay) on TxD. If CD* is asserted, the receiver also receives the incoming data.	00

Table 322: MPSCx Main Configuration Register Low (MMCRLx), Offset: 0x000A00, 0x008A00, 0x010A00, 0x018A00, 0x020A00, 0x028A00, 0x030A00, 0x038A00 (where x is the port number 0 to 7) (Continued)

Bits	Field Name	Function	Initial Value
10	NLM	Null Modem 0 - Normal operation. The MPSC uses the CD* and CTS* inputs to control the data flow 1 - Null Modem. The MPSC CD* and RTS* internal signals are always asserted. The external pin status can be still read from the Event Register. NOTE: For information about the behavior of the Event Register in different modes, see: <ul style="list-style-type: none"> • Section “The ESR register holds information on the transmit/receive channel condition.” on page 345. • Section 14.6.5.6 “CHR10 - BISYNC Event Status Register (ESR)” on page 360. • Section 14.7.5.7 “CHR10 - UART Event Status Register (ESR)” on page 371. • Section 14.8.2.3 “CHR10 - Transparent Event Status Register (ESR)” on page 378. 	0
11		Reserved.	0
12	TSYN	Transmitter Synchronize to Receiver Setting this bit synchronizes the transmitter to receiver byte boundaries. This is particularly important in the X.21 protocol. 0 - No synchronization assumed. 1 - Transmit bit stream is synchronized to the receive bit stream. This bit affects only a transparent transmitter. Transmitter will start transmission nx8 bit period after the receive data arrives. If CTS* is already asserted, the transparent transmitter will start transmit 8 clocks after the receiver starts to receive data. NOTE: Only this bit when transmit and receive clocks are equal and TCDV and RCDV are set to '00'.	0
13		Reserved.	0

Table 322: MPSCx Main Configuration Register Low (MMCRLx), Offset: 0x000A00, 0x008A00, 0x010A00, 0x018A00, 0x020A00, 0x028A00, 0x030A00, 0x038A00 (where x is the port number 0 to 7) (Continued)

Bits	Field Name	Function	Initial Value
15:14	TSNS	<p>Transmit Sense.</p> <p>Defines the number of bit times the internal sense signal will stay active after last transition on the RXD line occurs. It is useful for AppleTalk protocol to avoid the spurious CD* change interrupt that would otherwise occur during the frame synchronization sequence that precedes the opening flag. The delay is a function of RCDV (clock divider) setting.</p> <p>00 (RCDV = 0) - Infinite (Carrier Sense is always active - default) 00 (RCDV ≠ 0) - Infinite (Carrier Sense is always active - default) 01 (RCDV = 0) - 14 bit times 01 (RCDV ≠ 0) - 6.5 bit times 10 (RCDV = 0) - 4 bit times (normal AppleTalk) 10 (RCDV ≠ 0) - 2.5 bit times (normal AppleTalk) 11 (RCDV = 0) - 3 bit times 11 (RCDV ≠ 0) - 1 bit time</p>	0
16	TIDL	<p>Transmit Idles</p> <p>0 - TxD is encoded during data transmission (including preamble and flags/ sync patterns). TxD is in MARK during idle. (Default.) 1 - TxD is encoded all the time, even when idles are transmitted. Table 323 .</p>	0
17	RTSM	<p>RTS* Mode</p> <p>This bit may be changed on the fly.</p> <p>0 - Send IDLE between frames. RTS* is negated between frames. IDLE pattern is defined by the protocol and TIDL bit. 1- Send flags/syncs between frames according to the protocol. RTS* is always asserted. Refer to Table 323 .</p>	0
18		Reserved.	0
19	CTSS	<p>CTS* Sampling Mode</p> <p>0 - Asynchronous CTS*. CTS* is synchronized inside the GT-96100A. Transmission starts after synchronization is achieved with a few cycles delay to the external CTS*. (Default) 1 - Synchronous CTS*. CTS* is synchronized to the Rx clock. This mode is recommended when connecting an MPSC to a FlexTDM. NOTE: Synchronous CTS* must be used for ISDN D channels.</p>	0
20	CDS	<p>CD* Sampling mode</p> <p>0 - Asynchronous CD*. CD* is synchronized internally in the GT-96100A and then data is received. (Default) 1 - Synchronous CD*. CD* is synchronized to the Rx clock. This mode is recommended when connecting an MPSC to a Flex-TDM.</p>	0

Table 322: MPSCx Main Configuration Register Low (MMCRLx), Offset: 0x000A00, 0x008A00, 0x010A00, 0x018A00, 0x020A00, 0x028A00, 0x030A00, 0x038A00 (where x is the port number 0 to 7) (Continued)

Bits	Field Name	Function	Initial Value
21	CTSM	<p>CTS* Operating Mode</p> <p>0 - Normal mode. (Envelop Mode). CTS* should envelop the frame. Deassertion of CTS* during transmission will cause a CTS lost error.</p> <p>1- Pulse Mode. Once CTS* is sampled low, synchronization has been achieved. Further transitions of CTS* have no effect. CTS* synchronization will be lost when RTS* is deasserted.</p>	0
22	CDM	<p>CD* Operating Mode</p> <p>0- Normal mode. (Envelop Mode). CD* should envelop the frame. Deassertion of CD* during reception will cause a CD lost error.</p> <p>1- Pulse Mode. Once CD* is sampled low, synchronization has been achieved. Further transitions of CD* have no effect.</p>	0
25:23	CRCM	<p>CRC Mode</p> <p>000 - CRC16-CCITT (HDLC based protocols, e.g. X.25) (Default)</p> <p>001 - CRC-16 (BISYNC)</p> <p>010 - CRC32-CCITT (HDLC based protocols, e.g. LAP-D. Identical to the Ethernet CRC)</p> <p>011 - Reserved</p> <p>1XX- Reserved</p>	010
27:26		Reserved.	0
28	TRVD	<p>Transmit Reverse Data</p> <p>0 - Normal Mode. (Default)</p> <p>1 - Reverse Data Mode. MSB is shifted out first.</p>	0
29	RRVD	<p>Receive Reverse Data</p> <p>0 - Normal Mode. (Default)</p> <p>1 - Reverse Data Mode. MSB is shifted in first.</p>	0
30		Reserved.	0
31	GDE	<p>Glitch Detect Enable</p> <p>0 - Normal mode. No glitch detect. (Default)</p> <p>1 - When glitch is detect, a maskable interrupt is generated.</p> <p>When this bit is set the MPSC looks for glitches in the external receive and transmit clocks.</p> <p>NOTE: The GT-96100A tries to clean the input clocks by receiving them via a Schmitt trigger input buffer.</p>	0

The following table summarizes the relationship between the TIDL and RTSM

Table 323: TIDL/RTSM Relationship

RTSM/TIDL	TxD	RTS*	TxD	RTS*
00	'1' Not Encoded	1	Data Encoded	0
01	'1' Encoded	1	Data Encoded	0
10	Flags/Not Encoded	0	Data Encoded	0
11	Flags/Encoded	0	Data Encoded	0

14.2.2 MPSCx Main Configuration Register High (MMCRHx)

Table 324: MPSCx Main Configuration Register High (MMCRHx), Offset: 0x000A04, 0x008A04, 0x010A04, 0x018A04, 0x020A04, 0x028A04, 0x030A04, 0x038A04 (where x is the port number 0 to 7)

Bits	Field Name	Function	Initial Value
0	TCI	Transmit Clock Invert 0 - Normal operation - Data is shifted out on the falling edge. (Default.) 1 -The internal transmit clock is inverted by the MPSC before it is used. This allows the MPSC to clock data out half a cycle earlier on the rising edge of the clock.	0
1	TINV	Transmit bit stream inversion 0 - No invert. 1 - Invert the data before it is sent to the DPLL. Setting TINV to '1' generates FM1 from FM0, NRZI mark from NRZI space, etc. It also inverts the bit stream in NRZ mode.	0
4:2	TPL	Transmit Preamble Length Determines the number of preamble bytes the transmitter sends before it starts to transmit data. The send pattern is defined by the TPPT bits. 000 - No Preamble (Default) 001 - 1 byte 010 - 2 bytes 011 - 4 bytes 100 - 6 bytes 101 - 8 bytes 110 - 16 bytes 111 - Reserved	0
8:5	TPPT	Transmit Preamble Pattern Defines a character sent as a preamble sequence. Two TPPT characters form a preamble byte. The number of preamble bytes sent is defined by the TPL field. The receiving DPLL uses the preamble pattern to lock on the receiving signal.	0

Table 324: MPSCx Main Configuration Register High (MMCRHx), Offset: 0x000A04, 0x008A04, 0x010A04, 0x018A04, 0x020A04, 0x028A04, 0x030A04, 0x038A04 (where x is the port number 0 to 7) (Continued)

Bits	Field Name	Function	Initial Value
10:9	TCDV	<p>Transmit Clock Divider</p> <p>Defines the transmit clock divider. The transmit bit rate is the rate of the clock entering the MPSC Tx machine (from external pin or a BRG) divided by the TCDV field. For FM0, FM1, Manchester, and Differential Manchester, one of the 8x, 16x, or 32x options must be set.</p> <p>00 - 1x clock mode (Default. For NRZ and NRZI only.) 01 - 8x clock mode 10 - 16x clock mode 11 - 32x clock mode</p>	0
13:11	TDEC	<p>Transmit Encoder</p> <p>Specifies the encoding method for the dedicated Tx channel DPLL.</p> <p>000 - NRZ (default) 001 - NRZI (mark, can be set to Space by setting TINV bit) 010 - FM0 (can be set to FM1 by setting the TINV bit) 011 - Reserved 100 - Manchester 101 - Reserved 110 - Differential Manchester 111 - Reserved</p>	0
15:14		Reserved.	0
16	RINV	<p>Receive Bit Stream Inversion.</p> <p>0 - No invert. 1 - Inverts the data before it is sent from the DPLL to the MPSC data path. Setting RINV to '1' decodes FM1 and NRZI mark when the RENC field is programmed to FM0 and NRZI space etc. It also inverts the received bit stream in NRZ mode.</p>	0
20:17	GDW	<p>Clock Glitch Width</p> <p>When the GDE bit is set, the MPSC will consider Tx/Rx clock pulses that are narrower than GDW system clocks as a glitch.</p>	0
21		Reserved.	0

Table 324: MPSCx Main Configuration Register High (MMCRHx), Offset: 0x000A04, 0x008A04, 0x010A04, 0x018A04, 0x020A04, 0x028A04, 0x030A04, 0x038A04 (where x is the port number 0 to 7) (Continued)

Bits	Field Name	Function	Initial Value
22	RDW	<p>Receive Data Width</p> <p>0 - Normal mode. The MPSC data path is 16 bits wide. Upon receiving 16 bits, the data is transferred into the SDMA FIFOs. Buffers must be 64-bit word aligned. DMA bursts are enabled.</p> <p>NOTE: Normal Mode must be used for HDLC based protocols.</p> <p>1 - Low latency operation. Data is transferred to the FIFOs after 8 bits are received. Logical FIFO width is one byte.</p> <p>NOTE: This mode allows byte aligned buffers. This mode must be chosen for BISYNC and UART modes. DMA bursts are disabled. The SDMA writes one byte per DRAM access. Setting RDW also bypasses the receive FIFO threshold. The SDMA arbitrates for DMA access as soon as the FIFO has one byte in it.</p>	0
24:23	RSYL	<p>Receive Sync Length (BISYNC and Transparent Modes)</p> <p>00 - External sync (CD* assertion)</p> <p>01 - 4-bit sync</p> <p>10 - 8-bit sync (MonoSYNC)</p> <p>11 - 16-bit sync (BISYNC)</p>	0
26:25	RCDV	<p>Receive Clock Divider</p> <p>Defines the receive clock divider. The receive bit rate is the rate of the clock entering the MPSC Rx machine (from external pin or a BRG) divided by the RCDV field. For FM0, FM1, Manchester, and Differential Manchester, one of the 8x, 16x, or 32x options must be set.</p> <p>00 - 1x clock mode (Default. For NRZ and NRZI only.)</p> <p>01 - 8x clock mode</p> <p>10 - 16x clock mode</p> <p>11 - 32x clock mode</p>	0
29:27	RENC	<p>Receive Encoder</p> <p>Specifies the encoding method for the dedicated Rx channel DPLL.</p> <p>000 - NRZ (default)</p> <p>001 - NRZI (Mark, can be set to Space by setting RINV bit)</p> <p>010 - FM0 (can be set to FM1 by setting the RINV bit)</p> <p>011 - Reserved</p> <p>100 - Manchester</p> <p>101 - Reserved</p> <p>110 - Differential Manchester</p> <p>111 - Reserved</p>	0

Table 324: MPSCx Main Configuration Register High (MMCRHx), Offset: 0x000A04, 0x008A04, 0x010A04, 0x018A04, 0x020A04, 0x028A04, 0x030A04, 0x038A04 (where x is the port number 0 to 7) (Continued)

Bits	Field Name	Function	Initial Value
31:30	SEDC	Synchronization Clock Edge The clock edge used by the DPLL for adjusting the receive sample point due to drift in the receive signal. 00 - Both rising and falling edges. (Default.) 01 - Rising edge 10 - Falling edge 11 - No adjustment	0

14.3 MPSCx Protocol Configuration Registers (MPCRx)

Each MPSC has a dedicated Protocol Configuration Register (MPCRx).

The MPCRx registers are located at base+08 relative to the corresponding MPSC Main Configuration Register (MMCRx). The functionality of the MPCRx is protocol dependent. Detailed descriptions of the MPCRx are given in the following protocol sections.

14.4 Channel Registers (CHxRx)

Each MPSC and the ethernet controller has ten dedicated Channel Registers (CHxRx) to program the MPSC or ethernet controller.

The CHxRx registers are located at base+0xC0 through base+0x30 relative to the corresponding MPSC Main Configuration Register (MMCRx). The functionality of the CHxRx is protocol dependent. Detailed descriptions of the CHxRx are given in the following protocol sections.

14.5 HDLC Mode

14.5.1 HDLC Receive/Transmit Operation

In HDLC mode, an MPSC performs the following protocol functions:

- Flag generation and stripping
- Bit stuffing and stripping
- Address recognition (up to 16 bit addresses)
- CRC generation and checking
- Line condition monitoring
- LocalTalk preamble generation
- LocalTalk trailing abort generation

Figure 59: Typical HDLC Frame

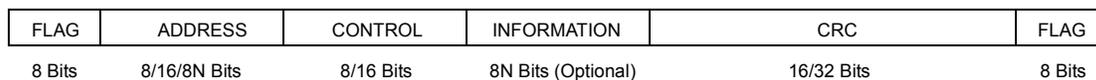
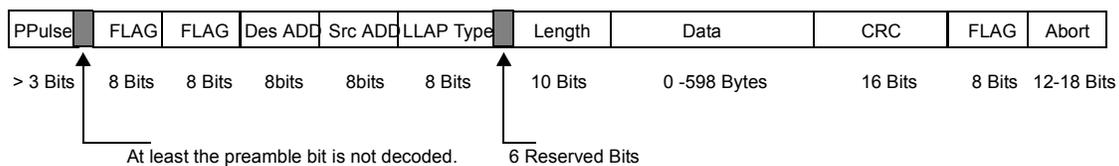


Figure 60: Typical LocalTalk Frame



14.5.2 SDMAx Command/Status Field for HDLC Mode

When an MPSC is in HDLC mode, the Command/Status field in the corresponding SDMAx descriptor has the following format:

Table 325: SDMAx Command/Status Field for HDLC Mode

Bit	Rx - Function	Tx - Function
0	CE - CRC Error	Reserved
1	CDL - CD Loss	CTSL - CTS Loss
2	DE - Decoding Error	Reserved
3	NO - Non Octet Frame	D-deferred. Transmission was deferred due to busy channel.
4	ABR - Abort Sequence/Residue[0]	Reserved
5	Residue[1]	Reserved
6	OR - Data Overrun/Residue[2]	UR - Data Underrun
7	MFL - Max Frame Length Err	Reserved
8	SF - Short Frame	RL - Retransmit Limit Error
9	Reserved	COL - Collision Occurred
13:10	Reserved	RC-Retransmit Count (LAN HDLC mode only)
14	Reserved	Reserved
15	ES - Error Summary ES = CE CDL DE NO ABR OR MFL SF ¹	Error Summary ES = CTSL UR RL ¹
16	L - last	L - Last
17	F - First	F - First
21:18	Reserved	Reserved
22	Reserved	GC - Generate CRC
23	EI - Enable Interrupt	EI - Enable Interrupt
29:24	Reserved	Reserved
30	AM - Auto Mode	AM - Auto Mode
31	O -Owner	O - Owner

1. “||” means logical OR.

14.5.3 MPSCx Protocol Configuration Register (MPCRx) for HDLC

Figure 61: MPSCx Protocol Configuration Register (MPCRx) for HDLC

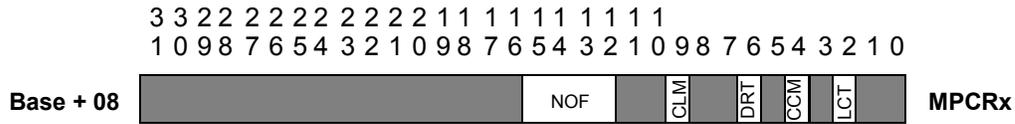


Table 326: MPSCx Protocol Configuration Register (MPCRx) for HDLC, Offset: 0x000A08, 0x008A08, 0x010A08, 0x018A08, 0x020A08, 0x028A08, 0x030A08, 0x038A08 (where x is the port number 0 to 7)

Bits	Field Name	Function	Initial Value
1:0		Reserved.	0
2	LCT	Local Talk When set, the following LocalTalk support is added to the HDLC controller: <ul style="list-style-type: none"> Two abort sequences will be generated at the end of frame following its closing flag. A preamble will be generated. No encoding will be done for the last preamble bit When working with LocalTalk, the FM0 Encoding Scheme should be set by writing '010' to RENC and TDEC in the MMCRx. The user should also set TPPT to 0xF and TPL to '1' (one byte preamble). The last preamble bit is not decoded. This must be done for LocalTalk RTS frames. Setting TPL to '0' leads to a frame without preamble. This can be used with LocalTalk data frames. Setting TPL to other values leads to unpredictable results.	0
3		Reserved.	0
4	CCM	CRC Compliance Mode. In HDLC, the TX side uses bit stuffing to prevent a data/CRC pattern from looking like an HDLC control flag. The CCM tells the Rx side how to handle frames that were received with mistakes in bit stuffing, when they occur immediately before the end flag. This is a borderline condition that may or may not present a problem in actual systems. 0 - Compatible Mode. If the Rx side receives a frame that is missing a stuffed bit that is supposed to be immediately before the End Flag, then mark in the descriptor that the frame has a good CRC, and pass the good CRC along to the buffer. 1 - Compliance Mode. If the Rx side receives a frame that is missing a stuffed bit that is supposed to immediately proceed the End Flag, then mark in the descriptor that the frame has a bad CRC, and pass the errored CRC to the buffer.	
5		Reserved.	0

Table 326: MPSCx Protocol Configuration Register (MPCRx) for HDLC, Offset: 0x000A08, 0x008A08, 0x010A08, 0x018A08, 0x020A08, 0x028A08, 0x030A08, 0x038A08 (where x is the port number 0 to 7) (Continued)

Bits	Field Name	Function	Initial Value
6	DRT	Disable Rx on Tx When DRT is set to '1' the Rx path is closed during Tx. This is useful in multidrop configurations when a user doesn't want to receive its own frames.	0
8:7		Reserved.	0
9	CLM	Collision Mode. When set to '1', the MPSC transceiver tries to retransmit a frame after a CTS lost. This mode allows automatic collision resolution for an ISDN LAP-D type channel.	0
11:10		Reserved.	0
15:12	NOF	Number of Flags Specifies the number of flags transmitted between consecutive frames. Setting NOF to '0' specifies shared flag mode. In shared flag mode, the closing flag of a frame is used as the opening flag of the following frame. This setting also puts the receiver in back-to-back mode. The default value is 1.	1
31:16		Reserved.	0

Unless otherwise specified:

- ‘1’ means set.
- ‘0’ means not set.
- ‘0’ is the default value after reset.

Table 327: CHxR1 - Sync/Abort Register (SYNR), Offset: 0x000A0C, 0x008A0C, 0x010A0C, 0x018A0C, 0x020A0C, 0x028A0C, 0x030A0C, 0x038A0C (where x is channel 0 to 7)

Bits	Field Name	Function	Initial Value
7:0	SYNC	Holds the synchronization pattern for the receive machine and opening/closing flag/sync-pattern for the transmit machine. This is an HDLC Abort Pattern so no additional programming is needed for the HDLC protocol.	7E
15:8		Reserved	0
23:16	Abort	The abort pattern is transmitted upon receiving an abort command or when an UnderRun event occurs. This is an HDLC Abort Pattern so no additional programming is needed for the HDLC protocol.	FE
31:24		Reserved	0

Table 328: CHxR2 - Command Register (CR), Offset: 0x000A10, 0x008A10, 0x010A10, 0x018A10, 0x020A10, 0x028A10, 0x030A10, 0x038A10 (where x is channel 0 and 7)

Bits	Field Name	Function	Reset Value
6:0		Reserved.	0
7	A	Abort Transmission Abort transmission immediately and go to IDLE. The descriptor is not closed or incremented. NOTE: Command is not synchronized to byte.	0
22:8		Reserved.	0
23	A	Abort Reception Abort receive immediately and go to IDLE. The descriptor is not closed or incremented. The processor must issue enter hunt command after abort command in order to enable reception. The bit is cleared upon entering IDLE state. After executing an Abort Reception, the CPU must disable the Tx SDMA channel. The CPU then needs to execute a normal initialization process to the MPSC.	0
30:24		Reserved.	0

Table 328: CHxR2 - Command Register (CR), Offset: 0x000A10, 0x008A10, 0x010A10, 0x018A10, 0x020A10, 0x028A10, 0x030A10, 0x038A10 (where x is channel 0 and 7)

Bits	Field Name	Function	Reset Value
31	EH	Enter Hunt Upon receiving the Enter Hunt command, the receive machine moves to HUNT state and continuously searches for an opening flag. If enter hunt mode command is issued during frame reception, the current descriptor is closed with CRC error ¹ . The EH bit is cleared upon entering Hunt state.	0
N/A	TD	Transmit Demand Fetch a descriptor and start transmission. Issued through the SDMAx Command Register.	
N/A	Stop	Stop Complete frame transmission and stop. (Go to IDLE). Issued through the SDMAx Command Register.	

1. The reception process for this purpose begins after proper address recognition is allowed. Before achieving an address match, the receiver goes to Enter Hunt state without closing the descriptor.

NOTES:The ET bit in the Main Configuration Register must be set to ‘1’ before issuing the following Transmit Demand, Stop Transmission, or Abort Transmission commands.

The ER bit in the Main Configuration Register must be set to ‘1’ before issuing the Enter Hunt or Abort Reception commands.

When the ET or ER bits are deasserted, the MPSCx transmit/receive channel is in low power mode (NO CLOCK). Issuing one of the above commands in this state will lead to unpredictable results.

Table 329: CHxR3 - Maximum Frame Length Register (MFLR), Offset: 0x000A14, 0x008A14, 0x010A14, 0x018A14, 0x020A14, 0x028A14, 0x030A14, 0x038A14 (where x is channel 0 and 7)

Bits	Field Name	Function	Initial Value
15:0	FLBR	Frame Length Buffer Register Holds the maximum allowed frame length. When a frame exceeds the number written in the FLBR, the remainder of the frame is discarded. The HDLC controller waits for a closing flag and then returns the frame status with bit 7 (MFLE) set to ‘1’.	0xFFFF
31:16		Reserved.	0

Table 330: CHxR4 - Address Filtering Register (ADFR), Offset: 0x000A18, 0x008A18, 0x010A18, 0x018A18, 0x020A18, 0x028A18, 0x030A18, 0x038A18 (where x is channel 0 and 7)

Bits	Field Name	Function	Initial Value
15:0	BCE	Bit Comparison Enable Bits Setting '1' in one of the BCE bits enables the address comparison for this bit: <ul style="list-style-type: none"> For 16-bit LAP-D like address recognition, write 0xFFFF in ADFR. For 8-bit HDLC/LAP-B like address recognition, write 0x00FF in ADFR. For reception of a predefined address group, write '0' to the appropriate bits to disable address comparison on these bits. 	0
28:16		Reserved.	0
29	N	Null Enable Enables the reception of HDLC NULL address (0x0000 or 0x00 depending on the BCE setting)	0
30		Reserved.	0
31	B	Broadcast Enable Enables the reception of HDLC broadcast address (0xFFFF or 0xFF, depending on the BCE setting).	0

Table 331: CHxR5 - Short Frame Register (SHFR), Offset: 0x000A1C, 0x008A1C, 0x010A1C, 0x018A1C, 0x020A1C, 0x028A1C, 0x030A1C, 0x038A1C (where x is channel 0 and 7)

Bits	Field Name	Function	Initial Value
2:0	SHFR	Short Frame Register Setting SHFR to '1' enables the Short Frame Error report. Short Frames are frames with byte count less than 3+SHFR.	0
31:3		Reserved.	0

Table 332: CHxR6 - Address 1 and 2 Register (ADLR), Offset: 0x000A20, 0x008A20, 0x010A20, 0x018A20, 0x020A20, 0x028A20, 0x030A20, 0x038A20 (where x is channel 0 and 7)

Bits	Field Name	Function	Reset Value
15:0	AD1	Address 1 A 16-bit address that can be used for receive address recognition.	0
31:16	AD2	Address 2 A 16-bit address used for receive address recognition.	0

Table 333: CHxR7 - Address 3 and 4 Register (ADHR), Offset: 0x000A24, 0x008A24, 0x010A24, 0x018A24, 0x020A24, 0x028A24, 0x030A24, 0x038A24 (where x is channel 0 and 7)

Bits	Field Name	Function	Reset Value
15:0	AD3	Address 3 A 16-bit address that can be used for receive address recognition.	0
31:16	AD4	Address 4 A 16-bit address that can be used for receive address recognition.	0

Table 334: CHxR8 - Reserved, Offset: 0x000A28, 0x008A28, 0x010A28, 0x018A28, 0x020A28, 0x028A28, 0x030A28, 0x038A28 (where x is channel 0 and 7)

Bits	Field Name	Function	Initial Value
31:0		Reserved. NOTE: Do not access this register in the HDLC mode.	0

Table 335: CHxR9 - Reserved, Offset: 0x000A2C, 0x008A2C, 0x010A2C, 0x018A2C, 0x020A2C, 0x028A2C, 0x030A2C, 0x038A2C (where x is channel 0 and 7)

Bits	Field Name	Function	Reset Value
31:0		Reserved. NOTE: Do not access this register in the HDLC mode.	0

The ESR register holds information on the transmit/receive channel condition.

CHR10 can be read by the CPU for channel condition resolution. Some changes in the channel condition can generate maskable interrupts, as shown below.

Table 336: CHxR10 - Event Status Register (ESR), Offset: 0x000A30, 0x008A30, 0x010A30, 0x018A30, 0x020A30, 0x028A30, 0x030A30, 0x038A30 (where x is channel 0 and 7)

Bits	Field Name	Function	Initial Value
0	CTS	Clear To Send Signal An interrupt is generated when this signal is deasserted during transmit.	0
1	CD	Carrier Detect Signal An interrupt is generated when this signal is deasserted during receive.	0
2		Reserved.	0
3	TIDLE	Tx in IDLE state. An interrupt is generated upon entering IDLE state.	0
4		Reserved.	0
5	RHS	Rx in HUNT state.	0
10:6		Reserved.	0
11	RLIDL	1 = Rx IDLE Line	0
12	DPCS	1 = DPLL Carrier Sense.	0
13	RRF	1 = Rx Receiving Flags.	0
31:14		Reserved.	0

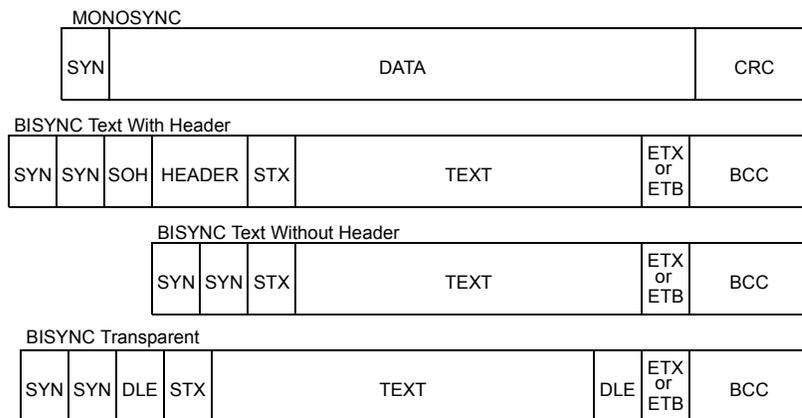
14.6 BISYNC Mode

The GT-96100A BISYNC controller was designed to reduce CPU overhead by executing most of the protocol requirements for BISYNC/MonoSYNC mode without CPU interference.

When Auto Transparent mode is enabled, the GT-96100A automatically switches to the transparent receive mode upon receiving a DLE STX sequence.

Other features are controlled by programming the bank of control registers.

Figure 63: Typical BISYNC/MonoSYNC Frames



14.6.1 BISYNC Transmit Operation

In BISYNC mode an MPSC handles the following protocol functions:

- Leading SYNC character transmission before a buffer with F bit set.
- Optional 32-bit transmission before the SYNC transmission.
- DLE transmission before a buffer with the TD bit set.
- BCC generation:
 - BCC (CRC-16, VRC/LRC and VRC/CRC-16) is calculated.
 - Buffers with BCE set to '0' are excluded from BCC calculation.
 - CRC reset is controlled from the RC bit in Tx descriptor.
 - The calculation of BCC is sent or discarded according to the GC bit in the Tx descriptor.
- Automatic stuffing of DLE when transmitting a transparent buffer (buffer with TR bit set).
- SYNC transmission if underrun occurs.

BISYNC transmission is descriptor chain oriented. Transmission starts when the CPU issues a Transmit Demand command and continues until the channel's SDMA reaches a NULL pointer or a 'not owned' descriptor.

14.6.2 BISYNC Receive Operation

There are two major operating modes in the BISYNC receiver.

Table 337: BISYNC Receiver Operating Modes

Mode	Function
Normal Mode	The CPU must monitor each received byte and manage each BISYNC operation (e.g., moving into transparent mode) manually.
Auto Transparent Mode	The GT-96100A handles transparent mode automatically. This mode reduces the CPU burden since it can monitor the incoming data buffer-by-buffer and not byte-by-byte.

14.6.2.1 BISYNC Normal Receive Mode

In Normal Mode, the BISYNC receiver handles the following protocol functions:

- BISYNC, MonoSYNC, NibbleSYNC or External SYNC synchronization.
- Auto SYNC stripping in text mode.
- Auto DLE-SYNC stripping in transparent text mode.
- Auto SYNC stripping after receiving DLE ITB in transparent mode.
- Automatic exit of transparent mode after receiving DLE-ETX/ETB (if RTR bit in the MPCR_x was cleared).
- Marking of buffers that contain transparent data by setting the TB bit in the descriptor.
- BCC generation:
 - BCC (CRC-16, VRC/LRC and VRC/CRC-16) is calculated.
 - In transparent text mode, CRC-16 always overrides the VRC.
 - SYNC (DLE-SYNC) is not included in the BCC calculation.
- Buffer closing at the reception of ETX, ETB, ITB and ENQ.
- Maintaining SYNC (stay in text mode) after ITB.
- Protocol correctness checking:
 - Test for '1' padding at the end of block reception. (The CPU should ignore a padding error reported after ITB, and can use it when testing for proper NAK or EOT.)
 - Test for DLE-CTL after receiving DLE-ITB in transparent text mode. If another sequence arrives (except SYNCs), buffer is closed with DLE error.

14.6.2.2 BISO SYNC Auto Transparent Receive Mode

In Auto Transparent Mode, the BISO SYNC receiver handles the following protocol functions:

- BISO SYNC, MonoSYNC, NibbleSYNC, or External SYNC synchronization.
- Auto SYNC stripping in text mode.
- Auto DLE-SYNC stripping in transparent text mode.
- Auto SYNC stripping after receiving DLE ITB in transparent mode.
- Automatic switch to transparent mode after receiving DLE-STX.
- Automatic exit of transparent mode after receiving DLE-ETX/ETB.
- Marking of buffers that contain transparent data by setting the TB bit in the descriptor.
- BCC generation:
 - BCC (CRC-16, VRC/LRC and VRC/CRC-16) is calculated.
 - In transparent text mode, CRC-16 always overrides the VRC.
 - SYNC (DLE-SYNC) is not included in the BCC calculation.
 - Opening STX/SOH (DLE-STX) are discarded from BCC calculations.
- Buffer closing at the reception of ETX, ETB, ITB, and ENQ.
- Maintaining SYNC (stay in text mode) after ITB.
- Buffer closing after SYN-SYN-DLE-CHAR (when char is not STX).
- Protocol correctness checking:
 - Test for '1' padding at the end of block reception. (The CPU should ignore a padding error reported after ITB, and can use it when testing for proper NAK or EOT.)
 - Test for DLE-CTL (CTL is a control character with B or H set) after receiving DLE-ITB in transparent text mode. If another sequence arrives (except SYNCs), buffer is closed with a DLE error.

The BISO SYNC receive process is block oriented. A block starts after a buffer was closed due to control character reception, overrun, protocol error, parity error, or line error (i.e. CD deassertion).

The first descriptor in a block is marked with F bit set to '1'. The last descriptor in block is marked with L bit set to '1'. The last descriptor also includes the actual status report for the block. Intermediate descriptors can be recognized by having both F and L bit set to '0'.

14.6.3 SDMAx Command/Status Field for BISYNC Mode

When an MPSC is in BISYNC mode the Command/Status field in the corresponding SDMAx descriptor has the following format:

Table 338: SDMAx Command/Status Field for BISYNC Mode

Bits	Rx - Function	Tx - Function
0	CE - CRC/LRC Error	Reserved
1	CDL - CD Loss	CTSL - CTS Loss
2	DE - Decoding Error	Reserved
3	DLE - DLE Error. While in transparent mode, this indicates a DLE was received and the following byte was not a valid control character.	Reserved
4	PR - Parity Error. Last byte in buffer has parity error.	Reserved
5	Reserved	Reserved
6	OR - Data Overrun	Reserved
8:7	Reserved	Reserved
9	PDR - Pad Report. This is set if there were no four consecutive '1's after the block reception.	Reserved
10	Reserved	Reserved
11	TB - Transparent Buffer. Buffer contains transparent data.	Reserved
12	Reserved	Reserved
13	C - Last bytes in buffer is a user defined control character.	Reserved
14	B - Last bytes in buffer are BCC.	Reserved
15	ES - Error Summary ES = CDL DE DLE PR OR	ES - Error Summary ES = CTSL
16	L - Last	L - Last NOTE: Transmit Bit 22 is used only if L bit is set to '1'. If L bit is set to '0', no BCC is sent at the end of this buffer transmission.
17	F - First	F - First

Table 339: MPSCx Protocol Configuration Register (MPCRx) for BISYNC, Offset: 0x000A08, 0x008A08, 0x010A08, 0x018A08, 0x020A08, 0x028A08, 0x030A08, 0x038A08 (where x is the port number 0 to 7)

Bits	Field Name	Function	Initial Value
1:0		Reserved.	0
2	ATM	Auto Transparent Mode 0 - Normal Mode. 1 - Receiver switches to transparent mode after receiving DLE-STX and exits transparent mode upon receiving a DLE-ETB or DLE-ETX sequence. When switching to transparent mode, new buffers are opened for transparent data. When the ATR bit is set to '1' the following characters should be programmed into CTL1-8: <ul style="list-style-type: none"> • CTL3 - STX • CTL4 - SOH NOTE: When entering transparent mode either automatically or by issuing an RTR command, the Receiver will strip automatically leading DLEs. The TB bit in the descriptor will be set to signal the software that the buffer contains transparent data.	0
5:3		Reserved.	0
6	DRT	Disable Rx on Tx When DRT is set to '1' the Rx path is closed during Tx. This is useful in a multidrop configuration when a user doesn't want to receive its own frames.	0
9:7		Reserved.	0
10	TRP	Trailing Pad When set, the BISYNC transmitter sends a PAD character (0xFF) at the end of each outgoing frame (i.e. after a buffer with L bit set.)	0
12:11		Reserved.	10
13	RTR	Receive Transparent Mode 0 - The receiver is placed in normal mode with sync stripping and control character recognition operative. 1 - The receiver is placed in transparent mode. Syncs DLEs and control characters are recognized only after leading DLE characters. CRC16 is calculated even in VRC/LRC mode while in transparent mode. NOTE: When entering transparent mode either automatically or by issuing an RTR command, the receiver automatically strips leading DLEs. The TB bit in the descriptor is set to signal the software that the buffer contains transparent data.	0

Unless otherwise specified:

- '1' means set
- '0' means unset.
- '0' is the default value after reset.

14.6.5.1 CHR1 - SYNC/DLE Register (SDR)

CHR1[7:0] holds the SYNC character and CHR1[23:16] holds the DLE character for the channel. After reset it holds the value of 7E in the SYNC field and FE in the DLE field. The user must write the appropriate values before enabling the Rx/Tx machines.

If bit 15 is set, the BISYNC receive machine discards the SYNC patterns received in a middle of a message.

NOTE: This usually happens when the transmitter experiences underrun.

If bit [15] is '0' the SYNC characters is transferred to the receive buffer.

If bit 31 is '1', the first DLE received in transparent mode is discarded. If bit 31 is '0', the BISYNC receiver is not discard DLE in transparent mode.

A BISYNC transmitter always stuffs the leading DLE before transmitting the DLE that is part of a transparent buffer (transmit descriptor with TR bit set). In order to send DLE ETX, for example, the CPU must either prepare a buffer that contains DLE ETX and set TR='0', or prepare a buffer with ETX and program the transmitter to send a leading DLE by setting the TD bit in the descriptor.

A BISYNC transmitter always transmits SYNC-SYNC at the beginning of a frame. This is true in MonoSYNC and NibbleSYNC modes.

When a BISYNC transmitter experiences underrun it transmits continuous SYNC patterns in text mode or DLE-SYNC in transparent mode. The BISYNC transmitter exits this state upon receiving new data or when the CPU issues a Stop or Abort command.

The receiver SYNC length is programmable. The actual length is determined according to the value of the RSYL bits in the MMCRx. If the RSYL bits equal #00b, the synchronization is done externally and the receiver will start receiving when CD* is asserted.

In NibbleSync mode, bits [7:4] are used by the receiver for sync recognition. Bits [3:0] should return the SYNC pattern in order to assure proper SYNC transmission.

14.6.5.2 CHR2 - Command Register (CR)

Table 340: CHxR2 - Command Register (CR), Offset: 0x000A10, 0x008A10, 0x010A10, 0x018A10, 0x020A10, 0x028A10, 0x030A10, 0x038A10 (where x is channel 0 and 7)

Bits	Field Name	Function	Initial Value
0	TEL	Tx Enable Longitudinal Redundancy Check 0 - LRC is disabled. 1 - LRC is enabled. (TEL default value is 0 and the CPU must write "1" to it in order to enable LRC). When set, TEL overrides the CRC mode that was programmed in the CRCM field in the MMCRx.	0
1	TEV	Tx Enable Vertical Redundancy Check (Parity Bit) 0 - VRC is disabled. 1 - VRC is enabled. (TEV default value is '0' and the CPU must write '1' to it in order to enable VRC).	0
3:2	TPM	Transmit Parity Mode 00 - Odd 01 - Low (always "0") 10 - Even 11 - High (always "1")	0
4	TLRM	Transmit Longitudinal Redundancy Mode 0 - Odd 1 - Even	1
6:5		Reserved.	0
7	A	Abort Transmission Abort transmission immediately and go to IDLE. The descriptor is not closed or incremented. NOTE: Command is not synchronized to byte.	0
15:8		Reserved.	0
16	REL	Rx Enable Longitudinal Redundancy Check. 0 - LRC is disabled. 1 - LRC is enabled. This is the normal mode for BISYNC. When set, REL overrides the CRC mode that was programmed in the CRCM field in the MMCRx.	0
17	REV	Rx Enable Vertical Redundancy Check (parity bit). 0 - VRC (parity) is disabled. 1 - VRC is enabled. This is the normal mode for BISYNC.	0
19:18	RPM	Receive Parity Mode 00 - Odd 01 - Low (always '0') 10 - Even 11 - High (always '1')	0

Table 340: CHxR2 - Command Register (CR), Offset: 0x000A10, 0x008A10, 0x010A10, 0x018A10, 0x020A10, 0x028A10, 0x030A10, 0x038A10 (where x is channel 0 and 7)

Bits	Field Name	Function	Initial Value
20	RLRM	Receive Longitudinal Redundancy Mode 0 - Odd 1 - Even	1
22:21		Reserved.	0
23	A	Abort Reception Abort receive immediately and go to IDLE. The descriptor is not closed or incremented. The processor must issue an enter hunt command after an abort command to enable reception. The A bit is cleared upon entering IDLE state.	0
24		Reserved.	0
25	CRD	Close Rx Descriptor When the CPU issues a CRD command the current receive descriptor is closed and the following received data is SDMA'd into a new buffer. If there is no active receive in process, no action takes place.	0
28:26		Reserved.	0
29	RBC	Reset BCC The CPU issues an RBC command in order to manually reset the CRC-LRC/VRC generator. The BCC calculation starts with the next byte. The GT-96100A clears the RBC bit after resetting BCC.	0
30		Reserved.	0
31	EH	Enter Hunt Upon receiving an enter hunt command, the receive machine will move to a hunt state and will continuously search for an opening SYNC or external SYNC. If an enter hunt mode command is issued during frame reception, the current descriptor will be closed with a CRC error. The EH bit will be cleared upon entering a hunt state.	0
NA	TD	Transmit Demand Fetch a descriptor and start transmission. Issued through the SDMAx Command Register.	
NA	Stop	Stop Transmission Complete frame transmission and stop. (Go to IDLE). Issued through the SDMA's Command Register.	

The ET bit in the Main Configuration Register must be set to '1' before issuing any of the following commands:

- Transmit Demand.
- Stop Transmission.
- Abort Transmission.

The ER bit in the Main Configuration Register must be set to '1' before issuing any of the following commands:

- Enter Hunt
- Reset BCC
- Close Rx Descriptor
- Abort Reception.

When the ET or ER bits are deasserted, the MPSCx transmit/receive channel is in low power mode (NO CLOCK).

NOTE: Issuing one of the above commands in this state will lead to unpredictable results.

Setting TEL='0', TEV='1' and CRCM='001', or setting REL='0', REV='1' and CRCM='001', will set the BISYNC transmitter/receiver to work in VRC+CRC16 mode. The calculated parity bit is considered part of the data that the CRC-16 checks.

When a BISYNC transmitter transmits a transparent buffer, it automatically switches to the CRC that was programmed in the CRCM field in MMCRx. When a receiver enters transparent mode, it automatically switches to the CRC that was programmed in CRCM field in MMCRx. In both cases, CRCM must be programmed to '001' in order to meet the BISYNC CRC-16 specifications.

14.6.5.3 CHR4 - Control Filtering Register (CFR)

Bits 7:0 of the CFR register are the Bit Comparison Enable bits. Setting '1' in one of the BCE bits enables the control comparison for this bit

14.6.5.4 CHR5-8 - BISYNC Control Character Registers

Figure 66 shows a BISYNC control register format.

The CHAR field holds the pattern for the control character while bits 8-15 are used to control the GT-96100A behavior when the control character is recognized.

Figure 66: BISYNC Control Character Register Format

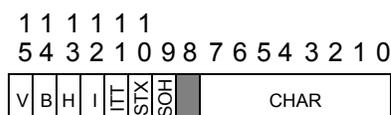


Table 341: BISYNC Control Character Register Format

Bits	Field Name	Function	Initial Value
7:0	CHAR	The Control Character To Sync On NOTE: Bit 7 must be programmed according to the parity method in use. See Table 340 .	0
8		Reserved.	0
9	SOH	SOH Character 0 - Normal Mode. 1 - SOH character. In Auto Transparent mode the characters following SOH including STX are part of the BCC calculations.	0
10	STX	STX Character 0 - Normal character. 1 - STX character. In Auto Transparent mode, an STX character is expected after the first DLE in order to enter transparent mode.	0
11	ITT	Ignore While Receiving in Text Mode 0 - Normal control character. 1 - Ignore this character after entering text mode (i.e. after receiving SYN-SYN-STX/SOH).	0
12	I	Interrupt 0 - No interrupt. 1 - Generate interrupt upon receiving this CHAR.	0
13	H	Hunt 0 - Close buffer and maintain SYNC. 1 - Close buffer and move to HUNT state.	0
14	B	BCC Next 0 - Close buffer. 1 - BCC is next. Receive BCC and than close buffer.	0
15	V	Valid. 0 - Entry is not valid. 1 - Entry is valid.	0

The BISYNC Control Character programming recommendations for Auto Transparent Mode and CPU Controlled Operation are shown in the following tables.

Table 342: Auto Transparent Programming

Control Character	V	B	H	I	ITT	STX	SOH
STX ¹	1	0	0	X	1	1	0
SOH ²	1	0	0	X	1	0	1
ETX	1	1	1	X	0	0	0
ITB	1	1	0	X	0	0	0
ETB	1	1	1	X	0	0	0
ENQ	1	0	1	X	0	0	0
EOT	1	0	1	X	1	0	0
NACK	1	0	1	X	1	0	0

1. CTL3 must be use to hold STX

2. CTL4 must be use to hold SOH

Table 343: CPU Controlled Operation

Control Character	V	B	H	I	ITT	STX	SOH
ETX	1	1	1	X	0	0	0
ITB	1	1	0	X	0	0	0
ETB	1	1	1	X	0	0	0
ENQ	1	0	1	X	0	0	0
EOT	1	0	1	X	1	0	0
NACK	1	0	1	X	1	0	0
Other Entry							
Other Entry							

14.6.5.5 CHR9 - Reserved

This register is reserved.

Do not access this register in the BISYNC mode.

14.6.5.6 CHR10 - BISYNC Event Status Register (ESR)

The ESR register holds information on the transmit/receive channel condition.

CHR10 can be read by the CPU for channel condition resolution. Some changes in the channel condition can generate maskable interrupts, as shown below.

Table 344: CHxR10 - BISYNC Event Status Register (ESR), Offset: 0x000A30, 0x008A30, 0x010A30, 0x018A30, 0x020A30, 0x028A30, 0x030A30, 0x038A30 (where x is channel 0 and 7)

Bits	Field Name	Event
0	CTS	Clear To Send Signal ¹
1	CD	Carrier Detect Signal ²
2		Reserved
3	TIDLE	Tx in Idle State ³
5	RHS	Rx in HUNT state
10:6		Reserved
11	RLIDL	1 = Rx IDLE Line ⁴
12	DPCS	1 = DPLL Carrier Sense
15:13		Reserved.
23:16	RCRn	Received Control Character n When the BISYNC receiver recognizes a control character it sets the corresponding RCRn bit. Bit 16 (RCR1) corresponds to CTL1. Bit 23 (RCR8) corresponds to CTL8. RCRn bits are cleared by writing '1' to the bit. RCRn is set if the corresponding control character arrives, and both its Valid bit and Interrupt bit are also set (e.g., bit 16 will be set if CTL1 arrives, and both CTL1's "V" bit is set, and CTL1's "I" bit is also set.)

1. Interrupt is generated when signal is deasserted during transmit
2. Interrupt is generated when signal is deasserted during receive
3. Interrupt is generated upon entering IDLE state
4. Interrupt is generated upon change in line status

NOTE: PERR will be set in transparent mode during SYN stripping when a non DLE or SYN character is received. This is a protocol violation. The receiver moves to hunt mode and a maskable interrupt is generated. The received character is discarded.

14.7 UART Mode

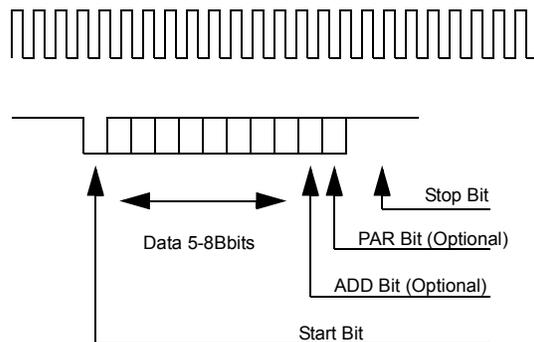
14.7.1 UART Receive/Transmit Operation

In UART mode an MPSC performs the following protocol functions:

- Start/Stop bit framing.
- Programmable data lengths (5-8 bits).
- Synchronous and asynchronous support.
- Message oriented data support.
- Parity detection and generation.
- Frame error, noise error, break, and idle detection.
- Support for HDLC over asynchronous control-octet transparency protocol.
- Multidrop operation with address recognition of up to two different addresses.

Figure 67 shows a typical UART frame format. A frame with a start bit is followed by 5-8 data bits. The address and parity bits are optional.

Figure 67: Typical UART Frame



At the end of a frame there are 1–2 stop bits before the transmitter can start to transmit the next frame. If there is nothing to transmit, a continuous ‘1’ is transmitted. This indicates that the line is idle.

The GT-96100A’s UART samples each bit three times near its central point to define the bit value. A new start bit can be recognized only after the last stop bit sample is received. For example, at a 16x clock rate, the receiver can receive a start bit after a 9/16 bit time long stop bit.

When in UART mode, the RDW bit in the MMCRx should be set to configure the MPSCx data path to 8 bits.

A UART transceiver can work in Asynchronous or Isochronous modes.

14.7.1.1 Asynchronous Mode

In Asynchronous mode, the DPLL encoding must be set to RNZ and the clock sampling rate is set to 8x, 16x, or 32x of the data rate. The DPLL is synchronized by the falling edge of the start bit. If no error occurs, it maintains synchronization until the last bit in a frame is received.

Each bit is sampled three times around it's middle point. The bit value is determined by a majority vote. This feature helps to filter out noise from received data.

14.7.1.2 Isochronous Mode

In Isochronous mode, the DPLL sampling rate will be 1x the data rate. The receive data must be synchronized to the receive clock.

14.7.2 SDMAx Command/Status Field for UART Mode

When an MPSC is in UART mode the Command/Status field in the corresponding SDMAx descriptor has the following format:

Table 345: SDMAx Command/Status Field for UART Mode

Bit	Rx - Function	Tx - Function
0	PE - Parity Error. Last byte in buffer has parity error.	Reserved
1	CDL - CD Loss	CTSL - CTS Loss
2	Reserved	Reserved
3	FE - Framing Error	Reserved
5:4	Reserved	Reserved
6	OR - Data Overrun	Reserved
8:7	Reserved	Reserved
9	BR - Break Received while receiving data into this buffer	Reserved
10	MI - Max Idle. Buffer was closed due to Max_Idle timer expiration. NOTE: When this bit is set, the status of bit 0 is disregarded.	Reserved
11	A - Address. First byte in the buffer is an address. (Valid only in multidrop mode, '0' in point to point mode.)	Reserved
12	AM - Address match. This bit will be set to '1' when a match occurred even if the V bit of the address is disabled.	Reserved
13	CT - The last byte in the buffer was precede by a transparency control octet.	Reserved
14	C - The last byte in a buffer is a user define control character.	Reserved
15	ES - Error Summary ES = PE CDL FE OR	ES - Error Summary ES = CTSL
16	L - Last	L- Last

Table 346: MPSCx Protocol Configuration Register (MPCRx) for UART Mode, Offset: 0x000A08, 0x008A08, 0x010A08, 0x018A08, 0x020A08, 0x028A08, 0x030A08, 0x038A08 (where x is the port number 0 to 7)

Bits	Field Name	Function	Initial Value
5:0		Reserved.	0
6	DRT	Disable Rx on Tx. When DRT is set to '1' the Rx path is closed during Tx. This is useful in multidrop configurations when a user doesn't want to receive its own frames	0
7	ISO	<p>Isochronous Mode</p> <p>0 - Asynchronous Mode. Start and stop bits are expected. RENC in the MMCRx should be programmed to NRZ and RCDV should be programmed to x8, x16 or x32 mode. (x16 is recommended for most applications).</p> <p>1 - Isochronous Mode. The receive bit stream is assumed to be synchronous to the receive clock. RCDV should be programmed to x1 mode.</p>	0
8	RZS	<p>Receive Zero Stop Bit</p> <p>0 - Normal Mode. At least one stop bit is expected.</p> <p>1 - Zero Stop Bit. The receiver continues reception when a stop bit is missing. If a '0' is received when stop bit is expected, this bit is considered a start bit. The FE (Framing Error) bit is set and the next bit to be received is considered to be data.</p>	0
9	FRZ	<p>Freeze Tx</p> <p>0 - Restart Tx after freeze (normal operation). Transmission continues from the place it stopped.</p> <p>1 - Freeze Tx at the end of the current character.</p>	0
11:10	UM	<p>UART Mode</p> <p>00 - Normal Mode. Multidrop is disabled and IDLE line wake up is selected. A UART receiver wakes up after entering hunt mode upon receiving an IDLE character (all one character).</p> <p>01 - Multi Drop Mode. In multidrop mode, there is an additional Address/Data bit in each character. Upon receiving an address character, the UART receiver compares it to two 8-bit addresses stored in its channel registers. If a match occurs, the receiver transfers the address and the following characters into a new buffer. If there is a no match, the character is discarded and the receiver is set to the hunt mode. If none of the addresses is valid (V bit in both address register is set to '0'), there is always a match and all the characters are transferred into the DRAM. Addresses are always be placed in a new buffer (Regardless of the V bit). The receiver receives characters until a new address is received, an abort character is received, an enter hunt command is issued, or until max idle counter expiration. Upon max idle counter expiration, the receiver is set to the hunt mode.</p> <p>10 - Reserved.</p> <p>11 - Reserved.</p>	0

Table 346: MPSCx Protocol Configuration Register (MPCRx) for UART Mode, Offset: 0x000A08, 0x008A08, 0x010A08, 0x018A08, 0x020A08, 0x028A08, 0x030A08, 0x038A08 (where x is the port number 0 to 7) (Continued)

Bits	Field Name	Function	Initial Value
13:12	CL	Character Length 00 - 5 data bits 01 - 6 data bits 10 - 7 data bits 11 - 8 data bits	01
14	SBL	Stop Bit Length 0 - One stop bit 1 - Two stop bits	0
15	FLC	Flow Control 0 - Normal Mode. The CTSM bit in the MMCRx determines the CTS* pin behavior. 1 - Asynchronous Mode. When CTS* is negative, transmission stops at the end of the current character. When CTS* is asserted again the transmission starts from the place it stopped. No CTS* lost is reported. Line is IDLE (MARK) during CTS* deassertion period.	0
31:16		Reserved	0

NOTE: When CD* is deasserted during frame reception UART behavior is different for multidrop and normal modes. In normal mode the UART hunts for an IDLE character (hunting starts when CD* is asserted again) before receiving valid start bit. In this mode, transmitting from a GT-96100A model to another should be with the 'P' bit in the buffer descriptor set. In multidrop mode, the UART receiver hunts for a start bit as soon as CD* is asserted again.

14.7.4 UART Stop Bit Reception and Framing Error

The UART receiver always expects to find a stop bit at the end of a character. If no stop bit is detected, the Framing Error (FE) bit is set in the receive descriptor. After a framing error, the reception process is controlled by the RZS and UM bits in the UART MPCRx. The various options are summarized in the table below.

Table 347: UART Stop Bit Reception and Framing Error

UM	RZS	Operation	Break Recognition
00	0	Go to hunt after missing a stop bit. The receiver is enabled after receiving a new IDLE char.	Single Break
00	1	The receiver tries to synchronize itself. The missing stop bit is considered as the following start bit and the reception process continues.	Two Break Sequence

Table 347: UART Stop Bit Reception and Framing Error (Continued)

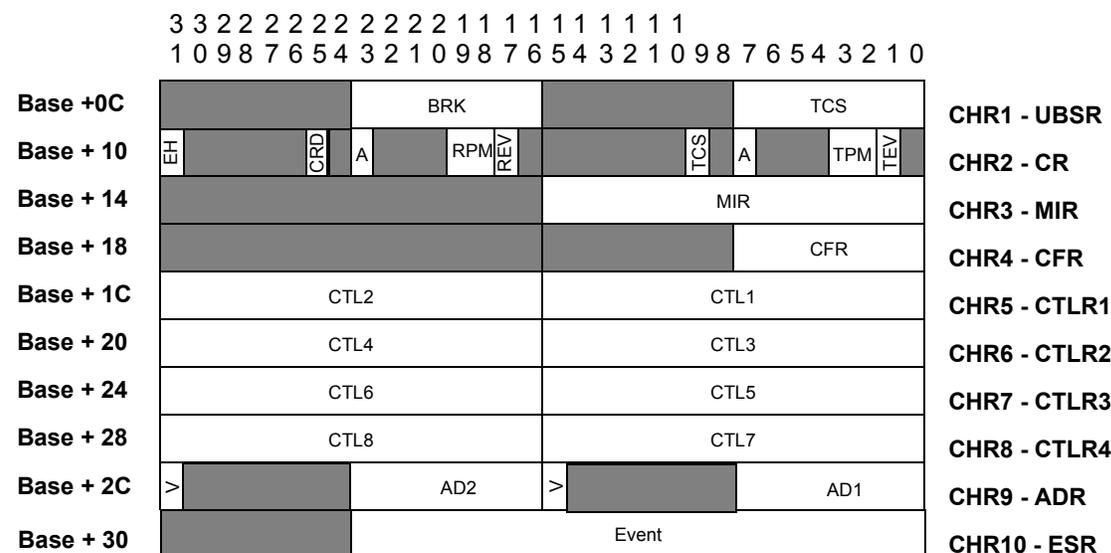
UM	RZS	Operation	Break Recognition
01	0	Goes to hunt after missing stop bit. The receiver is enabled after receiving new address character.	Single Break
01	1	The receiver tries to synchronize itself. The missing stop bit is considered as the following start bit and the reception process continues.	Two Break Sequence

14.7.5 Channel Registers (CHxRx) for UART Mode

The MPSCx Channel Registers (CHxRx) are protocol dependent.

Figure 69 shows the CHxRx format in UART mode.

Figure 69: Channel Registers (CHxRx) for UART Mode



Unless otherwise specified:

- ‘1’ means set.
- ‘0’ means unset.
- ‘0’ is the default value after reset.

14.7.5.1 CHR1 - UART Break/Stuff Register (UBSR)

The UART Break/Stuff register has two fields: Break Count (BRK)(CHR1[23:16]) and Control Stuff Character (TCS) (CHR1[7:0]).

With the BRK field, the UART transmitter will starts to transmit break characters after receiving an abort command. The number for the break character to send is programmed into the BRK field.

For example, when BRK equals ‘0’, no break character is transmitted. When BRK equals ‘1’, one break character is transmitted.

A break character is a character with all ‘0’s including it’s stop bit.

Upon issuing a TCS command, the transmitter sends a TCS character after the current transmitting character. This allows a transmitter to bypass the normal pipeline when a special control character must be send (e.g. XON/XOFF).

Upon receiving a break character, the UART stops the reception process and moves to the hunt state. In a point to point configuration, the receiver is hunting for a new IDLE character. In a multidrop configuration, the receiver hunts for a new address character.

When the UART is in RZS=0 mode after receiving a break sequence, the descriptor is closed with BR bit (bit 9) set. In addition, a “break descriptor” also has the FE bit (bit 3) set and, if in odd parity, the PE bit (bit 0) is also set.

When the UART in RZS=1 mode, two consecutive break sequences are needed for proper break recognition. The first break character is not recognized. Instead, the UART receiver closes the descriptor with the FE bit (bit 3) set and, if in odd parity, the PE bit (bit 0) will also be set. The second break will be recognized as a break and a descriptor will be closed with the BR bit (bit 9) set. In addition, a “break descriptor” will also have the FE bit (bit 3) set and, if in odd parity, the PE bit (bit 0) will also be set.

14.7.5.2 CHR2 - Command Register (CR)

Table 348: CHxR2 - Command Register (CR), Offset: 0x000A10, 0x008A10, 0x010A10, 0x018A10, 0x020A10, 0x028A10, 0x030A10, 0x038A10 (where x is channel 0 and 7)

Bits	Field Name	Function	Initial Value
0		Reserved.	0
1	TEV	Tx Enable Vertical Redundancy Check 0 - VRC (parity) is disabled. 1 - VRC is enabled.	0
3:2	TPM	Transmit Parity Mode 00 - Odd 01 - Low (always 0) 10 - Even 11 - High (always 1)	0
6:4		Reserved.	0
7	A	Transmit Abort Aborts the transmission immediately (on byte boundaries) and goes to IDLE. The descriptor is not closed or incremented. After receiving an abort command, the GT-96100A halts the transmit process and starts sending a break sequence according to the BRK field in CHR1. NOTE: Command is not synchronized to byte.	0
8		Reserved.	0
9	TCS	Transmit TCS Character. The TCS character is transmitted after the current transmitted character. The transmitter then continues with the normal Tx sequence. The TCS command can be used to send out of band characters such as XOFF and XON.	0
16:10		Reserved.	0
17	REV	Rx Enable Vertical Redundancy Check 0 - VRC (parity) is disabled. 1 - VRC is enabled.	0

Table 348: CHxR2 - Command Register (CR), Offset: 0x000A10, 0x008A10, 0x010A10, 0x018A10, 0x020A10, 0x028A10, 0x030A10, 0x038A10 (where x is channel 0 and 7)

Bits	Field Name	Function	Initial Value
19:18	RPM	Receive Parity Mode. 00 - Odd 01 - Low (always '0') 10 - Even 11 - High (always '1')	0
22:20		Reserved.	0
23	A	Receive Abort Abort receive immediately and go to IDLE. The descriptor is not closed or incremented. The processor must issue a enter hunt command after an abort command in order to enable reception. The A bit is cleared upon entering IDLE state.	0
24		Reserved.	0
25	CRD	Close Rx Descriptor When the CPU issues a CRD command the current receive descriptor is closed and subsequent received data is DMA'd into a new buffer. If there is no active receive process, no action takes place. The GT-96100A clears the CRD bit upon closing the buffer status. ¹	0
30:26		Reserved.	0
31	EH	Enter Hunt Upon receiving an enter hunt command, the receive machine moves to a hunt state and continuously searches for an opening character. An opening character is considered an IDLE char in point to point mode (UM=00) or a matched address in multidrop mode. The EH bit is cleared upon entering a hunt state.	0
N/A	TD	Transmit Demand Fetch a descriptor and start transmission. Issued through the SDMAx Command Register.	
N/A	Stop	Stop Complete frame transmission and stop. (Go to IDLE). Issued through the SDMAx Command Register.	

1. Usually, it takes a few cycles from the time the CRD bit is closed until the SDMAx actually closes the buffer. The SDMAx generates a maskable interrupt when closing a buffer if programmed to do so.

Table 349: UART Control Character Register Format

Bits	Field Name	Function	Reset Value
7:0	CHAR	The control character to sync on.	0
11:8		Reserved.	0
12	INT	Interrupt 0 - No interrupt. 1 - Generate interrupt upon receiving this CHAR.	0
13	CO	Control Octet (ISO 3309 Control Octet) Upon receiving a control octet, the control octet is discarded and the 6th bit (i.e. bit 5 in CHR5) of the following octet is complemented. The current buffer is closed with the CO bit asserted. NOTE: When the CO bit is set, the CHAR field must be programmed with '10111110' in order to be ISO-3309 compatible.	0
14	R	Reject 0 - Receive character and close the buffer. 1 - Reject character. The character is discarded, the buffer is closed and a maskable interrupt is generated.	0
15	V	Valid. 0 - Entry is not valid. 1 - Entry is valid.	0

14.7.5.6 CHR9 - Address Register (ADR)

CHR9 holds the UART addresses for multidrop operation. The GT-96100A UART supports up to 2 addresses.

Upon receiving an address, the UART transfers the previous frame status to the SDMA. This causes the SDMA to close the previous frame descriptor and to locate the address in a new buffer.

There are two modes for address recognition operation. The first mode, setting of '1', allows the address and following characters to be transferred to the SDMA only if there is a match. The second mode, setting of '0', allows all frames to be passed to the SDMA. The CPU can use the AM bit in the last frame descriptor to check if a match occurred.

14.7.5.7 CHR10 - UART Event Status Register (ESR)

The ESR register holds information on the transmit/receive channel condition. CHR10 can be read by the CPU for channel condition resolution.

Some changes in the channel condition can generate maskable interrupts, as shown in [Table 350](#).

Table 350: CHR10 - UART Event Status Register (ESR), Offset: 0x000A30, 0x008A30, 0x010A30, 0x018A30, 0x020A30, 0x028A30, 0x030A30, 0x038A30 (where x is channel 0 and 7)

Bits	Field Name	Event
0	CTS	Clear To Send Signal ¹
1	CD	Carrier Detect Signal ²
2		Reserved.
3	TIDLE	Tx in Idle State ³
4		Reserved.
5	RHS	Rx in HUNT State
6		Reserved.
7	RLS	Rx Line Status
10:8		Reserved.
11	RLIDL	1 = Rx IDLE Line ^c
15:12		Reserved.
23:16	RCRn	Received Control Char n When the UART receiver recognizes a control character it sets the corresponding RCRn bit. (bit 16 (RCR1) corresponds to CTL1... bit 23 (RCR8) corresponds to CTL8). RCRn bits are cleared by write a 1 to the bit.

1. Interrupt is generated when signal is deasserted during transmit.

2. Interrupt is generated when signal is deasserted during receive.

3. Interrupt is generated upon entering IDLE state.

14.8 Transparent Protocol

In transparent mode, the GT-96100A does not perform any protocol dependent data processing.

However, it gives the processor hardware assistance for bit reception, using the GT-96100A's powerful SDMA engines, and some assistance in synchronization, interrupt generation, and frame construction. The CPU also uses the built-in CRC engine for CRC generation and checking. In any case, CRC bits are transferred into memory for CPU use.

In transparent mode, the channel is fully configured from the MMCRx and no mode is defined by the channel registers.

A transparent channel is synchronous. If it is not serviced on time, underrun and overrun errors can occur.

The receiver can use external sync using the CD* input or synchronize itself on a SYNC sequence according to the RSYL bits in the MMCRx.

14.8.1 SDMAx Command/Status Field for Transparent Mode

When an MPSC is in Transparent mode the Command/Status field in the corresponding SDMAx descriptor has the following format:

Table 351: SDMAx Command/Status Field for Transparent Mode

Bit	Rx - Function	Tx - Function
0	CE - CRC/LRC Error	Reserved.
1	CDL - CD Loss	CTSL - CTS Loss
2	DE - Decoding Error	Reserved.
3	Reserved.	Reserved.
4	Reserved.	Reserved.
5	Reserved.	Reserved.
6	OR - Data Overrun	UR - Data Underrun
14:7	Reserved.	Reserved.
15	ES - Error Summary ES = CE CDL DE OR	ES - Error Summary ES = CTSL UR
16	L - Last	L - Last
17	F - First	F - First
21:18	Reserved.	Reserved.
22	Reserved.	GC - Generate BCC/LRC.
23	EI - Enable Interrupt	EI - Enable Interrupt
29:24	Reserved.	Reserved.
30	AM - Auto Mode	AM - Auto Mode
31	O - Owner	O - Owner

Table 352: Transparent Mode Synchronization Options

Synchronization Option	Description
External Synchronization	(RSYL = '00') The receiver starts to receive data whenever CD* is asserted and stops receiving data when CD* is deasserted (if CDM=0) or when the CPU issues an Enter Hunt Command.
Sync Hunt	RSYL = '01', '10', or '11' (nibble, byte or two bytes sync) The receiver hunts for the sync pattern, as defined by RSYL. When the synch pattern is recognized, the receiver starts to receive data. The receive process stops when CD* is deasserted and CDM=0 or when the CPU issues an enter hunt command. If bit 15 is set, there is no transfer of the SYNC characters to the receiver. The syncs are stripped until the first data character is received, and are not calculated in the packet CRC. If bit 15 is reset, sync characters that appear after the sync pattern is recognized are regarded as data. On the transmitter side, in sync hunt mode, two sync characters are always sent at the beginning of a frame. NOTE: When RSYL equals 01, the Sync pattern is defined by bits [7:4] of the Sync Register.

There are two mode of transmit synchronization in transparent mode. They are selected by setting TSYN[12} in the MMCR_x, see [Table 322](#).

Table 353: Transmitter Mode Synchronization Options

Synchronization Option	Description
TSYN = 0	Synchronization is achieved whenever CTS* is asserted.
TSYN=1	Synchronization is achieved after receiver synchronization and CTS* is asserted. The transmitter always starts to transmit on the receive byte boundaries. In external synchronization, when CTS* is asserted, the transmitter starts to transmit 8 bits after CD* assertion. In sync hunt mode, when CTS* is asserted, the transmitter starts to transmit 8 bits after sync recognition. If CTS* is deasserted after the receiver gains synchronization, the transmitter waits to the byte boundary before it starts to transmit.

14.8.2.2 CHR2 - Command Register (CR)

Table 354: CHxR2 - Command Register (CR), Offset: 0x000A10, 0x008A10, 0x010A10, 0x018A10, 0x020A10, 0x028A10, 0x030A10, 0x038A10 (where x is channel 0 and 7)

Bits	Field Name	Function	Initial Value
6:0	Reserved	Reserved.	0
7	A	Abort Transmission Aborts the transmission immediately (on byte boundaries) and goes to IDLE. The descriptor is not closed or incremented. NOTE: Command is not synchronized to byte.	0
22:8	Reserved	Reserved.	0
23	A	Abort Reception Abort receive immediately and go to IDLE. The descriptor is not closed or incremented. The processor must issue an enter hunt command after an abort command in order to enable reception. The A bit is cleared upon entering IDLE state.	0
24	Reserved	Reserved.	0
25	CRD	Close Rx Descriptor When the CPU issues a CRD command the current receive descriptor is closed and the following received data is SDMA'd into a new buffer. If there is no active receive in progress, no action takes place.	0
30:26	Reserved	Reserved.	0
31	EH	Enter Hunt Upon receiving an enter hunt command, the receive machine moves to a hunt state and continuously searches for an opening sync or an external sync. If the enter hunt command is received during a frame reception, the current descriptor is closed with a CRC error. The EH bit is cleared upon entering a hunt state.	0
N/A	TD	Transmit Demand Fetch a descriptor and start transmission. Issued at SDMAx Command Register.	
N/A	Stop	Stop Complete frame transmission and stop. (Go to IDLE). Issued at SDMAx Command Register.	

The ET bit in the Main Configuration Register must be set to '1' before issuing any of the following commands:

- Transmit Demand
- Stop Transmission
- Abort Transmission

The ER bit in the Main Configuration Register must be set to ‘1’ before issuing any of the following commands:

- Enter Hunt
- Close Rx Descriptor
- Abort Reception

When the ET or ER bits are deasserted, the MPSCx transmit/receive channel is in low power mode (NO CLOCK).

NOTE: Issuing one of the above commands in this state leads to unpredictable results.

14.8.2.3 CHR10 - Transparent Event Status Register (ESR)

The ESR register holds information on the transmit/receive channel condition. CHR10 can be read by the CPU for channel condition resolution. Some changes in the channel condition can generate maskable interrupts, as shown [Table 355](#).

Table 355: CHR10 - Transparent Event Status Register (ESR), Offset: 0x000A30, 0x008A30, 0x010A30, 0x018A30, 0x020A30, 0x028A30, 0x030A30, 0x038A30 (where x is channel 0 and 7)

Bits	Field Name	Event
0	CTS	Clear To Send Signal ¹
1	CD	Carrier Detect Signal ²
2		Reserved.
3	TIDLE	Tx in Idle State ³
4		Reserved.
5	RHS	Rx in HUNT state
11:6		Reserved.
12	DPCS	1 = DPLL Carrier Sense
15:13		Reserved.
23:16	RCRn	Received Control Character n When the transparent receiver recognizes a control character it sets the corresponded RCRn bit (bit 16 (RCR1) corresponds to CTL1... bit 23 (RCR8) correspond to CTL8). RCRn bits are cleared by writing ‘1’ to the bit.

1. Interrupt is generated when signal is deasserted during transmit

2. Interrupt is generated when signal is deasserted during receive

3. Interrupt is generated upon entering IDLE state

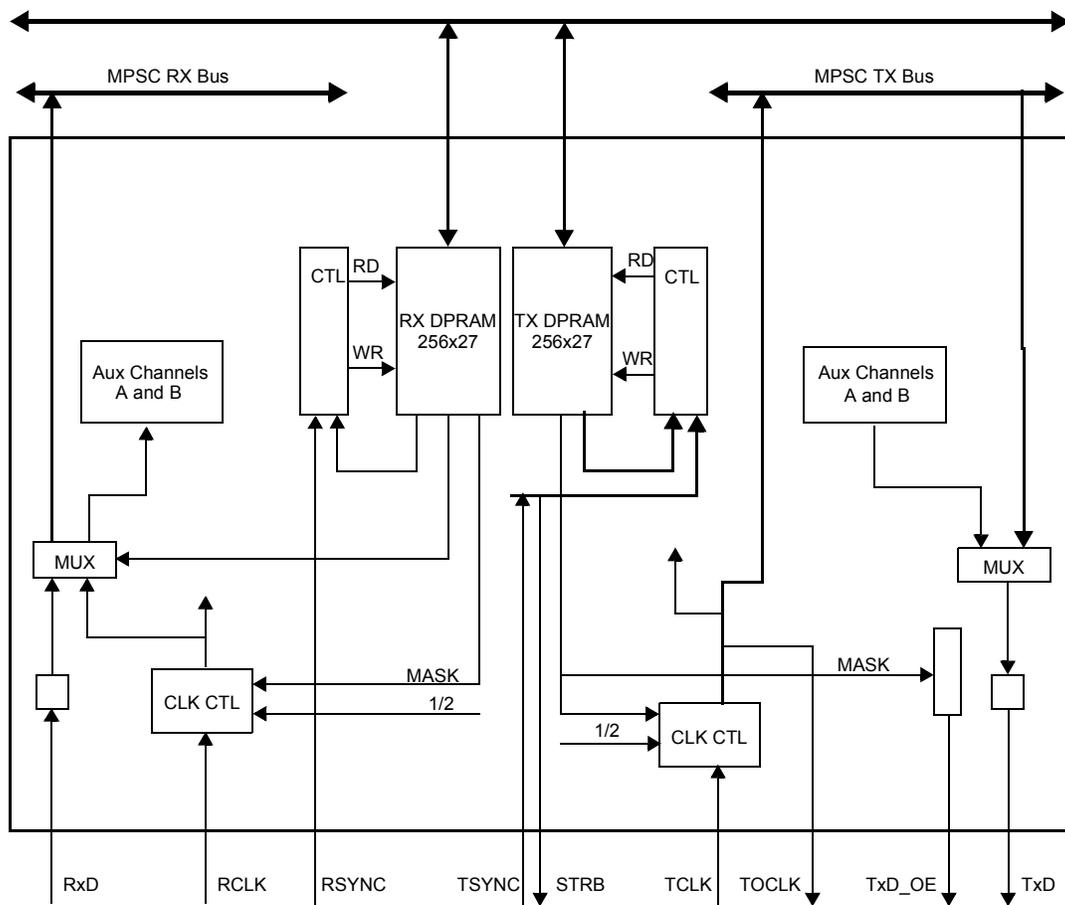
15. FLEXTDM UNITS (FTDM)

There are four FlexTDM units (also called time slot assigners) in the GT-96100A. Each unit is capable of supporting IOM-1/2 (GCI), PCM highway, T1/CEPT lines, as well as other proprietary time slot assigned buses.

The time slot assignment is configured by programming an internal dual port RAM (DPRAM). This DPRAM supports static and dynamic configurations. The DPRAM based design provides the flexibility to program the FlexTDM to any of the common time slot buses (i.e. GCI, PCM) or to a proprietary bus.

The FlexTDM unit consists of a transmit and a receive section. Each has a dedicated 256 entry dual port RAM. [Figure 72](#) shows a block diagram of a FlexTDM unit.

Figure 72: FlexTDM Architecture



15.1 FlexTDM Architecture

The FlexTDM architecture is based on two dual-port RAM (DPRAM) arrays. one RAM array is for receiving and one is for transmitting. The frame structure of a time slot assigned bus is configured by programming this DPRAM.

The FlexTDM incorporates two auxiliary channels: AUXA and AUXB. These channels are customized for the IOM Monitor and C/I channels.

The eight MPSCs and two auxiliary channels are time multiplexed on the TxD and RxD lines using two dedicated muxes. The mux select lines are controlled through DPRAM programming. A MPSC connected to a FlexTDM gets its receive and transmit signals from the FlexTDM. The actual bit rate of the MPSC is defined by the FlexTDM programming and the time slots that are assigned to this MPSC.

The FlexTDM supports independent transmit and receive clocks and frame syncs. Either 1x or 2x input clocks are allowed (the 2x clock is required for IOM bus interface). Selection of clock edges (rising/falling) for frame sync and data is supported as well.

15.2 FlexTDM DPRAM

The FlexTDM DPRAM is a 256x27 dual port RAM array that controls the FlexTDM behavior. The DPRAM can be configured dynamically during FlexTDM operation.

Table 356 shows the FlexTDM DPRAM field assignments.

Table 356: Flex TDM DPRAM Entry

Bits	Field Name	Function	Reset Value
26	FTINT	FlexTDM Interrupt An interrupt is generated if FTINT bit is set in the current entry and reset in the last TDM entry that was executed.	X
25	L	Last Entry in Frame The TDM read pointer returns to entry 0 or entry 128 (address 0 or 128 of the TDM DPRAM) after reading this entry. Control of which entry is executed next - either 0 or 128 - is done through R2HALF and T2HALF bits in TCR (see Section 15.4 "FlexTDM Configuration Register (TCR)" on page 384).	X
24:23	RPT	Number of times entry is repeated before moving to the next entry. The FlexTDM repeats execution of this entry according to the value programmed in RPT. ¹ 00 - Entry not repeated (i.e. it is executed once). 01 - Entry repeated once. 10 - Entry repeated twice. 11 - Entry repeated three times (i.e. it is executed four times).	X
22:21		Reserved.	X

Table 356: Flex TDM DPRAM Entry (Continued)

Bits	Field Name	Function	Reset Value
20:19	STRB	<p>This field controls the TDSTRB output of the TDM.</p> <p>00 - 0 01 - Z (tri-state) 10 - Toggle (strobe state is inverted every cycle this entry is executed) 11 - 1</p> <p>NOTE: The FlexTDM provides a single external strobe signal, which is shared between the receive and transmit sections. The strobe pin is driven according to the logical OR of the internally generated strobe signals. For example, if transmit strobe is Z and receive strobe is 0, external strobe will be 0. If transmit strobe is 1 and receive strobe is 0, external strobe will be 1.</p>	X
18	B	<p>Byte</p> <p>Defines byte/bit resolution for this entry.</p> <p>0 - Bit. Data width associated with this entry is 1 bit.</p> <p>1 - Byte. Data width associated with this entry is 1 byte (8 bits).</p> <p>NOTE: When B=1, indicating byte resolution, the DPRAM entry is executed 8 times (once for each bit in the byte).¹</p>	X
17:13		Reserved.	X
12:8	CH	<p>Channel Select</p> <p>Controls which of the serial controllers is assigned to the current data group.</p> <p>00000 - MPSC0 00001 - MPSC1 00010 - MPSC2 00011 - MPSC3 00100 - MPSC4 00101 - MPSC5 00110 - MPSC6 00111 - MPSC7 11110 - AUXA 11111 - AUXB</p> <p>NOTE: CH values not listed above are reserved and are not to be used.</p>	X

Table 356: Flex TDM DPRAM Entry (Continued)

Bits	Field Name	Function	Reset Value
7:0	MASK	<p>Mask pattern for the current data.</p> <p>Definition of this field is dependent on B (byte/bit) setting.</p> <p>B=1 (byte mode)</p> <p>In byte mode each bit in the MASK field defines the mask for the corresponding bit in the data.</p> <ul style="list-style-type: none"> • 1 - TDM transmit data is driven out, receive data is processed normally. • 0 - TDM transmit data is not driven out (transmit output is tri-stated), receive data is ignored. <p>B=0 (bit mode)</p> <p>In bit mode only bits 5:0 are valid. Bits 7:6 must be set to '0'.</p> <p>Bits 1:0 define the mask for the current bit:</p> <ul style="list-style-type: none"> • 00 - Z Transmit output is tri-state. Receive data is ignored. • 01 - 0 Transmit output is forced low. Receive data is ignored. • 10 - D Transmit data driven out. Receive data is processed normally. • 11 - 1 (transmit output is forced high, receive data is ignored) <p>Bit 2 serves as the D channel bit. Setting this bit to '1' marks this time slot as a D channel time slot. This bit identifies the MPSC to which the TDM must assert or de-assert the internal CTS signal based on the recent sampled value of the Dgrant bit (see below).</p> <ul style="list-style-type: none"> • If Dgrant is sampled low, CTS is asserted. • If Dgrant is sampled high, CTS is deasserted. • If Dgrant is deasserted while a D channel frame is being transmitted, the MPSC connected to the D channel stops transmission (due to CTS lost) and initiates a collision resolution procedure. <p>Bit 3 - Dgrant/Dreq</p> <p>In a received frame, this bit serves as the Dgrant bit. An IOM (GCI) PHY uses this bit to signal the GT-96100A that it has access to the D channel. In a transmit frame, it serves as the Dreq bit. The GT-96100A drives the Dreq bit to '0' when sending data on the D channel. When not sending data on the D channel, Dreq is driven according to the value programmed in MASK[1:0] bits.</p> <p>Bit 4 identifies the current bit as the IOM-2 MX bit. The GT-96100A recognizes an inactive-to-active transition of received MX bit as an indication that valid IOM-2 monitor data is driven by the PHY.</p> <p>Bit 5 identifies the current bit as the IOM-2 MR bit. The GT-96100A recognizes an inactive-to-active transition of received MR bit as an indication that IOM-2 monitor data, sent by the GT-96100A, have been sampled by the PHY.</p>	X

1. The total number of times that a DPRAM entry is actually repeated is set by both the RPT (bits 24:23) and B (bit 18) fields. The range is 1 (B=0, RPT=00) to 32 (B=1, RPT=11).

NOTES: When the FlexTDM is disabled, the CPU accesses the DPRAM for both reads and writes. When the FlexTDM is enabled, the FlexTDM DPRAM is write only.

After reset the dual port RAM entries are undefined. Users must explicitly program the required entries in order to ensure correct and consistent system operation.

15.3 FlexTDM Programming Modes

After enabling the FlexTDM, there are two ways the user can dynamically program the DPRAM: Single Array Mode and Split Array Mode.

In Single Array Mode, the FlexTDM read pointer ALWAYS returns to entry 0 after receiving a SYNC or reading the last entry (entry with L bit set). The entire range of 256 entries is available for programming a TDM frame. The user can dynamically make changes in the DPRAM but precautions must be made not to write to the same address from which the FlexTDM is reading. This can be done by checking the read pointer before accessing the DPRAM array.

In Split Array mode, the TDM frame is limited to 128 DPRAM entries. The user programs the FlexTDM frame in entries 0-127 and enables the FlexTDM. When a change in programming is needed, the user first programs the new frame in entries 128-255 and then sets the R2HALF (read as “Return To Half”) bit in the TCR. When the next SYNC or last entry occurs, the FlexTDM starts processing the frame as programmed in entries 128-255. The user can then re-program entries 0-127. This process of switching between one half of the DPRAM and the other half simplifies on-the-fly DPRAM changes.

NOTE: If a frame structure is changed (e.g. a change in frame length) while the FlexTDM is enabled, the FlexTDM can lose synchronization. This can lead to a CD lost error and a CTS lost error for all channels that are connected to the FlexTDM.

15.4 FlexTDM Configuration Register (TCR)

Table 357: FlexTDMx Configuration Register (TCR), Offset: 0x008B08, 0x018B08, 0x028B08, 0x038B08 (where x is FlexTDM 0 to 3)

Bits	Field Name	Function	Reset Value
6:0		Reserved	
7	TDDCHG	TDM Delay for D_Channel Grant, 0-low priority 1-high priority	1
10:8	TDTT	TDM Delay for Transparent Transmit TDTT must be set to '101' for proper operation.	101
13:11	TDTR	TDM Delay for Transparent Receive TDTR must be set to '100' for proper operation.	100
14	TTM	TDM Transparent Mode TTM must be set to '0' for proper operation.	0
15	RR2HALF	Receive Return to Half. This bit is used for dynamic programming of the TDM receive frame. 0 - Return to 0. After sync or last, the FlexTDM receive read pointer returns to entry 0 in the Rx DPRAM. 1 - Return to 128. After Sync or Last, the FlexTDM receive read pointer returns to entry 128 in the Rx DPRAM.	0
16	TR2HALF	Transmit Return to Half Used for dynamic programming of the TDM transmit frame. 0 - Return to 0. After sync or last, the FlexTDM transmit read pointer returns to entry 0 in the Tx DPRAM. 1 - Return to 128. After sync or last, the FlexTDM transmit read pointer returns to entry 128 in the Tx DPRAM.	0
19:17	TM	TDM Mode 000 - PCM 001 - Reserved 010 - IOM1 011 - Reserved 100 - IOM2-TE 101 - Reserved 110 - IOM2-LC 111 - Reserved	0

Table 357: FlexTDMx Configuration Register (TCR), Offset: 0x008B08, 0x018B08, 0x028B08, 0x038B08 (where x is FlexTDM 0 to 3) (Continued)

Bits	Field Name	Function	Reset Value
20	SE	Sync Edge (for TRSYNC and TTSYNC) 0 - Sync signals are sampled on falling edge of the clock. 1 - Sync signals are sampled on rising edge of the clock.	0
21	DE	Driving Edge. 0 - Transmit data is sent on rising edge, receive data is sampled on falling edge of the clock. 1 - Transmit data is sent on falling edge, receive data is sampled on rising edge of the clock.	0
22	STZ	Set Tx to Zero When set, the TDM transmit output (TTXD) is forced to zero until a serial clock is available. (IOM-2 mode)	0
23	CRT	Common Receive and Transmit pins 0 - Separate receive and transmit pins. 1 - Common receive and transmit pins. The transmit section of the FlexTDM uses the FlexTDM's Rx clock and Rx sync signals. In this mode, TTCLK and TTSYNC pins are used as general purpose pins.	0
24	CLKDV	Divide CLK by Two 0 - normal (1x) clock mode 1 - double (2x) clock mode Input clock is twice the bit clock. (IOM-2 mode)	0
26:25	TSD	Transmit Sync Delay Specifies the delay (in number of bits) between transmit sync and the first bit of the transmit frame. 00 - No bit delay (IOM-2 mode). 01 - 1-bit delay 10 - 2-bit delay 11 - 3-bit delay	0
28:27	RSD	Receive Sync Delay Specifies the delay (in number of bits) between receive sync and the first bit of the receive frame. 00 - No bit delay (IOM-2 mode) 01 - 1-bit delay 10 - 2-bit delay 11 - 3-bit delay	0

Table 357: FlexTDMx Configuration Register (TCR), Offset: 0x008B08, 0x018B08, 0x028B08, 0x038B08 (where x is FlexTDM 0 to 3) (Continued)

Bits	Field Name	Function	Reset Value
30:29	TDIAG	<p>TDM Diagnostic</p> <p>00 - Normal Operation The received input is connected to the FlexTDM receive pin (TRXD) and transmit output is connected to the FlexTDM transmit pin (TTXD).</p> <p>01 - Echo TDM receive input is echoed on TDM output with one clock delay. The received bit stream is processed normally according to DPRAM programming.</p> <p>10 - Loopback Transmit data is driven on TDM output, as in normal operation, and is also connected internally to the TDM receive line. The received bit stream is processed normally according to DPRAM programming. Transactions on TRXD are not seen by the FlexTDM.</p> <p>11 - Internal Loopback (transmit output is inactive) TDM transmit output is internally connected to the TDM receive input. This mode is useful for TDM loopback testing without affecting the external lines.</p> <p>NOTE: Proper operation of the echo and loopback modes requires that an identical clock is supplied to the TDM's transmit and receive sections. If a TDM entry in the DPRAM is in byte mode (bit 18, 'B' is set to 1) and configured for a MPSC channel in Transparent mode, bits 6 and 7 must never be masked simultaneously. For example, a MASK (bits 7:0 of the TDM entry) of 0b00xx xxxx (where x means 'don't care'), DPRAM entry for MPSC channel in Transparent mode is not allowed in byte mode.</p>	0
31	TEN	<p>Enable FlexTDM</p> <p>0 - FlexTDM disabled. Transmit output is Z.</p> <p>1 - FlexTDM enabled.</p>	0

15.5 FlexTDM Synchronization

After enable, the FlexTDM needs to receive one received sync to achieve synchronization. Until synchronization is achieved, the FlexTDM transmit and receive paths are disabled.

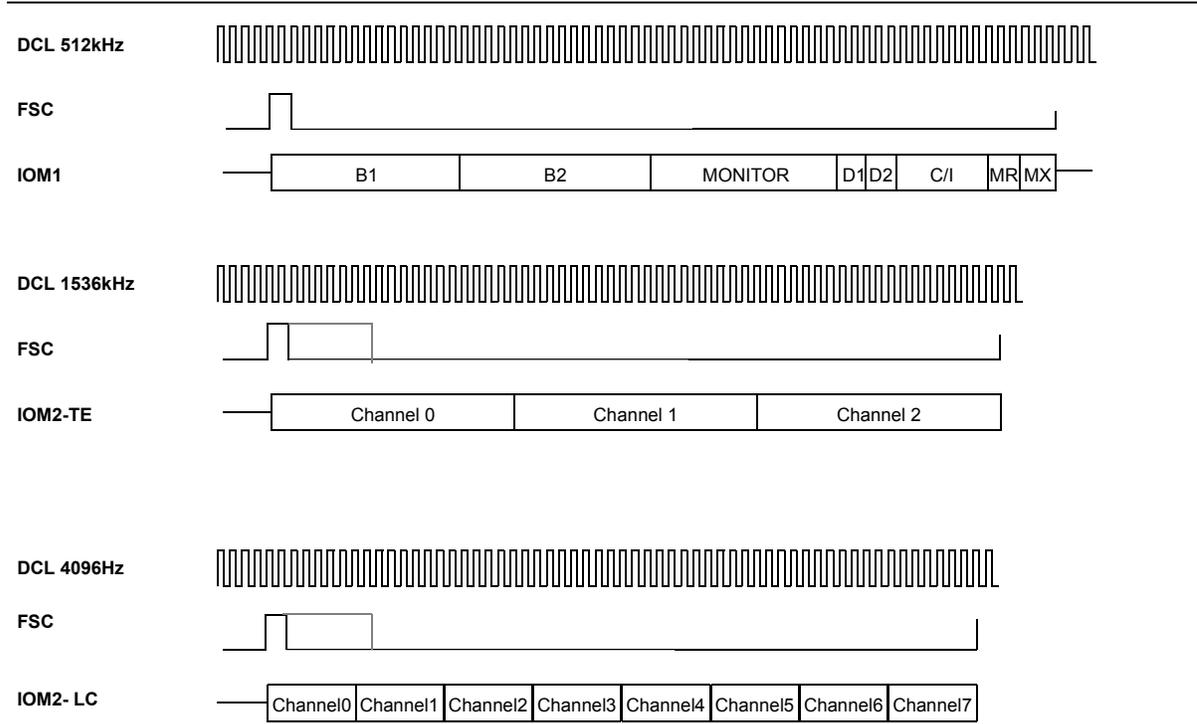
After gaining synchronization, the FlexTDM always predicts when the next sync is expected. If the sync signal is not asserted when expected, the FlexTDM executes a synchronization lost procedure and a maskable interrupt is generated. All the MPSCs that are connected to the FlexTDM experience a CD loss and a CTS loss and all the transmit and receive processes are halted until synchronization is gained again.

15.6 IOM (GCI) Mode

The GCI bus, also known as the IOM-2 bus, is a super set of an older bus known as IOM-1.

While IOM-1 is well suited for NT applications, it lacks the LAPD D channel collision resolution support which is crucial for TE implementation where multiple accesses to the D channel are allowed. The GT-96100A's FlexTDM supports both the IOM-1 and IOM-2 frame structures. Figure 73 depicts various IOM frame structures.

Figure 73: Typical IOM Structures



15.6.1 IOM-1 Frames

An IOM-1 frame is constructed from two 8-bit B channels, one 2-bit D channel, a monitor channel, and a C/I channel.

The monitor channel is used to transfer data between the CPU and an ISDN PHY device. This channel is also used for PHY chip programming and layer-2 message transferring. The MR and MX bits are used to control the data transfers on the monitor channel.

The C/I channel is used to transfer layer-1 messages between the CPU and the PHY chip.

In IOM-1 mode no D channel control is supported and the D channel is assumed to be always granted. This mode can be used when interfacing PHY chips such as the Siemens PEB-2081 and PEB-2086.

15.6.2 IOM-2-TE Frames

An IOM-2-TE frame consists of three IOM channels (channels 0,1,2).

The C/I channel of the third sub-frame (i.e. IOM channel 2) is used for TIC bus applications. The TIC bus is used by the PHY device to grant D channel access. Usually, bit 4 of C/I channel 2 is used as the D channel request/grant bit. However, this is programmable in the GT-96100A and it is possible to specify any desired bit as the request/grant bit.

The GT-96100A fully supports IOM-2-TE frames. In IOM-2-TE mode, the GT-96100A handles the D channel REQ/GNT protocol by itself. The GT-96100A accesses the D channel only when the received Dgrant bit is asserted low. The GT-96100A drives the Dreq bit to '0' when sending data over the D channel. Otherwise, it drives the value programmed in MASK[1:0] bits (see [Table 356](#) for more information).

15.6.3 IOM-2-LC Frames

The IOM-2 line card frame is used to connect up to eight ISDN devices on the same card. An IOM-2-LC frame consists of eight IOM channels.

The GT-96100A can be programmed to access any one of the IOM channels. The monitor and C/I channels refer to the selected IOM channel. However, other MPSCs can be used to access other time slots in the IOM-2-LC frame.

15.7 PCM Highway Mode

This is a free programming mode where each MPSC can be connected to any time slot in the programmed frame.

15.8 Data Rate Adoption

Since it is capable of accessing each bit of the serial TDM stream separately, the FlexTDM supports data rate adoption.

The FlexTDM DPRAM allows programmable routing of each bit (or byte) in the data to any of the MPSCs and supports masking of bits which are to be ignored. The clock pulse associated with a masked bit is “stolen” from the MPSC (or AUX channel). Thus, the serial controller is clocked at an effective rate that is appropriate for the logical data stream.

15.9 FlexTDM Auxiliary Channels A and B

The auxiliary channels were designed to support the IOM-2 monitor and C/I channels. These are simple channels accessible by reading and writing 8 bit registers.

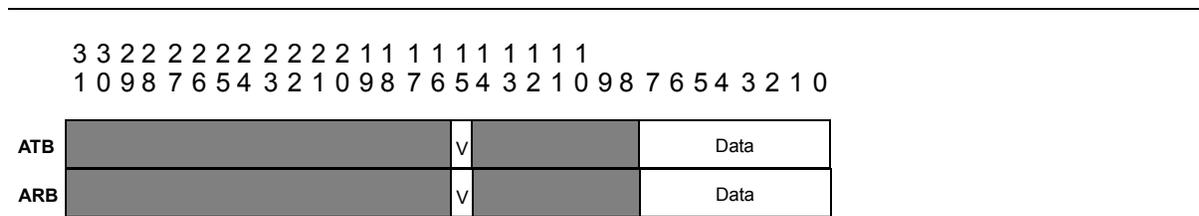
Table 358: Monitor Channel Handshaking Process

Monitor Channel	MX Outgoing Frame	MR Incoming Frame	Interrupt	MX Incoming Frame	MR Outgoing Frame	Interrupt
FF	1	1		1	1	
FF	1	1		1	1	
1st Byte	0	1		0	1	Rx
1st Byte	0	0	Tx	0	0	
2nd Byte	1	0		1	0	
2nd Byte	0	0		0	0	Rx
2nd Byte	0	1		0	1	
2nd Byte	0	0	Tx	0	0	
FF	1	0		1	0	
FF	1	0		1	0	
FF	1	1		1	1	
FF	1	1		1	1	
FF	1	1		1	1	
FF	1	1		1	1	

15.9.2 Auxiliary Channel B

Channel B is used to interface to the 32Kbps C/I channel. The C/I channel is used to transfer 4-bit layer-1 commands between the CPU and the PHY chip. A command is considered valid after it is received twice (i.e. after receiving 8 bits).

Figure 75: Channel B Control Register



The CPU loads ATB with the data it wants to transmit and sets the V bit to 1. Auxiliary channel B transmitter loads the new data into its internal shift register, clears the V bit and generates a maskable interrupt. The data is transmitted constantly until new data is loaded.

NOTE: The C/I channel messages are 4 bits wide. The message must be duplicated to create an 8-bit word, which is written to the 8-bit ATB register.

When the transmit FlexTDM loses synchronization (i.e. when the SYNC signal is asserted not where expected), the auxiliary channel B transmitter flushes its shift register and stop its transmit process. The transmit process restarts when the FlexTDM regains synchronization.

The auxiliary channel B receiver generates an interrupt after it loads into ARB register a *new* C/I command. A new command is recognized when it is different from the previous loaded command. The channel receiver sets the V bit whenever it loads data into the ARB register. The CPU clears the V bit when it reads ARB.

When the receive FlexTDM loses synchronization, the auxiliary channel B receiver flushes its shift register and stops its receive process. The receive process restarts when the FlexTDM regains synchronization. The ARB contents are not affected by synchronization lost.

NOTES: The GT-96100A sets the V bit whenever it loads data, even if no new command was received. However, the GT-96100A interrupts the CPU only when new command was received.

When the FlexTDM loses synchronization (i.e., when SYNC is not asserted where expected), the V bit is cleared, even if new data has been loaded into ATB register. This prevents the new data from being transmitted. The driver software must rewrite the data to the ATB register when there is loss of synchronization (Loss of synchronization can be recognized using an interrupt or by polling).

15.10 IOM Programming

Table 359, Table 360 and Table 361 provide the recommended DPRAM programming for the IOM bus. If IOM mode is selected, the programming recommendations must be followed for the appropriate collision resolution process.

NOTE: In these tables, B channels mask bits are set to '1'. However, the CPU must complete layer-2 negotiations before granting a MPSC access to one of the B channels.

Table 359: IOM-1 Programming

Entry Number	L	RPT	STRB	B	CH	MASK	Comments
0	0	00	xx	1	00000	11111111	B channel 0 to MPSC0 (1 byte).
1	0	00	xx	1	00001	11111111	B channel 1 to MPSC1 (1 byte).
2	0	00	xx	1	11110	11111111	Monitor to AUXA (1 byte).
3	0	01	xx	0	00010	00000110	D channel to MPSC2 (2 bits). NOTE: The MASK setting for D channel data.
4	0	11	xx	0	11111	00000010	C/I to AUXB (4 bits).

Table 359: IOM-1 Programming (Continued)

Entry Number	L	RPT	STRB	B	CH	MASK	Comments
5	0	00	xx	0	11110	00010001 or 00010011	Receive: MX bit to AUXA (1bit). Transmit: force 0 or 1 on MR using MASK[1:0] (1bit).
6	1	00	xx	0	11110	00100001 or 00100011	Receive: MR bit to AUXA (1bit). Transmit: force 0 or 1 on MX using MASK[1:0] (1bit). L indicates last entry in frame.

Table 360: IOM2-TE Programming

Entry No	L	RPT	STRB	B	CH	MASK	Comments
0	0	00	xx	1	00000	11111111	B channel 0 (in IOM channel 0) to MPSC0 (1 byte).
1	0	00	xx	1	00001	11111111	B channel 1 (in IOM channel 0) to MPSC1 (1 byte).
2	0	00	xx	1	11110	11111111	Monitor (in IOM channel 0) to AUXA (1 byte).
3	0	01	xx	0	00010	00000110	D channel (in IOM channel 0) to MPSC2 (2 bits). Note the MASK setting for D channel data.
4	0	11	xx	0	11111	00000010	C/I (in IOM channel 0) to AUXB (4 bits).
5	0	00	xx	0	11110	00010001 or 00010011	Receive: MX bit (in IOM channel 0) to AUXA (1bit). Transmit: force 0 or 1 on MR using MASK[1:0] (1bit).
6	0	00	xx	0	11110	00100001 or 00100011	Receive: MR bit (in IOM channel 0) to AUXA (1bit). Transmit: force 0 or 1 on MX using MASK[1:0] (1bit).
7	0	11	xx	1	11110	00000000	Skip IOM channel 1 (4 bytes).
8	0	10	xx	1	11110	00000000	Skip part of IOM channel 2 (3 bytes).
9	0	01	xx	0	11110	00000000	Skip more (2 bits).
10	0	00	xx	0	11110	00001000	D-grant/D-req bit in C/I channel of IOM channel 2 (1 bit).

Table 360: IOM2-TE Programming (Continued)

Entry No	L	RPT	STRB	B	CH	MASK	Comments
11	0	11	xx	0	11110	00000000	Skip (4 bits).
12	1	00	xx	0	11110	00000000	Skip to end of frame (1 bit). L is set to indicate last entry in frame.

Table 361: IOM2-LC (connected to channel 3) GCI

Entry No	L	RPT	STRB	B	CH	MASK	Comments
0	0	11	xx	1	11110	00000000	Skip IOM channel 0 (4 bytes).
1	0	11	xx	1	11110	00000000	Skip IOM channel 1 (4 bytes).
2	0	11	xx	1	11110	00000000	Skip IOM channel 2 (4 bytes).
3	0	00	xx	1	00000	11111111	B channel 0 (in IOM channel 3) to MPSC0 (1 byte).
4	0	00	xx	1	00001	11111111	B channel 1 (in IOM channel 3) to MPSC1 (1 byte).
5	0	00	xx	1	11110	11111111	Monitor (in IOM channel 3) to AUXA (1 byte).
6	0	01	xx	0	00010	00000110	D channel (in IOM channel 3) to MPSC2 (2 bits). NOTE: The MASK setting for D channel data.
7	0	11	xx	0	11111	00000010	C/I (in IOM channel 3) to AUXB (4 bits).
8	0	00	xx	0	11110	00010001 or 00010011	Receive: MX bit (in IOM channel 3) to AUXA (1bit). Transmit: force 0 or 1 on MR using MASK[1:0] (1bit).
9	0	00	xx	0	11110	00100010 or 00100011	Receive: MR bit (in IOM channel 3) to AUXA (1bit). Transmit: force 0 or 1 on MX using MASK[1:0] (1bit).
10	0	11	xx	1	11110	00000000	Skip IOM channel 4 (4 bytes).
11	0	11	xx	1	11110	00000000	Skip IOM channel 5 (4 bytes).

Table 361: IOM2-LC (connected to channel 3) GCI (Continued)

Entry No	L	RPT	STRB	B	CH	MASK	Comments
12	0	11	xx	1	11110	00000000	Skip IOM channel 6 (4 bytes).
13	1	11	xx	1	11110	00000000	Skip IOM channel 7 (4 bytes). L is set to indicate last entry in frame.

15.11 FlexTDM Registers

Table 362: FlexTDM Register Map

Description	Offset	Page Number
<i>FlexTDM0</i>		
FlexTDM0 Transmit Dual Port RAM (TDPR0), block 0	0x000B00 - 0x000BFF	
FlexTDM0 Transmit Dual Port RAM (TDPR0), block 1	0x001B00 - 0x001BFF	
FlexTDM0 Transmit Dual Port RAM (TDPR0), block 2	0x002B00 - 0x002BFF	
FlexTDM0 Transmit Dual Port RAM (TDPR0), block 3	0x003B00 - 0x003BFF	
FlexTDM0 Receive Dual Port RAM (RDPR0), block 0	0x004B00 - 0x004BFF	
FlexTDM0 Receive Dual Port RAM (RDPR0), block 1	0x005B00 - 0x005BFF	
FlexTDM0 Receive Dual Port RAM (RDPR0), block 2	0x006B00 - 0x006BFF	
FlexTDM0 Receive Dual Port RAM (RDPR0), block 3	0x007B00 - 0x007BFF	
FlexTDM0 Transmit Read Pointer (TRP0)	0x008B00	
FlexTDM0 Receive Read Pointer (TRP0)	0x008B04	
FlexTDM0 Configuration Register (TCR0)	0x008B08	page 384
FlexTDM0 AUX ChannelA TX Register (ATA0)	0x008B0C	
FlexTDM0 AUX ChannelA RX Register (ARA0)	0x008B10	
FlexTDM0 AUX ChannelB TX Register (ATB0)	0x008B14	
FlexTDM0 AUX ChannelB RX Register (ARB0)	0x008B18	
<i>FlexTDM1</i>		
FlexTDM1 Transmit Dual Port RAM (TDPR1), block 0	0x010B00 - 0x010BFF	
FlexTDM1 Transmit Dual Port RAM (TDPR1), block 1	0x011B00 - 0x011BFF	
FlexTDM1 Transmit Dual Port RAM (TDPR1), block 2	0x012B00 - 0x012BFF	
<i>FlexTDM1 (Continued)</i>		

Table 362: FlexTDM Register Map (Continued)

Description	Offset	Page Number
FlexTDM1 Transmit Dual Port RAM (TDPR1), block 3	0x013B00 - 0x013BFF	
FlexTDM1 Receive Dual Port RAM (RDPR1), block 0	0x014B00 - 0x014BFF	
FlexTDM1 Receive Dual Port RAM (RDPR1), block 1	0x015B00 - 0x015BFF	
FlexTDM1 Receive Dual Port RAM (RDPR1), block 2	0x016B00 - 0x016BFF	
FlexTDM1 Receive Dual Port RAM (RDPR1), block 3	0x017B00 - 0x017BFF	
FlexTDM1 Transmit Read Pointer (TRP1)	0x018B00	
FlexTDM1 Receive Read Pointer (TRP1)	0x018B04	
FlexTDM1 Configuration Register (TCR1)	0x018B08	page 384
FlexTDM1 AUX ChannelA TX Register (ATA1)	0x018B0C	
FlexTDM1 AUX ChannelA RX Register (ARA1)	0x018B10	
FlexTDM1 AUX ChannelB TX Register (ATB1)	0x018B14	
FlexTDM1 AUX ChannelB RX Register (ARB1)	0x018B18	
FlexTDM2		
FlexTDM2 Transmit Dual Port RAM (TDPR2), block 0	0x020B00 - 0x020BFF	
FlexTDM2 Transmit Dual Port RAM (TDPR2), block 1	0x021B00 - 0x021BFF	
FlexTDM2 Transmit Dual Port RAM (TDPR2), block 2	0x022B00 - 0x022BFF	
FlexTDM2 Transmit Dual Port RAM (TDPR2), block 3	0x023B00 - 0x023BFF	
FlexTDM2 Receive Dual Port RAM (RDPR2), block 0	0x024B00 - 0x024BFF	
FlexTDM2 Receive Dual Port RAM (RDPR2), block 1	0x025B00 - 0x025BFF	
FlexTDM2 Receive Dual Port RAM (RDPR2), block 2	0x026B00 - 0x026BFF	
FlexTDM2 Receive Dual Port RAM (RDPR2), block 3	0x027B00 - 0x027BFF	
FlexTDM2 Transmit Read Pointer (TRP2)	0x028B00	
FlexTDM2 Receive Read Pointer (TRP2)	0x028B04	
FlexTDM2 Configuration Register (TCR2)	0x028B08	page 384
FlexTDM2 AUX ChannelA TX Register (ATA2)	0x028B0C	
FlexTDM2 AUX ChannelA RX Register (ARA2)	0x028B10	
FlexTDM2 AUX ChannelB TX Register (ATB2)	0x028B14	
FlexTDM2 AUX ChannelB RX Register (ARB2)	0x028B18	
FlexTDM3		
FlexTDM3 Transmit Dual Port RAM (TDPR3), block 0	0x030B00 - 0x030BFF	

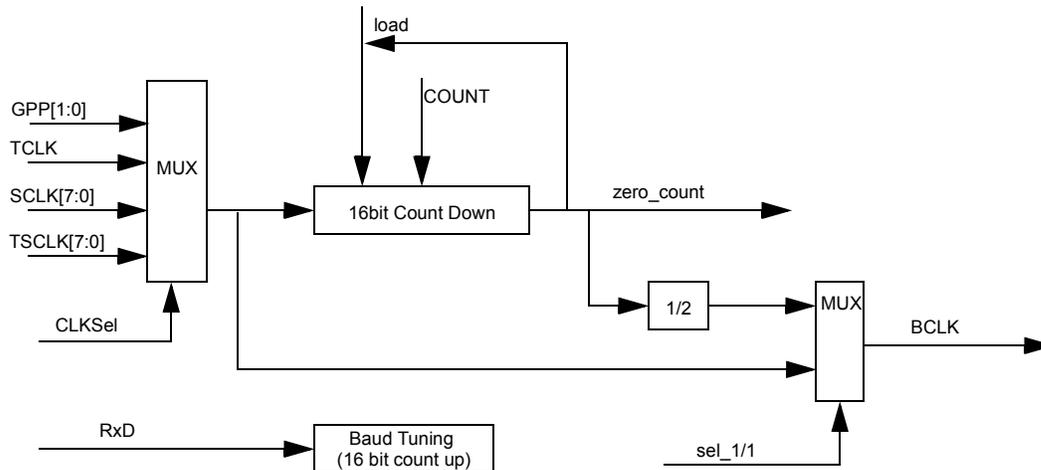
Table 362: FlexTDM Register Map (Continued)

Description	Offset	Page Number
FlexTDM3 Transmit Dual Port RAM (TDPR3), block 1	0x031B00 - 0x031BFF	
FlexTDM3 Transmit Dual Port RAM (TDPR3), block 2	0x032B00 - 0x032BFF	
FlexTDM3 Transmit Dual Port RAM (TDPR3), block 3	0x033B00 - 0x033BFF	
FlexTDM3 Receive Dual Port RAM (RDPR3), block 0	0x034B00 - 0x034BFF	
FlexTDM3 Receive Dual Port RAM (RDPR3), block 1	0x035B00 - 0x035BFF	
FlexTDM3 Receive Dual Port RAM (RDPR3), block 2	0x036B00 - 0x036BFF	
FlexTDM3 Receive Dual Port RAM (RDPR3), block 3	0x037B00 - 0x037BFF	
FlexTDM3 Transmit Read Pointer (TRP3)	0x038B00	
FlexTDM3 Receive Read Pointer (TRP3)	0x038B04	
FlexTDM3 Configuration Register (TCR3)	0x038B08	page 384
FlexTDM3 AUX ChannelA TX Register (ATA3)	0x038B0C	
FlexTDM3 AUX ChannelA RX Register (ARA3)	0x038B10	
FlexTDM3 AUX ChannelB TX Register (ATB3)	0x038B14	
FlexTDM3 AUX ChannelB RX Register (ARB3)	0x038B18	

16. BAUD RATE GENERATORS (BRGs)

There are eight baud rate generators (BRGs) in the GT-96100A. [Figure 76](#) shows a BRG block diagram.

Figure 76: Baud Rate Generator Block Diagram



16.1 BRG Inputs and Outputs

There are 19 clock inputs to the baud rate generators (BRGs). Two general purpose pins can be programmed to function as clock inputs for the BRGs: GPP[0] and GPP[1]. Additionally, each of the serial input clocks can be used as a BRG clock. Finally, the TCLK, which is the system parallel clock, is also an option.

When a BRG is enabled, it loads the Count Down Value (CDV), from the BRG configuration register, into its count down counter. When the counter expires (i.e. reaches zero), the BRG clock output, BCLK, is toggled and the counter reloads.

16.2 BRG Baud Tuning

A baud tuning mechanism can be used to adjust the generated clock rate to the receive clock rate.

When baud tuning is enabled, the baud tuning mechanism monitors for a start bit, i.e. High-to-Low transition. When a start bit is found, the baud tuning machine measures the bit length by counting up until the next Low-to-High transition. The count-up value of the BRG is then loaded into the Count Up Value (CUV) register and a maskable interrupt is generated signaling the CPU that the bit length value is available. The CPU reads the value from the CUV and adjusts the CDV to the requested value.

The CUV can be used to adjust the CDV, in the BRG configuration register, to the requested value.

16.3 BRG Registers

Table 363: BRG Registers Map

Register Name	Offset	Page
BRG0		
BRG0 Configuration Register (BCR0)	0x102A00	page 399
BRG0 Baud Tuning Register (BTR0)	0x102A04	page 400
BRG1		
BRG1 Configuration Register (BCR1)	0x102A08	For a description of the BRG1 registers, see the descriptions for the BRG0 registers.
BRG1 Baud Tuning Register (BTR1)	0x102A0C	
BRG2		
BRG2 Configuration Register (BCR2)	0x102A10	For a description of the BRG2 registers, see the descriptions for the BRG0 registers.
BRG2 Baud Tuning Register (BTR2)	0x102A14	
BRG3		
BRG3 Configuration Register (BCR3)	0x102A18	For a description of the BRG3 registers, see the descriptions for the BRG0 registers.
BRG3 Baud Tuning Register (BTR3)	0x102A1C	
BRG4		
BRG4 Configuration Register (BCR4)	0x102A20	For a description of the BRG4 registers, see the descriptions for the BRG0 registers.
BRG4 Baud Tuning Register (BTR4)	0x102A24	
BRG5		
BRG5 Configuration Register (BCR5)	0x102A28	For a description of the BRG5 registers, see the descriptions for the BRG0 registers.
BRG5 Baud Tuning Register (BTR5)	0x102A2C	
BRG6		
BRG6 Configuration Register (BCR6)	0x102A30	For a description of the BRG6 registers, see the descriptions for the BRG0 registers.
BRG6 Baud Tuning Register (BTR6)	0x102A34	
BRG7		
BRG7 Configuration Register (BCR7)	0x102A38	For a description of the BRG7 registers, see the descriptions for the BRG0 registers.
BRG7 Baud Tuning Register (BTR7)	0x102A3C	

Table 364: BRGx Configuration Register (BCR)

Bits	Name	Description	Reset Value
15:0	CDV	<p>Count Down Value.</p> <p>The user programs the CDV field to define the baud rate that the BRG generates. CDV is loaded into the BRG counter every time it reaches 0. The actual baud rate is:</p> $\text{BaudRate} = \frac{\text{InputClockRate}}{(\text{CDV}+1) \times 2}$ <p>When CDV is 0x0000, the generated baud rate is equal to the input clock rate.</p>	0
16	En	<p>Enable BRG</p> <p>0 - Disabled. Output clock is clamped to 0.</p> <p>1 - Enabled.</p>	0
17	RST	<p>Reset BRG</p> <p>0 - No Op.</p> <p>1 - Reset BRG counter to 0.</p>	0
22:18	CLKS	<p>Clock Source (input clock to the BRG)</p> <p>00000 - BCLK0 (from GPP[0] pin)</p> <p>00001 - BCLK1 (from GPP[1] pin)</p> <p>00010 - SCLK0 (from PA[5] pin)</p> <p>00011 - TSCLK0 (from PA[6] pin)</p> <p>00100 - SCLK1 (from PB[5] pin)</p> <p>00101 - TSCLK1 (from PB[6] pin)</p> <p>00110 - SCLK2 (from PC[5] pin)</p> <p>00111 - TSCLK2 (from PC[6] pin)</p> <p>01000 - SCLK3 (from PD[5] pin)</p> <p>01001 - TSCLK3 (from PD[6] pin)</p> <p>01010 - SCLK4 (from PE[5] pin)</p> <p>01011 - TSCLK4 (from PE[6] pin)</p> <p>01100 - SCLK5 (from PF[5] pin)</p> <p>01101 - TSCLK5 (from PF[6] pin)</p> <p>01110 - SCLK6 (from MII0[12] pin)</p> <p>01111 - TSCLK6 (from MII0[1] pin)</p> <p>10000 - SCLK7 (from MII0[13] pin)</p> <p>10001 - TSCLK7 (from MII0[6] pin)</p> <p>10010 - TCLK</p>	10010
24:23		Reserved.	0

Table 364: BRGx Configuration Register (BCR) (Continued)

Bits	Name	Description	Reset Value
25	BT	Baud Tuning 0 - Disabled 1 - Enabled Setting BT to 1 enables the baud tuning for the duration of one start bit. When the Start Bit Length calculation is done, the GT-96100A clears the BT bit.	0
31:24		Reserved	0

Table 365: BRGx Baud Tuning register (BTR)

NOTE: If the BRG is written for a clock source that is inactive, this register cannot be accessed, see [Table 364](#) bits [22:18].

Bits	Name	Description	Reset Value
15:0	CUV	Count Up Value NOTE: These bits are read only.	0
31:16		Reserved.	0

Table 366: Watchdog Configuration register (WDC), Offset: 0x101A80 (Continued)

Bits	Field Name	Function	Initial Value
30	WDE	Watchdog Expiration When the watchdog counter expires, this bit is asserted. The WDE* pin can be used to reset the entire system. This bit is read only.	1
31	EN	Enable 0 - Watchdog is disabled, counter is loaded with Preset_VAL. NMI and WDE are set to '1'. 1 - Watchdog is enabled. This bit is read only.	0

Table 367: Watchdog Value register (WDV), Offset: 0x101A84

Bits	Field Name	Function	Reset Value
23:0	NMI_VAL	NMI_VAL are the 24 least significant bits of a 32-bit value. The upper 8 bits are always '00'. When the Watchdog counter reaches a value equal to the NMI value NMI* pin is asserted. The actual NMI value is a 32-bit number equal to {0x00,NMI_VAL}.	0x000.0000
31:24		Reserved.	0

17.2 Watchdog Operation

After reset, the watchdog is disabled.

The watchdog must be serviced periodically in order to avoid NMI or reset (WDE*). Watchdog service is performed by writing '01' to CTL2, followed by writing '10' to CTL2. Upon watchdog service, the GT-96100A clears the NMI and WDE bits (if set) and reloads the Preset_VAL into the watchdog counter.

A write sequence of '01' followed by '10' into CTL1 disables/enables the watchdog. The watchdog's current status can be read in bit 31 of WDC. When disabled, the GT-96100A sets the NMI and WDE bits (if clear) and reloads the Preset_VAL into the watchdog counter.

Preset_VAL and NMI_VAL can be changed while the watchdog is enabled. However, Preset_VAL will affect the watchdog only after it is loaded into the watchdog counter (e.g. after watchdog service).

If the watchdog is not serviced before the counter reaches NMI_VAL, a non-maskable interrupt event occurs. a watchdog expiration event occurs. The NMI bit is reset, asserting low the NMI* pin.

In order to deassert the NMI* and/or WDE* pins, the watchdog must be serviced, disabled or the GT-96100A must be reset. The GT-96100A holds WDE* asserted for the duration of 16 system cycles after reset assertion.

18. TIMERS/COUNTERS

There are three 24-bit wide and one 32-bit wide timer/counters on the GT-96100A. Each one can be selected to operate as a timer or as a counter.

NOTE: The count frequency for the timer/counters is equal to TCLK frequency

In Counter mode, the counter counts down to terminal count, stops, and issue an interrupt.

In Timer mode, the timer counts down, issues an interrupt on terminal count, and resets itself to the programmed countdown value, and begins to count down.

Reads from the counter or timer are done from the counter itself, while writes are to its register. For example, note that even though the registers are programmed to an initial value of 0 the counters read 0xffffffff.

In order to reprogram a timer/counter:

1. Disable the timer/counter.
2. Load it with a new value.
3. Enable it as appropriate, counter or timer.

NOTE: There are no external input pins for enable/disable nor are there any output timer pins on the GT-96100A.

18.1 Timer / Counter Registers

Table 368: Timer/Counter 0, Offset: 0x850

Bits	Field Name	Function	Initial Value
31:0	TC0Value	The counter or timer initial value.	0xffffffff

Table 369: Timer/Counter 1, Offset: 0x854

Bits	Field Name	Function	Initial Value
23:0	TC1Value	The counter or timer initial value.	0xffffffff
31:24	Reserved	Reserved.	0x0

Table 370: Timer/Counter 2, Offset: 0x858

Bits	Field Name	Function	Initial Value
23:0	TC2Value	The counter or timer initial value.	0xffffffff
31:24	Reserved	Reserved.	0x0

Table 371: Timer/Counter 3, Offset: 0x85c

Bits	Field Name	Function	Initial Value
23:0	TC3Value	The counter or timer initial value.	0xffffffff
31:24	Reserved	Reserved.	0x0

Table 372: Timer/Counter Control, Offset: 0x864

Bits	Field Name	Function	Initial Value
0	EnTC0	The timer/counter counts only when it is enabled. 0 - Disable 1 - Enable	0x0
1	SelTC0	Timer or Counter Selection 0 - Counter 1 - Timer	0x0
2	EnTC1	The timer/counter counts only when it is enabled. 0 - Disable 1 - Enable	0x0
3	SelTC1	Timer or Counter Selection 0 - Counter 1 - Timer	0x0
4	EnTC2	The timer/counter counts only when it is enabled. 0 - Disable 1 - Enable	0x0
5	SelTC2	Timer or Counter Selection 0 - Counter 1 - Timer	0x0
6	EnTC3	The timer/counter counts only when it is enabled. 0 - Disable 1 - Enable	0x0
7	SelTC3	Timer or Counter Selection 0 - Counter 1 - Timer	0x0
31:8	Reserved	Reserved.	0x0

19. GENERAL PURPOSE PORTS

19.1 Overview

The GT-96100A supports up to 88 general purpose pins.

Most of these pins are shared (multiplexed) with the serial ports (ports A through F) and with the MII ports (MII0 and MII1). The result is that in most applications less than 88 pins can be configured as GPPs. However, the control of the ports in GT-96100A is made very flexible, providing the user with a pin level configuration capability. Thus, for example, if port A is used for serial communication but one of its pins is not needed (e.g. PA[6]), this single pin can be programmed to function as a general purpose pin, without affecting the other pins associated with the serial function.

GT-96100A also provides the capability of generating interrupts for each of the GPP pins.

19.2 General Purpose Control Registers

The General Purpose registers control the behavior of the pins associated with the GPP function. There are 12 registers, which are divided into 3 groups. The groups are defined as follows:

- Group 0 includes GPP, A and B ports.
- Group 1 includes C, D, E and F ports.
- Group 2 includes MII0 and MII1 ports.

Each group of registers comprises four registers, used to configure the group pins.

Table 373: Control Registers

Register	Description
General Purpose Configuration registers	GPC0, GPC1, and GPC2 These registers control the setting of the pins as either general-purpose I/O or peripheral function. When a pin is configured as peripheral, its exact functionality is set according to the port's connection setting. For example, if port C is connected to MPSC2, PC[0] is used as RXD2. If port C is tied to TDM0, PC[0] can be set to either TRXD0 or TTXD0.
General Purpose Input/Output registers	GPIO0, GPIO1, and GPIO2 These registers control the direction of the pins (either input or output). NOTE: GPIO registers affect pin functionality not only when the pin is configured as general purpose I/O, but also when the pin is configured as peripheral function. For example, If port C is tied to TDM0, setting PC[0] as TRXD0 (input) or TTXD0 (output) is done by programming GPIO1 accordingly.

Table 373: Control Registers (Continued)

Register	Description
General Purpose Data registers	<p>GPD0, GPD1, and GPD2</p> <p>These registers hold the data associated with the general purpose I/O pins. These registers' bits are read-only for general purpose inputs and are read-write for general purpose outputs. The data read from these registers reflect the pin state, while writing to these registers sets the data to be driven out (for output pins).</p> <p>When a GPP pin is configured as input and its associated GPD bit is 0, an interrupt is set. An interrupt is set regardless of the GPC state.</p>
General Purpose Level registers	<p>GPL0, GPL1, and GPL2</p> <p>These registers define the polarity associated with pins that are configured as general purpose input. For each pin defined as asserted low, the pin state is inverted before being processed in the device. Thus, if a pin is externally tied to '0', but is configured in the GPL register as negative polarity, the data read from the respective GPD bit is '1'.</p>

The registers are described in the following tables.

Table 374: GPP Registers Map

Description	Offset	Page Number
General Purpose Configuration		
General Purpose Configuration 0 (GPC0)	0x100A00	page 407
General Purpose Configuration 1 (GPC1)	0x100A04	page 409
General Purpose Configuration 2 (GPC2)	0x100A08	page 412
General Purpose Input/Output		
General Purpose Input/Output 0 (GPIO0)	0x100A20	page 407
General Purpose Input/Output 1 (GPIO1)	0x100A24	page 410
General Purpose Input/Output 2 (GPIO2)	0x100A28	page 412
General Purpose Data		
General Purpose Data 0 (GPD0)	0x100A40	page 408
General Purpose Data 1 (GPD1)	0x100A44	page 410
General Purpose Data 2 (GPD2)	0x100A48	page 413
General Purpose Level		
General Purpose Level 0 (GPL0)	0x100A60	page 408
General Purpose Level 1 (GPL1)	0x100A64	page 411
General Purpose Level 2 (GPL2)	0x100A68	page 413

Table 375: General Purpose Configuration 0 (GPC0), Offset: 0x100A00

Bits	Field Name	Function	Initial Value
15:0	GPPC	General Purpose Port Configuration Each bit controls the setting of the respective pin in the GPP port. 0 - pin is configured as General Purpose I/O. 1 - pin is configured as peripheral function.	0
22:16	PAC	Port A Configuration Each bit controls the setting of the respective pin in port A. 0 - pin is configured as General Purpose I/O. 1 - pin is configured as peripheral function.	0
23		Reserved.	0
30:24	PBC	Port B Configuration Each bit controls the setting of the respective pin in port B. 0 - pin is configured as General Purpose I/O. 1 - pin is configured as peripheral function.	0
31		Reserved.	0

Table 376: General Purpose Input/Output 0 (GPIO0), Offset: 0x100A20

Bits	Field Name	Function	Initial Value
15:0	GPPIO	General Purpose Port I/O Each bit controls the setting of the respective pin in the GPP port as input or output. 0 - pin is configured as input. 1 - pin is configured as output.	0
22:16	PAIO	Port A I/O Each bit controls the setting of the respective pin in port A as input or output. 0 - pin is configured as input. 1 - pin is configured as output.	0
23		Reserved.	0
30:24	PBIO	Port B I/O. Each bit controls the setting of the respective pin in port B as input or output. 0 - pin is configured as input. 1 - pin is configured as output.	0
31		Reserved.	0

Table 377: General Purpose Data 0 (GPD0), Offset: 0x100A40

Bits	Field Name	Function	Initial Value
15:0	GPPD	General Purpose Port Data Each bit holds the data of the respective pin in GPP port. If the pin is configured as input, this bit is read-only and reflects the signal status at the pin. If the pin is configured as output, this bit is read-write and controls the data driven out.	0
22:16	PAD	Port A Data Each bit holds the data of the respective pin in port A. If the pin is configured as input, this bit is read-only and reflects the signal status at the pin. If the pin is configured as output, this bit is read-write and controls the data driven out.	0
23		Reserved.	0
30:24	PBD	Port B Data Each bit holds the data of the respective pin in port B. If the pin is configured as input, this bit is read-only and reflects the signal status at the pin. If the pin is configured as output, this bit is read-write and controls the data driven out.	0
31		Reserved.	0

Table 378: General Purpose Level 0 (GPL0), Offset: 0x100A60

Bits	Field Name	Function	Initial Value
15:0	GPPL	General Purpose Port Level Each bit defines the polarity of the respective pin in GPP port. 0 - positive (normal) polarity. 1 - negative polarity. If the pin is configured as input, it is inverted before being processed inside the device.	0
22:16	PAL	Port A Level Each bit defines the polarity of the respective pin in port A. 0 - positive (normal) polarity. 1 - negative polarity. If the pin is configured as input, it is inverted before being processed inside the device.	0
23		Reserved.	0
30:24	PBL	Port B Level Each bit defines the polarity of the respective pin in port B. 0 - positive (normal) polarity. 1 - negative polarity. If the pin is configured as input, it is inverted before being processed inside the device.	0

Table 378: General Purpose Level 0 (GPL0), Offset: 0x100A60 (Continued)

Bits	Field Name	Function	Initial Value
31		Reserved.	0

Table 379: General Purpose Configuration 1 (GPC1), Offset: 0x100A04

Bits	Field Name	Function	Initial Value
6:0	PCC	Port C Configuration Each bit controls the setting of the respective pin in port C. 0 - pin is configured as General Purpose I/O. 1 - pin is configured as peripheral function.	0
7		Reserved.	0
14:8	PDC	Port D Configuration Each bit controls the setting of the respective pin in port D. 0 - pin is configured as General Purpose I/O. 1 - pin is configured as peripheral function.	0
15		Reserved.	0
22:16	PEC	Port E Configuration Each bit controls the setting of the respective pin in port E. 0 - pin is configured as General Purpose I/O. 1 - pin is configured as peripheral function.	0
23		Reserved.	0
30:24	PFC	Port F Configuration Each bit controls the setting of the respective pin in port F. 0 - pin is configured as General Purpose I/O. 1 - pin is configured as peripheral function.	0
31		Reserved.	0

Table 380: General Purpose Input/Output 1 (GPIO1), Offset: 0x100A24

Bits	Field Name	Function	Initial Value
6:0	PCIO	Port C I/O Each bit controls the setting of the respective pin in port C as input or output. 0 - pin is configured as input. 1 - pin is configured as output.	0
7		Reserved.	0
14:8	PDIO	Port D I/O Each bit controls the setting of the respective pin in port D as input or output. 0 - pin is configured as input. 1 - pin is configured as output.	0
15		Reserved.	0
22:16	PEIO	Port E I/O Each bit controls the setting of the respective pin in port E as input or output. 0 - pin is configured as input. 1 - pin is configured as output.	0
23		Reserved.	0
30:24	PFIO	Port F I/O Each bit controls the setting of the respective pin in port F as input or output. 0 - pin is configured as input. 1 - pin is configured as output.	0
31		Reserved.	0

Table 381: General Purpose Data 1 (GPD1), Offset: 0x100A44

Bits	Field Name	Function	Initial Value
6:0	PCD	Port C Data Each bit holds the data of the respective pin in port C. If the pin is configured as input, this bit is read-only and reflects the signal status at the pin. If the pin is configured as output, this bit is read-write and controls the data driven out.	0
7		Reserved	0
14:8	PDD	Port D Data Each bit holds the data of the respective pin in port D. If the pin is configured as input, this bit is read-only and reflects the signal status at the pin. If the pin is configured as output, this bit is read-write and controls the data driven out.	0

Table 381: General Purpose Data 1 (GPD1), Offset: 0x100A44 (Continued)

Bits	Field Name	Function	Initial Value
15		Reserved	0
22:16	PED	Port E Data Each bit holds the data of the respective pin in port E. If the pin is configured as input, this bit is read-only and reflects the signal status at the pin. If the pin is configured as output, this bit is read-write and controls the data driven out.	0
23		Reserved	0
30:24	PFD	Port F Data Each bit holds the data of the respective pin in port F. If the pin is configured as input, this bit is read-only and reflects the signal status at the pin. If the pin is configured as output, this bit is read-write and controls the data driven out.	0
31		Reserved	0

Table 382: General Purpose Level 1 (GPL1), Offset: 0x100A64

Bits	Field Name	Function	Initial Value
6:0	PCL	Port C Level Each bit defines the polarity of the respective pin in port C. 0 - positive (normal) polarity. 1 - negative polarity. If the pin is configured as input, it is inverted before being processed inside the device.	0
7		Reserved.	0
14:8	PDL	Port D Level Each bit defines the polarity of the respective pin in port D. 0 - positive (normal) polarity. 1 - negative polarity. If the pin is configured as input, it is inverted before being processed inside the device.	0
15		Reserved.	0
22:16	PEL	Port E Level Each bit defines the polarity of the respective pin in port E. 0 - positive (normal) polarity. 1 - negative polarity. If the pin is configured as input, it is inverted before being processed inside the device.	0
23		Reserved.	0

Table 382: General Purpose Level 1 (GPL1), Offset: 0x100A64 (Continued)

Bits	Field Name	Function	Initial Value
30:24	PFL	Port F Level Each bit defines the polarity of the respective pin in port F. 0 - positive (normal) polarity. 1 - negative polarity. If the pin is configured as input, it is inverted before being processed inside the device.	0
31		Reserved.	0

Table 383: General Purpose Configuration 2 (GPC2), Offset: 0x100A08

Bits	Field Name	Function	Initial Value
14:0	MII0C	Port MII0 Configuration Each bit controls the setting of the respective pin in port MII0. 0 - pin is configured as General Purpose I/O. 1 - pin is configured as peripheral function.	0
15		Reserved.	0
30:16	MII1C	Port MII1 Configuration Each bit controls the setting of the respective pin in port MII1. 0 - pin is configured as General Purpose I/O. 1 - pin is configured as peripheral function.	0
31		Reserved.	0

Table 384: General Purpose Input/Output 2 (GPIO2), Offset: 0x100A28

Bits	Field Name	Function	Initial Value
14:0	MII0IO	Port MII0 I/O Each bit controls the setting of the respective pin in port MII0 as input or output. 0 - pin is configured as input. 1 - pin is configured as output.	0
15		Reserved.	0
30:16	MII1IO	Port MII1 I/O Each bit controls the setting of the respective pin in port MII1 as input or output. 0 - pin is configured as input. 1 - pin is configured as output.	0

Table 384: General Purpose Input/Output 2 (GPIO2), Offset: 0x100A28 (Continued)

Bits	Field Name	Function	Initial Value
31		Reserved.	0

Table 385: General Purpose Data 2 (GPD2), Offset: 0x100A48

Bits	Field Name	Function	Initial Value
14:0	MII0D	Port MII0 Data Each bit holds the data of the respective pin in port MII0. If the pin is configured as input, this bit is read-only and reflects the signal status at the pin. If the pin is configured as output, this bit is read-write and controls the data driven out.	0
15		Reserved.	0
30:16	MII1D	Port MII1 Data Each bit holds the data of the respective pin in port MII1. If the pin is configured as input, this bit is read-only and reflects the signal status at the pin. If the pin is configured as output, this bit is read-write and controls the data driven out.	0
31		Reserved.	0

Table 386: General Purpose Level 2 (GPL2), Offset: 0x100A68

Bits	Field Name	Function	Initial Value
14:0	MII0L	Port MII0 Level. Each bit defines the polarity of the respective pin in port MII0. 0 - positive (normal) polarity. 1 - negative polarity. If the pin is configured as input, it is inverted before being processed inside the device.	0
15		Reserved.	0
30:16	MII1L	Port MII1 Level Each bit defines the polarity of the respective pin in port MII1. 0 - positive (normal) polarity. 1 - negative polarity. If the pin is configured as input, it is inverted before being processed inside the device.	0
31		Reserved.	0

20. PHYSICAL SIGNAL ROUTING

20.1 Signal Routing

There are six serial ports on the GT-96100A's Ports A–F and the MII0 port which are used to externally route the MPSCs, FlexTDMs, and the General Purpose Interface signals.

The actual physical routing of the MPSC and TDM signals are defined in the Main Routing Register (MRR), see [Table 387](#).

Table 387: MPSC Routing Register (MRR), Offset: 0X101A00

Bits	Field Name	Function	Initial Value
2:0	MR0	MPSC0 Routing 000 - Port A 001 - FlexTDM0 010 - FlexTDM1 011 - FlexTDM2 100 - FlexTDM3 101 - Reserved 110 - Reserved 111 - Unconnected (Default) When not connected to MPSC0, Port A is connected to the PCI Arbiter or used as GPP. See Section 19. "General Purpose Ports" on page 405 for more details.	111
5:3	MR1	MPSC1 Routing 000 - Port B 001 - FlexTDM0 010 - FlexTDM1 011 - FlexTDM2 100 - FlexTDM3 101 - Reserved 110 - Reserved 111 - Unconnected (Default) When not connected to MPSC1, Port B is connected to the PCI Arbiter or used as GPP. See Section 19. "General Purpose Ports" on page 405 for more details.	111

Table 387: MPSC Routing Register (MRR), Offset: 0X101A00 (Continued)

Bits	Field Name	Function	Initial Value
8:6	MR2	MPSC2 Routing 000 - Port C 001 - FlexTDM0 010 - FlexTDM1 011 - FlexTDM2 100 - FlexTDM3 101 - Reserved 110 - Reserved 111 - Unconnected (Default) When not connected to MPSC2, Port C is connected to TDM0 or used as GPP. See Section 19. "General Purpose Ports" on page 405 for more details.	111
11:9	MR3	MPSC3 Routing 000 - Port D 001 - FlexTDM0 010 - FlexTDM1 011 - FlexTDM2 100 - FlexTDM3 101 - Reserved 110 - Reserved 111 - Unconnected (Default) When not connected to MPSC3, Port D is connected to TDM1 or used as GPP. See Section 19. "General Purpose Ports" on page 405 for more details.	111
14:12	MR4	MPSC4 Routing 000 - Port E 001 - FlexTDM0 010 - FlexTDM1 011 - FlexTDM2 100 - FlexTDM3 101 - Reserved 110 - Reserved 111 - Unconnected (Default) When not connected to MPSC4, Port E is connected to TDM2 or used as GPP. See Section 19. "General Purpose Ports" on page 405 for more details.	111

Table 387: MPSC Routing Register (MRR), Offset: 0X101A00 (Continued)

Bits	Field Name	Function	Initial Value
17:15	MR5	MPSC5 Routing 000 - Port F 001 - FlexTDM0 010 - FlexTDM1 011 - FlexTDM2 100 - FlexTDM3 101 - Reserved 110 - Reserved 111 - Unconnected (Default) When not connected to MPSC5, Port F is connected to TDM3 or used as GPP. See Section 19. “General Purpose Ports” on page 405 for more details.	111
20:18	MR6	MPSC6 Routing 000 - Port MII0 001 - FlexTDM0 010 - FlexTDM1 011 - FlexTDM2 100 - FlexTDM3 101 - Reserved 110 - Reserved 111 - Unconnected (Default) When not connected to MPSC6, Port MII0 is connected to Ethernet 0 or used as GPP. See Section 19. “General Purpose Ports” on page 405 for more details. NOTE: Proper operation of MII interface for Ethernet 0 requires that fields MR6 and MR7 cannot be set to value ‘000’. Any other value is a valid setting. The RMII interface can be used for employing both MPSC6 and Ethernet 0. See Table 289 for more details.	111

Table 387: MPSC Routing Register (MRR), Offset: 0X101A00 (Continued)

Bits	Field Name	Function	Initial Value
23:21	MR7	MPSC7 Routing 000 - Port MII0 001 - FlexTDM0 010 - FlexTDM1 011 - FlexTDM2 100 - FlexTDM3 101 - Reserved 110 - Reserved 111 - Unconnected (Default) When not connected to MPSC7, Port MII0 is connected to Ethernet 0 or used as GPP. See Section 19. "General Purpose Ports" on page 405 for more details. NOTE: Proper operation of MII interface for Ethernet 0 requires that fields MR6 and MR7 cannot be set to value '000'. Any other value is a valid setting. The RMII interface can be used for employing both MPSC7 and Ethernet 0. See Table 289 for more details.	111
31:24		Reserved.	0

NOTE: When a MPSC is connected to a FlexTDM, it's clocks come from the FlexTDM Unit regardless of the content of CRR/CRT.

20.2 Clock Routing

The MPSCs' receive and transmit clocks use the baud rate generators or serial clock input signals. The routing of these signals is defined in the RX Clock Routing Register (RCRR) and the TX Clock Routing Register (TCRR).

Table 388: RX Clock Routing Register (RCRR), Offset: 0X101A10

Bits	Field Name	Function	Initial Value
3:0	CRR0	MPSC0 RX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK0 1001 to 1111 - Reserved	0000
7:4	CRR1	MPSC1 RX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK1 1001 to 1111 - Reserved	0000
11:8	CRR2	MPSC2 RX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK2 1001 to 1111 - Reserved	0000

Table 388: RX Clock Routing Register (RCRR), Offset: 0X101A10 (Continued)

Bits	Field Name	Function	Initial Value
15:12	CRR3	MPSC3 RX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK3 1001 to 1111 - Reserved	0000
19:16	CRR4	MPSC4 RX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK4 1001 to 1111 - Reserved	0000
23:20	CRR5	MPSC5 RX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK5 1001 to 1111 - Reserved	0000

Table 388: RX Clock Routing Register (RCRR), Offset: 0X101A10 (Continued)

Bits	Field Name	Function	Initial Value
27:24	CRR6	MPSC6 RX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK6 1001 to 1111 - Reserved	0000
31:28	CRR7	MPSC7 RX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK7 1001 to 1111 - Reserved	0000

Table 389: TX Clock Routing Register (TCRR), Offset: 0x101A20

Bits	Field Name	Function	Initial Value
3:0	CRT0	MPSC0 TX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK0 1001 - TSCLK0 1010 to 1111 - Reserved	0000

Table 389: TX Clock Routing Register (TCRR), Offset: 0x101A20 (Continued)

Bits	Field Name	Function	Initial Value
7:4	CRT1	MPSC1 TX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK1 1001 - TSCLK1 1010 to 1111 - Reserved	0000
11:8	CRT2	MPSC2 TX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK2 1001 - TSCLK2 1010 to 1111 - Reserved	0000
15:12	CRT3	MPSC3 TX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK3 1001 - TSCLK3 1010 to 1111 - Reserved	0000

Table 389: TX Clock Routing Register (TCRR), Offset: 0x101A20 (Continued)

Bits	Field Name	Function	Initial Value
19:16	CRT4	MPSC4 TX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK4 1001 - TSCLK4 1010 to 1111 - Reserved	0000
23:20	CRT5	MPSC5 TX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK5 1001 - TSCLK5 1010 to 1111 - Reserved	0000
27:24	CRT6	MPSC6 TX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK6 1001 - TSCLK6 1010 to 1111 - Reserved	0000

Table 389: TX Clock Routing Register (TCRR), Offset: 0x101A20 (Continued)

Bits	Field Name	Function	Initial Value
31:28	CRT7	MPSC7 TX Clock Routing 0000 - BRG0 0001 - BRG1 0010 - BRG2 0011 - BRG3 0100 - BRG4 0101 - BRG5 0110 - BRG6 0111 - BRG7 1000 - SCLK7 1001 - TSCLK7 1010 to 1111 - Reserved	0000

21. INTERRUPT CONTROLLER

The GT-96100A provides four interrupt pins that can be used for generating interrupts to the CPU:

- Interrupt0*
- Interrupt1*
- SerInt0*
- SerInt1*

The GT-96100A also integrates a programmable interrupt controller that is capable of routing each of the internal interrupt requests to one (or more) of the interrupt pins. The interrupt controller performs a logical OR of all internal interrupt events and asserts an interrupt to the CPU when at least one of the (unmasked) events is set.

The interrupt pins provided by the GT-96100A can be used in any system which supports multiple interrupt signals. If the CPU is capable of handling multiple interrupt inputs, connect the GT-96100A pins directly to the CPU. If the system requires a PCI directed interrupt, connect one of the pins to the PCI and the remaining pins to the CPU.

The two serial interrupt pins (SerInt0*, SerInt1*) allow separation of communication interrupts from other system events. Only events that originate within the communication unit can be routed to the serial interrupt pins.

21.1 Interrupt Cause Registers

There are two high-level interrupt cause registers, which serve to indicate the occurrence of certain events. One cause register (Main_Cause register) consists of events originating in the GT-96100A's system controller logic. This register is located at offset 0x000C18.

The other cause register (High_Cause register) consists of events originating in the PCI_1 unit and the communication unit. This register is located at offset 0x000C1C.

NOTE: There is another cause register dedicated to communication events. This register (Serial_Cause register) is located at 0x103A00.

When an interrupt event occurs, a bit in one of the cause registers is set. All bits in the cause register(s) are ORed together, and the result is driven on one of the Interrupt* lines. The Interrupt* causes the CPU to read the interrupt cause registers and run a service routine depending on the interrupt being serviced.

The interrupt is acknowledged by the CPU resetting bits in the cause register. The specific bit that is reset depends on the interrupt event being serviced. Reset a bit by writing '0' to this bit and '1' to all other bits.

NOTE: An exception to the above is the CPUInt ([25:22]) and PCIInt ([29:26]) bits, which are intended for generating PCItoCPU and CPUtoPCI interrupts. Set these bits by writing '0' from the interrupt originating side. Clear these bits by writing '0' from the interrupt destination side. For example, if one of the PCI agents needs to assert an interrupt to the CPU, it can write '0' to one of the PCIInt bits in the Main_Cause register. Assuming that this bit is not masked, interrupt0* will be asserted. After servicing this interrupt the CPU should clear the interrupt bit by writing zero to it.

21.1.1 Communication Unit Cause Registers

The GT-96100A includes 16 second level cause registers used to trap events generated within the communication unit.

Each interrupt source in the communication unit is tied to one of these second level cause registers. Each of these registers is tied to specific bits in the High_Cause register and in the Serial_Cause register. These bits function as summary bits, and are set when specific bits in the cause register are asserted (i.e. each summary bit is equivalent to a logical OR of some bits in one of the cause registers). These summary bits are read-only.

When an interrupt event occurs in the communication unit, a bit is set in one of the second level cause registers. This bit, if not masked, asserts one of the summary bits in the High_Cause register and in the Serial_Cause register. Following that, one (or more) of the interrupt lines is asserted.

The CPU recognizes an interrupt that is due to the communication unit when one of the summary bits in the High_Cause register is set or when one of the summary bits in the Serial_Cause register is set. Based on the specific bit set, the CPU reads the second level cause register associated with this bit in order to identify the actual interrupt event that generated the interrupt. In order to acknowledge the interrupt, the CPU must write zero to this bit in the cause register.

NOTE: The CPU cannot reset the summary bits directly, because these bits are read-only. A summary bit is automatically reset when the CPU resets the bits in the related second level cause register.

21.2 Interrupt Mask Registers

The GT-96100A provides interrupt mask registers for each of the internal cause registers. These mask registers allow masking of certain events, so that only specific events (as selected by the user) actually cause assertion of one of the interrupts.

There are two mask registers associated with Interrupt0* and two mask registers associated with Interrupt1*. These mask registers are used for enabling events that cause the assertion of either of these interrupts. For Interrupt0*, these mask registers are located at 0x000C1C and at 0x000C9C. For Interrupt1*, the mask registers are located at 0x000C24 and at 0x000C38. Programming '0' in a mask register bit disables the associated event from asserting interrupt. Programming '1' allows the interrupt event to cause interrupt signal assertion.

In addition, there is one mask register associated with each of the serial interrupt pins. For SerInt0* the mask register is located at 0x103A80 and for SerInt1* the mask register is located at 0x103A88.

21.3 Interrupt Summaries

The GT-96100A provides the following three interrupt summary bits in the main cause register:

- IntSum (bit [0] in the Main_Cause register) is the logical OR of all interrupt bits in both the Main_Cause and High_Cause registers. This OR is not affected by the state of the mask bits and can be used for event polling.
- Int0*Sum (bit [30] in the Main_Cause register) is the logical OR of all interrupt bits in both the Main_Cause and High_Cause registers masked by Interrupt0* mask registers. It serves as an indication that at least one of the (unmasked) Interrupt0* events is set.
- Int1*Sum (bit [31] in the Main_Cause register) is the logical OR of all interrupt bits in both the Main_Cause and High_Cause registers masked by Interrupt1* mask registers. It serves as an indication that at least one of the (unmasked) Interrupt1* events is set.

21.4 Interrupt Select Registers

There are two interrupt select registers that can be used to optimize interrupt service routines. One select register is associated with Interrupt0* (offset 0x000C70) and another register is associated with interrupt1* (offset 0x000C74).

These select registers optimize service routines in the following manner:

- Instead of checking BOTH the Main_Cause register and the High_Cause register, when interrupted, the CPU has the option to read the appropriate select register.
- The select register will reflect the Cause register bits of either the Main_Cause or the High_Cause registers, depending on which register has active unmasked interrupt bits.
- Bit [30] of the select register indicates which of the cause registers (Main or High) is selected and bits [29:0] reflect the state of the interrupt bits of the selected cause register. For example, if bit [5] of the High_Cause register is set (and is unmasked), and no (unmasked) bit in the Main_Cause register is set, then bit [5] of the select register is set as well. In addition, bit [30] of the select register is set, indicating that the High_Cause register is currently being selected.

In case both the Main_Cause and the High_Cause registers have interrupt bits set, the select register reflects the state of the Main_Cause register (and bit [30] is therefore reset). However, in order to indicate that both cause registers are active, bit [31] of the select register is also set, in this case.

21.5 Interrupt Registers Tables

Table 390: Interrupt Registers Map

Register Name	Offset	Page Number
Interrupt Main Cause register	0x000C18	page 428
Interrupt0* Main Mask register	0x000C1C	page 432
Interrupt1* Main Mask register	0x000C24	page 434
Interrupt High Cause register	0x000C98	page 430
Interrupt0* High Mask register	0x000C9C	page 433
Interrupt1* High Mask register	0x000CA4	page 435
Interrupt0* Select register	0x000C70	page 431
Interrupt1* Select register	0x000C74	page 432
Serial Cause register	0x103A00	page 436
SerInt0* Mask register	0x103A80	page 438
SerInt1* Mask register	0x103A88	page 439
Ethernet0 Cause register	0x084850	page 440
Ethernet0 Mask register	0x084858	page 440
Ethernet1 Cause register	0x088850	page 441
Ethernet1 Mask register	0x088858	page 441
SDMA Cause register	0x103A10	page 441
SDMA Mask register	0x103A90	page 441
MPSC0 Cause register	0x103A20	page 444
MPSC0 Mask register	0x103AA0	page 444
MPSC1 Cause register	0x103A24	page 445
MPSC1 Mask register	0x103AA4	page 445
MPSC2 Cause register	0x103A28	page 445
MPSC2 Mask register	0x103AA8	page 445
MPSC3 Cause register	0x103A2C	page 446
MPSC3 Mask register	0x103AAC	page 446
MPSC4 Cause register	0x103A30	page 446
MPSC4 Mask register	0x103AB0	page 446
MPSC5 Cause register	0x103A34	page 446
MPSC5 Mask register	0x103AB4	page 446

Table 390: Interrupt Registers Map (Continued)

Register Name	Offset	Page Number
MPSC6 Cause register	0x103A38	page 446
MPSC6 Mask register	0x103AB8	page 446
MPSC7 Cause register	0x103A3C	page 446
MPSC7 Mask register	0x103ABC	page 446
FlexTDM Cause register	0x103A40	page 447
FlexTDM Mask register	0x103AC0	page 447
BRG Cause register	0x103A48	page 448
BRG Mask register	0x103AC8	page 448
GPP0 Cause register	0x103A50	page 448
GPP0 Mask register	0x103AD0	page 448
GPP1 Cause register	0x103A54	page 449
GPP1 Mask register	0x103AD4	page 449
GPP2 Cause register	0x103A58	page 449
GPP2 Mask register	0x103AD8	

Table 391: Interrupt Main Cause Register, Offset: 0x000C18

Bits	Field Name	Function	Initial Value
0	IntSum	Interrupt Summary Logical OR of all the interrupt bits, regardless of the Mask registers' values. This bit is read-only.	0
1	MemOut	Asserts when the CPU or PCI accesses an address out of range in the memory decoding or a burst access to 8- 16-bit devices.	0
2	DMAOut	Asserts when the DMA or Communication unit accesses an address out of range.	0
3	CPUOut	Asserts when the CPU accesses an address out of range.	0
4	DMA0Comp	Asserts at completion of DMA Channel 0 transfer.	0
5	DMA1Comp	Asserts at completion of DMA Channel 1 transfer.	0
6	DMA2Comp	Asserts at completion of DMA Channel 2 transfer.	0
7	DMA3Comp	Asserts at completion of DMA Channel 3 transfer.	0
8	T0Exp	Asserts when Timer 0 expires.	0

Table 391: Interrupt Main Cause Register, Offset: 0x000C18 (Continued)

Bits	Field Name	Function	Initial Value
9	T1Exp	Asserts when Timer 1 expires.	0
10	T2Exp	Asserts when Timer 2 expires.	0
11	T3Exp	Asserts when Timer 3 expires.	0
12	MasRdErr0	Asserts when the GT-96100A detects a parity error during a PCI_0 master read operation.	0
13	SlvWrErr0	Asserts when the GT-96100A detects a parity error during a PCI_0 slave write operation.	0
14	MasWrErr0	Asserts when the GT-96100A detects a parity error during a PCI_0 master write operation.	0
15	SlvRdErr0	Asserts when the GT-96100A detects a parity error during a PCI_0 slave read operation.	0
16	AddrErr0	Asserts when the GT-96100A detects a parity error on the PCI_0 address lines.	0
17	MemErr	Asserts when a memory parity error is detected.	0
18	MasAbort0	Asserts upon PCI_0 master abort.	0
19	TarAbort0	Asserts upon PCI_0 target abort.	0
20	RetryCtr0	Asserts when the PCI_0 retry counter expires.	0
21	PMCInt0	If Power Management is enabled this bit functions as PMC0 interrupt, otherwise it functions as one of the CPUInt bits. <ul style="list-style-type: none"> • PMCInt: Asserts when power state bits in PMCSR0 register are updated from PCI. • CPUInt: Set by the CPU by writing '0' to generate an interrupt on the PCI bus. Cleared when the PCI writes '0'. 	0
25:22	CPUInt	These bits are set by the CPU by writing '0' to generate an interrupt on the PCI bus. They are cleared when the PCI writes '0'. This requires that Interrupt1* is used as a PCI interrupt signal.	0
29:26	PCIInt	These bits are set by the PCI by writing '0' to generate an interrupt on the CPU. They are cleared when the CPU writes '0'.	0
30	Int0*Sum	Interrupt Summary Logical OR of all interrupt bits in the main and high Cause registers masked by Interrupt0* mask registers. This bit is read-only.	0
31	Int1*Sum	Interrupt Summary Logical OR of all interrupt bits in the main and high Cause registers masked by Interrupt1* mask registers. This bit is read-only.	0

Table 392: Interrupt High Cause Register, Offset: 0x000C98

Bits	Field Name	Function	Initial Value
0	Ether0Sum	Ethernet port 0 Interrupt Summary Logical OR of all unmasked interrupt bits in the Ethernet0_Cause register. This bit is read-only.	0
1	Ether1Sum	Ethernet Port 1 Interrupt Summary Logical OR of all unmasked interrupt bits in the Ethernet1_Cause register. This bit is read-only.	0
3:2		Reserved.	0
4	SdmaSum	SDMA Interrupt Summary Logical OR of all unmasked interrupt bits in the SDMA_Cause register. This bit is read-only.	0
5	MpscSum	MPSC Interrupt Summary Logical OR of all unmasked interrupt bits in all the MPSC_Cause registers. This bit is read-only.	0
6	FtdmSum	FlexTDM Interrupt Summary Logical OR of all unmasked interrupt bits in the FTDM_Cause register. This bit is read-only.	0
7	BrgSum	Baud Rate Generators Interrupt Summary Logical OR of all unmasked interrupt bits in the BRG_Cause register. This bit is read-only.	0
8	GPP0Sum	GPP0 Interrupt Summary Logical OR of all unmasked interrupt bits in the GPP0_Cause register. This bit is read-only.	0
9	GPP1Sum	GPP1 Interrupt Summary Logical OR of all unmasked interrupt bits in the GPP1_Cause register. This bit is read-only.	0
10	GPP2Sum	GPP2 Interrupt Summary Logical OR of all unmasked interrupt bits in the GPP2_Cause register. This bit is read-only.	0
11		Reserved.	0
12	MasRdErr1	Asserts when the GT-96100A detects a parity error during a PCI_1 master read operation.	0
13	SlvWrErr1	Asserts when the GT-96100A detects a parity error during a PCI_1 slave write operation.	0

Table 392: Interrupt High Cause Register, Offset: 0x000C98 (Continued)

Bits	Field Name	Function	Initial Value
14	MasWrErr1	Asserts when the GT-96100A detects a parity error during a PCI_1 master write operation.	0
15	SlvRdErr1	Asserts when the GT-96100A detects a parity error during a PCI_1 slave read operation.	0
16	AddrErr1	Asserts when the GT-96100A detects a parity error on the PCI_1 address lines.	0
17		Reserved.	0
18	MasAbort1	Asserts upon PCI_1 master abort.	0
19	TarAbort1	Asserts upon PCI_1 target abort.	0
20	RetryCtr1	Asserts when the PCI_1 retry counter expires.	0
21	PMCInt1	If Power Management is enabled, this bit functions as PMC1 interrupt, otherwise it is reserved. PMC1 asserts when power state bits in PMCSR1 register are updated from PCI.	0
23:22		Reserved.	0
24	PciArb0	PCI_0 Arbiter Interrupt This bit is set when a “broken master” condition is detected by PCI_0 arbiter.	0
25	PciArb1	PCI_1 Arbiter Interrupt This bit is set when a “broken master” condition is detected by PCI_1 arbiter.	0
31:26		Reserved.	0

Table 393: Interrupt0* Select Register, Offset: 0x000C70

Bits	Field Name	Function	Initial Value
29:0	AliasedBits	Aliased to bits [29:0] of the selected Cause register.	0
30	SelectCause	Selected Cause Register 0 - Main Cause Register 1 - High Cause Register	0
31	LowAndHigh	Interrupt is set in both Main and High Cause Registers.	0

Table 394: Interrupt1* Select Register, Offset: 0x000C74

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for Interrupt0* Select register.	0

Table 395: Interrupt0* Main Mask Register, Offset: 0x000C1C

Bits	Field Name	Function	Initial Value
0		Reserved.	0
1	MemOutMask	Masks MemOut Interrupt to Interrupt0*	0
2	DMAOutMask	Masks DMAOut Interrupt to Interrupt0*	0
3	CPUOutMask	Masks CPUOut Interrupt to Interrupt0*	0
4	DMA0CompMask	Masks DMA0Comp Interrupt to Interrupt0*	0
5	DMA1CompMask	Masks DMA1Comp Interrupt to Interrupt0*	0
6	DMA2CompMask	Masks DMA2Comp Interrupt to Interrupt0*	0
7	DMA3CompMask	Masks DMA3Comp Interrupt to Interrupt0*	0
8	T0ExpMask	Masks T0Exp Interrupt to Interrupt0*	0
9	T1ExpMask	Masks T1Exp Interrupt to Interrupt0*	0
10	T2ExpMask	Masks T2Exp Interrupt to Interrupt0*	0
11	T3ExpMask	Masks T3Exp Interrupt to Interrupt0*	0
12	MasRdErr0Mask	Masks MasRdErr0 Interrupt to Interrupt0*	0
13	SlvWrErr0Mask	Masks SlvWrErr0 Interrupt to Interrupt0*	0
14	MasWrErr0Mask	Masks MasWrErr0 Interrupt to Interrupt0*	0
15	SlvRdErr0Mask	Masks SlvRdErr0 Interrupt to Interrupt0*	0
16	AddrErr0Mask	Masks AddrErr0 Interrupt to Interrupt0*	0
17	MemErrMask	Masks MemErr Interrupt to Interrupt0*	0
18	MasAbort0Mask	Masks MasAbort0 Interrupt to Interrupt0*	0
19	TarAbort0Mask	Masks TarAbort0 Interrupt to Interrupt0*	0
20	RetryCtr0Mask	Masks RetryCtr0 Interrupt to Interrupt0*	0
21	PMC0Mask	If Power Management is enabled, masks PMC0 interrupt to Interrupt0*, otherwise it is reserved (Read Only 0).	0
25:22		Reserved.	0
29:26	PCIIntMask	Masks PCIInt Interrupt to Interrupt0*	0

Table 395: Interrupt0* Main Mask Register, Offset: 0x000C1C (Continued)

Bits	Field Name	Function	Initial Value
31:30		Reserved.	0

Table 396: Interrupt0* High Mask Register, Offset: 0x000C9C

Bits	Field Name	Function	Initial Value
0	Ether0SumMask	Masks Ether0Sum to Interrupt0*	0
1	Ether1SumMask	Masks Ether1Sum to Interrupt0*	0
3:2		Reserved.	0
4	SdmaSumMask	Masks SdmaSum to Interrupt0*	0
5	MpscSumMask	Masks MpscSum to Interrupt0*	0
6	FtdmSumMask	Masks FtdmSum to Interrupt0*	0
7	BrgSumMask	Masks BrgSum to Interrupt0*	0
8	GPP0SumMask	Masks GPP0Sum to Interrupt0*	0
9	GPP1SumMask	Masks GPP1Sum to Interrupt0*	0
10	GPP2SumMask	Masks GPP2Sum to Interrupt0*	0
11		Reserved.	0
12	MasRdErr1Mask	Masks MasRdErr1 to Interrupt0*	0
13	SlvWrErr1Mask	Masks SlvWrErr1 to Interrupt0*	0
14	MasWrErr1Mask	Masks MasWrErr1 to Interrupt0*	0
15	SlvRdErr1Mask	Masks SlvRdErr1 to Interrupt0*	0
16	AddrErr1Mask	Masks AddrErr1 to Interrupt0*	0
17		Reserved.	0
18	MasAbort1Mask	Masks MasAbort1 to Interrupt0*	0
19	TarAbort1Mask	Masks TarAbort1 to Interrupt0*	0
20	RetryCtr1Mask	Masks RetryCtr1 to Interrupt0*	0
21	PMC1Mask	If Power Management is enabled, masks PMC1 interrupt to Interrupt0*, otherwise it is reserved (Read Only 0).	0
23:22		Reserved.	
24	PciArb0Mask	Masks PciArb0 to Interrupt0*	0
25	PciArb1Mask	Masks PciArb1 to Interrupt0*	0

Table 396: Interrupt0* High Mask Register, Offset: 0x000C9C (Continued)

Bits	Field Name	Function	Initial Value
31:26		Reserved.	0

Table 397: Interrupt1* Main Mask Register, Offset: 0x000C24

Bits	Field Name	Function	Initial Value
0		Reserved	0
1	MemOutMask	Masks MemOut Interrupt to Interrupt1*	0
2	DMAOutMask	Masks DMAOut Interrupt to Interrupt1*	0
3	CPUOutMask	Masks CPUOut Interrupt to Interrupt1*	0
4	DMA0CompMask	Masks DMA0Comp Interrupt to Interrupt1*	0
5	DMA1CompMask	Masks DMA1Comp Interrupt to Interrupt1*	0
6	DMA2CompMask	Masks DMA2Comp Interrupt to Interrupt1*	0
7	DMA3CompMask	Masks DMA3Comp Interrupt to Interrupt1*	0
8	T0ExpMask	Masks T0Exp Interrupt to Interrupt1*	0
9	T1ExpMask	Masks T1Exp Interrupt to Interrupt1*	0
10	T2ExpMask	Masks T2Exp Interrupt to Interrupt1*	0
11	T3ExpMask	Masks T3Exp Interrupt to Interrupt1*	0
12	MasRdErr0Mask	Masks MasRdErr0 Interrupt to Interrupt1*	0
13	SivWrErr0Mask	Masks SivWrErr0 Interrupt to Interrupt1*	0
14	MasWrErr0Mask	Masks MasWrErr0 Interrupt to Interrupt1*	0
15	SivRdErr0Mask	Masks SivRdErr0 Interrupt to Interrupt1*	0
16	AddrErr0Mask	Masks AddrErr0 Interrupt to Interrupt1*	0
17	MemErrMask	Masks MemErr Interrupt to Interrupt1*	0
18	MasAbort0Mask	Masks MasAbort0 Interrupt to Interrupt1*	0
19	TarAbort0Mask	Masks TarAbort0 Interrupt to Interrupt1*	0
20	RetryCtr0Mask	Masks RetryCtr0 Interrupt to Interrupt1*	0
21	PMC0Mask	If Power Management is enabled, masks PMC0 interrupt to Interrupt1*, otherwise it masks CPUInt Interrupt to Interrupt1*	0
25:22	CPUInt	Masks CPUInt Interrupt to Interrupt1*	0
31:26		Reserved	0

Table 398: Interrupt1* High Mask Register, Offset: 0x000CA4

Bits	Field Name	Function	Initial Value
0	Ether0SumMask	Masks Ether0Sum to Interrupt1*	0
1	Ether1SumMask	Masks Ether1Sum to Interrupt1*	0
3:2		Reserved.	0
4	SdmaSumMask	Masks SdmaSum to Interrupt1*	0
5	MpscSumMask	Masks MpscSum to Interrupt1*	0
6	FtdmSumMask	Masks FtdmSum to Interrupt1*	0
7	BrgSumMask	Masks BrgSum to Interrupt1*	0
8	GPP0SumMask	Masks GPP0Sum to Interrupt1*	0
9	GPP1SumMask	Masks GPP1Sum to Interrupt1*	0
10	GPP2SumMask	Masks GPP2Sum to Interrupt1*	0
11		Reserved.	0
12	MasRdErr1Mask	Masks MasRdErr1 to Interrupt1*	0
13	SivWrErr1Mask	Masks SivWrErr1 to Interrupt1*	0
14	MasWrErr1Mask	Masks MasWrErr1 to Interrupt1*	0
15	SivRdErr1Mask	Masks SivRdErr1 to Interrupt1*	0
16	AddrErr1Mask	Masks AddrErr1 to Interrupt1*	0
17		Reserved.	0
18	MasAbort1Mask	Masks MasAbort1 to Interrupt1*	0
19	TarAbort1Mask	Masks TarAbort1 to Interrupt1*	0
20	RetryCtr1Mask	Masks RetryCtr1 to Interrupt1*	0
21	PMC1Mask	If Power Management is enabled, masks PMC1 interrupt to Interrupt1*. Otherwise, it is reserved (Read Only 0).	0
23:22		Reserved.	
24	PciArb0Mask	Masks PciArb0 to Interrupt1*	0
25	PciArb1Mask	Masks PciArb1 to Interrupt1*	0
31:26		Reserved.	0

Table 399: Serial Cause Register, Offset: 0x103A00

Bits	Field Name	Function	Initial Value
0	Ether0Sum	Ethernet Port 0 Interrupt Summary Logical OR of all unmasked interrupt bits in the Ethernet0_Cause register. This bit is read-only.	0
1	Ether1Sum	Ethernet Port 1 Interrupt Summary Logical OR of all unmasked interrupt bits in the Ethernet1_Cause register. This bit is read-only.	0
3:2		Reserved.	0
4	SdmaSum	SDMA Interrupt Summary Logical OR of all unmasked bits in the SDMA_Cause register. This bit is read-only.	0
5	MpscSum	MPSC Interrupt Summary Logical OR of all unmasked interrupt bits in the MPSC_Cause registers. This bit is read-only.	0
6	FtdmSum	FlexTDM Interrupt Summary Logical OR of all unmasked interrupt bits in the FTDM_Cause register. This bit is read-only.	0
7	BrgSum	Baud Rate Generators Interrupt Summary Logical OR of all unmasked interrupt bits in the BRG_Cause register. This bit is read-only.	0
8	Sdma0Sum	SDMA0 Interrupt Summary Logical OR of unmasked bits [3:0] in the SDMA_Cause register. This bit is read-only.	0
9	Mpsc0Sum	MPSC0 Interrupt Summary Logical OR of all unmasked interrupt bits in the MPSC0_Cause register. This bit is read-only.	0
10	Sdma1Sum	SDMA1 Interrupt Summary Logical OR of unmasked bits [7:4] in the SDMA_Cause register. This bit is read-only.	0

Table 399: Serial Cause Register, Offset: 0x103A00 (Continued)

Bits	Field Name	Function	Initial Value
11	Mpsc1Sum	MPSC1 Interrupt Summary Logical OR of all unmasked interrupt bits in the MPSC1_Cause register. This bit is read-only.	0
12	Sdma2Sum	SDMA2 Interrupt Summary Logical OR of unmasked bits [11:8] in the SDMA_Cause register. This bit is read-only.	0
13	Mpsc2Sum	MPSC2 Interrupt Summary Logical OR of all unmasked interrupt bits in the MPSC2_Cause register. This bit is read-only.	0
14	Sdma3Sum	SDMA2 Interrupt Summary Logical OR of unmasked bits [15:12] in the SDMA_Cause register. This bit is read-only.	0
15	Mpsc3Sum	MPSC3 Interrupt Summary Logical OR of all unmasked interrupt bits in the MPSC3_Cause register. This bit is read-only.	0
16	Sdma4Sum	SDMA4 Interrupt Summary Logical OR of unmasked bits [19:16] in the SDMA_Cause register. This bit is read-only.	0
17	Mpsc4Sum	MPSC4 Interrupt Summary Logical OR of all unmasked interrupt bits in the MPSC4_Cause register. This bit is read-only.	0
18	Sdma5Sum	SDMA5 Interrupt Summary Logical OR of unmasked bits [23:20] in the SDMA_Cause register. This bit is read-only.	0
19	Mpsc5Sum	MPSC5 Interrupt Summary Logical OR of all unmasked interrupt bits in the MPSC5_Cause register. This bit is read-only.	0
20	Sdma6Sum	SDMA6 Interrupt Summary Logical OR of unmasked bits [27:24] in the SDMA_Cause register. This bit is read-only.	0

Table 399: Serial Cause Register, Offset: 0x103A00 (Continued)

Bits	Field Name	Function	Initial Value
21	Mpsc6Sum	MPSC6 Interrupt Summary Logical OR of all unmasked interrupt bits in the MPSC6_Cause register. This bit is read-only.	0
22	Sdma7Sum	SDMA7 Interrupt Summary Logical OR of unmasked bits [31:28] in the SDMA_Cause register. This bit is read-only.	0
23	Mpsc7Sum	MPSC7 Interrupt Summary Logical OR of all unmasked interrupt bits in the MPSC7_Cause register. This bit is read-only.	0
24	GPP0Sum	GPP0 Interrupt Summary Logical OR of all unmasked interrupt bits in the GPP0_Cause register. This bit is read-only.	0
25	GPP1Sum	GPP1 Interrupt Summary Logical OR of all unmasked interrupt bits in the GPP1_Cause register. This bit is read-only.	0
26	GPP2Sum	GPP2 Interrupt Summary Logical OR of all unmasked interrupt bits in the GPP2_Cause register. This bit is read-only.	0
31:27		Reserved.	0

Table 400: SerInt0* Mask Register, Offset: 0x103A80

Bits	Field Name	Function	Initial Value
0	Ether0SumMask	Mask Ether0Sum interrupt to SerInt0*	0
1	Ether1SumMask	Mask Ether1Sum interrupt to SerInt0*	0
3:2		Reserved.	0
4	SdmaSumMask	Mask SdmaSum interrupt to SerInt0*	0
5	MpscSumMask	Mask MpscSum interrupt to SerInt0*	0
6	FtdmSumMask	Mask FtdmSum interrupt to SerInt0*	0

Table 400: SerInt0* Mask Register, Offset: 0x103A80 (Continued)

Bits	Field Name	Function	Initial Value
7	BrgSumMask	Mask BrgSum interrupt to SerInt0*	0
8	Sdma0SumMask	Mask Sdma0Sum interrupt to SerInt0*	0
9	Mpsc0SumMask	Mask Mpsc0Sum interrupt to SerInt0*	0
10	Sdma1SumMask	Mask Sdma1Sum interrupt to SerInt0*	0
11	Mpsc1SumMask	Mask Mpsc1Sum interrupt to SerInt0*	0
12	Sdma2SumMask	Mask Sdma2Sum interrupt to SerInt0*	0
13	Mpsc2SumMask	Mask Mpsc2Sum interrupt to SerInt0*	0
14	Sdma3SumMask	Mask Sdma3Sum interrupt to SerInt0*	0
15	Mpsc3SumMask	Mask Mpsc3Sum interrupt to SerInt0*	0
16	Sdma4SumMask	Mask Sdma4Sum interrupt to SerInt0*	0
17	Mpsc4SumMask	Mask Mpsc4Sum interrupt to SerInt0*	0
18	Sdma5SumMask	Mask Sdma5Sum interrupt to SerInt0*	0
19	Mpsc5SumMask	Mask Mpsc5Sum interrupt to SerInt0*	0
20	Sdma6SumMask	Mask Sdma6Sum interrupt to SerInt0*	0
21	Mpsc6SumMask	Mask Mpsc6Sum interrupt to SerInt0*	0
22	Sdma7SumMask	Mask Sdma7Sum interrupt to SerInt0*	0
23	Mpsc7SumMask	Mask Mpsc7Sum interrupt to SerInt0*	0
24	GPP0SumMask	Mask GPP0Sum interrupt to SerInt0*	0
25	GPP1SumMask	Mask GPP1Sum interrupt to SerInt0*	0
26	GPP2SumMask	Mask GPP2Sum interrupt to SerInt0*	0
31:27		Reserved.	0

Table 401: SerInt1* Mask Register, Offset: 0x103A88

Bits	Field Name	Function	Initial Value
0	Ether0SumMask	Mask Ether0Sum interrupt to SerInt1*	0
1	Ether1SumMask	Mask Ether1Sum interrupt to SerInt1*	0
3:2		Reserved.	0
4	SdmaSumMask	Mask SdmaSum interrupt to SerInt1*	0
5	MpscSumMask	Mask MpscSum interrupt to SerInt1*	0

Table 401: SerInt1* Mask Register, Offset: 0x103A88 (Continued)

Bits	Field Name	Function	Initial Value
6	FtdmSumMask	Mask FtdmSum interrupt to SerInt1*	0
7	BrgSumMask	Mask BrgSum interrupt to SerInt1*	0
8	Sdma0SumMask	Mask Sdma0Sum interrupt to SerInt1*	0
9	Mpsc0SumMask	Mask Mpsc0Sum interrupt to SerInt1*	0
10	Sdma1SumMask	Mask Sdma1Sum interrupt to SerInt1*	0
11	Mpsc1SumMask	Mask Mpsc1Sum interrupt to SerInt1*	0
12	Sdma2SumMask	Mask Sdma2Sum interrupt to SerInt1*	0
13	Mpsc2SumMask	Mask Mpsc2Sum interrupt to SerInt1*	0
14	Sdma3SumMask	Mask Sdma3Sum interrupt to SerInt1*	0
15	Mpsc3SumMask	Mask Mpsc3Sum interrupt to SerInt1*	0
16	Sdma4SumMask	Mask Sdma4Sum interrupt to SerInt1*	0
17	Mpsc4SumMask	Mask Mpsc4Sum interrupt to SerInt1*	0
18	Sdma5SumMask	Mask Sdma5Sum interrupt to SerInt1*	0
19	Mpsc5SumMask	Mask Mpsc5Sum interrupt to SerInt1*	0
20	Sdma6SumMask	Mask Sdma6Sum interrupt to SerInt1*	0
21	Mpsc6SumMask	Mask Mpsc6Sum interrupt to SerInt1*	0
22	Sdma7SumMask	Mask Sdma7Sum interrupt to SerInt1*	0
23	Mpsc7SumMask	Mask Mpsc7Sum interrupt to SerInt1*	0
24	GPP0SumMask	Mask GPP0Sum interrupt to SerInt1*	0
25	GPP1SumMask	Mask GPP1Sum interrupt to SerInt1*	0
26	GPP2SumMask	Mask GPP2Sum interrupt to SerInt1*	0
31:27		Reserved.	0

Table 402: Ethernet0 Cause Register, Offset: 0x084850 and Ethernet0 Mask Register, Offset: 0x084858

Bits	Field Name	Function	Initial Value
0	RxBuffer	Rx Buffer Return	0
1		Reserved.	0
2	TxBufferHigh	Tx Buffer for High priority queue.	0

Table 402: Ethernet0 Cause Register, Offset: 0x084850 and Ethernet0 Mask Register, Offset: 0x084858 (Continued)

Bits	Field Name	Function	Initial Value
3	TxBufferLow	Tx Buffer for low priority queue.	0
5:4		Reserved.	0
6	TxEndHigh	Tx End for high priority queue.	0
7	TxEndLow	Tx End for low priority queue.	0
8	RxError	Rx Resource Error. Indicates a Rx resource error event.	0
9		Reserved.	0
10	TxErrorHigh	Tx Resource Error for high priority queue.	0
11	TxErrorLow	Tx Resource Error for low priority queue.	0
12	RxOVR	Rx Overrun	0
13	TxUdr	Tx Underrun	0
27:14		Reserved.	0
28	MIIPhySTC	MII PHY Status Change.	0
29	SMIdone	SMI Command Done	0
30		Reserved.	0
31	EtherIntSum	Ethernet Interrupt Summary This bit is a logical OR of the (unmasked) bits [30:4] in the Interrupt Cause register.	0

Table 403: Ethernet1 Cause Register, Offset: 0x088850 and Ethernet1 Mask Register, Offset: 0x088858

Bits	Field Name	Function	Initial Value
31:0	various	Same as for Ethernet0 Cause register.	0

Table 404: SDMA Cause Register, Offset: 0x103A10 and SDMA Mask Register, Offset: 0x103A90

Bits	Field Name	Function	Initial Value
0	Sdma0RxBuf	SDMA Channel 0 Rx Buffer Return Indicates that SDMA0 Rx closed a descriptor and returned the associated buffer to CPU ownership.	0

**Table 404: SDMA Cause Register, Offset: 0x103A10 and
SDMA Mask Register, Offset: 0x103A90 (Continued)**

Bits	Field Name	Function	Initial Value
1	Sdma0RxErr	SDMA Channel 0 Rx Error Indicates that a Rx resource error occurred.	0
2	Sdma0TxBuf	SDMA Channel 0 Tx Buffer Return Indicates that SDMA0 Tx closed a descriptor and returned the associated buffer to CPU ownership.	0
3	Sdma0TxEnd	SDMA Channel 0 Tx End Indicates that a Tx resource error occurred or that the Tx DMA moved to IDLE after a stop command. Also set when Tx retransmit limit is reached in HDLC mode.	0
4	Sdma1RxBuf	SDMA Channel 1 Rx Buffer Return Indicates that SDMA1 Rx closed a descriptor and returned the associated buffer to CPU ownership.	0
5	Sdma1RxErr	SDMA Channel 1 Rx Error Indicates that a Rx resource error occurred.	0
6	Sdma1TxBuf	SDMA Channel 1 Tx Buffer Return Indicates that SDMA1 Tx closed a descriptor and returned the associated buffer to CPU ownership.	0
7	Sdma1TxEnd	SDMA Channel 1 Tx End Indicates that a Tx resource error occurred or that the Tx DMA moved to IDLE after a stop command. Also set when Tx retransmit limit is reached in HDLC mode.	0
8	Sdma2RxBuf	SDMA Channel 2 Rx Buffer Return Indicates that SDMA2 Rx closed a descriptor and returned the associated buffer to CPU ownership.	0
9	Sdma2RxErr	SDMA Channel 2 Rx Error Indicates that a Rx resource error occurred.	0
10	Sdma2TxBuf	SDMA Channel 2 Tx Buffer Return Indicates that SDMA2 Tx closed a descriptor and returned the associated buffer to CPU ownership.	0
11	Sdma2TxEnd	SDMA Channel 2 Tx End Indicates that a Tx resource error occurred or that the Tx DMA moved to IDLE after a stop command. Also, set when Tx retransmit limit is reached in HDLC mode.	0
12	Sdma3RxBuf	SDMA Channel 3 Rx Buffer Return Indicates that SDMA3 Rx closed a descriptor and returned the associated buffer to CPU ownership.	0

**Table 404: SDMA Cause Register, Offset: 0x103A10 and
SDMA Mask Register, Offset: 0x103A90 (Continued)**

Bits	Field Name	Function	Initial Value
13	Sdma3RxErr	SDMA Channel 3 Rx Error Indicates that a Rx resource error occurred.	0
14	Sdma3TxBuf	SDMA Channel 3 Tx Buffer Return Indicates that SDMA3 Tx closed a descriptor and returned the associated buffer to CPU ownership.	0
15	Sdma3TxEnd	SDMA Channel 3 Tx End Indicates that a Tx resource Error occurred or that the Tx DMA moved to IDLE after a stop command. Also, set when Tx retransmit limit is reached in HDLC mode.	0
16	Sdma4RxBuf	SDMA Channel 4 Rx Buffer Return Indicates that SDMA4 Rx closed a descriptor and returned the associated buffer to CPU ownership.	0
17	Sdma4RxErr	SDMA Channel 4 Rx Error Indicates that a Rx resource error occurred.	0
18	Sdma4TxBuf	SDMA Channel 4 Tx Buffer Return Indicates that SDMA4 Tx closed a descriptor and returned the associated buffer to CPU ownership.	0
19	Sdma4TxEnd	SDMA Channel 4 Tx End Indicates that a Tx resource error occurred or that the Tx DMA moved to IDLE after a stop command. Also, set when Tx retransmit limit is reached in HDLC mode.	0
20	Sdma5RxBuf	SDMA Channel 5 Rx Buffer Return Indicates that SDMA5 Rx closed a descriptor and returned the associated buffer to CPU ownership.	0
21	Sdma5RxErr	SDMA Channel 5 Rx Error Indicates that a Rx resource error occurred.	0
22	Sdma5TxBuf	SDMA Channel 5 Tx Buffer Return Indicates that SDMA5 Tx closed a descriptor and returned the associated buffer to CPU ownership.	0
23	Sdma5TxEnd	SDMA Channel 5 Tx End Indicates that a Tx resource error occurred or that the Tx DMA moved to IDLE after a stop command. Also, set when Tx retransmit limit is reached in HDLC mode.	0
24	Sdma6RxBuf	SDMA Channel 6 Rx Buffer Return Indicates that SDMA6 Rx closed a descriptor and returned the associated buffer to CPU ownership.	0

**Table 404: SDMA Cause Register, Offset: 0x103A10 and
SDMA Mask Register, Offset: 0x103A90 (Continued)**

Bits	Field Name	Function	Initial Value
25	Sdma6RxErr	SDMA Channel 6 Rx Error Indicates that a Rx resource error occurred.	0
26	Sdma6TxBuf	SDMA Channel 6 Tx Buffer Return Indicates that SDMA6 Tx closed a descriptor and returned the associated buffer to CPU ownership.	0
27	Sdma6TxEnd	SDMA Channel 6 Tx End Indicates that a Tx resource error occurred or that the Tx DMA moved to IDLE after a stop command. Also, set when Tx retransmit limit is reached in HDLC mode.	0
28	Sdma7RxBuf	SDMA Channel 7 Rx Buffer Return Indicates that SDMA7 Rx closed a descriptor and returned the associated buffer to CPU ownership.	0
29	Sdma7RxErr	SDMA Channel 7 Rx Error Indicates that a Rx resource error occurred.	0
30	Sdma7TxBuf	SDMA Channel 7 Tx Buffer Return Indicates that SDMA7 Tx closed a descriptor and returned the associated buffer to CPU ownership.	0
31	Sdma7TxEnd	SDMA Channel 7 Tx End Indicates that a Tx resource Error occurred or that the Tx DMA moved to IDLE after a stop command. Also, set when Tx retransmit limit is reached in HDLC mode.	0

**Table 405: MPSC0 Cause Register, Offset: 0x103A20 and
MPSC0 Mask Register, Offset: 0x103AA0**

Bits	Field Name	Function	Initial Value
0	Mpsc0Rx	MPSC0 Normal Rx Interrupt Summary Logical OR of (unmasked) bits 4-7 below. This bit is read only.	0
1	Mpsc0RxErr	MPSC0 Rx Error Interrupt Summary Logical OR of (unmasked) bits 8-11 below. This bit is read only.	0
2		Reserved.	0
3	Mpsc0TxErr	MPSC0 Tx Error Interrupt Summary Logical OR of (unmasked) bits 13-15 below. This bit is read only.	0

**Table 405: MPSC0 Cause Register, Offset: 0x103A20 and
MPSC0 Mask Register, Offset: 0x103AA0 (Continued)**

Bits	Field Name	Function	Initial Value
4	Mpsc0RLSC	MPSC0 Rx Line Status Change (from to IDLE)	0
5	Mpsc0RHNT	MPSC0 Rx Entered HUNT State	0
6	Mpsc0RFSC/ Mpsc0RCC	MPSC0 Rx Flag Status Change (HDLC mode) MPSC0 Received Control Character (Bisync, Uart modes)	0
7	Mpsc0RCSC	MPSC0 Rx Carrier Sense Change (DPLL decoded carriers sense)	0
8	Mpsc0ROVR	MPSC0 Rx Overrun	0
9	Mpsc0RCDL	MPSC0 Rx Carrier Detect Loss	0
10	Mpsc0RCKG	MPSC0 Rx Clock Glitch	0
11	MPsc0BPER	MPSC0 Bisync Protocol Error (valid only in Bisync mode)	0
12	Mpsc0TEIDL	MPSC0 Tx Entered IDLE State	0
13	Mpsc0TUDR	MPSC0 Tx Underrun	0
14	Mpsc0TCTSL	MPSC0 Tx Clear To Send Loss	0
15	Mpsc0TCKG	MPSC0 Tx Clock Glitch	0
31:16		Reserved.	0

**Table 406: MPSC1 Cause Register, Offset: 0x103A24 and
MPSC1 Mask Register, Offset: 0x103AA4**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for MPSC0 cause register.	0

**Table 407: MPSC2 Cause Register, Offset: 0x103A28 and
MPSC2 Mask Register, Offset: 0x103AA8**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for MPSC0 cause register.	0

**Table 408: MPSC3 Cause Register, Offset: 0x103A2C and
MPSC3 Mask Register, Offset: 0x103AAC**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for MPSC0 cause register.	0

**Table 409: MPSC4 Cause Register, Offset: 0x103A30 and
MPSC4 Mask Register, Offset: 0x103AB0**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for MPSC0 cause register.	0

**Table 410: MPSC5 Cause Register, Offset: 0x103A34 and
MPSC5 Mask Register, Offset: 0x103AB4**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for MPSC0 cause register.	0

**Table 411: MPSC6 Cause Register, Offset: 0x103A38 and
MPSC6 Mask Register, Offset: 0x103AB8**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for MPSC0 cause register.	0

**Table 412: MPSC7 Cause Register, Offset: 0x103A3C and
MPSC7 Mask Register, Offset: 0x103ABC**

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for MPSC0 cause register.	0

Table 413: FlexTDM Cause Register, Offset: 0x103A40 and FlexTDM Mask Register, Offset: 0x103AC0

Bits	Field Name	Function	Initial Value
0	Ftdm0RAUXA	FlexTDM0 Rx Interrupt from AUX channel A	0
1	Ftdm0RAUXB	FlexTDM0 Rx Interrupt from AUX channel B	0
2	Ftdm0Rint	FlexTDM0 Rx Interrupt (programmed in dual port ram)	0
3	Ftdm0RSL	FlexTDM0 Rx Synchronization Loss	0
4	Ftdm0TAUXA	FlexTDM0 Tx Interrupt from AUX channel A	0
5	Ftdm0TAUXB	FlexTDM0 Tx Interrupt from AUX channel B	0
6	Ftdm0Tint	FlexTDM0 Tx Interrupt (programmed in dual port ram)	0
7	Ftdm0TSL	FlexTDM0 Tx Synchronization Loss	0
8	Ftdm1RAUXA	FlexTDM1 Rx Interrupt from AUX channel A	0
9	Ftdm1RAUXB	FlexTDM1 Rx Interrupt from AUX channel B	0
10	Ftdm1Rint	FlexTDM1 Rx Interrupt (programmed in dual port ram)	0
11	Ftdm1RSL	FlexTDM1 Rx Synchronization Loss	0
12	Ftdm1TAUXA	FlexTDM1 Tx Interrupt from AUX channel A	0
13	Ftdm1TAUXB	FlexTDM1 Tx Interrupt from AUX channel B	0
14	Ftdm1Tint	FlexTDM1 Tx Interrupt (programmed in dual port ram)	0
15	Ftdm1TSL	FlexTDM1 Tx Synchronization Loss	0
16	Ftdm2RAUXA	FlexTDM2 Rx Interrupt from AUX channel A	0
17	Ftdm2RAUXB	FlexTDM2 Rx Interrupt from AUX channel B	0
18	Ftdm2Rint	FlexTDM2 Rx Interrupt (programmed in dual port ram)	0
19	Ftdm2RSL	FlexTDM2 Rx Synchronization Loss	0
20	Ftdm2TAUXA	FlexTDM2 Tx Interrupt from AUX channel A	0
21	Ftdm2TAUXB	FlexTDM2 Tx Interrupt from AUX channel B	0
22	Ftdm2Tint	FlexTDM2 Tx Interrupt (programmed in dual port ram)	0
23	Ftdm2TSL	FlexTDM2 Tx Synchronization Loss	0
24	Ftdm3RAUXA	FlexTDM3 Rx Interrupt from AUX channel A	0
25	Ftdm3RAUXB	FlexTDM3 Rx Interrupt from AUX channel B	0
26	Ftdm3Rint	FlexTDM3 Rx Interrupt (programmed in dual port ram)	0
27	Ftdm3RSL	FlexTDM3 Rx Synchronization Loss	0
28	Ftdm3TAUXA	FlexTDM3 Tx Interrupt from AUX channel A	0

Table 413: FlexTDM Cause Register, Offset: 0x103A40 and FlexTDM Mask Register, Offset: 0x103AC0 (Continued)

Bits	Field Name	Function	Initial Value
29	Ftdm3TAUXB	FlexTDM3 Tx Interrupt from AUX channel B	0
30	Ftdm3Tint	FlexTDM3 Tx Interrupt (programmed in dual port ram)	0
31	Ftdm3TSL	FlexTDM3 Tx Synchronization Loss	0

Table 414: BRG Cause Register, Offset: 0x103A48 and BRG Mask Register, Offset: 0x103AC8

Bits	Field Name	Function	Initial Value
0	BTR0	Baud Tuning 0 interrupt	0
1	BTR1	Baud Tuning 1 interrupt	0
2	BTR2	Baud Tuning 2 interrupt	0
3	BTR3	Baud Tuning 3 interrupt	0
4	BTR4	Baud Tuning 4 interrupt	0
5	BTR5	Baud Tuning 5 interrupt	0
6	BTR6	Baud Tuning 6 interrupt	0
7	BTR7	Baud Tuning 7 interrupt	0
31:24		Reserved.	0

Table 415: GPP0 Cause Register, Offset: 0x103A50 and GPP0 Mask Register, Offset: 0x103AD0

Bits	Field Name	Function	Initial Value
31:0	GPInt[31:0]	General Purpose Interrupt Bits A bit in this cause register is set when the value latched in the GPD register bit is '0'. NOTE: Interrupts also occur when the associated GPC pin is configured as a functional input.	0

**Table 416: GPP1 Cause Register, Offset: 0x103A54 and
GPP1 Mask Register, Offset: 0x103AD4**

Bits	Field Name	Function	Initial Value
31:0	GPInt[63:32]	<p>General Purpose Interrupt Bits</p> <p>A bit in this cause register is set when the value latched in the GPD register bit is '0'.</p> <p>NOTE: Interrupts also occur when the associated GPC pin is configured as a functional input.</p>	0

**Table 417: GPP2 Cause Register, Offset: 0x103A58 and
GPP2 Mask Register, Offset: 0x103AD8**

Bits	Field Name	Function	Initial Value
31:0	GPInt[95:64]	<p>General Purpose Interrupt Bits</p> <p>A bit in this cause register is set when the value latched in the GPD register bit is '0'.</p> <p>NOTE: Interrupts also occur when the associated GPC pin is configured as a functional input.</p>	0

Table 418: SErr0* Mask, PCI_0 Events Offset: 0xc28

Bits	Field Name	Function	Initial Value
0	AddrErr0	<p>Mask bit.</p> <p>When this bit is set and the GT-96100A detects a parity error on PCI_0 address lines, SErr0* is asserted.</p>	0x0
1	MasWrErr0	<p>Mask bit.</p> <p>When this bit is set and the GT-96100A detects a parity error during a PCI_0 master write operation, SErr0* is asserted.</p>	0x0
2	MasRdErr0	<p>Mask bit.</p> <p>When this bit is set and the GT-96100A detects a parity error during a PCI_0 master read operation, SErr0* is asserted.</p>	0x0
3	MemErr	<p>Mask bit.</p> <p>When this bit is set and a memory parity error has been detected, SErr0* is asserted.</p>	0x0
4	MasAbor0t	<p>Mask bit.</p> <p>When this bit is set and the GT-96100A performs a PCI_0 master abort, SErr0* is asserted.</p>	0x0

Table 418: SErr0* Mask, PCI_0 Events Offset: 0xc28 (Continued)

Bits	Field Name	Function	Initial Value
5	TarAbort0	Mask bit. When this bit is set and the GT-96100A detects a PCI_0 target abort, SErr0* is asserted.	0x0
31:6	Reserved		0x0

**Table 419: SErr1* Mask, PCI_1 Events Offset: 0xca8
(RESERVED if configured for only PCI_0)**

Bits	Field Name	Function	Initial Value
0	AddrErr1	Mask bit. When this bit is set and the GT-96100A detects a parity error on the PCI_1 address lines, SErr1* is asserted.	0x0
1	MasWrErr1	Mask bit. When this bit is set and the GT-96100A detects a parity error during a PCI_1 master write operation, SErr1* is asserted.	0x0
2	MasRdErr1	Mask bit. When this bit is set and the GT-96100A detects a parity error during a PCI_1 master read operation, SErr1* is asserted.	0x0
3	MemErr	Mask bit. When this bit is set and a memory parity error has been detected, SErr1* is asserted.	0x0
4	MasAbort1	Mask bit. When this bit is set and the GT-96100A performs a PCI_1 master abort, SErr1* is asserted.	0x0
5	TarAbort1	Mask bit. When this bit is set and the GT-96100A detects a PCI_1 target abort, SErr1* is asserted.	0x0
31:6	Reserved		0x0

22. RESET CONFIGURATION

The GT-96100A must acquire some knowledge about the system before it is configured by the software.

Special modes of operation are sampled on RESET in order to enable the GT-96100A to function as required. Certain pins must be pulled up to VCC3.3 or down to GND (4.7K Ohm recommended) externally to accomplish this.

The following configuration pins are continuously sampled from Rst* assertion until three TClk cycles after Rst* is deasserted. This does not apply to Frame1*/Req64* that requires zero hold time in respect to RESET rise (as defined in PCI spec).

NOTE: Rst* must be de-asserted for at least 10 PClk cycles before any CPU transactions are generated.

Table 420: Reset Configuration

Pin	Configuration Function
DAdr[2], Frame1*/Req64*:	PCI Bus Configuration
	00-Only PCI_0 is used as 64-bit. 01-Only PCI_0 is used as 32-bit, PCI_1 is NOT used. 10-Reserved. 11-Both PCI_0 and PCI_1 are used as 32-bit.
Interrupt0*:	CPU Data Endianess
	0-Big endian 1-Little endian
Ready*, CSTiming*	Multi-GT-96100A Address ID
	00-GT responds to SysAD[26,25]= 00 01-GT responds to SysAD[26,25]= 01 10-GT responds to SysAD[26,25]= 10 11- GT responds to SysAD[26,25]= 11 NOTE: Boot GT-96100A should be programmed to 11.
BankSel[0]:	PCI Class Code Default Select
	0-Memory Controller (0x580) 1-Network Controller (0x280)
DAdr[10]:	Multiple GT-96100A Support
	0-Not supported. 1-Supported.
DAdr[9]:	66 MHz PCI
	0-Disable. 1-Enable.
DAdr[8]:	I2O Support
	0-Enable. 1-Disable.

Table 420: Reset Configuration (Continued)

Pin	Configuration Function
DAdr[7]:	UMA Support
	0- Enable - DMAReq[0]*/MREQ* functions as MREQ*. 1- Disable - DMAReq[0]*/MREQ* functions as DMAReq[0]*.
DAdr[6]:	Programming Conditional PCI Retry
	0- Enable. 1- Disable.
DAdr[5]:	Expansion ROM Enable
	0- Enable. 1- Disable.
DAdr[4:3]:	Device Boot Bus Width (Controlled by BootCS*) AND CS[3]* Device Width.
	00- 8 bits 01- 16 bits 10- 32 bits 11- 64 bits
DAdr[0]:	Autoload
	0- Enable. 1- Disable.
SDQM[1]:	PCI_1 Power Management
	0- Disable. 1- Enable.
SDQM[0]:	PCI_0 Power Management
	0- Disable. 1- Enable.
DMAReq[3]*	Duplicate ALE
	0- Do not Duplicate ALE. 1- Duplicate ALE output on ADP[1] (no ECC in system).
DMAReq[1]*	Duplicate SDRAM Control Signals
	0- Do not Duplicate SRAS*, SCAS* and DWr*. 1- Duplicate SRAS*, SCAS* and DWr* on ADP[7], ADP[6] and ADP[3] (no ECC in system).
ADP[7:4]	Reserved
	Design in the option to pull-up or pull-down these pins.

Table 420: Reset Configuration (Continued)

Pin	Configuration Function
DAdr[1]	Reserved
	Must pull LOW during Reset.
DMAReq[2]*	Reserved
	Must pull LOW during Reset.
SDQM[2]	Reserved
	Must pull LOW during Reset.
SDQM[3]	Reserved
	Must pull LOW during Reset.

23. CONNECTING THE MEMORY CONTROLLER TO SDRAM AND DEVICES

In order to connect the memory (SDRAM and Devices), follow the pin connections for the appropriate SDRAM and devices listed in this section's tables.

23.1 SDRAM

The GT-96100A supports both 64-bit and 32-bit SDRAM.

Table 421: 64-bit SDRAM

Connection	Connect...	To...
SDRAM Address	DAdr[12:0]	A[10:0] A[11] (64/128/256 Mbit only) A[12] (256 Mbit only)
SDRAM Bank Address	BankSel[1:0]	BA0 BA1 (64/128/256 Mbit only)
SDRAM Data	AD[63:0]	D[63:0], SDRAM Data pins
SDRAM Control Pins	SRAS* SCAS* DW _r * ¹	SDRAM Row Address Strobe SDRAM Column Address Strobe Write Enable
Chip Selects	SCS[0]* SCS[1]* SCS[2]* SCS[3]*	Chip Select, Bank 0 Chip Select, Bank 1 Chip Select, Bank 2 Chip Select, Bank 3
Byte Enables	SDQM[0]* SDQM[1]* SDQM[2]* SDQM[3]* SDQM[4]* SDQM[5]* SDQM[6]* SDQM[7]*	D[7:0] Byte Enable D[15:8] Byte Enable D[23:16] Byte Enable D[31:24] Byte Enable D[39:32] Byte Enable D[47:40] Byte Enable D[55:48] Byte Enable D[63:56] Byte Enable
ECC Bits	ADP[7:0]	D[63:0] ECC byte
Clock	Same Clock Output used for TCik	Clock Input

1. SRAS*, SCAS*, and DW_r* can be duplicated on ADP[7], ADP[6] and ADP[3] if programmed on RESET.

Table 422: 32-bit SDRAM

Connection	Connect...	To...
SDRAM Address	DAdr[11:0]	A[10:0] A[11] (64/128 Mbit only)
SDRAM Bank Address	BankSel[1:0]	BA0 BA1 (64/128 Mbit only)
SDRAM Even Data SDRAM Odd Data	AD[31:0] AD[63:32]	Even D[31:0] SDRAM Data pins Odd D[31:0] SDRAM Data pins
SDRAM Control Pins	SRAS* SCAS* DWr* ¹	SDRAM Row Address Strobe SDRAM Column Address Strobe Write Enable
Chip Selects	SCS[0]* SCS[1]* SCS[2]* SCS[3]*	Chip Select, Bank 0 Chip Select, Bank 1 Chip Select, Bank 2 Chip Select, Bank 3
Byte Enables	SDQM[0]* SDQM[1]* SDQM[2]* SDQM[3]* SDQM[4]* SDQM[5]* SDQM[6]* SDQM[7]*	Even D[7:0] Byte Enable Even D[15:8] Byte Enable Even D[23:16] Byte Enable Even D[31:24] Byte Enable Odd D[7:0] Byte Enable Odd D[15:8] Byte Enable Odd D[23:16] Byte Enable Odd D[31:24] Byte Enable
ECC Bits	Not supported for 32-bit SDRAM.	
Clock	Same Clock Output used for TClk.	Clock Input

1. SRAS*, SCAS*, and DWr* can be duplicated on ADP[7], ADP[6] and ADP[3] if programmed on RESET.

23.2 Devices

The GT-96100A supports 64-, 32-, 16- and 8-bit devices.

Table 423: 64-bit Devices

Connection	Connect...	To...
Device Address	BAdr[2:0] AD[31:6] ALE Latch Outputs	To the device's LSB address bits. Address Latch Inputs Address LE Device Address Bits [28:3]

Table 423: 64-bit Devices (Continued)

Connection	Connect...	To...
Device Data	AD[63:0]	Device Data Pins [63:0]
Device Control Pins	ALE AD[41:32] Control latch bit[41] output Control latch bit[40] output Control latch bit[39:36] outputs Control latch bit[35:32] outputs	Control latch LE Control Latch Inputs Becomes DevRW* Becomes BootCS* Becomes CS[3:0]* Becomes DMAAck[3:0]*
Write Strobes	Wr[0]* Wr[1]* Wr[2]* Wr[3]* Wr[4]* Wr[5]* Wr[6]* Wr[7]*	D[7:0] Write Strobe D[15:8] Write Strobe D[23:16] Write Strobe D[31:24] Write Strobe D[39:32] Write Strobe D[47:40] Write Strobe D[55:48] Write Strobe D[63:56] Write Strobe
ECC Bits	Not supported for Devices.	

Table 424: 32-bit Devices

Connection	Connect...	To...
Device Address	BAdr[2:0] AD[31:5] ALE Latch Outputs	To the device's LSB address bits. Address Latch Inputs Address LE Device Address Bits [29:3]
Device Data	AD[31:0] AD[63:32]	Device Even Data Pins [31:0] Device Odd Data Pins[31:0]
Device Control Pins	ALE AD[41:32] Control latch bit[41] output Control latch bit[40] output Control latch bit[39:36] outputs Control latch bit[35:32] outputs	Control latch LE Control Latch Inputs Becomes DevRW* Becomes BootCS* Becomes CS[3:0]* Becomes DMAAck[3:0]*

Table 424: 32-bit Devices (Continued)

Connection	Connect...	To...
Write Strobes	Wr[0]* Wr[1]* Wr[2]* Wr[3]* Wr[4]* Wr[5]* Wr[6]* Wr[7]*	Even D[7:0] Write Strobe Even D[15:8] Write Strobe Even D[23:16] Write Strobe Even D[31:24] Write Strobe Odd D[7:0] Write Strobe Odd D[15:8] Write Strobe Odd D[23:16] Write Strobe Odd D[31:24] Write Strobe
ECC Bits	Not supported for Devices.	

Table 425: 16-bit Devices

Connection	Connect...	To...
Device Address	BAdr[2:0] AD[31:4] ALE Latch Outputs	To the device's LSB address bits. Address Latch Inputs Address LE Device Address Bits [30:3]
Device Data	AD[15:0] AD[47:32]	Device Even Data Pins [15:0] Device Odd Data Pins[15:0]
Device Control Pins	ALE AD[41:32] Control latch bit[41] output Control latch bit[40] output Control latch bit[39:36] outputs Control latch bit[35:32] outputs	Control latch LE Control Latch Inputs Becomes DevRW* Becomes BootCS* Becomes CS[3:0]* Becomes DMAAck[3:0]*
Write Strobes	Wr[0]* Wr[1]* Wr[4]* Wr[5]*	Even D[7:0] Write Strobe Even D[15:8] Write Strobe Odd D[7:0] Write Strobe Odd D[15:8] Write Strobe
ECC Bits	Not supported for Devices.	

Table 426: 8-bit Devices

Connection	Connect...	To...
Device Address	BAdr[2:0] AD[31:3] ALE Latch Outputs	To the device's LSB address bits. Address Latch Inputs Address LE Device Address Bits [31:3]
Device Data	AD[7:0] AD[39:32]	Device Even Data Pins [7:0] Device Odd Data Pins[7:0]
Device Control Pins	ALE AD[41:32] Control latch bit[41] output Control latch bit[40] output Control latch bit[39:36] outputs Control latch bit[35:32] outputs	Control latch LE Control Latch Inputs Becomes DevRW* Becomes BootCS* Becomes CS[3:0]* Becomes DMAAck[3:0]*
Write Strobes	Wr[0]* Wr[4]*	Even D[7:0] Write Strobe Odd D[7:0] Write Strobe
ECC Bits	Not supported for Devices.	

24. JTAG INTERFACE

24.1 IEEE Standard 1149.1

The GT-96100A supports test mode operation through its JTAG boundary scan interface to enable board testing. The JTAG interface is IEEE 1149.1 standard compliant. It supports mandatory and optional boundary scan instructions.

24.2 TAP Controller

The Test Access Port (TAP) is constructed with a 5-pin interface and a 16-state Finite State Machine (FSM), as defined by IEEE JTAG standard 1149.1.

To place the GT-96100A in functional mode, the JTAG interface must be disabled by resetting the JTAG state machine.

According to the IEEE 1149.1 standard, the JTAG state machine is not reset when the GT-96100A RESET is asserted. The JTAG state machine can only be reset by one of the following methods:

- Asserting TRST* (JTAG[4]).
- By setting TMS (JTAG[1]) for at least five TCK (JTAG[0]) cycles.

To place the GT-96100A in the boundary scan test mode, the JTAG state machine must be moved to its control states. The TMS & TDI inputs control the state transitions of the JTAG state machine, as specified in the 1149.1 standard. The JTAG state machine has various states for shifting instructions to the Instruction Register and for shifting data to and from the boundary scan, identification, or bypass registers.

NOTE: Although the JTAG state machine is not reset, the GT-96100A RESET must be de-asserted (pulled up) when the GT-96100A is in the boundary scan test mode.

In order to meet this requirement, the BSDL file defines the RESET within a compliant pattern.

24.3 Instruction Register (IR)

The Instruction register (IR) is a 4-bit two-stage register. It contains the command that is shifted in when the TAP FSM is in the Shift-IR state. When the TAP FSM is in the Capture-IR state, the IR outputs all four bits in parallel.

The GT-96100A supports the following instructions:

Table 427: Supported JTAG Instructions

Instruction	Code	Description
HIGHZ	0011	Select the Bypass Register between TDI and TDO. Sets the GT-96100A output pins to high-impedance state.
IDCODE	0010	Selects the Identification Register between TDI and TDO. This 32-bit register is used to identify the GT-96100A device. See Table 428
EXTEST	0000	Selects the Boundary Scan Register between TDI and TDO. Output boundary scan register cells drive the output pins of the GT-96100A. Input boundary scan register cell sample the input pin of the GT-96100A.
SAMPLE/PRE-LOAD	0001	Selects the Boundary Scan Register between TDI and TDO. Sample input pins of the GT-96100A to input boundary scan register cells. Preload output boundary scan register cells with the Boundary Scan Register value.
BYPASS	1111	Selects the single bit Bypass Registers between TDI and TDO. This allows for rapid data movement through an untested device.

NOTE: Bi-directional pins can be programmed to be either input or output, depending on their control bit in the Boundary Scan Register.

24.4 Bypass Register (BR)

The Bypass Register (BR) is a one-bit serial shift register that connects TDI to TDO when the IR holds the Bypass command, and the TAP FSM is in the Shift-DR state. Data that is driven on the TDI input pin is shifted out one cycle later on the TDO output pin. The Bypass Register is loaded with "0" when the TAP FSM is in the Capture-DR state.

24.5 JTAG Scan Chain

The JTAG Scan Chain is a serial shift register that is used to sample and drive all of the GT-96100A pins during the JTAG tests. It is a 216-bit deep shift register in the GT-96100A, thereby allowing it to sequentially access all of the data pins. For further details, go to JTAG Scan Chain in BSDL format ([Hash://www.GalileoT.com/library/raclib.htm](http://www.GalileoT.com/library/raclib.htm)) for a full description.

24.6 ID Register

The ID Register is a 32-bit deep serial shift register. The ID Register is loaded with vendor and device information when the TAP FSM is in the Capture-DR state. In the GT-96100A, the ID Register is loaded with "00011001011000010000000101010111" during Capture_DR. The Identification code format of the ID Register is shown in [Table 428](#) which describes the various ID Code fields.

Table 428: IDCODE Register Map

Bits	Value	Description
31:28	0010	Version
27:12	1001011001010011	Part number
11:1	00010101011	Manufacturer ID
0	1	Mandatory

When the JTAG interface is not connected in a JTAG chain on the board:

- It's input pins must be pulled to VCC3.3, or GND (4.7K Ohm recommended).
- It's output pin (TDO) must be left unconnected.

JTAG TMS & TDI input pins must be pulled HIGH, and TCLK & TRST* input pins must be pulled LOW.

25. BIG AND LITTLE ENDIAN

25.1 Background

NOTE: For a description of big and little endian and how it is used in Galileo Technology system controllers, go to Endianess Explained! (<http://www.GalileoT.com/library/syslib.htm>) on the Galileo Technology Website.

There are three bits in the GT-96100A which control byte swapping on the CPU and PCI interfaces. One bit is located in the CPU Interface Configuration Register (0x000) bit 12. The other two bits are in PCI Internal Command register (0xc00) bits 0 and 16.

All these bits are given the same value as sampled at RESET on Interrupt0* pin. These bits can also be programmed after reset is de-asserted.

If all bits are set to 1, the GT-96100A assumes Little-endian data format and NO byte swapping is done within the device.

Additionally, there are three WORD-SWAP bits in GT-96100A which controls 32-bit word swap on access to/from PCI:

- Bit 10 controls PCI master interface word swap.
- Bit 11 controls PCI target interface word swap when accessed through non-swap BARs.
- Bit 12 controls PCI target interface word swap when accessed through swap BARs.

Since the PCI bus is 32-bit wide and the GT-96100A data path is 64-bit wide, byte swap is not good enough in case of working in a BIG endian PCI bus configuration. These three bits are used for endianess compensation for this case.

On top of the above, the GT-96100A supports byte swapping on serial data transferred between the communication unit agents and memory/PCI. Each of the serial DMA channels has two configuration bits associated with it that control byte swapping - one bit controls swapping of incoming (receive) data, and the other bit controls swapping of outgoing (transmit) data. Refer to the Ethernet and SDMA sections for more details about serial DMA configuration options.¹

The nomenclature for this section is shown in [Table 429](#).

1. DMA descriptors that are used by the serial DMA channels are not considered data, and therefore are not affected by the setting of receive/transmit swap bits in the DMA configuration registers. Descriptors swapping is controlled via one bit in the CIU configuration register. Refer to CIU section for details.

Table 429: Nomenclature

Name	Definition
W, Word	32-bits of data, R4600 terminology
DW, Double Word	64-bits of data, R4600 terminology
Even Address	Address of which A[2] == 0 In Little-endian format, this address points to the LEAST significant W of a DW. In Big-endian format, this address points to the MOST significant W of a DW.
Odd Address	Address of which A[2] == 1. In Little-endian format, this address points to the MOST significant W of a DW. In Big-endian format, this address points to the LEAST significant W of a DW.
Even Word	LEAST significant W of a DW.
Odd Word	MOST significant W of a DW.

25.1.1 Bit 12 of the CPU Interface Configuration register

Bit 12 of the CPU Interface Configuration register (0x000) affects the following:

- Setting this bit to 1 (Little-endian mode) means there is no byte swapping within the CPU Interface unit on any data transfer.
- Setting this bit to 0 (Big endian mode) means that byte swapping takes place on data transfers from CPU to the GT-96100A internal registers (including Configuration Data register, offset 0xcfc). No byte swapping takes place on data transfers for which the source/target is external.

25.1.2 Bits 0 and 16 of the PCI Internal Command register

Bit 0 of the PCI Internal Command register (0xc00) controls byte swapping of GT-96100A PCI master interface. Bit 16 controls byte swapping of GT-96100A PCI target interface. These bits affect the following:

- Setting these bits to 1 means there is no byte swapping within the PCI Interface unit on any data transfer.
- Setting these bits to 0 means that no byte swapping takes place on data transfers from PCI to/from the GT-96100A internal registers. Byte swapping does take place on data transfers for which the source/target is external

25.1.3 Bits 10-12 of the PCI Internal Command register

Setting these bits to 0 means there is no word swapping.

Setting these bits to 1 means that no word swapping takes place on data transfers from PCI to/from the GT-96100A internal registers. Word swapping does take place on data transfers for which the source/target is external.

NOTE: Only the 32-bit PCI interface supports word swapping.

25.2 Configuring a System for Big and Little Endian

Table 430 shows the basic combinations of the resources and swapping bits with sample data.

- CPU bit = Bit 12 of the CPU Interface Configuration register (0x000).
- PCI byte swap bit = Bits 0 and 16 of the PCI Internal Command register (0xc00).
- PCI word swap bit = Bits 10-12 of the PCI Internal Command register (0xc00).

NOTE: The sample data is 0x04030201.

Table 430: Configuring for Big and Little Endian

Resource	Swap Bits (CPU bit: PCI byte swap bit: PCI word swap bit)			
	110	001	010	101
Internal Registers (CPU access)	04030201	01020304	01020304	04030201
Internal Registers (PCI access)	04030201	04030201	04030201	04030201
Internal PCI Configuration Registers (CPU access)	04030201	01020304	01020304	04030201
Internal PCI Configuration Registers (PCI access)	04030201	04030201	04030201	04030201
External PCI Configuration Registers	04030201	04030201	01020304	01020304
Memory (DRAM and Devices) (CPU access)	04030201	04030201	04030201	04030201
Memory (DRAM and Devices) (PCI access)	04030201	01020304	04030201	01020304
CPU to PCI (Except external PCI Configuration Registers)	04030201	01020304	04030201	01020304

26. USING THE GT-96100A WITHOUT THE CPU INTERFACE

Table 431 lists the pins that must be strapped when the GT-96100A is used without the CPU interface (i.e., PCI Memory Controller only).

NOTE: Rst* and TClk must always be connected in any system. Each pin must be strapped with a separate resistor unless otherwise noted.

Table 431: CPU-less Pin Strapping

Pin	Strapping ¹
ValidOut*	Pulled up to VCC through a resistor.
Release*	Pulled up to VCC through a resistor.
SysAD[63:0] ²	Pulled up to VCC through a resistor.
SysCmd[8:0] ³	Pulled up to VCC through a resistor.
SysADC[8:0]	No Connect.
ValidIn*	No Connect.
WrRdy*	No Connect.
Interrupt*	Sampled at RESET, see Section 22. “Reset Configuration” on page 452.
Hit	Pulled down to GND.
ScTCE*	Pulled up to VCC.

1. Galileo Technology recommends using 4.7KOhm resistors.
2. SysAD[63:0] can be pulled up through a single resistor instead of 64 separate resistors.
3. SysCmd[8:0] can be pulled up through a single resistor instead of 9 separate resistors.

27. USING THE GT-96100A IN DIFFERENT PCI CONFIGURATIONS

The PCI interface of the GT-96100A can be used in 4 different modes:

- No PCI.
- PCI_0 as 32-bit PCI.
- PCI_0 and PCI_1 as 32-bit PCI.
- PCI_0 as 64-bit PCI.

Table 432 lists what must be done with the pins when the GT-96100A is used without any PCI interface.

NOTE: Rst* must be connected. Most pins should be strapped HIGH or LOW through a resistor. Galileo Technology recommends using 4.7 KOhm resistors.

Table 432: No PCI Interface

Pin	Pin Usage
VREF0	VREF0
PCIk0	Pulled up to VCC through a resistor.
DevSel0*	Pulled up to VCC through a resistor.
Stop0*	Pulled up to VCC through a resistor.
Par0	No Connect.
PErr0*	Pulled up to VCC through a resistor.
Frame0*	Pulled up to VCC through a resistor.
IRdy0*	Pulled up to VCC through a resistor.
TRdy0*	Pulled up to VCC through a resistor.
Gnt0*	Pulled down to GND through a resistor.
IdSel0	Pulled down to GND through a resistor.
SErr0*	Pulled up to VCC through a resistor.
Req0*	No Connect.
Int0*	Pulled up to VCC through a resistor.
Lock0*	Pulled up to VCC through a resistor.
PAD0[31:0]	No Connect.
CBE0[3:0]*	No Connect.
VREF1	Tie directly to 3V or 5V power plane.
PCIk1	Pulled up to VCC through a resistor.
DevSel1*/Ack64*	Pulled up to VCC through a resistor.
Stop1*	Pulled up to VCC through a resistor.
Par1/Par64	No Connect.

Table 432: No PCI Interface (Continued)

Pin	Pin Usage
PErr1*	Pulled up to VCC through a resistor.
Frame1*/Req64*	Pulled up to VCC through a resistor.
IRdy1*	Pulled up to VCC through a resistor.
TRdy1*	Pulled up to VCC through a resistor.
Gnt1*	Pulled down to GND through a resistor.
IdSel1	Pulled down to GND through a resistor.
SErr1*	Pulled up to VCC through a resistor.
Req1*	No Connect.
PAD1[31:0]/PAD0[63:32]	No Connect.
CBE1[3:0]*/CBE0[7:4]*	No Connect.

Table 433 lists what must be done with the pins when the GT-96100A is used with PCI_0 as a 32-bit PCI interface only (i.e., no PCI_1).

NOTE: When the GT-96100A is configured to a single 32-bit PCI_0 interface, the GT-96100A drives all PCI_1 interface signals to a random value. Therefore, there is no need to put pull ups or pull downs on PCI_1 interface signals.

Table 433: PCI_0 as 32-bit PCI Only

Pin	Pin Usage
VREF0	VREF0
PClk0	PClk0
DevSel0*	DevSel0*
Stop0*	Stop0*
Par0	Par0
PErr0*	PErr0*
Frame0*	Frame0*
IRdy0*	IRdy0*
TRdy0*	TRdy0*
Gnt0*	Gnt0*
IdSel0	IdSel0
SErr0*	SErr0*
Req0*	Req0*

Table 433: PCI_0 as 32-bit PCI Only (Continued)

Pin	Pin Usage
Int0*	Int0*
Lock0*	Lock0*
PAD0[31:0]	PAD0[31:0]
CBE0[3:0]*	CBE0[3:0]*
VREF1	Tie directly to 3V or 5V power plane.
PClk1	Pulled up to VCC through a resistor.
DevSel1*/Ack64*	Pulled up to VCC through a resistor.
Stop1*	Pulled up to VCC through a resistor.
Par1/Par64	No Connect.
PErr1*	Pulled up to VCC through a resistor.
Frame1*/Req64*	Pulled up to VCC through a resistor.
IRdy1*	Pulled up to VCC through a resistor.
TRdy1*	Pulled up to VCC through a resistor.
Gnt1*	Pulled down to GND through a resistor or pulled up to VCC through a resistor.
IdSel1	Pulled down to GND through a resistor.
SErr1*	Pulled up to VCC through a resistor.
Req1*	No Connect.
PAD1[31:0]/PAD0[63:32]	No Connect.
CBE1[3:0]*/CBE0[7:4]*	No Connect.

Table 434 lists what must be done with the pins when the GT-96100A is used with PCI_0 as a 32-bit PCI interface and PCI_1 as a 32-bit PCI interface.

Table 434: PCI_0 as 32-bit PCI and PCI_1 as 32-bit PCI

Pin	Pin Usage
VREF0	VREF0
PCIk0	PCIk0
DevSel0*	DevSel0*
Stop0*	Stop0*
Par0	Par0
PErr0*	PErr0*
Frame0*	Frame0*
IRdy0*	IRdy0*
TRdy0*	TRdy0*
Gnt0*	Gnt0*
IdSel0	IdSel0
SErr0*	SErr0*
Req0*	Req0*
Int0*	Int0*
Lock0*	Lock0*
PAD0[31:0]	PAD0[31:0]
CBE0[3:0]*	CBE0[3:0]*
VREF1	VREF1
PCIk1	PCIk1
DevSel1*/Ack64*	DevSel1*
Stop1*	Stop1*
Par1/Par64	Par1
PErr1*	PErr1*
Frame1*/Req64*	Frame1*
IRdy1*	IRdy1*
TRdy1*	TRdy1*
Gnt1*	Gnt1*
IdSel1	IdSel1
SErr1*	SErr1*
Req1*	Req1*

Table 434: PCI_0 as 32-bit PCI and PCI_1 as 32-bit PCI (Continued)

Pin	Pin Usage
PAD1[31:0]/PAD0[63:32]	PAD1[31:0]
CBE1[3:0]*/CBE0[7:4]*	CBE1[3:0]*

Table 435 lists what must be done with the pins when the GT-96100A is used with PCI_0 as a 64-bit PCI interface only (i.e. no PCI_1).

Table 435: PCI_0 as 64-bit PCI Only

Pin	Pin Usage
VREF0	VREF0
PClk0	PClk0
DevSel0*	DevSel0*
Stop0*	Stop0*
Par0	Par0
PErr0*	PErr0*
Frame0*	Frame0*
IRdy0*	IRdy0*
TRdy0*	TRdy0*
Gnt0*	Gnt0*
IdSel0	IdSel0
SErr0*	SErr0*
Req0*	Req0*
Int0*	Int0*
Lock0*	Lock0*
PAD0[31:0]	PAD0[31:0]
CBE0[3:0]*	CBE0[3:0]*
VREF1	Tie directly to 3V or 5V power plane (same as VREF0).
PClk1	Tie directly to PClk0.
DevSel1*/Ack64*	Ack64*
Stop1*	Pulled up to VCC through a resistor.
Par1/Par64	Par64
PErr1*	Pulled up to VCC through a resistor.

Table 435: PCI_0 as 64-bit PCI Only (Continued)

Pin	Pin Usage
Frame1*/Req64*	Req64*
IRdy1*	Pulled up to VCC through a resistor.
TRdy1*	Pulled up to VCC through a resistor.
Gnt1*	Pulled up to VCC through a resistor.
IdSel1	Pulled down to GND through a resistor.
SErr1*	Pulled up to VCC through a resistor.
Req1*	No Connect.
PAD1[31:0]/PAD0[63:32]	PAD0[63:32]
CBE1[3:0]*/CBE0[7:4]*	CBE0[7:4]*

28. PHASED LOCKED LOOP (PLL) APPLICATION NOTES

NOTE: Future revisions of this datasheet will contain new information and guidelines about using PLL.

The GT-96100A contains a Phase Locked Loop (PLL) logic. It is used to improve AC timing of the GT-96100A TClk output signals.

The following sections describe the special care the PLL requires from the system designer.

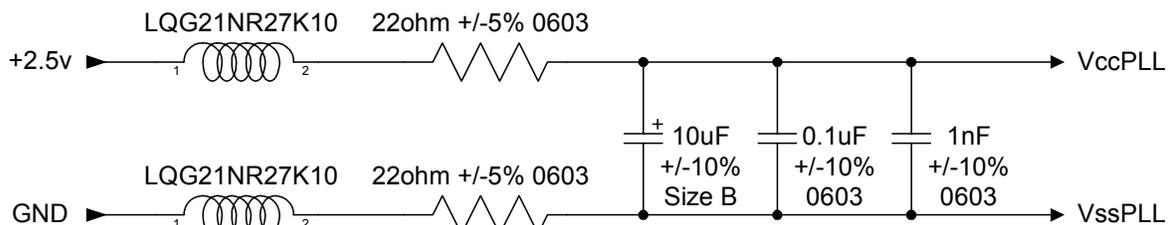
28.1 PLL Power Supply

The GT-96100A’s internal PLL has a separate power supply. There are two dedicated pins for this purpose - VccPLL and VssPLL. See [Section 33. “Pinout Table, 492 Pin BGA” on page 533](#) for exact pin numbers.

These analog power supplies must be isolated from the digital power supply pads and be noise filtered.

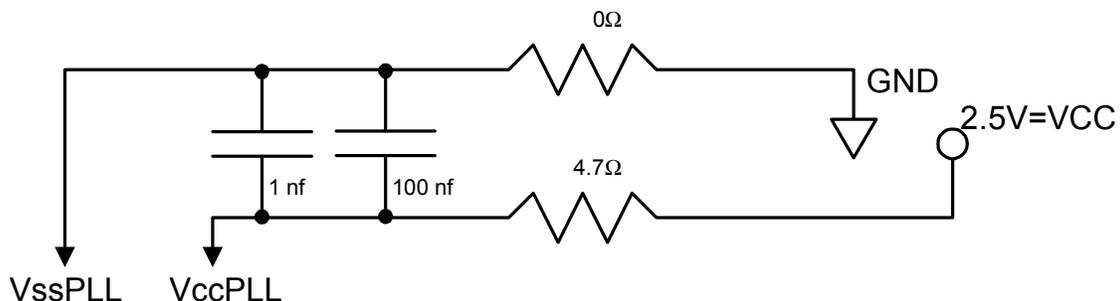
The internal PLL has two parallel capacitors values and two serial resistors used for VccPLL and VssPLL, see [Figure 79](#). The values for the capacitors are 0Ω and 4.7Ω, respectively. The two resistors have a value of 1nf and 100nf, respectively.

Figure 78: Filtering Circuit



NOTE: At this stage, the above recommendations are based on simulations. Galileo uses the above values for testing the GT-96100A. However, some changes to the above values might be required due to difference in package and the physical design of the devices.

Figure 79: Resistor and Capacitor Values for VccPLL and VssPLL



NOTE: The capacitors and the resistor must be located as close as possible to the GT-96100A and only have a minimum distance between them. Also, the capacitors must be assembled so that the 1nf capacitor is closest to the GT-96100A.

28.2 PLL Characteristics

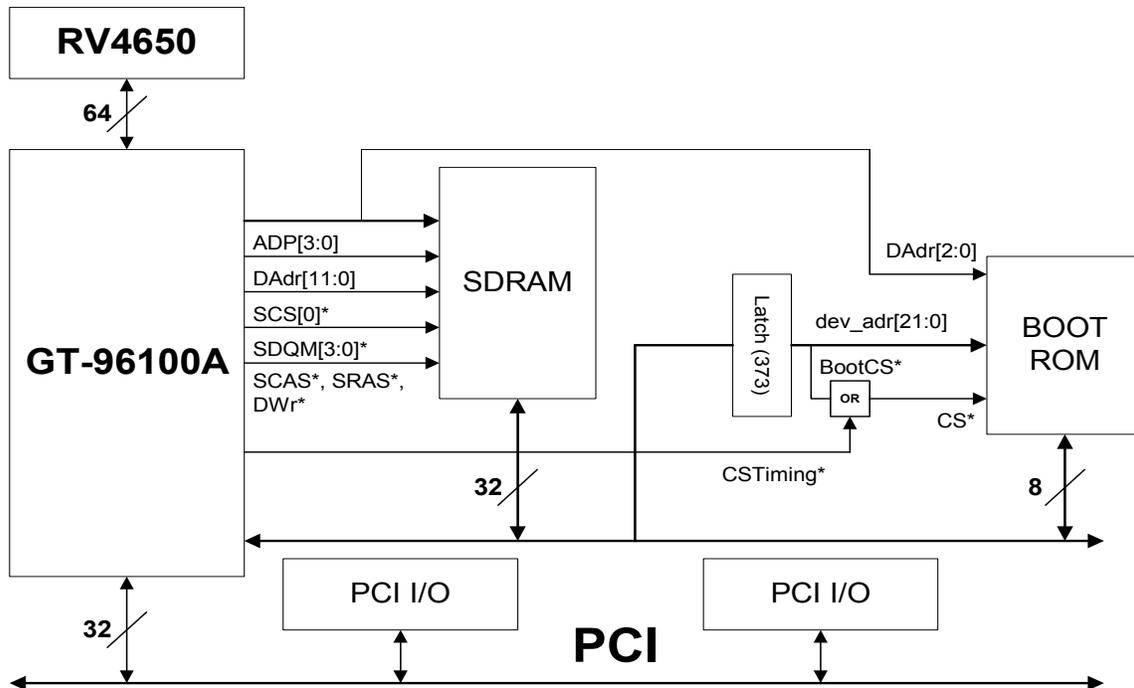
PLL maximum pull-in time PLUS locking time is 0.5 ms. This means that the reset pin must be kept asserted for at least 0.5ms after TClk is on.

29. SYSTEM CONFIGURATIONS

29.1 Minimal System Configuration

- Low Cost RV4650 CPU
- 32-bit SDRAM
- 8-bit Boot EPROM
- 32-bit PCI

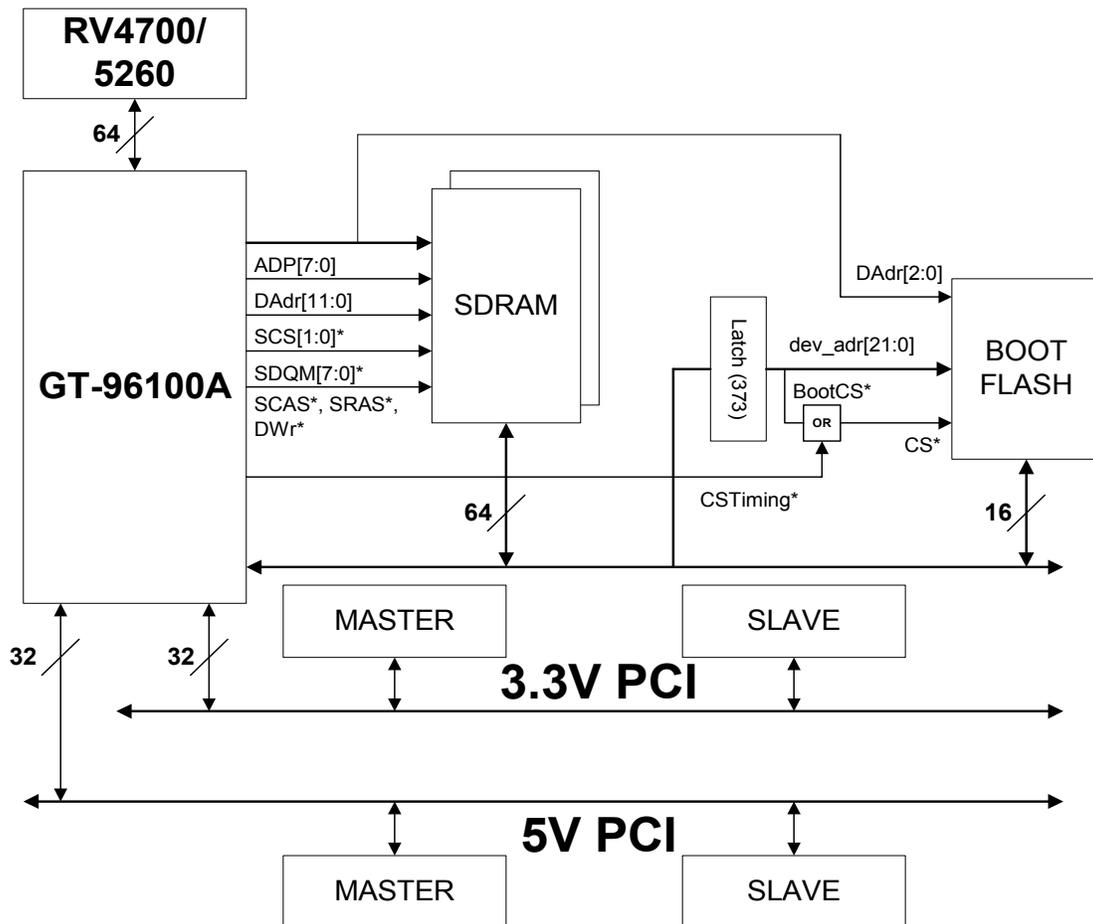
Figure 80: Minimal System Configuration



29.2 Typical System Configuration

- Support for RV4700/5260 CPU.
- Support for 16-bit Devices.
- Support for 64-bit SDRAM.
- Two 32-bit PCI buses (3.3 and 5V support).

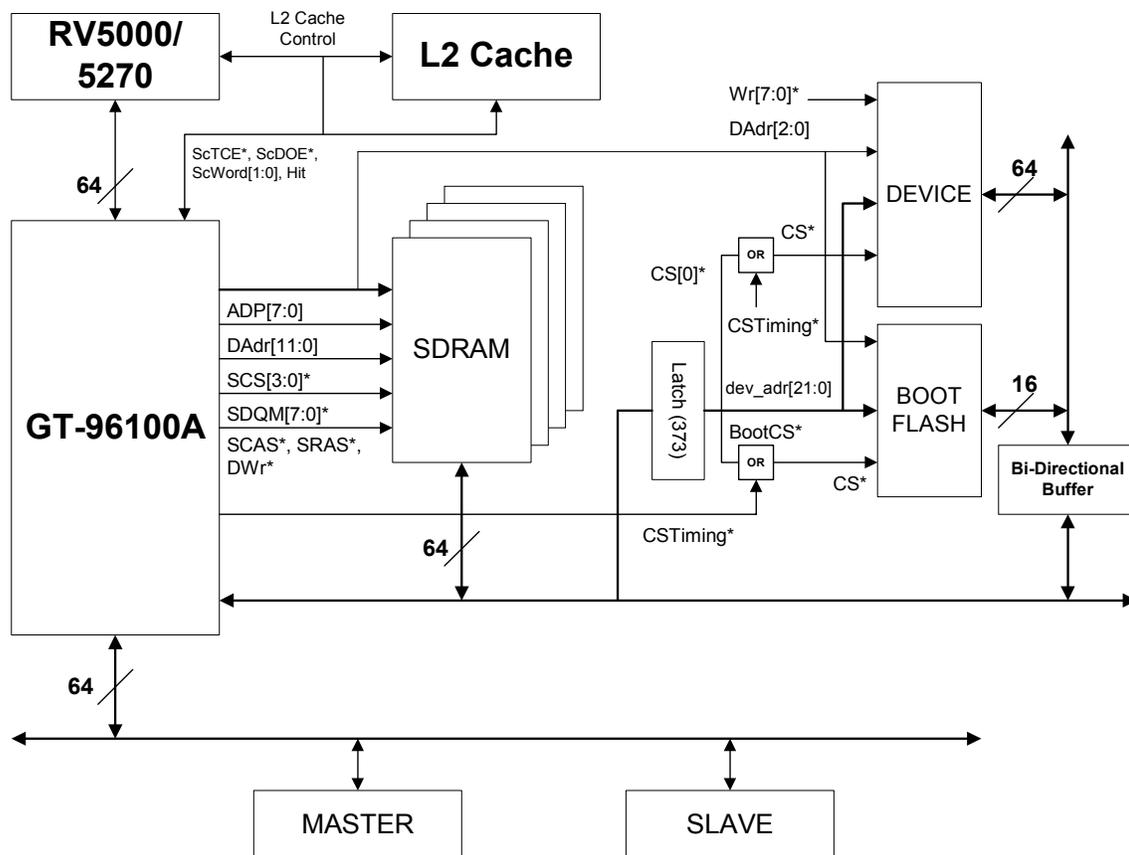
Figure 81: Typical System Configuration



29.3 High Performance System

- Support for RV5000/5270 CPU.
- Support for 2nd Level Cache.
- Support for 64-bit Devices.
- Support for 64-bit PCI.
- Multiple Banks of SDRAM.
- Buffer used for Large AD loading.

Figure 82: High Performance System



30. REGISTER TABLES

The GT-96100A's internal registers are accessed by the CPU or from the PCI bus.

The registers are memory-mapped for the CPU and memory- or I/O-mapped for the PCI.

The registers' address is comprised of the value in the Internal Space Decode register and the register Offset. The value in the Internal Space Decode register [14:0] is matched against bits [35:21] of the actual address; therefore, this value must be the actual address bits [35:21] shifted right once.

For example, to access "Channel 0 DMA Byte Count" register (offset 0x800) immediately after Reset:

- The full address is the default value in the Internal Space Decode register;
- this value is 0x0a0 shifted left once, which gives 0x140, two zero's and the offset 0x800, to become a 32-bit address of 0x14000800.

The location of the registers in the memory space can be changed by changing the value programmed into the Internal Space Decode register. For example, after changing the value in the Internal Space Decode register by writing to 0x14000068 a value of 0bd, an access to the "Channel 0 DMA Byte Count" register is with 0x17a00800.

When writing to the internal registers from the PCI with Byte Enable = 0xF, the write is ignored (as per PCI specifications).

If a write occurs to the following registers with at least one CBE* pin asserted, the entire 32-bit word is written:

- CPU Interface
- Processor Address Space Decoders
- Device Address Space Decoders
- All SDRAM and Device registers
- All DMA registers
- All Communication unit registers
- Timer/Counter

The following internal registers are CBE* sensitive:

- PCI Internal registers
- PCI Configuration registers
- Interrupt Registers

30.1 Access to On-Chip PCI Configuration Space Registers

An access from the CPU to one of the GT-96100A PCI configuration registers is performed differently than accesses to all other registers. The access is performed indirectly by writing the PCI configuration register offset into the Configuration Address register and then reading, or writing, the data from/to the Configuration Data register.

For example, to read data from the Status and Command register, the register offset "0x004" is written into the Configuration Address register, offset 0xcf8 (or full address from the previous example 0xbd000cf8). Then, reading from the Configuration Data register (offset 0xcfc), returns the data of the Status and Command register.

30.2 Register Maps

Table 436: CPU Registers Map

Description	Offset	Page Number
<i>CPU Configuration</i>		
CPU Interface Configuration	0x000000	page 85
Multi-GT Register	0x000120	page 87
<i>CPU Address Decode</i>		
SCS[1:0]* Low Decode Address	0x000008	page 87
SCS[1:0]* High Decode Address	0x000010	page 87
SCS[3:2]* Low Decode Address	0x000018	page 88
SCS[3:2]* High Decode Address	0x000020	page 88
CS[2:0]* Low Decode Address	0x000028	page 88
CS[2:0]* High Decode Address	0x000030	page 88
CS[3]* & Boot CS* Low Decode Address	0x000038	page 88
CS[3]* & Boot CS* High Decode Address	0x000040	page 89
PCI_0 I/O Low Decode Address	0x000048	page 89
PCI_0 I/O High Decode Address	0x000050	page 89
PCI_0 Memory 0 Low Decode Address	0x000058	page 89
PCI_0 Memory 0 High Decode Address	0x000060	page 89
PCI_0 Memory 1 Low Decode Address	0x000080	page 90
PCI_0 Memory 1 High Decode Address	0x000088	page 90
PCI_1 I/O Low Decode Address	0x000090	page 90
PCI_1 I/O High Decode Address	0x000098	page 90
PCI_1 Memory 0 Low Decode Address	0x0000A0	page 90
PCI_1 Memory 0 High Decode Address	0x0000A8	page 91
PCI_1 Memory 1 Low Decode Address	0x0000B0	page 91
PCI_1 Memory 1 High Decode Address	0x0000B8	page 91
Internal Space Decode	0x000068	page 91
SCS[1:0]* Address Remap	0x0000D0	page 91
SCS[3:2]* Address Remap	0x0000D8	page 92
CS[2:0]* Remap	0x0000E0	page 92
CS[3]* & Boot CS* Remap	0x0000E8	page 92

Table 436: CPU Registers Map (Continued)

Description	Offset	Page Number
<i>CPU Address Decode (Continued)</i>		
PCI_0 I/O Remap	0x0000F0	page 92
PCI_0 Memory 0 Remap	0x0000F8	page 92
PCI_0 Memory 1 Remap	0x000100	page 93
PCI_1 I/O Remap	0x000108	page 93
PCI_1 Memory 0 Remap	0x000110	page 93
PCI_1 Memory 1 Remap	0x000118	page 93
<i>CPU Errors Report</i>		
CPU Error Address (Low)	0x070	page 151
CPU Error Address (High)	0x078	page 151
CPU Error Data (Low)	0x128	page 151
CPU Error Data (High)	0x130	page 151
CPU Error Parity	0x138	page 151
<i>CPU Sync Barrier</i>		
PCI_0 Sync Barrier Virtual Register	0x0000C0	page 94
PCI_1 Sync Barrier Virtual Register	0x0000C8	page 94

Table 437: SDRAM Registers Map

Description	Offset	Page Number
<i>SDRAM and Device Address Decode</i>		
SCS[0]* Low Decode Address	0x000400	page 130
SCS[0]* High Decode Address	0x000404	page 130
SCS[1]* Low Decode Address	0x000408	page 130
SCS[1]* High Decode Address	0x00040C	page 130
SCS[2]* Low Decode Address	0x000410	page 131
SCS[2]* High Decode Address	0x000414	page 131
SCS[3]* Low Decode Address	0x000418	page 131
SCS[3]* High Decode Address	0x00041C	page 131
CS[0]* Low Decode Address	0x000420	page 131
CS[0]* High Decode Address	0x000424	page 132

Table 437: SDRAM Registers Map (Continued)

Description	Offset	Page Number
<i>SDRAM and Device Address Decode (Continued)</i>		
CS[1]* Low Decode Address	0x000428	page 132
CS[1]* High Decode Address	0x00042C	page 132
CS[2]* Low Decode Address	0x000430	page 132
CS[2]* High Decode Address	0x000434	page 132
CS[3]* Low Decode Address	0x000438	page 133
CS[3]* High Decode Address	0x00043C	page 133
Boot CS* Low Decode Address	0x000440	page 133
Boot CS* High Decode Address	0x000444	page 133
Address Decode Error	0x000470	page 133
<i>SDRAM Configuration</i>		
SDRAM Configuration	0x000448	page 134
SDRAM Operation Mode	0x000474	page 135
SDRAM Burst Mode	0x000478	page 135
SDRAM Address Decode	0x00047C	page 136
<i>SDRAM Parameters</i>		
SDRAM Bank0 Parameters	0x00044C	page 137
SDRAM Bank1 Parameters	0x000450	page 138
SDRAM Bank2 Parameters	0x000454	page 138
SDRAM Bank3 Parameters	0x000458	page 139
<i>ECC</i>		
ECC Upper Data	0x000480	page 139
ECC Lower Data	0x000484	page 139
ECC from Memory	0x000488	page 139
ECC Calculated	0x00048C	page 139
ECC Error report	0x000490	page 140

Table 437: SDRAM Registers Map (Continued)

Description	Offset	Page Number
<i>Device Parameters</i>		
Device Bank0 Parameters	0x00045C	page 140
Device Bank1 Parameters	0x000460	page 141
Device Bank2 Parameters	0x000464	page 141
Device Bank3 Parameters	0x000468	page 141
Device Boot Bank Parameters	0x00046C	page 142

Table 438: DMA Registers Map

Description	Offset	Page Number
<i>DMA Record</i>		
Channel 0 DMA Byte Count	0x000800	page 232
Channel 1 DMA Byte Count	0x000804	page 232
Channel 2 DMA Byte Count	0x000808	page 232
Channel 3 DMA Byte Count	0x00080C	page 233
Channel 0 DMA Source Address	0x000810	page 233
Channel 1 DMA Source Address	0x000814	page 233
Channel 2 DMA Source Address	0x000818	page 233
Channel 3 DMA Source Address	0x00081C	page 233
Channel 0 DMA Destination Address	0x000820	page 233
Channel 1 DMA Destination Address	0x000824	page 234
Channel 2 DMA Destination Address	0x000828	page 234
Channel 3 DMA Destination Address	0x00082C	page 234
Channel 0 Next Record Pointer	0x000830	page 234
Channel 1 Next Record Pointer	0x000834	page 234
Channel 2 Next Record Pointer	0x000838	page 235
Channel 3 Next Record Pointer	0x00083C	page 235
Channel 0 Current Descriptor Pointer	0x000870	page 235
Channel 1 Current Descriptor Pointer	0x000874	page 235
Channel 2 Current Descriptor Pointer	0x000878	page 235
Channel 3 Current Descriptor Pointer	0x00087C	page 236

Table 438: DMA Registers Map (Continued)

Description	Offset	Page Number
<i>DMA Record (Continued)</i>		
Channel 0 Control	0x000840	page 236
Channel 1 Control	0x000844	page 239
Channel 2 Control	0x000848	page 239
Channel 3 Control	0x00084C	page 239
<i>DMA Arbiter</i>		
Arbiter Control	0x000860	page 240

Table 439: Timer/Counter Registers Map

Description	Offset	Page Number
Timer /Counter 0	0x000850	page 403
Timer /Counter 1	0x000854	page 403
Timer /Counter 2	0x000858	page 403
Timer /Counter 3	0x00085C	page 404
Timer /Counter Control	0x000864	page 404

Table 440: PCI Registers Map

Description	Offset	Page Number
<i>PCI Internal</i>		
PCI_0 Command	0x000C00	page 172
PCI_1 Command	0x000C80	page 174
PCI_0 Time Out & Retry	0x000C04	page 174
PCI_1 Time Out & Retry	0x000C84	page 174
PCI_0 SCS[1:0]* Bank Size	0x000C08	page 175
PCI_1 SCS[1:0]* Bank Size	0x000C88	page 175
PCI_0 SCS[3:2]* Bank Size	0x000C0C	page 175
PCI_1 SCS[3:2]* Bank Size	0x000C8C	page 176
PCI_0 CS[2:0]* Bank Size	0x000C10	page 176
PCI_1 CS[2:0]* Bank Size	0x000C90	page 176

Table 440: PCI Registers Map (Continued)

Description	Offset	Page Number
<i>PCI Internal (Continued)</i>		
PCI_0 CS[3]* & Boot CS* Bank Size	0x000C14	page 177
PCI_1 CS[3]* & Boot CS* Bank Size	0x000C94	page 177
PCI_0 Base Address Registers' Enable	0x000C3C	page 178
PCI_1 Base Address Registers' Enable	0x000CBC	page 179
PCI_0 Prefetch/Max Burst Size	0x000C40	page 179
PCI_1 Prefetch/Max Burst Size	0x000CC0	page 179
PCI_0 SCS[1:0]* Base Address Remap	0x000C48	page 180
PCI_1 SCS[1:0]* Base Address Remap	0x000CC8	page 180
PCI_0 SCS[3:2]* Base Address Remap	0x000C4C	page 181
PCI_1 SCS[3:2]* Base Address Remap	0x000CCC	page 179
PCI_0 CS[2:0]* Base Address Remap	0x000C50	page 182
PCI_1 CS[2:0]* Base Address Remap	0x000CD0	page 182
PCI_0 CS[3]* & Boot CS* Address Remap	0x000C54	page 182
PCI_1 CS[3]* & Boot CS* Address Remap	0x000CD4	page 182
PCI_0 Swapped SCS[1:0]* Base Address Remap	0x000C58	page 180
PCI_1 Swapped SCS[1:0]* Base Address Remap	0x000CD8	page 180
PCI_0 Swapped SCS[3:2]* Base Address Remap	0x000C5C	page 181
PCI_1 Swapped SCS[3:2]* Base Address Remap	0x000CDC	page 181
PCI_0 Swapped CS[3]* & BootCS* Base Address Remap	0x000C64	page 183
PCI_1 Swapped CS[3]* & BootCS* Base Address Remap	0x000CE4	page 183
PCI_0 Configuration Address	0x000CF8	page 183
PCI_1 Configuration Address	0x000CF0	page 184
PCI_0 Configuration Data Virtual Register	0x000CFC	page 184
PCI_1 Configuration Data Virtual Register	0x000CF4	page 184
PCI_0 Interrupt Acknowledge Virtual Register	0x000C34	page 184
PCI_1 Interrupt Acknowledge Virtual Register	0x000C30	page 184

Table 440: PCI Registers Map (Continued)

Description	Offset	Page Number
PCI Configuration		
PCI_0 Device and Vendor ID	0x000000	page 185
PCI_1 Device and Vendor ID	0x000080	page 185
PCI_0 Status and Command	0x000004	page 186
PCI_1 Status and Command	0x000084	page 187
PCI_0 Class Code and Revision ID	0x000008	page 188
PCI_1 Class Code and Revision ID	0x000088	page 188
PCI_0 BIST, Header Type, Latency Timer, Cache Line	0x00000C	page 188
PCI_1 BIST, Header Type, Latency Timer, Cache Line	0x00008C	page 189
PCI_0 SCS[1:0]* Base Address	0x000010	page 190
PCI_1 SCS[1:0]* Base Address	0x000090	page 190
PCI_0 SCS[3:2]* Base Address	0x000014	page 191
PCI_1 SCS[3:2]* Base Address	0x000094	page 191
PCI_0 CS[2:0]* Base Address	0x000018	page 191
PCI_1 CS[2:0]* Base Address	0x000098	page 192
PCI_0 CS[3]* & Boot CS* Base Address	0x00001C	page 192
PCI_1 CS[3]* & Boot CS* Base Address	0x00009C	page 192
PCI_0 Internal Registers Memory Mapped Base Address	0x000020	page 193
PCI_1 Internal Registers Memory Mapped Base Address	0x0000A0	page 193
PCI_0 Internal Registers I/O Mapped Base Address	0x000024	page 193
PCI_1 Internal Registers I/O Mapped Base Address	0x0000A4	page 193
PCI_0 Subsystem ID and Subsystem Vendor ID	0x00002C	page 194
PCI_1 Subsystem ID and Subsystem Vendor ID	0x0000AC	page 194
Expansion ROM Base Address Register	0x000030	page 194
PCI_0 Interrupt Pin and Line	0x00003C	page 195
PCI_1 Interrupt Pin and Line	0x0000BC	page 195

Table 440: PCI Registers Map (Continued)

Description	Offset	Page Number
<i>PCI Configuration, Function 1</i>		
PCI_0 Swapped SCS[1:0]* Base Address	0x000110	page 198
PCI_1 Swapped SCS[1:0]* Base Address	0x000190	page 198
PCI_0 Swapped SCS[3:2]* Base Address	0x000114	page 198
PCI_1 Swapped SCS[3:2]* Base Address	0x000194	page 199
PCI_0 Swapped CS[3]* & Boot CS* Base Address	0x00011C	page 199
PCI_1 Swapped CS[3]* & Boot CS* Base Address	0x00019C	page 200

Table 441: Interrupts Registers Map

Description	Offset	Page Number
Interrupt Main Cause register	0x000C18	page 428
Interrupt0* Main Mask register	0x000C1C	page 432
Interrupt1* Main Mask register	0x000C24	page 434
Interrupt High Cause register	0x000C98	page 430
Interrupt0* High Mask register	0x000C9C	page 433
Interrupt1* High Mask register	0x000CA4	page 435
Interrupt0* Select register	0x000C70	page 431
Interrupt1* Select register	0x000C74	page 432
Serial Cause register	0x103A00	page 436
SerInt0* Mask register	0x103A80	page 438
SerInt1* Mask register	0x103A88	page 439
Ethernet0 Cause register	0x084850	page 440
Ethernet0 Mask register	0x084858	page 440
Ethernet1 Cause register	0x088850	page 441
Ethernet1 Mask register	0x088858	page 441
SDMA Cause register	0x103A10	page 441
SDMA Mask register	0x103A90	page 441
MPSC0 Cause register	0x103A20	page 444
MPSC0 Mask register	0x103AA0	page 444
MPSC1 Cause register	0x103A24	page 445

Table 441: Interrupts Registers Map (Continued)

Description	Offset	Page Number
MPSC1 Mask register	0x103AA4	page 445
MPSC2 Cause register	0x103A28	page 445
MPSC2 Mask register	0x103AA8	page 445
MPSC3 Cause register	0x103A2C	page 446
MPSC3 Mask register	0x103AAC	page 446
MPSC4 Cause register	0x103A30	page 446
MPSC4 Mask register	0x103AB0	page 446
MPSC5 Cause register	0x103A34	page 446
MPSC5 Mask register	0x103AB4	page 446
MPSC6 Cause register	0x103A38	page 446
MPSC6 Mask register	0x103AB8	page 446
MPSC7 Cause register	0x103A3C	page 446
MPSC7 Mask register	0x103ABC	page 446
FlexTDM Cause register	0x103A40	page 447
FlexTDM Mask register	0x103AC0	page 447
BRG Cause register	0x103A48	page 448
BRG Mask register	0x103AC8	page 448
GPP0 Cause register	0x103A50	page 448
GPP0 Mask register	0x103AD0	page 448
GPP1 Cause register	0x103A54	page 449
GPP1 Mask register	0x103AD4	page 449
GPP2 Cause register	0x103A58	page 449
GPP2 Mask register	0x103AD8	page 449
PCI_0 SErr0 Mask	0x000C28	page 449
PCI_1 SErr1 Mask	0x000CA8	page 451

Table 442: I₂O Support Registers Map

Description	Offset	Page Number
Inbound Message Register 0	0x00010	page 211
Inbound Message Register 1	0x00014	page 211
Outbound Message Register 0	0x00018	page 211
Outbound Message Register 1	0x0001C	page 211
Inbound Doorbell Register	0x00020	page 212
Inbound Interrupt Cause Register	0x00024	page 212
Inbound Interrupt Mask Register	0x00028	page 213
Outbound Doorbell Register	0x0002C	page 213
Outbound Interrupt Cause Register	0x00030	page 214
Outbound Interrupt Mask Register	0x00034	page 214
Inbound Queue Port Virtual Register	0x00040	page 215
Outbound Queue Port Virtual Register	0x00044	page 215
Queue Control Register	0x00050	page 215
Queue Base Address Register	0x00054	page 216
Inbound Free Head Pointer Register	0x00060	page 216
Inbound Free Tail Pointer Register	0x00064	page 216
Inbound Post Head Pointer Register	0x00068	page 216
Inbound Post Tail Pointer Register	0x0006C	page 217
Outbound Free Head Pointer Register	0x00070	page 217
Outbound Free Tail Pointer Register	0x00074	page 217
Outbound Post Head Pointer Register	0x00078	page 218
Outbound Post Tail Pointer Register	0x0007C	page 218

NOTE: I₂O registers can be accessed from the CPU and PCI_0 sides (unless stated otherwise). If accessed from the PCI_0 side, address offset is with respect to the PCI_0 SCS[1:0]* Base Address register contents. If accessed from CPU side, the address offset is with respect to the CPU Internal Space Base Register + 0x1c00.

Table 443: Communication Unit Register Map

Description	Offset	Page Number
Ethernet Ports		
Ethernet PHY Address Register (EPAR)	0X080800	page 285
Ethernet SMI Register (ESMIR)	0X080810	page 286
Ethernet0 Ports		
Ethernet0 Port Configuration Register (E0PCR)	0X084800	page 286
Ethernet0 Port Configuration Extend Register (E0PCXR)	0X084808	page 288
Ethernet0 Port Command Register (E0PCMR)	0X084810	page 291
Ethernet0 Port Status Register (E0PSR)	0X084818	page 291
Ethernet0 Serial Parameters Register (E0SPR)	0X084820	page 292
Ethernet0 Hash Table Pointer Register (E0HTPR)	0X084828	page 293
Ethernet0 Flow Control Source Address Low (E0FCSAL)	0X084830	page 293
Ethernet0 Flow Control Source Address High (E0FCSAH)	0X084838	page 294
Ethernet0 SDMA Configuration Register (E0SDCR)	0X084840	page 294
Ethernet0 SDMA Command Register (E0SDCMR)	0X084848	page 295
Ethernet0 Interrupt Cause Register (E0ICR)	0X084850	page 296
Ethernet0 Interrupt Mask Register (E0IMR)	0X084858	page 299
Ethernet0 IP Differentiated Services CodePoint to Priority0 low (E0DSCP2P0L)	0x84860	page 299
Ethernet0 IP Differentiated Services CodePoint to Priority0 high (E0DSCP2P0H)	0x84864	page 299
Ethernet0 IP Differentiated Services CodePoint to Priority1 low (E0DSCP2P1L)	0x84868	page 299
Ethernet0 IP Differentiated Services CodePoint to Priority1 high (E0DSCP2P1H)	0x8486c	page 299
Ethernet0 VLAN Priority Tag to Priority (E0VPT2P)	0x88870	page 299
Ethernet0 First Rx Descriptor Pointer 0 (E0FRDP0)	0X084880	page 263
Ethernet0 First Rx Descriptor Pointer 1 (E0FRDP1)	0X084884	
Ethernet0 First Rx Descriptor Pointer 2 (E0FRDP2)	0X084888	
Ethernet0 First Rx Descriptor Pointer 3 (E0FRDP3)	0X08488C	
Ethernet0 Current Rx Descriptor Pointer 0 (E0CRDP0)	0X0848A0	
Ethernet0 Current Rx Descriptor Pointer 1 (E0CRDP1)	0X0848A4	
Ethernet0 Current Rx Descriptor Pointer 2 (E0CRDP2)	0X0848A8	
Ethernet0 Current Rx Descriptor Pointer 3 (E0CRDP3)	0X0848AC	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
<i>Ethernet0 Ports (Continued)</i>		
Ethernet0 Current Tx Descriptor Pointer 0 (E0CTDP0)	0X0848E0	page 255
Ethernet0 Current Tx Descriptor Pointer 1 (E0CTDP1)	0X0848E4	
Ethernet0 MIB Counters	0X085800 - 0X0858FF	page 302
<i>Ethernet1 Ports</i>		
Ethernet1 Port Configuration Register (E1PCR)	0X088800	page 286
Ethernet1 Port Configuration Extend Register (E1PCXR)	0X088808	page 288
Ethernet1 Port Command Register (E1PCMR)	0X088810	page 291
Ethernet1 Port Status Register (E1PSR)	0X088818	page 291
Ethernet1 Serial Parameters Register (E1SPR)	0X088820	page 292
Ethernet1 Hash Table Pointer Register (E1HTPR)	0X088828	page 293
Ethernet1 Flow Control Source Address Low (E1FCSAL)	0X088830	page 293
Ethernet1 Flow Control Source Address High (E1FCSAH)	0X088838	page 294
Ethernet1 SDMA Configuration Register (E1SDCR)	0X088840	page 294
Ethernet1 SDMA Command Register (E1SDCMR)	0X088848	page 295
Ethernet1 Interrupt Cause Register (E0ICR)	0X088850	page 296
Ethernet1 Interrupt Mask Register (E0IMR)	0X088858	page 299
Ethernet IP Differentiated Services CodePoint to Priority0 low (E0DSCP2P0L)	0x84860	page 299
Ethernet IP Differentiated Services CodePoint to Priority0 high (E0DSCP2P0H)	0x84864	page 299
Ethernet IP Differentiated Services CodePoint to Priority1 low (E0DSCP2P1L)	0x84868	page 299
Ethernet IP Differentiated Services CodePoint to Priority1 high (E0DSCP2P1H)	0x8486c	page 299
Ethernet VLAN Priority Tag to Priority (E0VPT2P)	0x84870	page 299
Ethernet1 First Rx Descriptor Pointer 0 (E1FRDP0)	0X088880	page 263
Ethernet1 First Rx Descriptor Pointer 1 (E1FRDP1)	0X088884	
Ethernet1 First Rx Descriptor Pointer 2 (E1FRDP2)	0X088888	
Ethernet1 First Rx Descriptor Pointer 3 (E1FRDP3)	0X08888C	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
Ethernet1 Current Rx Descriptor Pointer 0 (E1CRDP0)	0X0888A0	page 263
Ethernet1 Current Rx Descriptor Pointer 1 (E1CRDP1)	0X0888A4	
Ethernet1 Current Rx Descriptor Pointer 2 (E1CRDP2)	0X0888A8	
Ethernet1 Current Rx Descriptor Pointer 3 (E1CRDP3)	0X0888AC	
Ethernet1 Current Tx Descriptor Pointer 0 (E1CTDP0)	0X0888E0	page 255
Ethernet1 Current Tx Descriptor Pointer 1 (E1CTDP1)	0X0888E4	
Ethernet1 MIB Counters	0X089800 - 0X0898FF	page 302
SDMAs		
SDMA Group Configuration Register	0X101AF0	page 312
SDMA Group 0, Channel0		
Channel0 Configuration Register (S0DC0)	0X000900	page 312
Channel0 Command Register (S0DCM0)	0X000908	page 314
Channel0 Rx Descriptor	0X008900 - 0X00890F	Not to be accessed during normal operation.
Channel0 Current Rx Descriptor Pointer (S0CRDP0)	0X008910	page 316
Channel0 Tx Descriptor	0X00C900 - 0X00C90F	Not to be accessed during normal operation.
Channel0 Current Tx Descriptor Pointer (S0CTDP0)	0X00C910	page 316
Channel0 First Tx Descriptor Pointer (S0FTDP0)	0X00C914	page 316

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
<i>SDMA Group 0, Channel1</i>		
Channel1 Configuration Register (S0DC1)	0X010900	For a description of the Channel1 registers, see the descriptions for the Channel0 registers.
Channel1 Command Register (S0DCM1)	0X010908	
Channel1 Rx Descriptor	0X018900 - 0X01890F	
Channel1 Current Rx Descriptor Pointer (S0CRDP1)	0X018910	
Channel1 Tx Descriptor	0X01C900 - 0X01C90F	
Channel1 Current Tx Descriptor Pointer (S0CTDP1)	0X01C910	
Channel1 First Tx Descriptor Pointer (S0FTDP1)	0X01C914	
<i>SDMA Group 0, Channel2</i>		
Channel2 Configuration Register (S0DC2)	0X020900	For a description of the Channel2 registers, see the descriptions for the Channel0 registers.
Channel2 Command Register (S0DCM2)	0X020908	
Channel2 Rx Descriptor	0X028900 - 0X02890F	
Channel2 Current Rx Descriptor Pointer (S0CRDP2)	0X028910	
Channel2 Tx Descriptor	0X02C900 - 0X02C90F	
Channel2 Current Tx Descriptor Pointer (S0CTDP2)	0X02C910	
Channel2 First Tx Descriptor Pointer (S0FTDP2)	0X02C914	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
<i>SDMA Group 0, Channel3</i>		
Channel3 Configuration Register (S0DC3)	0X030900	For a description of the Channel3 registers, see the descriptions for the Channel0 registers.
Channel3 Command Register (S0DCM3)	0X030908	
Channel3 Rx Descriptor	0X038900 - 0X03890F	
Channel3 Current Rx Descriptor Pointer (S0CRDP3)	0X038910	
Channel3 Tx Descriptor	0X03C900 - 0X03C90F	
Channel3 Current Tx Descriptor Pointer (S0CTDP3)	0X03C910	
Channel3 First Tx Descriptor Pointer (S0FTDP3)	0X03C914	
<i>SDMA Group 0, Channel4</i>		
Channel4 Configuration Register (S0DC4)	0X040900	For a description of the Channel4 registers, see the descriptions for the Channel0 registers.
Channel4 Command Register (S0DCM4)	0X040908	
Channel4 Rx Descriptor	0X048900 - 0X04890F	
Channel4 Current Rx Descriptor Pointer (S0CRDP4)	0X048910	
Channel4 Tx Descriptor	0X04C900 - 0X04C90F	
Channel4 Current Tx Descriptor Pointer (S0CTDP4)	0X04C910	For a description of the Channel4 registers, see the descriptions for the Channel0 registers.
Channel4 First Tx Descriptor Pointer (S0FTDP4)	0X04C914	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
SDMA Group 0, Channel5		
Channel5 Configuration Register (S0DC5)	0X050900	For a description of the Channel5 registers, see the descriptions for the Channel0 registers.
Channel5 Command Register (S0DCM5)	0X050908	
Channel5 Rx Descriptor	0X058900 - 0X05890F	
Channel5 Current Rx Descriptor Pointer (S0CRDP5)	0X058910	
Channel5 Tx Descriptor	0X05C900 - 0X05C90F	
Channel5 Current Tx Descriptor Pointer (S0CTDP5)	0X05C910	
Channel5 First Tx Descriptor Pointer (S0FTDP5)	0X05C914	
SDMA Group 0, Channel6		
Channel6 Configuration Register (S0DC6)	0X060900	For a description of the Channel6 registers, see the descriptions for the Channel0 registers.
Channel6 Command Register (S0DCM6)	0X060908	
Channel6 Rx Descriptor	0X068900 - 0X06890F	
Channel6 Current Rx Descriptor Pointer (S0CRDP6)	0X068910	
Channel6 Tx Descriptor	0X06C900 - 0X06C90F	
Channel6 Current Tx Descriptor Pointer (S0CTDP6)	0X06C910	
Channel6 First Tx Descriptor Pointer (S0FTDP6)	0X06C914	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
<i>SDMA Group 0, Channel7</i>		
Channel7 Configuration Register (S0DC7)	0X070900	For a description of the Channel7 registers, see the descriptions for the Channel0 registers.
Channel7 Command Register (S0DCM7)	0X070908	
Channel7 Rx Descriptor	0X078900 - 0X07890F	
Channel7 Current Rx Descriptor Pointer (S0CRDP7)	0X078910	
Channel7 Tx Descriptor	0X07C900 - 0X07C90F	
Channel7 Current Tx Descriptor Pointer (S0CTDP7)	0X07C910	
Channel7 First Tx Descriptor Pointer (S0FTDP7)	0X07C914	
<i>SDMA Group 1, Channel0</i>		
Channel1 Configuration Register (S1DC0)	0X100900	page 312
Channel1 Command Register (S1DCM0)	0X100908	page 314
Channel1 Rx Descriptor	0X108900 - 0X10890F	Not to be accessed during normal operation.
Channel1 Current Rx Descriptor Pointer (S1CRDP0)	0X108910	page 316
Channel1 Tx Descriptor	0X10C900 - 0X10C90F	Not to be accessed during normal operation.
Channel1 Current Tx Descriptor Pointer (S1CTDP0)	0X10C910	page 316
Channel1 First Tx Descriptor Pointer (S1FTDP0)	0X10C914	page 316

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
<i>SDMA Group 1, Channel1</i>		
Channel1 Configuration Register (S1DC1)	0X110900	For a description of the Channel1 registers, see the descriptions for the Channel0 registers.
Channel1 Command Register (S1DCM1)	0X110908	
Channel1 Rx Descriptor	0X118900 - 0X11890F	
Channel1 Current Rx Descriptor Pointer (S1CRDP1)	0X118910	
Channel1 Tx Descriptor	0X11C900 - 0X11C90F	
<i>SDMA Group 1, Channel1 (Continued)</i>		
Channel1 Current Tx Descriptor Pointer (S1CTDP1)	0X11C910	For a description of the Channel1 registers, see the descriptions for the Channel0 registers.
Channel1 First Tx Descriptor Pointer (S1FTDP1)	0X11C914	
<i>SDMA Group 1, Channel2</i>		
Channel2 Configuration Register (S1DC2)	0X120900	For a description of the Channel2 registers, see the descriptions for the Channel0 registers.
Channel2 Command Register (S1DCM2)	0X120908	
Channel2 Rx Descriptor	0X128900 - 0X12890F	
Channel2 Current Rx Descriptor Pointer (S1CRDP2)	0X128910	
Channel2 Tx Descriptor	0X12C900 - 0X12C90F	
Channel2 Current Tx Descriptor Pointer (S1CTDP2)	0X12C910	
Channel2 First Tx Descriptor Pointer (S1FTDP2)	0X12C914	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
<i>SDMA Group 1, Channel3</i>		
Channel3 Configuration Register (S1DC3)	0X130900	For a description of the Channel3 registers, see the descriptions for the Channel0 registers.
Channel3 Command Register (S1DCM3)	0X130908	
Channel3 Rx Descriptor	0X138900 - 0X13890F	
Channel3 Current Rx Descriptor Pointer (S1CRDP3)	0X138910	
Channel3 Tx Descriptor	0X13C900 - 0X13C90F	
Channel3 Current Tx Descriptor Pointer (S1CTDP3)	0X13C910	
Channel3 First Tx Descriptor Pointer (S1FTDP3)	0X13C914	
<i>SDMA Group 1, Channel4</i>		
Channel4 Configuration Register (S1DC4)	0X140900	For a description of the Channel4 registers, see the descriptions for the Channel0 registers.
Channel4 Command Register (S1DCM4)	0X140908	
Channel4 Rx Descriptor	0X148900 - 0X14890F	
Channel4 Current Rx Descriptor Pointer (S1CRDP4)	0X148910	
Channel4 Tx Descriptor	0X14C900 - 0X14C90F	
Channel4 Current Tx Descriptor Pointer (S1CTDP4)	0X14C910	
Channel4 First Tx Descriptor Pointer (S1FTDP4)	0X14C914	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
SDMA Group 1, Channel5		
Channel5 Configuration Register (S1DC5)	0X150900	For a description of the Channel5 registers, see the descriptions for the Channel0 registers.
Channel5 Command Register (S1DCM5)	0X150908	
Channel5 Rx Descriptor	0X158900 - 0X15890F	
Channel5 Current Rx Descriptor Pointer (S1CRDP5)	0X158910	
Channel5 Tx Descriptor	0X15C900 - 0X15C90F	
Channel5 Current Tx Descriptor Pointer (S1CTDP5)	0X15C910	
Channel5 First Tx Descriptor Pointer (S1FTDP5)	0X15C914	
SDMA Group 1, Channel6		
Channel6 Configuration Register (S1DC6)	0X160900	For a description of the Channel6 registers, see the descriptions for the Channel0 registers.
Channel6 Command Register (S1DCM6)	0X160908	
Channel6 Rx Descriptor	0X168900 - 0X16890F	
Channel6 Current Rx Descriptor Pointer (S1CRDP6)	0X168910	
Channel6 Tx Descriptor	0X16C900 - 0X16C90F	
Channel6 Current Tx Descriptor Pointer (S1CTDP6)	0X16C910	For a description of the Channel6 registers, see the descriptions for the Channel0 registers.
Channel6 First Tx Descriptor Pointer (S1FTDP6)	0X16C914	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
SDMA Group 1, Channel7		
Channel7 Configuration Register (S1DC7)	0X170900	For a description of the Channel7 registers, see the descriptions for the Channel0 registers.
Channel7 Command Register (S1DCM7)	0X170908	
Channel7 Rx Descriptor	0X178900 - 0X17890F	
Channel7 Current Rx Descriptor Pointer (S1CRDP7)	0X178910	
Channel7 Tx Descriptor	0X17C900 - 0X17C90F	
Channel7 Current Tx Descriptor Pointer (S1CTDP7)	0X17C910	
Channel7 First Tx Descriptor Pointer (S1FTDP7)	0X17C914	
MPSC0		
MPSC0 Main Configuration Low (MMCRL0)	0X000A00	page 329
MPSC0 Main Configuration High (MMCRH0)	0X000A04	page 333
MPSC0 Protocol Configuration (MPCR0)	0X000A08	page 339
Channel0 Register1 (CH0R1)	0X000A0C	page 337
Channel0 Register2 (CH0R2)	0X000A10	
Channel0 Register3 (CH0R3)	0X000A14	
Channel0 Register4 (CH0R4)	0X000A18	
Channel0 Register5 (CH0R5)	0X000A1C	
Channel0 Register6 (CH0R6)	0X000A20	
Channel0 Register7 (CH0R7)	0X000A24	
Channel0 Register8 (CH0R8)	0X000A28	
Channel0 Register9 (CH0R9)	0X000A2C	
Channel0 Register10 (CH0R10)	0X000A30	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
MPSC1		
MPSC1 Main Configuration Low (MMCRL1)	0X008A00	For a description of the MPSC1 registers, see the descriptions for the MPSC0 registers on page 499 .
MPSC1 Main Configuration High (MMCRH1)	0X008A04	
MPSC1 Protocol Configuration (MPCR1)	0X008A08	
Channel1 Register1 (CH1R1)	0X008A0C	
Channel1 Register2 (CH1R2)	0X008A10	
Channel1 Register3 (CH1R3)	0X008A14	
Channel1 Register4 (CH1R4)	0X008A18	
Channel1 Register5 (CH1R5)	0X008A1C	
Channel1 Register6 (CH1R6)	0X008A20	
Channel1 Register7 (CH1R7)	0X008A24	
Channel1 Register8 (CH1R8)	0X008A28	
Channel1 Register9 (CH1R9)	0X008A2C	
Channel1 Register10 (CH1R10)	0X008A30	
Channel1 Register11 (CH1R11)	0X008A34	
MPSC2		
MPSC2 Main Configuration Low (MMCRL2)	0X010A00	For a description of the MPSC1 registers, see the descriptions for the MPSC0 registers on page 499 .
MPSC2 Main Configuration High (MMCRH2)	0X010A04	
MPSC2 Protocol Configuration (MPCR2)	0X010A08	
Channel2 Register1 (CH2R1)	0X010A0C	
Channel2 Register2 (CH2R2)	0X010A10	
Channel2 Register3 (CH2R3)	0X010A14	
Channel2 Register4 (CH2R4)	0X010A18	
Channel2 Register5 (CH2R5)	0X010A1C	
Channel2 Register6 (CH2R6)	0X010A20	
Channel2 Register7 (CH2R7)	0X010A24	
Channel2 Register8 (CH2R8)	0X010A28	
Channel2 Register9 (CH2R9)	0X010A2C	
Channel2 Register10 (CH2R10)	0X010A30	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
MPSC3		
MPSC3 Main Configuration Low (MMCRL3)	0X018A00	For a description of the MPSC3 registers, see the descriptions for the MPSC0 registers on page 499 .
MPSC3 Main Configuration High (MMCRH3)	0X018A04	
MPSC3 Protocol Configuration (MPCR3)	0X018A08	
Channel3 Register1 (CH3R1)	0X018A0C	
Channel3 Register2 (CH3R2)	0X018A10	
Channel3 Register3 (CH3R3)	0X018A14	
Channel3 Register4 (CH3R4)	0X018A18	
Channel3 Register5 (CH3R5)	0X018A1C	
Channel3 Register6 (CH3R6)	0X018A20	
Channel3 Register7 (CH3R7)	0X018A24	
Channel3 Register8 (CH3R8)	0X018A28	
Channel3 Register9 (CH3R9)	0X018A2C	
Channel3 Register10 (CH3R10)	0X018A30	
MPSC4		
MPSC4 Main Configuration Low (MMCRL4)	0X020A00	For a description of the MPSC4 registers, see the descriptions for the MPSC0 registers on page 499 .
MPSC4 Main Configuration High (MMCRH4)	0X020A04	
MPSC4 Protocol Configuration (MPCR4)	0X020A08	
Channel4 Register1 (CH4R1)	0X020A0C	
Channel4 Register2 (CH4R2)	0X020A10	
Channel4 Register3 (CH4R3)	0X020A14	
Channel4 Register4 (CH4R4)	0X020A18	
Channel4 Register5 (CH4R5)	0X020A1C	
Channel4 Register6 (CH4R6)	0X020A20	
Channel4 Register7 (CH4R7)	0X020A24	
Channel4 Register8 (CH4R8)	0X020A28	
Channel4 Register9 (CH4R9)	0X020A2C	
Channel4 Register10 (CH4R10)	0X020A30	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
MPSC5		
MPSC5 Main Configuration Low (MMCR5L)	0X028A00	For a description of the MPSC5 registers, see the descriptions for the MPSC0 registers on page 499 .
MPSC5 Main Configuration High (MMCR5H)	0X028A04	
MPSC5 Protocol Configuration (MPCR5)	0X028A08	
Channel5 Register1 (CH5R1)	0X028A0C	
Channel5 Register2 (CH5R2)	0X028A10	
Channel5 Register3 (CH5R3)	0X028A14	
Channel5 Register4 (CH5R4)	0X028A18	
Channel5 Register5 (CH5R5)	0X028A1C	
Channel5 Register6 (CH5R6)	0X028A20	
Channel5 Register7 (CH5R7)	0X028A24	
Channel5 Register8 (CH5R8)	0X028A28	
Channel5 Register9 (CH5R9)	0X028A2C	
Channel5 Register10 (CH5R10)	0X028A30	
MPSC6		
MPSC6 Main Configuration Low (MMCR6L)	0X030A00	For a description of the MPSC6 registers, see the descriptions for the MPSC0 registers on page 499 .
MPSC6 Main Configuration High (MMCR6H)	0X030A04	
MPSC6 Protocol Configuration (MPCR6)	0X030A08	
Channel6 Register1 (CH6R1)	0X030A0C	
Channel6 Register2 (CH6R2)	0X030A10	
Channel6 Register3 (CH6R3)	0X030A14	
Channel6 Register4 (CH6R4)	0X030A18	
Channel6 Register5 (CH6R5)	0X030A1C	
Channel6 Register6 (CH6R6)	0X030A20	
Channel6 Register7 (CH6R7)	0X030A24	
Channel6 Register8 (CH6R8)	0X030A28	
Channel6 Register9 (CH6R9)	0X030A2C	
Channel6 Register10 (CH6R10)	0X030A30	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
MPSC7		
MPSC7 Main Configuration Low (MMCRL7)	0X038A00	For a description of the MPSC7 registers, see the descriptions for the MPSC0 registers on page 499 .
MPSC7 Main Configuration High (MMCRH7)	0X038A04	
MPSC7 Protocol Configuration (MPCR7)	0X038A08	
Channel7 Register1 (CH7R1)	0X038A0C	
Channel7 Register2 (CH7R2)	0X038A10	
Channel7 Register3 (CH7R3)	0X038A14	
Channel7 Register4 (CH7R4)	0X038A18	
Channel7 Register5 (CH7R5)	0X038A1C	
Channel7 Register6 (CH7R6)	0X038A20	
Channel7 Register7 (CH7R7)	0X038A24	
Channel7 Register8 (CH7R8)	0X038A28	
Channel7 Register9 (CH7R9)	0X038A2C	
Channel7 Register10 (CH7R10)	0X038A30	
FlexTDM0		
FlexTDM0 Transmit Dual Port RAM (TDPR0), block 0	0X000B00 - 0X000BFF	
FlexTDM0 Transmit Dual Port RAM (TDPR0), block 1	0X001B00 - 0X001BFF	
FlexTDM0 Transmit Dual Port RAM (TDPR0), block 2	0X002B00 - 0X002BFF	
FlexTDM0 Transmit Dual Port RAM (TDPR0), block 3	0X003B00 - 0X003BFF	
FlexTDM0 Receive Dual Port RAM (RDPR0), block 0	0X004B00 - 0X004BFF	
FlexTDM0 Receive Dual Port RAM (RDPR0), block 1	0X005B00 - 0X005BFF	
FlexTDM0 Receive Dual Port RAM (RDPR0), block 2	0X006B00 - 0X006BFF	
FlexTDM0 Receive Dual Port RAM (RDPR0), block 3	0X007B00 - 0X007BFF	
FlexTDM0 Transmit Read Pointer (TTRP0)	0X008B00	
FlexTDM0 Receive Read Pointer (TRRP0)	0X008B04	
FlexTDM0 Configuration Register (TCR0)	0X008B08	page 384
FlexTDM0 AUX ChannelA Tx Register (ATA0)	0X008B0C	
FlexTDM0 AUX ChannelA Rx Register (ARA0)	0X008B10	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
<i>FlexTDM0 (Continued)</i>		
FlexTDM0 AUX ChannelB Tx Register (ATB0)	0X008B14	
FlexTDM0 AUX ChannelB Rx Register (ARB0)	0X008B18	
<i>FlexTDM1</i>		
FlexTDM1 Transmit Dual Port RAM (TDPR1), block 0	0X010B00 - 0X010BFF	
FlexTDM1 Transmit Dual Port RAM (TDPR1), block 1	0X011B00 - 0X011BFF	
FlexTDM1 Transmit Dual Port RAM (TDPR1), block 2	0X012B00 - 0X012BFF	
FlexTDM1 Transmit Dual Port RAM (TDPR1), block 3	0X013B00 - 0X013BFF	
FlexTDM1 Receive Dual Port RAM (RDPR1), block 0	0X014B00 - 0X014BFF	
FlexTDM1 Receive Dual Port RAM (RDPR1), block 1	0X015B00 - 0X015BFF	
FlexTDM1 Receive Dual Port RAM (RDPR1), block 2	0X016B00 - 0X016BFF	
FlexTDM1 Receive Dual Port RAM (RDPR1), block 3	0X017B00 - 0X017BFF	
FlexTDM1 Transmit Read Pointer (TTRP1)	0X018B00	
FlexTDM1 Receive Read Pointer (TRRP1)	0X018B04	
FlexTDM1 Configuration Register (TCR1)	0X018B08	page 384
FlexTDM1 AUX ChannelA Tx Register (ATA1)	0X018B0C	
FlexTDM1 AUX ChannelA Rx Register (ARA1)	0X018B10	
FlexTDM1 AUX ChannelB Tx Register (ATB1)	0X018B14	
FlexTDM1 AUX ChannelB Rx Register (ARB1)	0X018B18	
<i>FlexTDM2</i>		
FlexTDM2 Transmit Dual Port RAM (TDPR2), block 0	0X020B00 - 0X020BFF	
FlexTDM2 Transmit Dual Port RAM (TDPR2), block 1	0X021B00 - 0X021BFF	
FlexTDM2 Transmit Dual Port RAM (TDPR2), block 2	0X022B00 - 0X022BFF	
FlexTDM2 Transmit Dual Port RAM (TDPR2), block 3	0X023B00 - 0X023BFF	
FlexTDM2 Receive Dual Port RAM (RDPR2), block 0	0X024B00 - 0X024BFF	
FlexTDM2 Receive Dual Port RAM (RDPR2), block 1	0X025B00 - 0X025BFF	
FlexTDM2 Receive Dual Port RAM (RDPR2), block 2	0X026B00 - 0X026BFF	
FlexTDM2 Receive Dual Port RAM (RDPR2), block 3	0X027B00 - 0X027BFF	
FlexTDM2 Transmit Read Pointer (TTRP2)	0X028B00	
FlexTDM2 Receive Read Pointer (TRRP2)	0X028B04	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
<i>FlexTDM2 (Continued)</i>		
FlexTDM2 Configuration Register (TCR2)	0X028B08	page 384
FlexTDM2 AUX ChannelA Tx Register (ATA2)	0X028B0C	
FlexTDM2 AUX ChannelA Rx Register (ARA2)	0X028B10	
FlexTDM2 AUX ChannelB Tx Register (ATB2)	0X028B14	
FlexTDM2 AUX ChannelB Rx Register (ARB2)	0X028B18	
<i>FlexTDM3</i>		
FlexTDM3 Transmit Dual Port RAM (TDPR3), block 0	0X030B00 - 0X030BFF	
FlexTDM3 Transmit Dual Port RAM (TDPR3), block 1	0X031B00 - 0X031BFF	
FlexTDM3 Transmit Dual Port RAM (TDPR3), block 2	0X032B00 - 0X032BFF	
FlexTDM3 Transmit Dual Port RAM (TDPR3), block 3	0X033B00 - 0X033BFF	
FlexTDM3 Receive Dual Port RAM (RDPR3), block 0	0X034B00 - 0X034BFF	
FlexTDM3 Receive Dual Port RAM (RDPR3), block 1	0X035B00 - 0X035BFF	
FlexTDM3 Receive Dual Port RAM (RDPR3), block 2	0X036B00 - 0X036BFF	
FlexTDM3 Receive Dual Port RAM (RDPR3), block 3	0X037B00 - 0X037BFF	
FlexTDM3 Transmit Read Pointer (TTRP3)	0X038B00	
FlexTDM3 Receive Read Pointer (TRRP3)	0X038B04	
FlexTDM3 Configuration Register (TCR3)	0X038B08	page 384
FlexTDM3 AUX ChannelA Tx Register (ATA3)	0X038B0C	
FlexTDM3 AUX ChannelA Rx Register (ARA3)	0X038B10	
FlexTDM3 AUX ChannelB Tx Register (ATB3)	0X038B14	
FlexTDM3 AUX ChannelB Rx Register (ARB3)	0X038B18	

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
Baud Rate Generators		
BRG0 Configuration Register (BCR0)	0X102A00	page 399
BRG0 Baud Tuning Register (BTR0)	0X102A04	page 400
BRG1 Configuration Register (BCR1)	0X102A08	For BRG1 through BRG7 configuration and tuning registers, see BRG0
BRG1 Baud Tuning Register (BTR1)	0X102A0C	
BRG2 Configuration Register (BCR2)	0X102A10	
BRG2 Baud Tuning Register (BTR2)	0X102A14	
BRG3 Configuration Register (BCR3)	0X102A18	
BRG3 Baud Tuning Register (BTR3)	0X102A1C	
BRG4 Configuration Register (BCR4)	0X102A20	
BRG4 Baud Tuning Register (BTR4)	0X102A24	
BRG5 Configuration Register (BCR5)	0X102A28	
BRG5 Baud Tuning Register (BTR5)	0X102A2C	
BRG6 Configuration Register (BCR6)	0X102A30	
BRG6 Baud Tuning Register (BTR6)	0X102A34	
BRG7 Configuration Register (BCR7)	0X102A38	
BRG7 Baud Tuning Register (BTR7)	0X102A3C	
Routing Registers		
Main Routing Register (MRR)	0X101A00	page 414
Receive Clock Routing Register (RCRR)	0X101A10	page 418
Transmit Clock Routing Register (TCRR)	0X101A20	page 420
General Purpose Ports		
General Purpose Configuration 0 (GPC0)	0X100A00	page 407
General Purpose Configuration 1 (GPC1)	0X100A04	page 409
General Purpose Configuration 2 (GPC2)	0X100A08	page 412
General Purpose Input/Output 0 (GPIO0)	0X100A20	page 407
General Purpose Input/Output 1 (GPIO1)	0X100A24	page 410
General Purpose Input/Output 2 (GPIO2)	0X100A28	page 412
General Purpose Data 0 (GPD0)	0X100A40	page 408
General Purpose Data 1 (GPD1)	0X100A44	page 410

Table 443: Communication Unit Register Map (Continued)

Description	Offset	Page Number
General Purpose Ports (Continued)		
General Purpose Data 2 (GPD2)	0X100A48	page 413
General Purpose Level 0 (GPL0)	0X100A60	page 408
General Purpose Level 1 (GPL1)	0X100A64	page 411
General Purpose Level 2 (GPL2)	0X100A68	page 413
Watchdog		
Watchdog Configuration Register (WDC)	0X101A80	page 401
Watchdog Value Register (WDV)	0X101A84	page 402
Communication Unit Arbiter		
Comm Unit Arbiter Configuration Register (CUACR)	0X101AC0	page 251
PCI Arbiters		
PCI_0 Arbiter Configuration Register	0X101AE0	page 243
PCI_1 Arbiter Configuration Register	0X101AE4	page 245

31. DC CHARACTERISTICS

Table 444: Absolute Maximum Ratings

Symbol	Parameter	Min.	Max.	Unit
$V_{CC2.5}$	Core Supply Voltage	-0.3	3.0	V
$V_{CC3.3}$	IO Supply Voltage	-0.3	4.0	V
V_i	Input Voltage	-0.3	5.5	V
I_{ik}	Input Protect Diode Current		+20	mA
I_{ok}	Output Protect Diode Current		+20	mA
T_c	Operating Case Temperature	0	110	C
T_{stg}	Storage Temperature	-40	125	C

NOTE: Operation at or beyond the maximum ratings is not recommended or guaranteed. Extended exposure at the maximum rating for extended periods of time may adversely affect device reliability.

Table 445: Recommended Operating Conditions

Symbol	Parameter	Min.	Typ.	Max.	Unit
$V_{CC2.5}$	Core Supply Voltage	2.375	2.5	2.625	V
$V_{CC3.3}$	I/O Supply Voltage	3.15	3.3	3.45	V
V_i	Input Voltage	-0.3		5.5	V
V_o	Output Voltage	0		$V_{CC3.3}$	V
T_c	Case Temperature	0		70	C

Table 446: Pin Capacitance

Symbol	Parameter	Min.	Typ.	Max.	Unit
C_{in}	Input Capacitance PCI PAD		2.5	4.5	pF
	Input Capacitance Non PCI PAD		2	3.5	
C_{out}	Output Capacitance PCI PAD		2.5	4.5	pF
	Output Capacitance Non PCI PAD		2	3.5	

31.1 DC Electrical Characteristics Over Operating Range

(T_c=0-70°C; V_{CC}3.3=+3.3V, +/-5%, V_{CC}2.5= +2.5V, +/-5%)

Table 447: DC Electrical Characteristics Over Operating Range

Symbol	Parameter	Test Condition	Min.	Max.	Unit	Loading
V _{ih}	Input HIGH level	Guaranteed Logic HIGH level	2.0V		V	
V _{il}	Input LOW level	Guaranteed Logic LOW level		0.8V	V	
V _{oh}	Output HIGH Voltage:	I _{oH} = 8,12,16,24 ¹ mA	2.4		V	50pF
V _{ol}	Output LOW Voltage:	I _{oL} = 8,12,16,24 ¹ mA		0.4	V	50pF
I _{ih}	Input HIGH Current			+1	uA	
I _{il}	Input LOW Current			+1	uA	
I _{ozh}	High Impedance Output Current			+1	uA	
I _{ozl}	High Impedance Output Current			+1	uA	
I _{cc}	Operating Current	I/O V _{CC} 3.3=3.465V f = 100MHz TCik/ 66Mhz PCik		440	mA	
		Core V _{CC} 2.5=2.625V f = 100MHz TCik/ 66Mhz PCik		880	mA	

1. See Table 448, "Driving Pad Characteristics," on page 510.

Table 448: Driving Pad Characteristics

Output Current	Pads Name
PCI Pads	CBE0_[3:0], CBE1_[3:0], DevSel0_, DevSel1_, Frame0_, Frame1_, Gnt0_, Gnt1_, IDSel0, IDSel1, Interrupt1_, Irdy0_, Irdy1_, Lock0_, PAD0[31:0], PAD1[31:0], Par0, Par1, PCIk0, PCIk1, PErr0_, PErr1_, Req0_, Req1_, SErr0_, SErr1_, Stop0_, Stop1_, Trdy0_, Trdy1_, Reset_
4mA Pads	JTAG[3]
8mA Pads	GPP[15:0], MDC, MDIO, MII0[14:0], MII1[14:0], NMI_, PORTA[6:0], PORTB[6:0], PORTC[6:0], PORTD[6:0], PORTE[6:0], PORTF[6:0], WDE_
16mA Pads	AD[63:0], ADP[1], ADP[3:2], ADP[7:6], ALE, BypsOE_, ClkOutPLL, CSTiming_, DMAReq_[0], Interrupt0_, ScDOE_, SCS_[3:0], ScWord[1:0], SDQM[7:0], SerInt0_, SerInt1_, SysAD[63:0], SysADC[7:0], SysCmd[8:0], ValidIn_, WrRdy_
24mA Pads	ADP[0], ADP[5:4], BankSel, DAdr[10:0], DMAReq_[3:1], DWr_, SCAS_, SRAS_

31.1.1 Power Sequencing Notes

The voltage power must be turned on sequentially so that the highest voltage power is always the preceding one; i.e. the highest voltage power must be turned on first, then the second highest voltage power and so on and so forth.

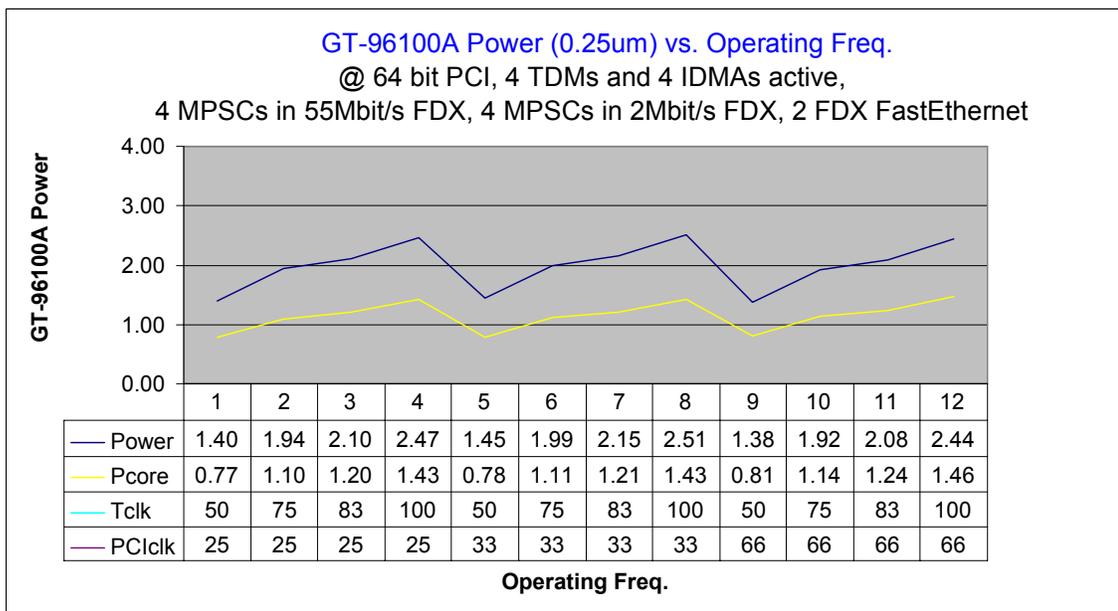
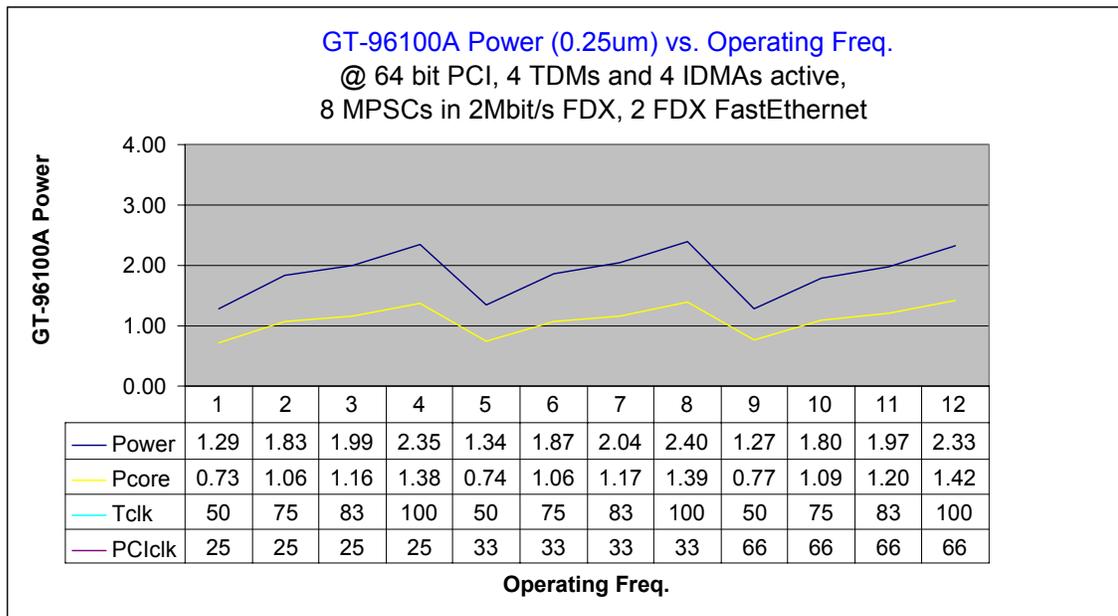
This power up sequence must be used due to the presence of protection diodes between the two power rails. If the power is turned on in incorrect sequence and if the tie-high terminal becomes tie low, these diodes leak current.

Likewise, when powering down, turn off the lowest voltage first. Turn off the next highest and so on and so forth.

31.1.2 Power Consumption

Figure 83 illustrate the GT-96100A power consumption in various common configurations and frequencies.

Figure 83: Power vs. Operating Frequency



31.2 Thermal Data

Table 449 shows the package thermal data for the GT-96100A.

Using the commercial grade device, Galileo Technology recommends the use of heatsink for most systems, especially those with little or no airflow.

Use an adequate airflow, layout, and other means to meet the recommended operating conditions listed in Table 449.

Table 449: Thermal data for GT-96100A in BGA 492

Airflow	0 m/s	1 m/s	2 m/s
θ_{ja}	18.5 c/w	16.6 c/w	15 c/w
Ψ_{jt}	0.37 c/w	0.43 c/w	0.53 c/w
θ_{jc}	5.9 c/w		

32. AC TIMING

NOTE: The following targets are subject to change.

Table 450: AC Timing Measurement Formulas

Commercial	Industrial
I/O Supply Voltage: TCase= 70° C; VCC= +3.3V, +/- 5% Core Supply Voltage: TCase= 70° C; VCC= +2.5V, +/- 5%	NOTE: AC timing specifications for the 66Mhz industrial grade part will be included in future revisions of this datasheet.

Table 451: AC Commercial Grade Timing

All Delays, Setup, and Hold times are referred to TCik RISING edge, unless stated otherwise.

Signals	Description	83 Mhz		100 Mhz		Unit	Loading
		Min.	Max.	Min.	Max.		
Clk							
TCik	Pulse Width High	4.8		4		ns	
TCik	Pulse Width Low	4.8		4		ns	
TCik	Clock Period	12		10		ns	
TCik	Rise Time		TBD		TBD	V/ns	
TCik	Fall Time		TBD		TBD	V/ns	
Reset**	Active Device Reset	10		10		TCik Cycle	
CPU Interface							
SysAD[63:0], SysCMD[8:0], Valid-Out*, Release*, ScTCE*	Setup	3.5		2.5		ns	
ScMatch	Setup	5		2.5			
SysAD[63:0], SysCMD[8:0], Valid-Out*, Release*, ScTCE*, ScMatch	Hold	1		1		ns	

Table 451: AC Commercial Grade Timing (Continued)

All Delays, Setup, and Hold times are referred to TClk RISING edge, unless stated otherwise.

Signals	Description	83 Mhz		100 Mhz		Unit	Loading
		Min.	Max.	Min.	Max.		
CPU Interface (Continued)							
SysAD[63:0], SysCMD[8:0], SysADC[7:0], ScDOE*, ValidIn*, WrRdy*, ScWord[1:0], Interrupt0*	Output Delay	2	7	2	5	ns	50pF
PCI Interface							
Pclk0, Pclk1	Clock Period	15		15		ns	
All Bussed Inputs	Setup	4		3.5		ns	
All Point to Point Inputs	Setup	7		6			
All Inputs	Hold	0		0		ns	
All Outputs, except Req*	Output Delay	2	9	2	8	ns	
Req*	Output Delay	2	11	2	10	ns	
Memory Interface							
AD[63:0]	Setup	3		2		ns	
AD[63:0]	Hold	1		1		ns	
AD[63:0], SCS[3:0]*	Output Delay	2	7	2	5.5	ns	50pF
DAdr[10:0]	Output Delay	2	7	2	5	ns	50pF
DAdr[2:0]	Output Delay (Device Burst)	2	7	2	5	ns	50pF
DAdr[10:3]	Output Delay from TClk Fall- ing (Device Write)	2	7	2	6	ns	50pF
ADP[7:0]	Output Delay (Parity)	2	7	2	5.5	ns	50pF
ADP[7:0]	Setup (Parity)	3		2		ns	50pF
ADP[7:0]	Hold (Parity)	1		1		ns	
ADP[3:0]/EOT[3:0]*	Setup (EOT[3:0]*)	4		3		ns	

Table 451: AC Commercial Grade Timing (Continued)

All Delays, Setup, and Hold times are referred to TCik RISING edge, unless stated otherwise.

Signals	Description	83 Mhz		100 Mhz		Unit	Loading
		Min.	Max.	Min.	Max.		
Memory Interface (Continued)							
ADP[3:0]/EOT[3:0]*	Hold (EOT[3:0]*)	1		1		ns	
ADP[1]/ALE	Output Delay (ALE)	2	7	2	5.5	ns	30pF
ADP[1]/ALE	Output Delay, TCik Falling (ALE)	2	7	2	6	ns	30pF
ADP[3]/DWr*	Output Delay (DWr*)	2	7	2	6	ns	50pF
ADP[5:4]/{Dadr[11], BankSel[1]}	Output Delay (Dadr[11], BankSel[1])	2	7	2	6	ns	50pF
ADP[7:6]/{SRAS*, SCAS*}	Output Delay (SRAS* and SCAS*)	2	7	2	6	ns	50pF
SDQM[7:0]	Output Delay	2	7	2	6		
Ready*	Setup	6		5		ns	
Ready*	Hold	1		1		ns	
DMAReq[0]*/MREQ*/SRAS*	Setup (DMAReq[0]/MREQ*)	4.5		3.5		ns	
DMAReq[0]*/MREQ*/SRAS*	Hold (DMAReq[0]/MREQ*)	1		1		ns	
DMAReq[0]*/MREQ*/SRAS*	Output Delay (SRAS*)	2	7	2	6	ns	50pF
DMAReq[1]*/BankSel[1]	Setup (DMAReq[1]*)	4.5		3.5		ns	
DMAReq[1]*/BankSel[1]	Hold (DMAReq[1]*)	1		1		ns	
DMAReq[1]*/BankSel[1]	Output Delay (BankSel[1])	2	7	2	5	ns	50pF
DMAReq[2]*/DAdr[11]	Setup (DMAReq[2]*)	4.5		3.5		ns	

Table 451: AC Commercial Grade Timing (Continued)

All Delays, Setup, and Hold times are referred to TClk RISING edge, unless stated otherwise.

Signals	Description	83 Mhz		100 Mhz		Unit	Loading
		Min.	Max.	Min.	Max.		
Memory Interface (Continued)							
DMAReq[2]*/DAdr[11]	Hold (DMAReq[2]*)	1		1		ns	
DMAReq[2]*/DAdr[11]	Output Delay (DAdr[11])	1.3	7	1.3	5	ns	50pF
DMAReq[3]*/SCAS*/EOT[0]*	Setup (DMAReq[3]*/EOT[0]*)	4.5		3.5		ns	
DMAReq[3]*/SCAS*/EOT[0]*	Hold (DMAReq[3]*/EOT[0]*)	1		1		ns	
DMAReq[3]*/SCAS*/EOT[0]*	Output Delay (SCAS*)	2	7.5	2	6	ns	50pF
MGnt*/ByPsOE*	Output Delay (MGnt*)	2	7	2	6	ns	30pF
MGnt*/ByPsOE*	Output Delay (Bypass OE*)	2	7.5	2	6.5	ns	30pF
SRAS*, SCAS*, DWr*	Output Delay	2	7	2	5	ns	50pF
ALE	Output Delay	2	7	2	5.5	ns	30pF
ALE	Output Delay, TClk Falling	2	7	2	5.5	ns	30pF
CSTiming*	Output Delay	2	7	2	5.5	ns	30pF

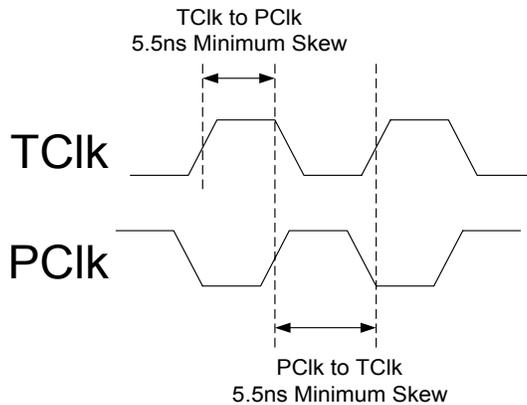
32.1 TClk/PClk Restrictions

TClk cycle must be smaller than PClk cycle by at least 1ns ($T_{pclk} > T_{tclk} + 1ns$). This restriction applies to all sync modes.

There is one exception to this restriction. TClk and PClk can run at the same frequency if the following conditions are met:

- The two clocks are synchronized (derived from the same clock source).
- If running at sync mode 1, a minimum skew of 5.5ns must be observed between rising edge of TClk and PClk, as shown in [Figure 84](#). Galileo Technology recommends using an inverted TClk as PClk in order to guarantee this skew.
- If running at sync mode 2 or 3, a maximum skew of 2ns between rising edge of TClk and PClk must be met.

Figure 84: TCik = PCik, in Sync Mode = 1, Skew Requirement



In addition to the above restriction, there are few sync modes specific restrictions, summarized in [Table 452](#).

Table 452: TCik/PCik Restrictions

Sync Mode	PCik Frequency Range	Restrictions
0,4	from DC up to TCik	$T_{pclk} > T_{tclk} + 1ns$
1	from TCik/2 up to TCik	<p>$T_{pclk} < 2T_{tclk} - 1ns$, unless running with synchronized $T_{pclk} = 2T_{tclk}$ and minimum skew of 5.5ns between PCik rise and TCik rise is guaranteed, as shown in Figure 84. For example, if $T_{tclk} = 15ns$ (66MHz), T_{pclk} should be smaller than 29ns (unless running with synchronized clocks).</p> <p>NOTE: To guarantee this skew, Galileo Technology recommends using an inverted TCik as PCik.</p>
2,3	from TCik/2 up to TCik	TCik and PCik are synchronized , and a maximum skew of 2ns between PCik rise and TCik rise is guaranteed.
5	from TCik/3 up to TCik/2	<p>$T_{pclk} < 3T_{tclk} - 1ns$, unless running with synchronized $T_{pclk} = 3T_{tclk}$ and minimum skew of 5.5ns between PCik rise and TCik rise is guaranteed. For example, if $T_{tclk} = 13.3ns$ (75MHz), T_{pclk} should be smaller than 39ns (unless running with synchronized clocks).</p>
6,7	from TCik/3 up to TCik/2	TCik and PCik are synchronized , and a maximum skew of 2ns between PCik rise and TCik rise is guaranteed.

The sync mode can be programmed by the CPU, the PCI or during autoloading. For sync mode information, see [Section 7.14.1 “PCI Internal Registers” on page 172](#).

32.2 Serial (Communication) Clock Domain AC Characteristic

(TC= 0 - 70°C; VDDI/O= +3.3V, +/- 5%; VDDC = +2.5V/-5%; External Load=50pf)

NOTE: For information on the settings for Receive Sync Delay (RSD), Transmit Sync Delay(TSD), Driving Edge (DE), and Sync Edge (SE), see [Section 15.4 “FlexTDM Configuration Register \(TCR\)”](#) on page 384.

Table 453: Flex-TDM Receive Timing - Normal Clock

Symbol	Description	Min	Max	Unit
t20	TRCLK Cycle Time	18		ns
t21	TRCLK Width Low	7.2		ns
t22	TRCLK Width High	7.2		ns
t23	TRSYNC Setup Time	5		ns
t24	TRSYNC Hold Time	5		ns
t25	TRSYNC Rise and Fall Time		15	ns
t27	TRXD Setup Time	5		ns
t28	TRXD Hold Time	5		ns

Figure 85: Flex-TDM Receive Timing - Normal Clock Waveform

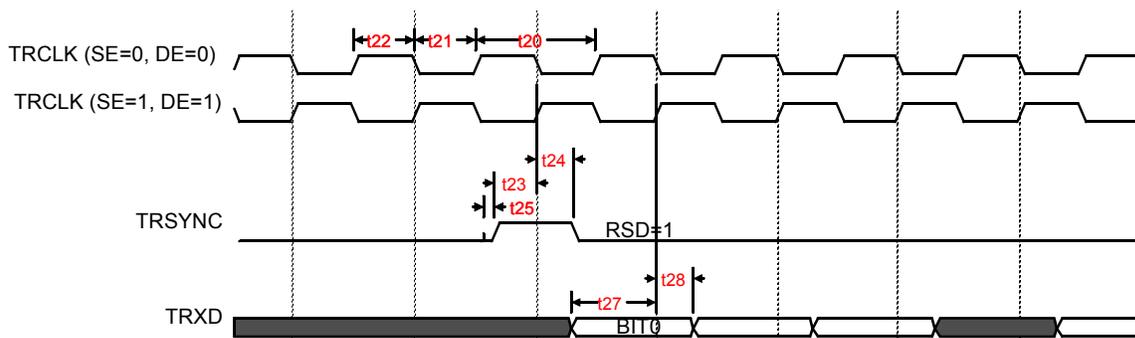


Table 454: Flex-TDM Transmit Timing - Normal Speed Clock

Symbol	Description	Min	Max	Unit
t40	TTCLK Cycle Time	18		ns
t41	TTCLK Width Low	7.2		ns
t42	TTCLK Width High	7.2		ns
t43	TTSYNC Setup Time	5		ns
t44	TTSYNC Hold Time	5		ns
t45	TTSYNC Rise and Fall Time		15	ns
t46	TDSTRB Output Delay		13	ns
t47	TTCLK to TDCLK Delay Time		8	ns
t48	TTXD Output Delay		13	ns

Figure 86: Flex-TDM Transmit Timing - Normal Speed Clock Waveform

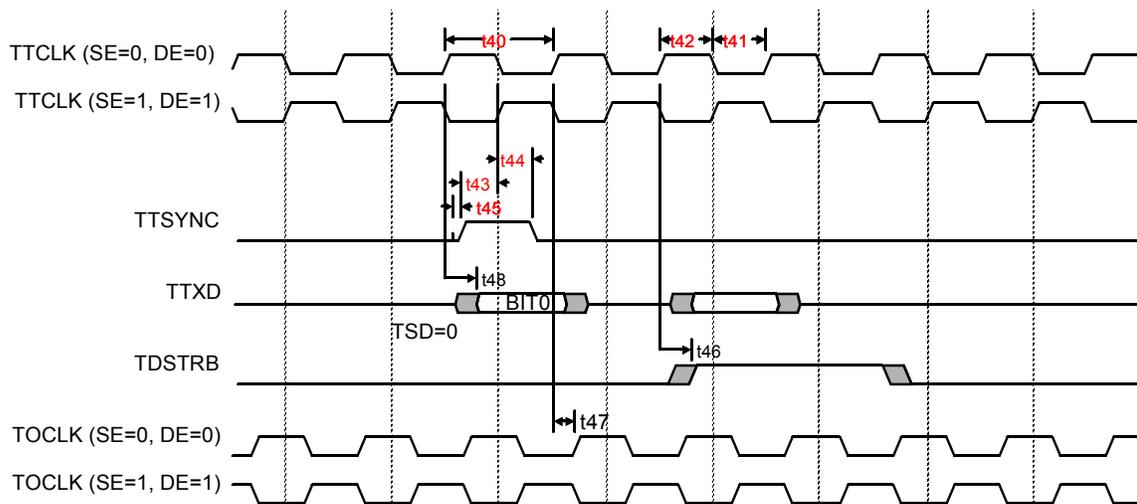


Table 455: Flex-TDM Receive Timing - Double Speed Clock

Symbol	Description	Min	Max	Unit
t30	TRCLK Cycle Time	18		ns
t31	TRCLK Width Low	7.2		ns
t32	TRCLK Width High	7.2		ns
t33	TRSYNC Setup Time	5		ns
t34	TRSYNC Hold Time	5		ns
t35	TRSYNC Rise and Fall Time		15	ns
t37	TRXD Setup Time	5		ns
t38	TRXD Hold Time	5		ns

Figure 87: Flex-TDM Receive Timing - Double Speed Clock Waveform

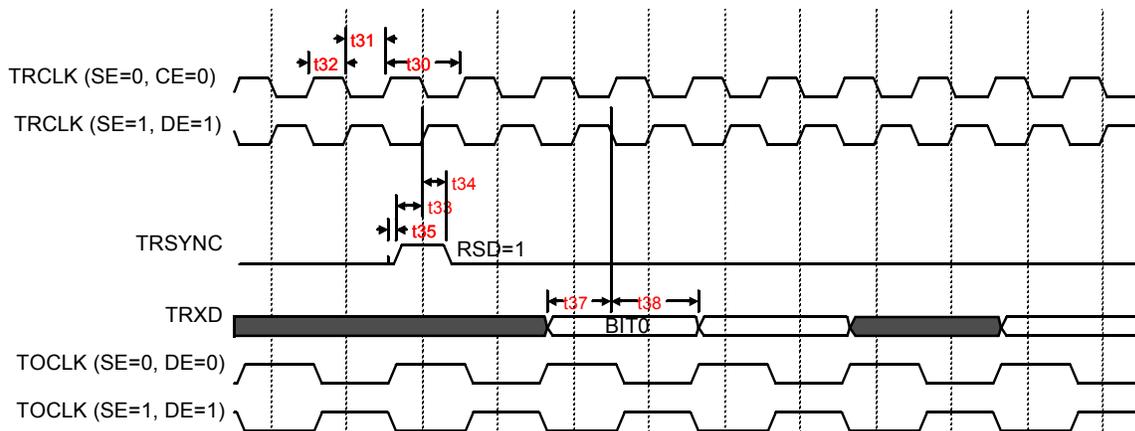


Figure 88: Flex-TDM Receive Timing - Double Speed Clock Waveform

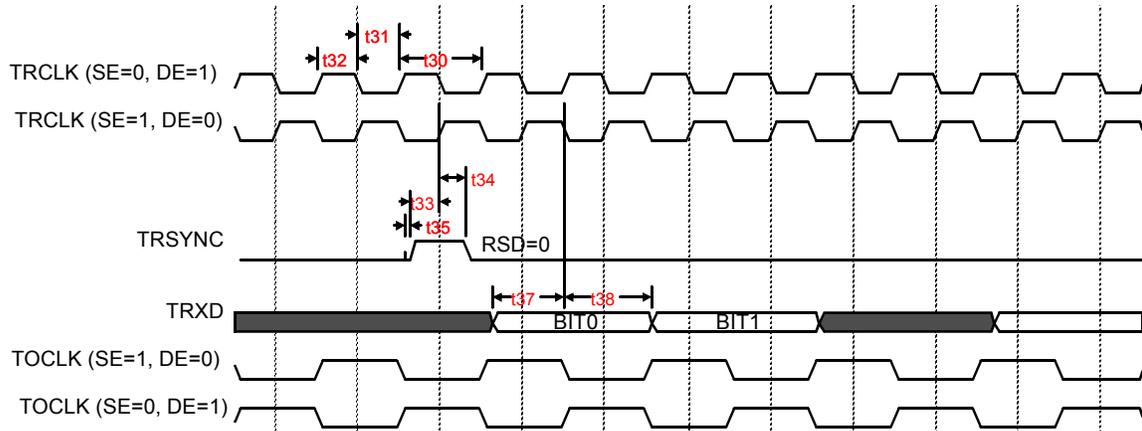


Table 456: Flex-TDM Transmit Timing - Double Speed Clock

Symbol	Description	Min	Max	Unit
t50	TTCLK Cycle Time	18		ns
t51	TTCLK Width Low	7.2		ns
t52	TTCLK Width High	7.2		ns
t53	TTSYNC Setup Time	5		ns
t54	TTSYNC Hold Time	5		ns
t55	TTSYNC Rise and Fall Time		15	ns
t56	TDSTRB Output Delay		13	ns
t57	TTCLK to TDCLK Delay		8	ns
t58	TTXD Output Delay		13	ns

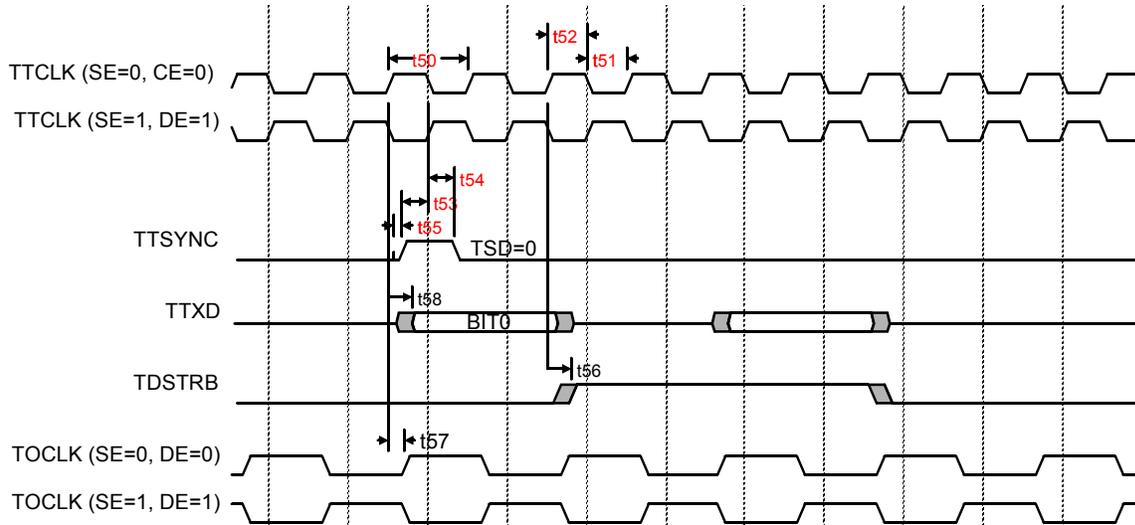
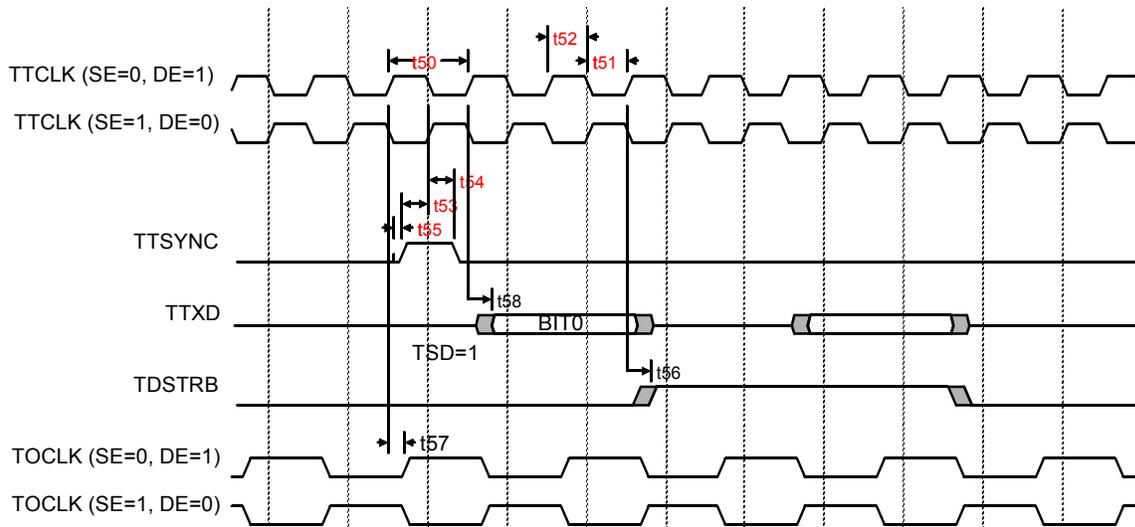
Figure 89: Flex-TDM Transmit Timing - Double Speed Clock Waveform

Figure 90: Flex-TDM Transmit Timing - Double Speed Clock Waveform


Table 457: MPSC Receive Timing

Symbol	Description	Min	Max	Unit
t60	RCLK Cycle Time	18		ns
t61	RCLK Width Low	7.2		ns
t62	RCLK Width High	7.2		ns
t63	RXD Setup Time	5		ns
t64	RXD Hold Time	5		ns
t65	CD* Setup Time	5		ns
t66	CD* Hold Time	5		ns

Figure 91: MPSC Receive Timing

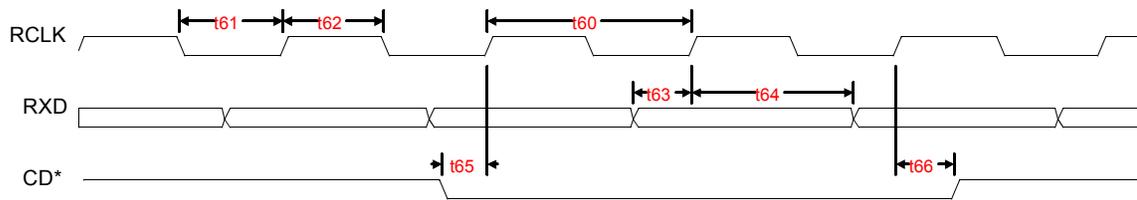
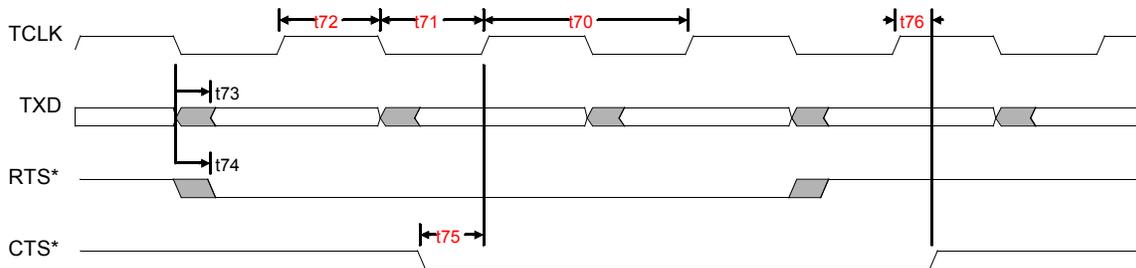


Table 458: MPSC Transmit Timing

Symbol	Description	Min	Max	Unit
t70	TCLK Cycle Time	18		ns
t71	TCLK Width Low	7.2		ns
t72	TCLK Width High	7.2		ns
t73	TXD Delay Time		13	ns
t74	RTS* Delay Time		13	ns
t75	CTS* Setup Time	5		ns
t76	CTS* Hold Time	5		ns

Figure 92: MPSC Transmit Timing



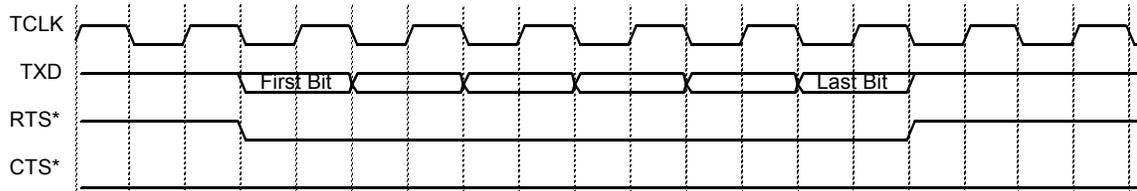
32.3 MPSC Waveforms

32.3.1 Output Delay From RTS*

Figure 93: Output Delay From RTS*, Asynchronous CTS*
(CTSS=0 in MMCRLx) Waveform



Figure 94: Output Delay From RTS*, Synchronous CTS*
(CTSS=1 in MMCRLx) Waveform

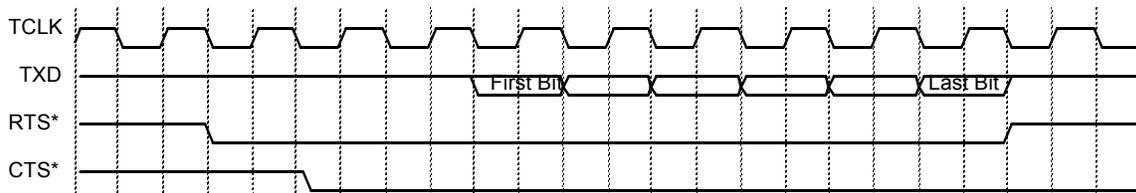


32.3.2 Output Delay From CTS*

Figure 95: Output Delay From CTS*, Asynchronous CTS* (CTSS=0 in MMCRLx) Waveform

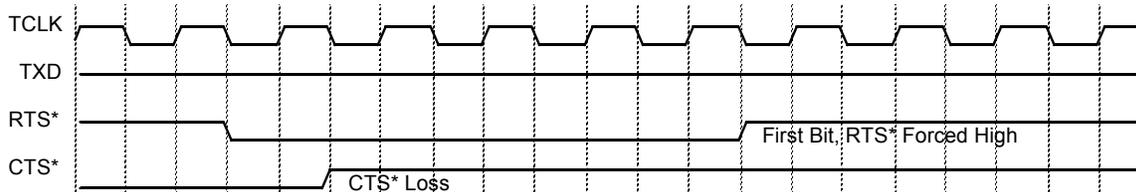


Figure 96: Output Delay From CTS*, Synchronous CTS* (CTSS=1 in MMCRLx) Waveform

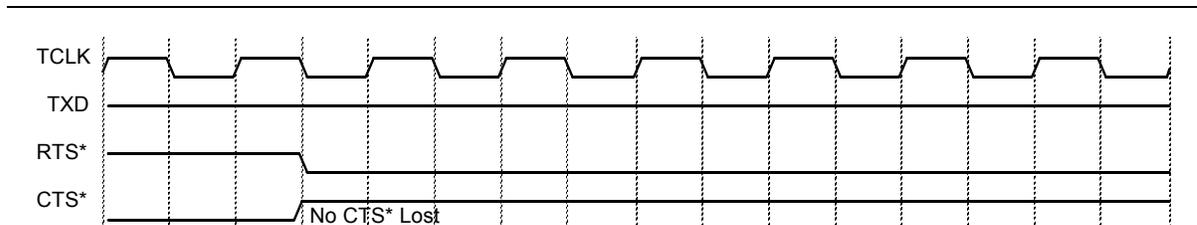


32.3.3 CTS* Loss In Synchronous Protocol

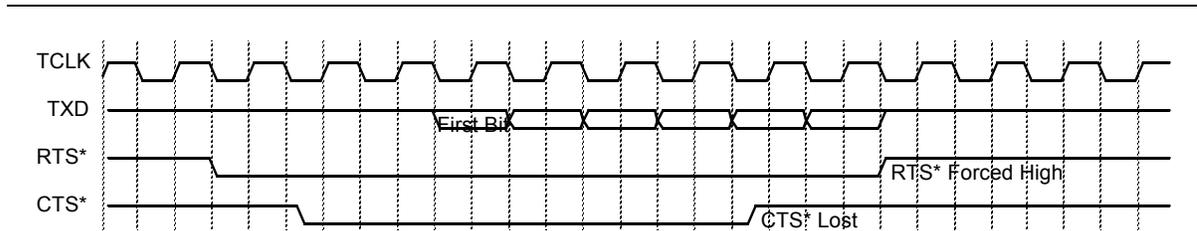
Figure 97: CTS* Loss In Synchronous Protocol: Start of Frame Waveform With CTS Lost



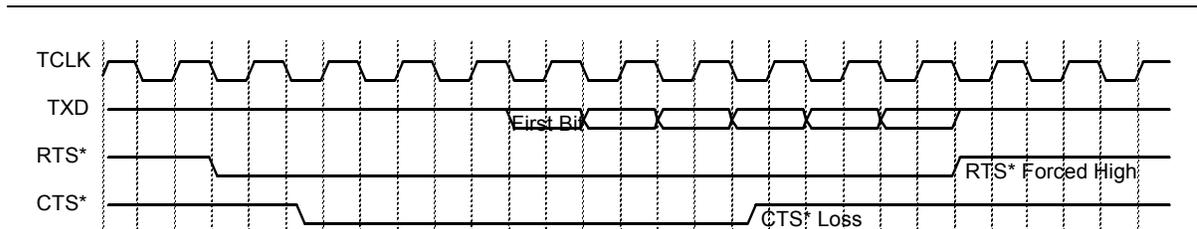
**Figure 98: CTS* Loss In Synchronous Protocol:
Start of Frame Waveform Without CTS Lost**



**Figure 99: CTS* Loss In Synchronous Protocol, Synchronous
CTS* (CTSS=1 in MMCRLx) Waveform**

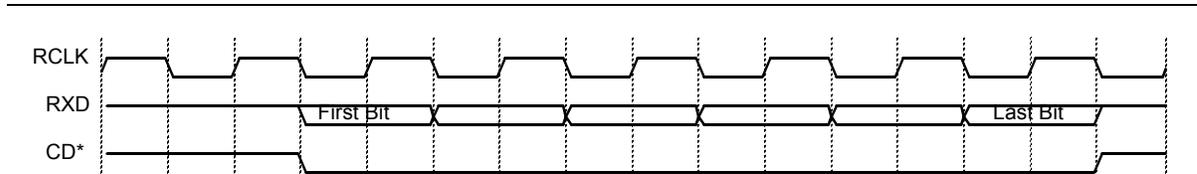


**Figure 100: CTS* Loss In Synchronous Protocol, Asynchronous
CTS* (CTSS=0 in MMCRLx) Waveform**



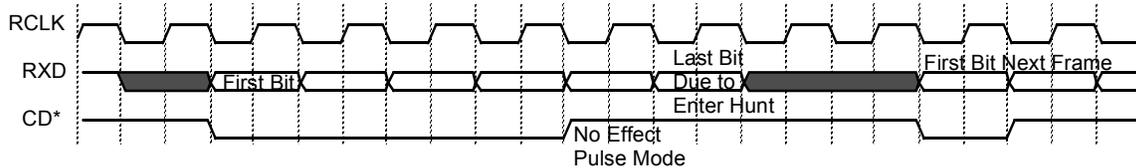
32.3.4 Reception Control Using CD*

Figure 101: Reception Control Using CD* Waveform



32.3.5 External Sync

Figure 102: External Sync (RSYL=0 in MMCRHx), CD* Pulse Mode (CDM=0 in MMCRLx) Waveform



32.3.6 Transmit Synchronize to Receive

Figure 103: Transmit Synchronize to Receive (TSYN=1 in MMCRLx), External Sync (RSYL=0 in MMCRHx) Waveform

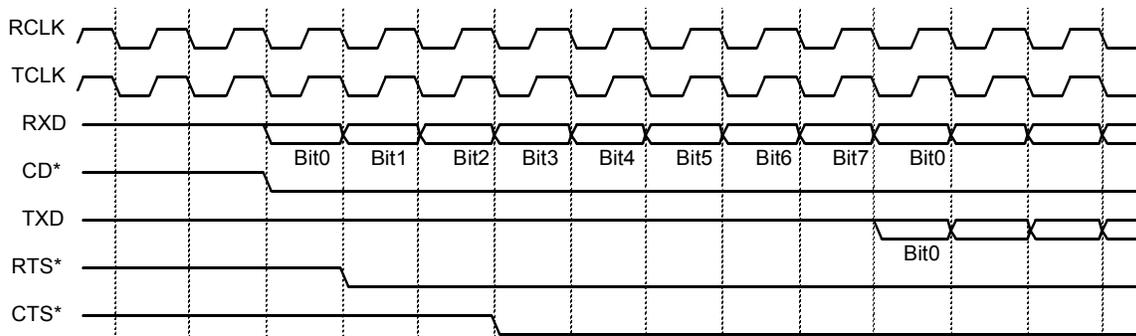
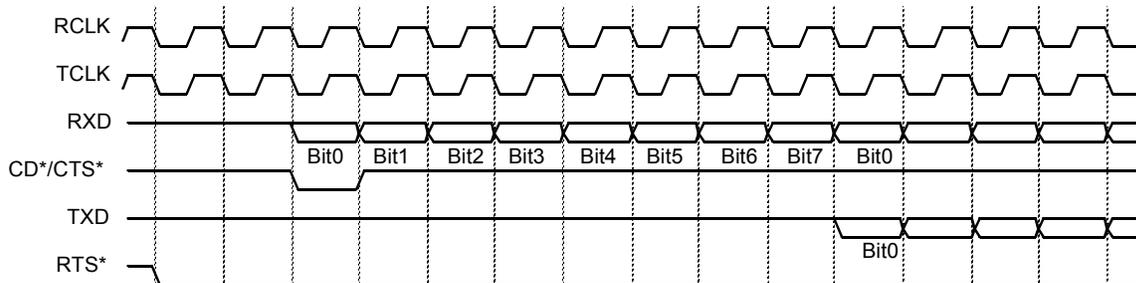


Figure 104: Transmit Synchronize to Receive (TSYN=1 in MMCRLx), External Sync (RSYL=0 in MMCRHx), CD* and CTS* Pulse Mode (CTSM=1 and CDM=1 in MMCRLx), Synchronous CTS* (CTSS=1 in MMCRLx) Waveform

NOTE: CD* and CTS* are connected to the same signal.



32.4 MII Waveforms

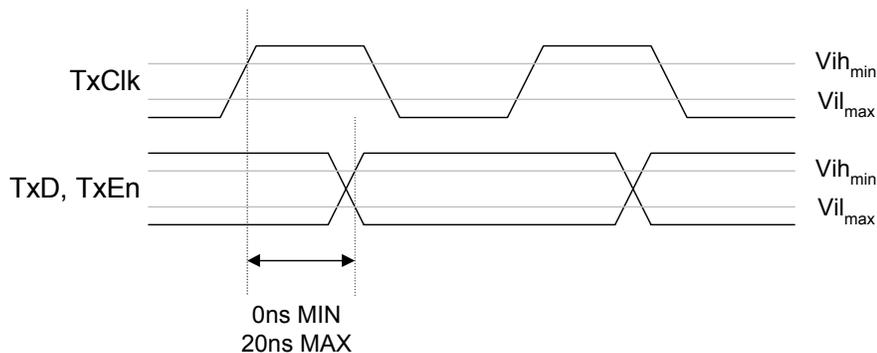
32.4.1 Transmit Timing

Table 459: MII Transmit Timing

Symbol	Description	Min	Typ	Max	Unit
Tmtxcc	MII TxCLK Cycle		40t ¹		ns
Tmtxch	MII TxCLK High	15t		25t	ns
Tmtxcl	MII TxCLK Low	15t		25t	ns
Tmtxv	MII TxCLK rising to TXD, TXEN valid			20	ns
Tmtxh	MII TXD, TXEN hold after TxCLK rising	5			ns

1. t=1 for 100Mb/s operation, t=10 for 10Mb/s operation

Figure 105: MII Port Transmit Signals Timing



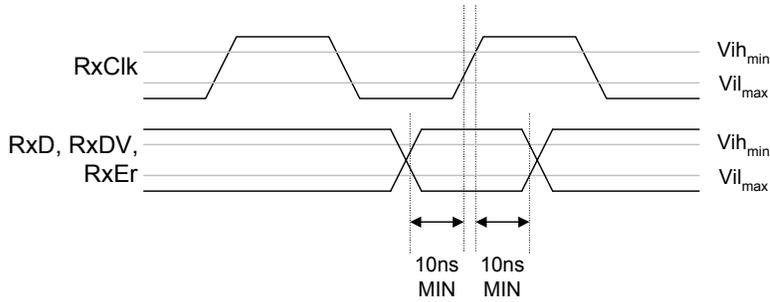
32.4.2 Receive Timing

Table 460: MII Receive Timing

Symbol	Description	Min	Typ	Max	Unit
Tmrxcc	MII RxCLK Cycle		40t ¹		ns
Tmrxch	MII RxCLK High	15t		25t	ns
Tmrxcl	MII RxCLK Low	15t		25t	ns
Tmrxs	MII RXD, RXDV setup before RxCLK rising	10			ns
Tmrxh	MII RXD, RXDV hold after RxCLK rising	5			ns

1. t=1 for 100Mb/s operation, t=10 for 10Mb/s operation

Figure 106:MII Port Receive Signals Timing

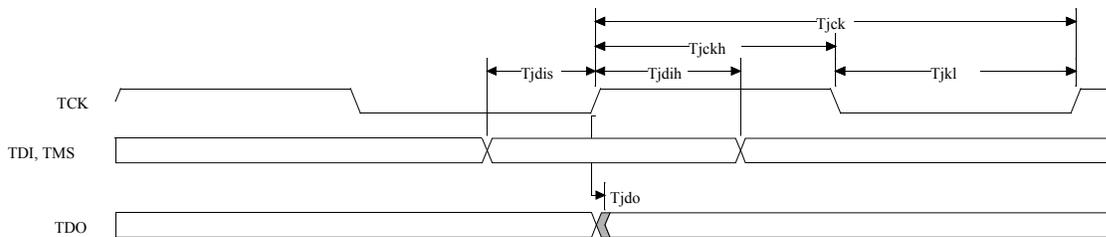


32.5 JTAG AC Characteristics

Table 461: JTAG AC Characteristics

Symbol	Description	Min	Max	Unit
Tjck	Jtag Clock Period	1000		ns
Tjckh	TCK High Period	400		ns
Tjckl	TCK Low Period	400		ns
Tjdis	TDI and TMS Setup Time	20		ns
Tjdih	TDI and TMS Hold Time	20		ns
Tjdo	TDO Output Delay	2	20	ns

Figure 107:JTAG AC Timing



32.6 Additional Delay Due to Capacitive Loading

Some applications may require additional capacitive loading on different output pins of the GT-96100A. For example, when using multiple GT-96100A devices connected to the same SysAD bus, the 50pF load specification may be exceeded. This additional loading affects the output delays of the signals, depending on the drive strength of the output driver.

The following section describes how to calculate the affects of additional loading on the output drivers.

32.6.1 Calculating the Maximum Delay due to Loading

The basic equation for calculating the maximum delay is:

$$T_{max} = [A_{typa} + (B_{typ} * C_r)] * 1.6$$

where:

- T_{max} is the maximum delay in nanoseconds.
- A_{typa} must be calculated by the designer as shown in [Section 32.6.1.1 “Calculating \$A_{typa}\$ ” on page 531](#).
- B_{typ} is a parameter according to the specific output buffer from [Table 462](#).
- C_r is the capacitance required, in picofarads.

32.6.1.1 Calculating A_{typa}

A_{typa} can be calculated by using the given values in the AC Timing Parameters table. We start with the equation:

$$T_{spec} = [A_{typa} + (B_{typ} * C_{ds})] * 1.6$$

and solving for A_{typa} :

$$A_{typa} = (T_{spec}/1.6) - (B_{typ} * C_{ds})$$

where:

- T_{spec} is the maximum delay parameter from the AC Timing Parameters Table, in nanoseconds.
- B_{typ} is a parameter according to the specific output buffer from [Table 462](#).
- C_{ds} is the capacitance parameter from the AC Timing Parameters Table, in picofarads.

NOTE: 1.6 is the worst case derating factor.

32.6.2 Calculating the Minimum Delay due to Loading

The basic equation for calculating the maximum delay is:

$$T_{min} = [A_{typb} + (B_{typ} * C_r)] * 0.7$$

where:

- T_{min} is the minimum delay in nanoseconds
- A_{typb} must be calculated by the designer as shown in [Section 32.6.2.1 “Calculating \$A_{typb}\$ ” on page 532](#)
- B_{typ} is a parameter according to the specific output buffer from [Table 462](#)
- C_r is the capacitance required, in picofarads.

32.6.2.1 Calculating Atpb

Atpb can be calculated by using the given values in the AC Timing Parameters table. We start with the equation:

$$T_{spec} = [Atpb + (Btyp * Cds)] * 0.7$$

and solving for Atpb:

$$Atpb = (T_{spec}/0.7) - (Btyp * Cds)$$

where:

- Tspec is the maximum delay parameter from the AC Timing Parameters Table, in nanoseconds
- Btyp is a parameter according to the specific output buffer from [Table 462](#)
- Cds is the capacitance parameter from the AC Timing Parameters Table, in picofarads.

NOTE: 0.7 is the worst case derating factor.

32.6.3 Btyp Values

[Table 462](#) lists the Btyp values for the different output buffers of the GT-96100A. See the DC Parameters Section for the corresponding pin and output driver.

Table 462: Btyp Values

Output Driver	Low to High Btyp Value	High to Low Btype Value
4mA	0.06	0.077
8mA	0.031	0.039
12mA	0.021	0.028
16mA	0.018	0.022
All PCI Outputs	0.015	0.018

32.6.4 Tmax Calculating Example

The following is an example of how to calculate the maximum delay on the AD[0] line for a 75pF load.

From the AC Timing Parameters Table, for a 50pF load, the maximum output delay on the AD[0] line is specified as 7ns for 83 Mhz. Looking at [Table 462](#), Btyp for AD[0], which is an 8mA driver, is = 0.031 (Low to High transition).

Substituting these values into:

$$Atpa = (T_{spec}/1.6) - (Btyp * Cds)$$

gives Atpa = 2.83.

Substituting Atpa of 3.45, Btyp of 0.031 and Cr of 75pF in:

$$T_{max} = [Atpa + (Btyp * Cr)] * 1.6$$

gives Tmax = 8.25. This means that the maximum output delay of AD[0] with a 75pF load is 8.25ns.

33. PINOUT TABLE, 492 PIN BGA

NOTE: The following table is sorted by ball number.

Table 463: GT-96100A Pinout Table

Ball #	Signal Name				
A01–A26		B01–B26		C01–C26	
A01	Req0_	B01	PAD0[28]	C01	PAD0[24]
A02	GPP[14]	B02	JTAG[3]	C02	PAD0[30]
A03	GPP[13]	B03	JTAG[0]	C03	GNT0_
A04	GPP[6]	B4	VssPLL	C04	JTAG[1]
A05	VccPLL	B05	GPP[7]	C05	GPP[12]
A06	MII1[14]	B06	GPP[2]	C06	GPP[9]
A07	MII1[10]	B07	MDC	C07	GPP[3]
A08	MII1[4]	B08	MII1[13]	C08	GPP[0]
A09	MII1[3]	B09	MII1[9]	C09	MII1[12]
A10	NC	B10	MII1[5]	C10	MII1[6]
A11	MII0[12]	B11	MII1[1]	C11	MII1[8]
A12	MII0[11]	B12	MII0[13]	C12	NC
A13	MII0[8]	B13	MII0[9]	C13	MII0[6]
A14	MII0[2]	B14	MII0[5]	C14	MII0[4]
A15	MII0[1]	B15	NC	C15	PORTF[5]
A16	PORTF[6]	B16	PORTF[3]	C16	PORTF[1]
A17	PORTF[4]	B17	PORTE[6]	C17	PORTE[3]
A18	PORTF[2]	B18	PORTE[4]	C18	PORTD[6]
A19	PORTE[5]	B19	PORTD[5]	C19	NC
A20	PORTE[1]	B20	PORTD[0]	C20	PORTC[5]
A21	PORTD[4]	B21	PORTC[6]	C21	PORTC[1]
A22	NC	B22	PORTC[0]	C22	PORTB[4]
A23	PORTC[3]	B23	PORTB[0]	C23	PORTA[3]
A24	PORTB[1]	B24	PORTA[2]	C24	DMAReq[0]_
A25	PORTA[4]	B25	Ready_	C25	AD[0]
A26	PORTA[0]	B26	AD[33]	C26	AD[2]

Table 463: GT-96100A Pinout Table (Continued)

Ball #	Signal Name				
<i>D01–D26</i>		<i>E01–E26</i>		<i>F01–10, F17–F26</i>	
D01	PAD0[17]	E01	Irdy0_	F01	PErr0_
D02	PAD0[25]	E02	PAD0[20]	F02	Frame0_
D03	PAD0[29]	E03	PAD0[23]	F03	PAD0[18]
D04	JTAG[2]	E04	PAD0[26]	F04	PAD0[22]
D05	JTAG[4]	E05	PAD0[31]	F05	PAD0[27]
D06	GPP[8]	E06	GPP[15]	F06	GND
D07	GPP[5]	E07	GPP[10]	F07	GND
D08	GPP[1]	E08	GPP[11]	F08	VCC 2.5
D09	MDIO	E09	GPP[4]	F09	VCC 3.3
D10	MII1[11]	E10	NC	F10	VCC 3.3
D11	MII1[2]	E11	MII1[7]	F17	VCC 3.3
D12	MII0[14]	E12	MII1[0]	F18	VCC 2.5
D13	MII0[7]	E13	MII0[10]	F19	VCC 2.5
D14	MII0[0]	E14	MII0[3]	F20	GND
D15	PORTF[0]	E15	NC	F21	GND
D16	PORTE[2]	E16	PORTD[2]	F22	AD[1]
D17	PORTE[0]	E17	PORTD[3]	F23	AD[36]
D18	PORTD[1]	E18	PORTC[2]	F24	AD[4]
D19	PORTC[4]	E19	PORTB[2]	F25	AD[39]
D20	PORTB[6]	E20	PORTB[3]	F26	AD[11]
D21	PORTB[5]	E21	PORTA[5]	G01–G06	
D22	PORTA[6]	E22	PORTA[1]	G01	PAD0[14]
D23	ALE	E23	BypsOE_	G02	DevSel0_
D24	CSTiming_	E24	AD[34]	G03	CBE0_[2]
D25	AD[32]	E25	AD[5]	G04	PAD0[21]
D26	AD[37]	E26	AD[9]	G05	IDSel0
				G06	GND

Table 463: GT-96100A Pinout Table (Continued)

Ball #	Signal Name				
G21–G26		J21–J26		L11–L16, L22–L26	
G21	GND	J21	VCC 3.3	L11	GND
G22	AD[35]	J22	AD[38]	L12	GND
G23	AD[6]	J23	AD[10]	L13	GND
G24	AD[7]	J24	AD[12]	L14	GND
G25	AD[42]	J25	AD[45]	L15	GND
G26	AD[13]	J26	ADP[4]	L16	GND
H01–H06, H21–H26		K01–K06, K21–K26		L22	AD[46]
H01	PAD0[10]	K01	PAD0[6]	L23	ADP[1]
H02	Par0	K02	PAD0[9]	L24	AD[15]
H03	Trdy0_	K03	PAD0[13]	L25	ADP[5]
H04	PAD0[16]	K04	CBE0_[1]	L26	SDQM[5]
H05	CBE0_[3]	K05	Lock0_	M01–M05, M11–M16, M22–M26	
H06	VCC 2.5	K06	VCC 3.3	M01	PAD0[1]
H21	VCC 2.5	K21	VCC 3.3	M02	PAD0[2]
H22	AD[3]	K22	AD[41]	M03	PAD0[5]
H23	AD[8]	K23	AD[44]	M04	CBE0_[0]
H24	AD[40]	K24	AD[14]	M05	PAD0[3]
H25	AD[43]	K25	AD[47]	M11	GND
H26	ADP[0]	K26	SDQM[0]	M12	GND
J01–J06		J01–J05		M13	GND
J01	PAD0[8]	L01	PAD0[4]	M14	GND
J02	PAD0[12]	L02	PAD0[7]	M15	GND
J03	PAD0[15]	L03	VREF0	M16	GND
J04	Stop0_	L04	PAD0[11]	M22	DWr_
J05	PAD0[19]	L05	SErr0_	M23	SDQM[4]
J06	VCC 2.5			M24	SCAS_
				M25	SDQM[1]
				M26	SCS_[0]

Table 463: GT-96100A Pinout Table (Continued)

Ball #	Signal Name				
N01–N05, N11–N16, N22–N26		P22–P26		T11–T16, T22–T26	
N01	PAD0[0]	P22	DAdr[4]	T11	GND
N02	TCIk	P23	DAdr[7]	T12	GND
N03	NC	P24	DAdr[3]	T13	GND
N04	PCIk0	P25	DAdr[2]	T14	GND
N05	NC	P26	DAdr[5]	T15	GND
N11	GND	R01–R05, R11–R16, R22–R26		T16	GND
N12	GND	R01	PAD1[30]	T22	AD[17]
N13	GND	R02	PAD1[25]	T23	SDQM[7]
N14	GND	R03	CBE1_[3]	T24	DMAReq_[1]
N15	GND	R04	PAD1[24]	T25	NC
N16	GND	R05	PAD1[27]	T26	DAdr[10]
N22	SCS_[1]	R11	GND	U01–U06, U21–U26	
N23	DAdr[0]	R12	GND	U01	PAD1[26]
N24	DAdr[1]	R13	GND	U02	PAD1[22]
N25	SRAS_	R14	GND	U03	PAD1[23]
N26	NC	R15	GND	U04	PAD1[19]
P01–P05, P11–P16		R16	GND	U05	PErr1_
P01	PAD1[28]	R22	DAdr[9]	U06	VCC 3.3
P02	PCIk1	R23	SCS_[2]	U21	VCC 3.3
P03	ClkOutPLL	R24	DMAReq_[2]	U22	AD[48]
P04	NC	R25	DAdr[8]	U23	ADP[6]
P05	PAD1[29]	R26	DAdr[6]	U24	SDQM[3]
P11	GND	T01–T05		U25	SCS_[3]
P12	GND	T01	PAD1[31]	U26	DMAReq_[3]
P13	GND	T02	VREF1		
P14	GND	T03	PAD1[17]		
P15	GND	T04	IDsel1		
P16	GND	T05	PAD1[16]		

Table 463: GT-96100A Pinout Table (Continued)

Ball #	Signal Name				
V01–V06, V21–V26		Y01–Y06, Y21–Y26		AA23–AA26	
V01	PAD1[20]	Y01	CBE1_[2]	AA23	AD[55]
V02	PAD1[18]	Y02	DevSel1_	AA24	AD[22]
V03	Frame1_	Y03	PAD1[13]	AA25	AD[51]
V04	Stop1_	Y04	PAD1[14]	AA26	AD[16]
V05	PAD1[15]	Y05	PAD1[4]	AB01–AB24	
V06	VCC 3.3	Y06	GND	AB01	SErr1_
V21	VCC 2.5	Y21	GND	AB02	PAD1[9]
V22	AD[53]	Y22	AD[56]	AB03	PAD1[5]
V23	AD[49]	Y23	AD[23]	AB04	SysAD[4]
V24	ADP[3]	Y24	AD[20]	AB05	SysAD[46]
V25	SDQM[6]	Y25	AD[18]	AB06	SysAD[45]
V26	BankSel[0]	Y26	ADP[2]	AB07	SysAD[12]
W01–W06, W21–W26		AA01–AA10, AA17–AA22		AB08	SysAD[33]
W01	PAD1[21]	AA01	Trdy1_	AB09	SysADC[1]
W02	Irdy1_	AA02	PAD1[12]	AB10	NC
W03	Par1	AA03	PAD1[11]	AB11	SysAD[62]
W04	CBE1_[1]	AA04	PAD1[10]	AB12	SysAD[52]
W05	CBE1_[0]	AA05	PAD1[0]	AB13	NC
W06	VCC 2.5	AA06	GND	AB14	NC
W21	VCC 2.5	AA07	GND	AB15	SysAD[20]
W22	AD[25]	AA08	VCC 2.5	AB16	SysAD[26]
W23	AD[52]	AA09	VCC 2.5	AB17	SysCMD[1]
W24	AD[50]	AA10	VCC 3.3	AB18	ScDOE_
W25	ADP[7]	AA17	VCC 3.3	AB19	Interrupt1_
W26	SDQM[2]	AA18	VCC 3.3	AB20	Interrupt0_
		AA19	VCC 2.5	AB21	ByPassPLL
		AA20	GND	AB22	AD[29]
		AA21	GND	AB23	AD[58]
		AA22	AD[27]	AB24	AD[24]

Table 463: GT-96100A Pinout Table (Continued)

Ball #	Signal Name				
AB25–AB26		AD01–AD26		AE01–AE26	
AB25	AD[54]	AD01	PAD1[6]	AE01	PAD1[7]
AB26	AD[19]	AD02	PAD1[2]	AE02	Req1_
AC01–AC26		AD03	SysAD[40]	AE03	SysAD[10]
AC01	PAD1[8]	AD04	SysAD[37]	AE04	SysAD[13]
AC02	PAD1[1]	AD05	SysAD[41]	AE05	SysAD[11]
AC03	PAD1[3]	AD06	SysAD[15]	AE06	SysAD[9]
AC04	Gnt1_	AD07	SysAD[38]	AE07	SysAD[35]
AC05	SysAD[8]	AD08	SysAD[34]	AE08	SysAD[3]
AC06	SysAD[39]	AD09	SysAD[0]	AE09	SysAD[1]
AC07	SysAD[14]	AD10	SysADC[3]	AE10	SysADC[0]
AC08	SysAD[44]	AD11	SysADC[6]	AE11	SysADC[2]
AC09	SysADC[5]	AD12	SysAD[49]	AE12	SysAD[30]
AC10	SysADC[7]	AD13	SysAD[16]	AE13	SysAD[29]
AC11	SysAD[63]	AD14	SysAD[17]	AE14	SysAD[28]
AC12	SysAD[51]	AD15	SysAD[24]	AE15	SysAD[25]
AC13	SysAD[19]	AD16	SysAD[22]	AE16	SysAD[21]
AC14	SysAD[18]	AD17	SysAD[23]	AE17	SysAD[53]
AC15	SysAD[50]	AD18	SysCMD[6]	AE18	SysAD[56]
AC16	SysAD[57]	AD19	ValidOut_	AE19	SysCMD[5]
AC17	SysCMD[7]	AD20	Release_	AE20	SysCMD[3]
AC18	SysCMD[2]	AD21	NC	AE21	WrRdy_
AC19	ValidIn_	AD22	SerInt0_	AE22	ScWord[1]
AC20	ScMatch	AD23	NMI_	AE23	Reset_
AC21	ScWord[0]	AD24	AD[30]	AE24	WDE_
AC22	AD[62]	AD25	AD[60]	AE25	AD[31]
AC23	AD[63]	AD26	AD[57]	AE26	AD[59]
AC24	AD[28]				
AC25	AD[26]				
AC26	AD[21]				

Table 463: GT-96100A Pinout Table (Continued)

Ball #	Signal Name	
AF01–AF26		
AF01	SysAD[36]	
AF02	SysAD[7]	
AF03	SysAD[43]	
AF04	SysAD[6]	
AF05	SysAD[5]	
AF06	SysAD[42]	
AF07	SysAD[47]	
AF08	SysAD[2]	
AF09	SysAD[32]	
AF10	SysADC[4]	
AF11	SysAD[31]	
AF12	SysAD[48]	
AF13	SysAD[60]	
AF14	SysAD[61]	
AF15	SysAD[54]	
AF16	SysAD[27]	
AF17	SysAD[55]	
AF18	SysAD[59]	
AF19	SysAD[58]	
AF20	SysCMD[8]	
AF21	SysCMD[4]	
AF22	SysCMD[0]	
AF23	ScTCE_	
AF24	SerInt1_	
AF25	OutModePLL	
AF26	AD[61]	

NOTE: All NC pins must be connected to ground for future devices compatibility. In order to use these pins in a future device, Galileo Technology recommends that these pins be connected by zero ohm resistors.

Figure 108:GT-96100A Pinout Map (top view, left side)

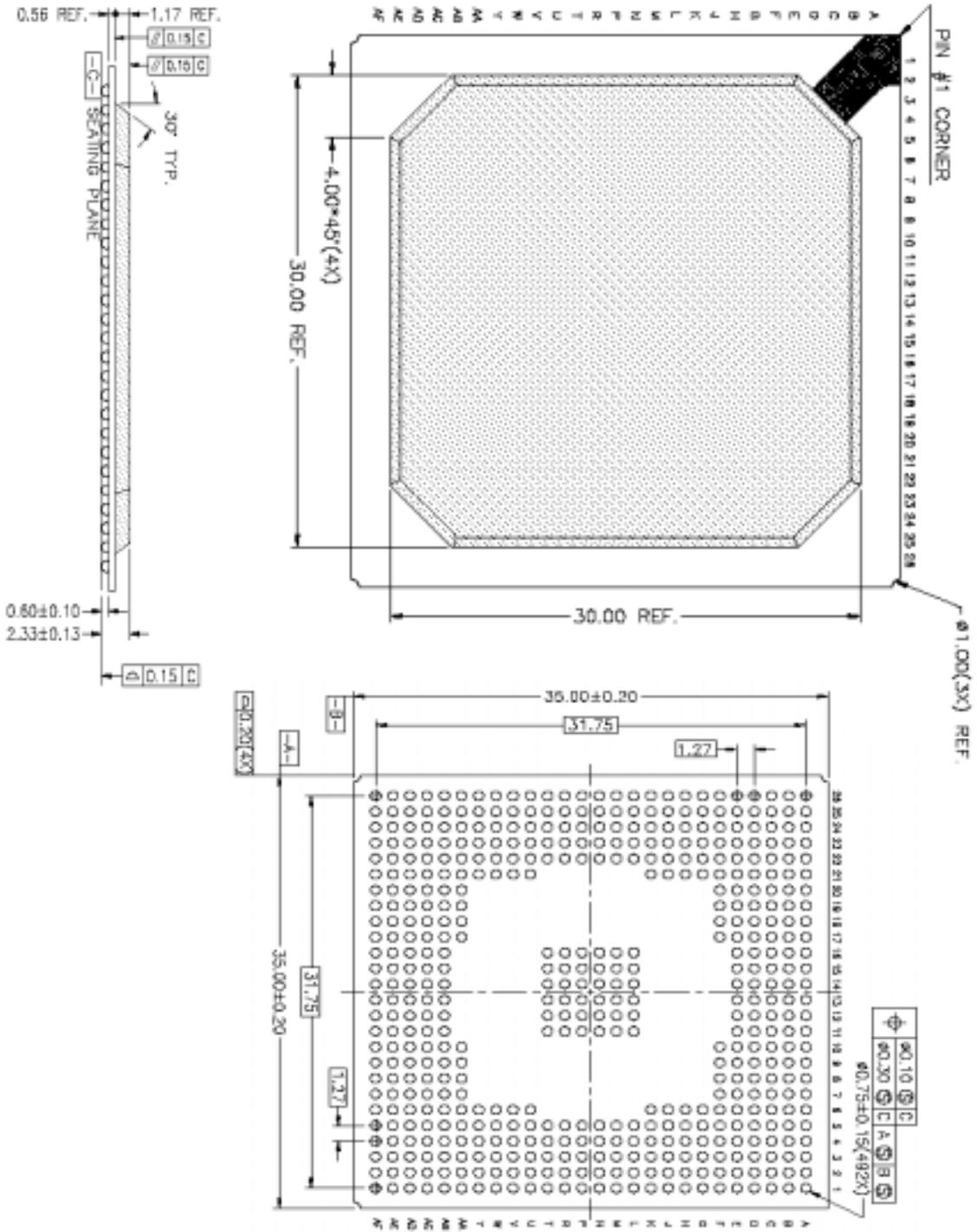
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	Req0_	GPP[14]	GPP[13]	GPP[6]	VccPLL	MII1[14]	MII1[10]	MII1[4]	MII1[3]	NC	MII0[12]	MII0[11]	MII0[8]	MII0[2]
B	PAD0[28]	JTAG[3]	JTAG[0]	VssPLL	GPP[7]	GPP[2]	MDC	MII1[13]	MII1[9]	MII1[5]	MII1[1]	MII0[13]	MII0[9]	MII0[5]
C	PAD0[24]	PAD0[30]	Gnt0_	JTAG[1]	GPP[12]	GPP[9]	GPP[3]	GPP[0]	MII1[12]	MII1[6]	MII1[8]	NC	MII0[6]	MII0[4]
D	PAD0[17]	PAD0[25]	PAD0[29]	JTAG[2]	JTAG[4]	GPP[8]	GPP[5]	GPP[1]	MDIO	MII1[11]	MII1[2]	MII0[14]	MII0[7]	MII0[0]
E	Irdy0_	PAD0[20]	PAD0[23]	PAD0[26]	PAD0[31]	GPP[15]	GPP[10]	GPP[11]	GPP[4]	NC	MII1[7]	MII1[0]	MII0[10]	MII0[3]
F	PErr0_	Frame0_	PAD0[18]	PAD0[22]	PAD0[27]	VSS	VSS	VCC2.5	VCC3.3	VCC3.3				
G	PAD0[14]	DevSel0_	CBE0_[2]	PAD0[21]	IDSel0	VSS								
H	PAD0[10]	Par0	Trdy0_	PAD0[16]	CBE0_[3]	VCC2.5								
J	PAD0[8]	PAD0[12]	PAD0[15]	Stop0_	PAD0[19]	VCC2.5								
K	PAD0[6]	PAD0[9]	PAD0[13]	CBE0_[1]	Lock0_	VCC3.3								
L	PAD0[4]	PAD0[7]	VREF0	PAD0[11]	SErr0_						VSS	VSS	VSS	VSS
M	PAD0[1]	PAD0[2]	PAD0[5]	CBE0_[0]	PAD0[3]						VSS	VSS	VSS	VSS
N	PAD0[0]	TCIk	NC	PCIk0	NC						VSS	VSS	VSS	VSS
P	PAD1[28]	PCIk1	ClkOutPLL	NC	PAD1[29]						VSS	VSS	VSS	VSS
R	PAD1[30]	PAD1[25]	CBE1_[3]	PAD1[24]	PAD1[27]						VSS	VSS	VSS	VSS
T	PAD1[31]	VREF1	PAD1[17]	IDSel1	PAD1[16]						VSS	VSS	VSS	VSS
U	PAD1[26]	PAD1[22]	PAD1[23]	PAD1[19]	PErr1_	VCC3.3								
V	PAD1[20]	PAD1[18]	Frame1_	Stop1_	PAD1[15]	VCC3.3								
W	PAD1[21]	Irdy1_	Par1	CBE1_[1]	CBE1_[0]	VCC2.5								
Y	CBE1_[2]	DevSel1_	PAD1[13]	PAD1[14]	PAD1[4]	VSS								
AA	Trdy1_	PAD1[12]	PAD1[11]	PAD1[10]	PAD1[0]	VSS	VSS	VCC2.5	VCC2.5	VCC3.3				
AB	SErr1_	PAD1[9]	PAD1[5]	SysAD[4]	SysAD[46]	SysAD[45]	SysAD[12]	SysAD[33]	SysADC[1]	NC	SysAD[62]	SysAD[52]	NC	NC
AC	PAD1[8]	PAD1[1]	PAD1[3]	Gnt1_	SysAD[8]	SysAD[39]	SysAD[14]	SysAD[44]	SysADC[5]	SysADC[7]	SysAD[63]	SysAD[51]	SysAD[19]	SysAD[18]
AD	PAD1[6]	PAD1[2]	SysAD[40]	SysAD[37]	SysAD[41]	SysAD[15]	SysAD[38]	SysAD[34]	SysAD[0]	SysADC[3]	SysADC[6]	SysAD[49]	SysAD[16]	SysAD[17]
AE	PAD1[7]	Req1_	SysAD[10]	SysAD[13]	SysAD[11]	SysAD[9]	SysAD[35]	SysAD[3]	SysAD[1]	SysADC[0]	SysADC[2]	SysAD[30]	SysAD[29]	SysAD[28]
AF	SysAD[36]	SysAD[7]	SysAD[43]	SysAD[6]	SysAD[5]	SysAD[42]	SysAD[47]	SysAD[2]	SysAD[32]	SysADC[4]	SysAD[31]	SysAD[48]	SysAD[60]	SysAD[61]

Figure 109:GT-96100A Pinout Map (top view, right side)

15	16	17	18	19	20	21	22	23	24	25	26	
MI0[1]	PORTF[6]	PORTF[4]	PORTF[2]	PORTE[5]	PORTE[1]	PORTD[4]	NC	PORTC[3]	PORTB[1]	PORTA[4]	PORTA[0]	A
NC	PORTF[3]	PORTE[6]	PORTE[4]	PORTD[5]	PORTD[0]	PORTC[6]	PORTC[0]	PORTB[0]	PORTA[2]	Ready_	AD[33]	B
PORTF[5]	PORTF[1]	PORTE[3]	PORTD[6]	NC	PORTC[5]	PORTC[1]	PORTB[4]	PORTA[3]	DMAReq[0]_	AD[0]	AD[2]	C
PORTF[0]	PORTE[2]	PORTE[0]	PORTD[1]	PORTC[4]	PORTB[6]	PORTB[5]	PORTA[6]	ALE	CSTiming_	AD[32]	AD[37]	D
NC	PORTD[2]	PORTD[3]	PORTC[2]	PORTB[2]	PORTB[3]	PORTA[5]	PORTA[1]	BypsOE_	AD[34]	AD[5]	AD[9]	E
		VDD3.3	VDD2.5	VDD2.5	VSS	VSS	AD[1]	AD[36]	AD[4]	AD[39]	AD[11]	F
						VSS	AD[35]	AD[6]	AD[7]	AD[42]	AD[13]	G
						VDD2.5	AD[3]	AD[8]	AD[40]	AD[43]	ADP[0]	H
						VDD3.3	AD[38]	AD[10]	AD[12]	AD[45]	ADP[4]	J
						VDD3.3	AD[41]	AD[44]	AD[14]	AD[47]	SDQM[0]	K
VSS	VSS						AD[46]	ADP[1]	AD[15]	ADP[5]	SDQM[5]	L
VSS	VSS						DWr_	SDQM[4]	SCAS_	SDQM[1]	SCS_[0]	M
VSS	VSS						SCS_[1]	DAdr[0]	DAdr[1]	SRAS_	NC	N
VSS	VSS						DAdr[4]	DAdr[7]	DAdr[3]	DAdr[2]	DAdr[5]	P
VSS	VSS						DAdr[9]	SCS_[2]	DMAReq_[2]	DAdr[8]	DAdr[6]	R
VSS	VSS						AD[17]	SDQM[7]	DMAReq_[1]	NC	DAdr[10]	T
		VDD3.3	VDD3.3	VDD2.5	VSS	VSS	AD[48]	ADP[6]	SDQM[3]	SCS_[3]	DMAReq_[3]	U
		VDD2.5	VDD2.5	VDD2.5	VSS	VSS	AD[53]	AD[49]	ADP[3]	SDQM[6]	BankSel[0]	V
		VDD2.5	VDD2.5	VDD2.5	VSS	VSS	AD[25]	AD[52]	AD[50]	ADP[7]	SDQM[2]	W
		VSS	VSS	VSS	VSS	VSS	AD[56]	AD[23]	AD[20]	AD[18]	ADP[2]	Y
		VDD3.3	VDD3.3	VDD2.5	VSS	VSS	AD[27]	AD[55]	AD[22]	AD[51]	AD[16]	AA
SysAD[20]	SysAD[26]	SysCMD[1]	ScDOE_	Interrupt1_	Interrupt0_	ByPassPLL	AD[29]	AD[58]	AD[24]	AD[54]	AD[19]	AB
SysAD[50]	SysAD[57]	SysCMD[7]	SysCMD[2]	ValidIn_	ScMatch	ScWord[0]	AD[62]	AD[63]	AD[28]	AD[26]	AD[21]	AC
SysAD[24]	SysAD[22]	SysAD[23]	SysCMD[6]	ValidOut_	Release_	NC	SerInt0_	NMI_	AD[30]	AD[60]	AD[57]	AD
SysAD[25]	SysAD[21]	SysAD[53]	SysAD[56]	SysCMD[5]	SysCMD[3]	WrRdy_	ScWord[1]	Reset_	WDE_	AD[31]	AD[59]	AE
SysAD[54]	SysAD[27]	SysAD[55]	SysAD[59]	SysAD[58]	SysCMD[8]	SysCMD[4]	SysCMD[0]	ScTCE_	SerInt1_	OutModePLL	AD[61]	AF

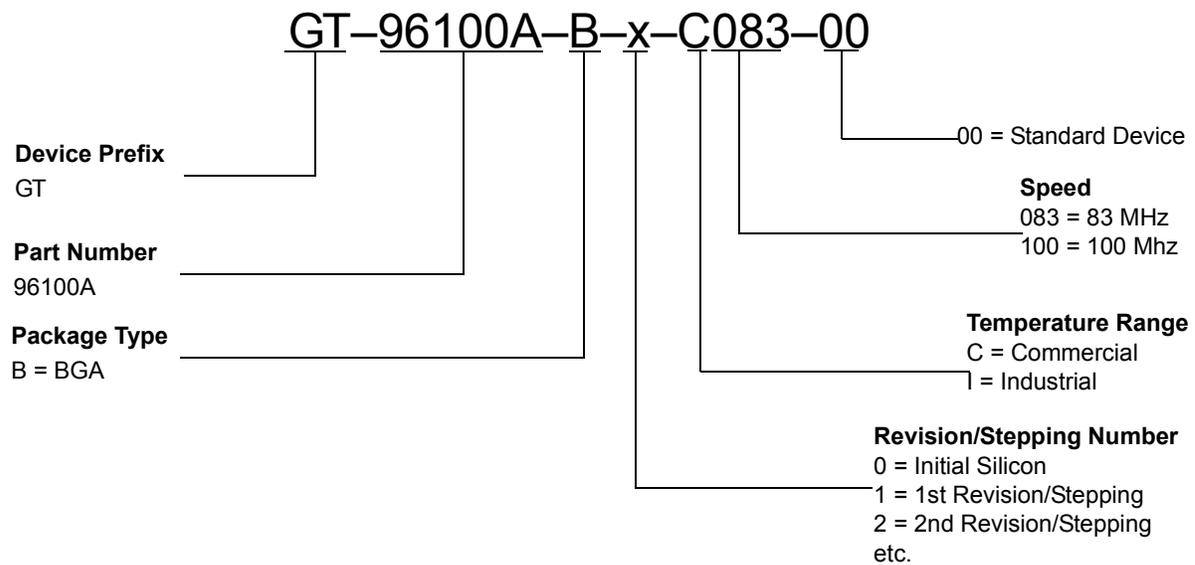
34. 492 BGA PACKAGE MECHANICAL INFORMATION

Figure 110:492 BGA Package Mechanical Information



35. GT-96100A PART NUMBERING

Figure 111: Sample Part Number



35.1 Standard Part Number

The standard part number for the GT-96100A is: GT-96100A-B-x.

Without the -YYYY-ZZ suffix, this part number indicates that it is the commercial temperature grade, 100MHz version. In other words, GT-96100A-B-x is the same as GT-96100A-B-x-C100-00, although it is not marked with the suffix information.

35.2 Valid Part Numbers

The following part numbers are the only valid part numbers that can be used when ordering the GT-96100A:

- GT-96100A-B-x, Commercial Temperature, 100MHz
- GT-96100A-B-x-C083-00, Commercial Temperature, 83MHz

36. ABBREVIATIONS

b	bit
B	byte
Gbps	gigabits per second
KHz	kilohertz
mA	milliampere
MHz	megahertz
ns	nanosecond
V	volt
GB	gigabytes
Gb	gigabits
KB	kilobytes
Kb	kilobits
MB	megabytes
Mb	megabits

37. REVISION HISTORY

Table 464: Document History

Document Type	Rev. Number	Date	Comments
Product Preview	0.1	April 17, 2000	Preliminary Release
Datasheet	1.0	3 October, 2000	
<p>1. In Table 107, "SDRAM Burst Mode", the following changes were made: bit 10 & bit 11; initial value changed from 0x1 to 0x0. Bits 31:12; initial value changed from 0x1 to X.</p> <p>2. In Table 281 "Hash Table Entry Fields", the following changes were made: Bits 52:51 was added. Bits 63:51 changed to bits 63:53.</p> <p>3. In Table 463 "GT-96100A Pinout Table", the following changes were made: AD21 changed from VSS to NC. The note at the end of the table was modified.</p> <p>4. In the Features section 83MHz for CPU frequency support was changed to 100MHz.</p> <p>5. In Section 13.7.4 "Transmit SDMA Notes" on page 317, information added.</p> <p>6. In Table 270 "PCI_0 Arbiter Configuration Register, Offset: 0x101AE0", a note was added in Priority Arbitration enable field</p> <p>7. In Section 12.1 "Functional Overview" on page 253, added information.</p> <p>8. In Table 14 "Interrupt Interface Pin Assignments", the following changes were made: SoR section deleted. GPP is defined as 15:0, no reserved bits. Interrupt1* notation as SoR type was deleted.</p> <p>9. In Table 181 "PCI_1 SCS[1:0]* Base Address, Offset: 0x090 from PCI_0 or CPU; 0x010 from PCI_1", initial value changed to 0x08 from 0x04.</p> <p>10. Table 183 "PCI_1 SCS[3:2]* Base Address, Offset: 0x094 from PCI_0 or CPU; 0x014 from PCI_1", initial value changed to 0x01000008 from 0x01000004.</p> <p>11. In Table 375, Table 376, Table 377 and Table 378, GPP changed to 15:0.</p> <p>12. In Table 200 "PCI_1 PMC Register, Offset: 0x0c0 from PCI_0 or CPU, 0x040 from PCI_11", initial value changed to 0x00090001 from 0x91.</p>			

Table 464: Document History (Continued)

Document Type	Rev. Number	Date	Comments
Datasheet	1.0	3 October, 2000	
			<p>13. Table 204 "Function 1 PCI_1 Swapped SCS[1:0]* Base Address, Offset: 0x190 from PCI_0 or CPU; 0x110 from PCI_1", initial value changed to 0x08 from 0x04.</p> <p>14. Table 206 "Function 1 PCI_1 Swapped SCS[3:2]* Base Address, Offset: 0x194 from PCI_0 or CPU; 0x114 from PCI_1", initial value changed to 0x01000008 from 0x01000004.</p> <p>15. In Section 32. "AC Timing" on page 513, the following changes were made: Rst* changed to Reset**. "10 TCik Cycle" to "0.5 mSec".</p> <p>16. Table 415, Table 416 and Table 417 were changed as follows: "...the value latched in the GPD register bit is '1'" changed to "...the value latched in the GPD register bit is '0'".</p> <p>17. In Table 292 "Serial Parameters Register (SPR), Offset: 0x084820 for Ethernet_0; 0x088820 for Ethernet_1", The following changes were made: Initial value changed to "11001 (64 bit time)" from "10000 (64 bit time). The function description was modified.</p> <p>18. Section 31.1.1 "Power Sequencing Notes" on page 510 was added.</p> <p>19. Section Table 327: "CHxR1 - Sync/Abort Register (SYNR), Offset: 0x000A0C, 0x008A0C, 0x010A0C, 0x018A0C, 0x020A0C, 0x028A0C, 0x030A0C, 0x038A0C (where x is channel 0 to 7)" on page 342 was modified.</p> <p>20. In Section 14.1.3 "Receive DPLL Clock Recovery" on page 328, the text was modified.</p> <p>21. In Section 14.7.1.1 "Asynchronous Mode" on page 361, "The DPLL sampling rate" changed to "the DPLL encoding must be set to NRZ and the clock ampling rate".</p> <p>22. In Table 6 "PCI Bus 0 Pin Assignments", the following changes were made: <ul style="list-style-type: none"> • "Req0*/PARB0_GNT0" changed to "Req0*/PARB0_GNT1" • "PCI_0 arbiter output grant 0" changed to "PCI_0 arbiter output grant 1" • "functions as the arbiter's grant 0 output" changed to "functions as the arbiter's grant 1 output" • "Gnt0*/PARB0_REQ0" changed to "Gnt0*/PARB0_REQ1" • "PCI_0 arbiter input request 0" changed to "PCI_0 arbiter input request 1" • "functions as the arbiter's request 0 input" changed to "functions as the arbiter's request 1 input". </p> <p>23. In Table 7 "PCI Bus 1 Pin Assignments", the following changes were made: <ul style="list-style-type: none"> • "Req1*/PARB1_GNT0" changed to "Req1*/PARB1_GNT1" • "PCI_1 arbiter output grant 0" changed to "PCI_1 arbiter output grant 1" • "functions as the arbiter's grant 0 output" changed to "functions as the arbiter's grant 1 output" • "Gnt1*/PARB1_REQ0" changed to "Gnt1*/PARB1_REQ1" • "PCI_1 arbiter input request 0" changed to "PCI_1 arbiter input request 1" • "functions as the arbiter's request 0 input" changed to "functions as the arbiter's request 1 input". </p>

Table 464: Document History (Continued)

Document Type	Rev. Number	Date	Comments
Datasheet	1.0	3 October, 2000	
<p>24. In Table 11 "WAN Pin Assignments", the following changes were made:</p> <ul style="list-style-type: none"> • "PARB0_GNT1" changed to "PARB0_GNT2" • "PCI_0 arbiter output grant 1" changed to "PCI_0 arbiter output grant 2" • "PARB0_REQ1" changed to "PARB0_REQ2" • "PCI_0 arbiter input request 1" changed to "PCI_0 arbiter input request 2" • "PARB0_REQ2" changed to "PARB0_REQ3" • "PCI_0 arbiter input request 2" changed to "PCI_0 arbiter input request 3" • "PARB0_GNT2" changed to "PARB0_GNT3" • "PCI_0 arbiter output grant 2" changed to "PCI_0 arbiter output grant 3" • "PARB0_GNT3" changed to "PARB0_GNT4" • "PCI_0 arbiter output grant 3" changed to "PCI_0 arbiter output grant 4" • "PARB0_REQ3" changed to "PARB0_REQ4" • "PCI_0 arbiter input request 3" changed to "PCI_0 arbiter input request 4" • "PARB0_REQ4" changed to "PARB0_REQ5" • "PCI_0 arbiter input request 4" changed to "PCI_0 arbiter input request 5" • "PARB1_GNT1" changed to "PARB1_GNT2" • "PCI_1 arbiter output grant 1" changed to "PCI_1 arbiter output grant 2" • "PARB1_REQ1" changed to "PARB1_REQ2" • "PCI_1 arbiter input request 1" changed to "PCI_1 arbiter input request 2" • "PARB1_REQ2" changed to "PARB1_REQ3" • "PCI_1 arbiter input request 2" changed to "PCI_1 arbiter input request 3" • "PARB1_GNT2" changed to "PARB1_GNT3" • "PCI_1 arbiter output grant 2" changed to "PCI_1 arbiter output grant 3" • "PARB0_GNT5" changed to "PARB0_GNT6" • "PCI_0 arbiter output grant 5" changed to "PCI_0 arbiter output grant 6" • "PARB1_GNT3" changed to "PARB1_GNT4" • "PCI_1 arbiter output grant 3" changed to "PCI_1 arbiter output grant 4" • "PARB0_REQ5" changed to "PARB0_REQ6" • "PCI_0 arbiter input request 5" changed to "PCI_0 arbiter input request 6" • "PARB1_REQ3" changed to "PARB1_REQ4" • "PCI_1 arbiter input request 3" changed to "PCI_1 arbiter input request 4" • "PARB0_GNT4" changed to "PARB0_GNT5" • "PCI_0 arbiter output grant 4" changed to "PCI_0 arbiter output grant 5" <p>25. In Table 270 "PCI_0 Arbiter Configuration Register, Offset: 0x101AE0", the following changes were made: PARB0_G5, PARB0_R5 changed to PARB0_GNT1, PARB0_REQ1 "0 - this pin to function as all PSc1k1." changed to "0 - this pin to function as OTSLK1."</p>			

Table 464: Document History (Continued)

Document Type	Rev. Number	Date	Comments
Datasheet	1.0	3 October, 2000	
<p>26. In Table 271 "PCI_1 Arbiter Configuration Register, Offset: 0x101AE4", "(except for bit 29 is reserved)" changed to "(except for bits 30:29 which are reserved)."</p> <p>27. In Table 16 "Test Interface Pin Assignments", the following changes were made: Jtag[2] type changed from "O" to "I" Jtag[3] type changed from "I" to "O".</p> <p>28. In Table 463 "GT-96100A Pinout Table", AD21 signal name changed from "Vss" to "NC".</p> <p>29. In Section 12.4.1.5 "Backoff Algorithm Options" on page 277, "Port_Configuration_Extend<Limit4>" changed to "Serial Parameters Register<Limit4>".</p> <p>30. In Table 292 "Serial Parameters Register (SPR), Offset: 0x084820 for Ethernet_0; 0x088820 for Ethernet_1", the following changes were made: Bit 22 was added and bits 31:23 were modified to 'Reserved'.</p> <p>31. In Table 453 "Flex-TDM Receive Timing - Normal Clock", the following changes were made: t21 & t22 max value were removed. t23 & t27 min value changed from 8 to 5.</p> <p>32. In Table 454 "Flex-TDM Transmit Timing - Normal Speed Clock", the following changes were made: t41 & t42 max value were removed t43 min value changed from 8 to 5 t46 max value CHANGED from 10 to 13. t48, was added at the end of the table.</p> <p>33. In Table 455 "Flex-TDM Receive Timing - Double Speed Clock", the following changes were made: t31 & t32 max value wre removed. t33 & t37 min value changed from 8 to 5.</p> <p>34. In Table 456 "Flex-TDM Transmit Timing - Double Speed Clock", the following changes were made: t51 & t52 max value were removed. t53 min value changed from 8 to 5 t56 & t58 max value changed from 10 to 13.</p> <p>35. In Table 457 "MPSC Receive Timing", the following changes were made: t61 & t62 max value were removed. t63 & t65 min value changed from 8 to 5.</p>			

Table 464: Document History (Continued)

Document Type	Rev. Number	Date	Comments
Datasheet	1.0	3 October, 2000	
			<p>36. In Table 458 "MPSC Transmit Timing", the following changes were made: t71 & t72 max value were removed. t73 max value changed from 10 to 13. t75 min value changed from 8 to 5.</p> <p>37. New Section 36, Abbreviations was added.</p> <p>38. In Section 5.11.1 "SDRAM and Device Address Decode" on page 130, table 86 through to table 103 were changed as follows: Bits 7:0 changed to 11:0 and bits 31:8 were changed to 31:12.</p> <p>39. Section 4.8.2 "MultiGT Bit In The CPU Configuration Register" on page 82 was modified.</p> <p>40. Section 4.8.4 "Multi-GT Restrictions" on page 83 was modified.</p> <p>41. In Table 36 "CPU Interface Configuration, Offset: 0x000", the following changes were made: pins 8:0 & 9 & 10 were changed to "Reserved (Must be zero)". Description added to Endianess "NOTE: affects only the internal registers, and the PCI Configuration data register".</p> <p>42. Figure 44 modified.</p> <p>43. In Table 17, pin OutModePLL description modified to "...pin is in hi-z state."</p> <p>44. In Table 316 the following changes were made: The initial value for bits 5:2 was changed to '1111'.</p> <p>45. In Table 172, the initial value was changed from 0x9652 to 0x9653.</p> <p>46. In Table 173, the initial value was changed from 0x965211ab to 0x965311ab.</p> <p>47. In Table 173, the initial value was changed from 0x02ab to 0x03.</p> <p>48. Table 403. The title was corrected from 'Ethernet1 Cause Register...' to 'Ethernet1 Mask Register...'.</p> <p>49. Table 449 "Thermal data for GT-96100A in BGA 492" was modified.</p>