GT–64120A

System Controller For RC4650/4700/5000 and RM526X/527X/7000 CPUs

Datasheet
Revision 1.1
JAN 10, 2001

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

FEATURES

- Integrated system controller with PCI interface for high-performance embedded control applications.
- Supports all 64-bit bus MIPS CPUs:
  - RM526X, RM527X and RM7000 from QED.
  - RC4650 through RC5000 from IDT.
  - R5000 compatibles from various vendors.
- 100MHz CPU bus frequency.
- 3.3V CPU bus interface
- Support for multiple GT–64120A devices on the same SysAD bus (up to 4).
- 64 byte CPU write posting buffer.
  - 64-bit wide, 8 levels deep.
  - Accepts CPU writes with zero wait-states.
- CPU address remapping to resources.
- Backward Software Compatibility with GT-64120, GT-64010A and GT-64111.
- Synchronous DRAM controller.
  - 512MB address space.
  - Supports 16/64/128/256Mbit SDRAM.
  - Supports 2-way & 4-way SDRAM bank interleaving.
  - Up to 256MB bank address space, 1MB granularity.
  - 1- 4 banks supported.
  - 32/64-bit data width.
  - Support 64-bit registered SDRAM.
  - 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 512MB address space.
  - Optional external wait-state support.
  - 8-,16-,32- and 64-bit width device support.
  - Support for boot ROMs.
- Four channel DMA controller:
  - 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 allowing two transfers to take place concurrently.
  - 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.
- One 32-bit wide timer/counter, three 24-bit wide timer/counters.

- 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.
  - Operates either synchronous or asynchronous to 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.
- I_2O and Plug and Play Support
  - Industry Standard I_2O messaging unit on primary 32-bit PCI interface (also available in 64-bit mode).
  - PCI configuration registers can be accessed from both CPU and PCI side.
- Plug and Play support.
  - Plug and Play compatible configuration registers.
- PCI Power Management compliant.
- PCI Hot-Plug and CompactPCI Hot-Swap capable compliant.
- 2.5v core, 3.3v I/O supply voltage.
  - All inputs are 5v tolerant.
- 388 Ball BGA package.

Part Number: GT–64120A
Publication Revision: 1.1

Galileo Technology, Inc.
142 Charcot Avenue
San Jose, California 95131
Phone: 1 408 367-1400
Fax: 1 (408) 367-1401
www.galileoT.com/support

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

LIST OF TABLES

LIST OF TABLES (CONTINUED)

LIST OF TABLES (CONTINUED)

NOTE: Use the Register Maps to locate a specific table.

LIST OF TABLES (CONTINUED)

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

1. OVERVIEW

The GT–64120A provides a single-chip solution for designers building systems with any of the following high performance 64-bit MIPS CPUs.

Table 1: Compatible CPUs

Vendors	CPU Models
QED	• RM526X • RM527X • RM7000
IDT	• RC4650 • RC4700 • RC5000
Other vendors	R5000 compatible CPUs

The architecture of the GT–64120A supports several system implementations for different applications and cost/performance points. It is also possible to design a powerful system with minimal glue logic or add commodity logic (controlled by the GT–64120A) for differentiated system architectures that attain higher performance.

The GT–64120A 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 CPU bus can write to the on-chip write buffer, a DMA agent can move data from SDRAM to its own buffers, and a PCI device can write into an on-chip FIFO, all simultaneously.

1.1 CPU Bus Interface

The GT–64120A 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 up to 8 words (sub-word, 2 word, and 4 word are also supported). With a maximum frequency of 100MHz, the CPU can transfer in excess of 800 Mbytes/sec.

The GT–64120A allows one CPU interface to connect to up to four GT–64120A devices. This increases the address space and flexibility of system design significantly.

NOTE: The increased loading has a small effect on the system's maximum operating frequency.

The GT–64120A supports CPU address remapping to resources and can be configured to operate in little or big endian mode.

1.2 SDRAM and Device Interface

The GT–64120A has a SDRAM controller with a 64-bit wide interface.

The SDRAM controller supports 16, 64, 128, and 256Mbit SDRAMs. It is 3.3V and 5V tolerant, works at frequencies up to 100MHz, and can address up to 512MBytes. Up to four banks of SDRAM may be connected, and the controller supports 2 bank interleaving for 16 Mbit SDRAMs and 2 or 4 bank interleaving for 64, 128, 256 Mbit SDRAMs. It also supports 64-bit registered SDRAMs.

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

The GT–64120A device controller supports different types of memory and I/O devices. It has the control signals and the timing programmability to support devices such as Flash, EPROMs, FIFOs, and I/O controllers. Device widths from 8-bits to 64-bits are supported.

The GT–64120A supports ECC on 64-bit wide SDRAM. It supports detection and correction of one error, detection of two errors, and detection of three and four errors, if they are in the same nibble. GT–64120A also drives even parity to CPU on read transaction.

1.3 PCI Interface

The GT–64120A interfaces directly with the PCI bus, operating at a maximum frequency of 66MHz, and can be configured to have either of the following:

- Two 32-bit PCI devices (PCI_0 and PCI_1).
- One 64-bit PCI device (PCI_0).

NOTE: See the PCI AC timing parameters (Section 22. “AC Timing” on page 255) when designing PCI systems that run faster than 33MHz.

Each GT–64120A PCI interface can be either a master initiating a PCI bus operation or a target responding to a PCI bus operation. The GT–64120A 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.

The GT–64120A becomes a PCI bus master when the CPU bus 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.

The GT–64120A 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 and all configuration and I/O cycles, in the case of internal registers. It is possible to program the PCI slave to retry all PCI transactions targeted to the GT–64120A. The PCI slave performs PCI address remapping to resources.

The GT–64120A contains the required PCI configuration registers. All internal registers, including the PCI configuration registers, are accessible from both the CPU bus and the PCI bus. The GT–64120A configuration register set is PC Plug and Play compatible with industry standard I_2O support.

The GT–64120A supports PCI Power Management, PCI Hot-Plug, and CompactPCI Hot Swap Capable requirements.

The GT–64120A can also act as a PCI to Memory bridge, even without the presence of the CPU.

1.4 DMA Engines

The GT–64120A incorporates four high performance DMA engines.

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

Source and destination addresses are non-aligned on any byte address boundary. The DMA channels are programmed by the CPU bus, or by the PCI masters, or without CPU bus intervention via a linked list of records. This linked list is loaded by the DMA controller into the channel's working set when a DMA transaction ends. The DMA supports increment/decrement/hold on source and destination addresses independently, and alignment of addresses towards source and destination. In addition, the GT–64120A provides an override capability of source/destination/record address mapping to force access to PCI_0 or PCI_1.

The DMA can be initiated by the CPU writing to a register, an external request via a DMAReq* pin, or an internal timer/counter. Four End of Transfer pins act as inputs to the GT–64120A allow ending a DMA transfer on a certain channel. In case of chained mode, after the transfer is ended, it is possible to transfer the descriptor to CPU ownership which can then calculate the number of remaining bytes in the buffer associated with the closed descriptor.

Fly-by is also supported, allowing data to be transferred directly between two residents on the device/memory bus without having to go into a DMA FIFO.

1.5 Differences Between the GT-64120 and the GT–64120A

Table 2: Differences between the GT-64120 and the GT–64120A

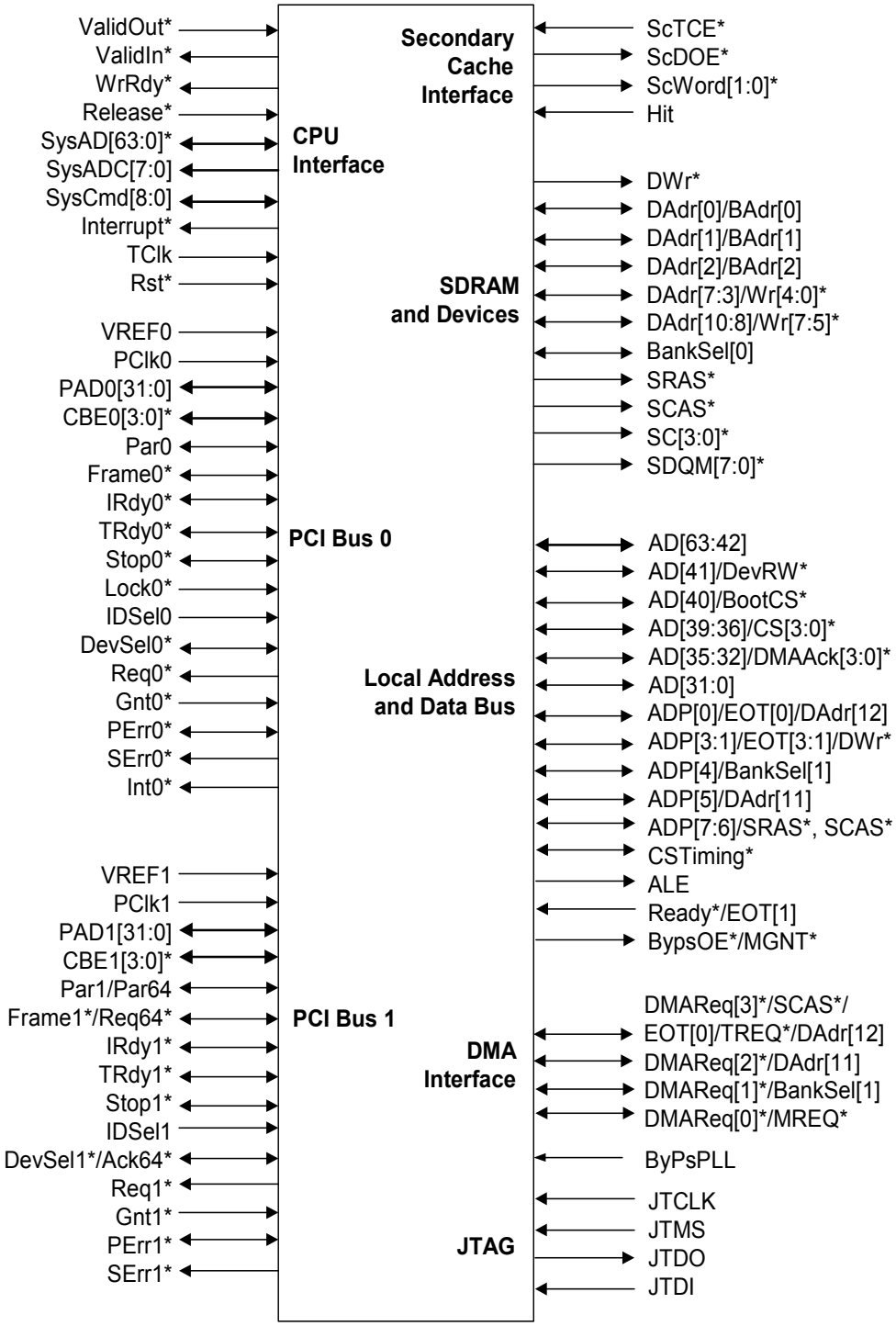
GT-64120	GT–64120A
• Manufactured on a 3,3V, 0.35u process.	• Manufactured on a 0.25u process • The core voltage operates on a 2.5V supply (VCC2.5) and the I/O operates on a 3.3V supply (VCC3.3). **NOTE:** All inputs are 5V tolerant.

Table 2: Differences between the GT-64120 and the GT–64120A (Continued)

GT-64120	GT–64120A
• Runs at a maximum CPU/SDRAM clock frequency of 75MHz.	• Runs at a maximum CPU/SDRAM clock frequency of 100MHz. • also includes an internal PLL for improved device timing.
• Supports parity on both SDRAM and Devices.	• Supports ECC on 64-bit SDRAM only. There is no parity nor ECC support for 32-bit SDRAM and for Devices. **NOTE:** It is possible to support parity in systems for 32-bit SDRAM or devices by using external ‘511 devices and some logic for interrupt generation.
• Does not support registered SDRAM DIMMs	• Supports 64-bit registered SDRAM DIMMs
• Does not support PCI power management in the system.	• Contains the required registers and logic to support PCI power management in the system.

2. PIN INFORMATION

Figure 1: Pin List



2.1 Pin Assignment Tables

Table 3: CPU Interface Pin Assignments

Pin Name/ Ball #	I/O	Full Name	Description
TClk P01	I	Clock	The input clock to the GT–64120A (up to 100MHz). TClk is used for both the SysAD and Device interface. TClk must be driven for all applications, including those that do not use the CPU bus.
Release* B04	I	Release	Signals to the GT–64120A that the CPU has released the SysAD and SysCmd buses for completion of a read request.
WrRdy* C06	O	Write Ready	The GT–64120A signals that it can accept a CPU write request (i.e., there is room in the write posting FIFO.)
ValidIn* A05	O	Valid Input	The GT–64120A signals that it is driving valid data on the SysAD bus and a valid data identifier on the SysCmd bus.
ValidOut* B05	I	Valid Output	Signals that the CPU is driving valid address or data on the SysAD bus and a valid command or data identifier on the SysCmd bus.
SysAD[63:0]	I/O	System Address/Data Bus	A 64-bit multiplexed address and data bus for communication between the CPU and GT–64120A.
[63:54] K01, J03, K02, J01, J02, H03, H01, J04, H02, G03 [53:44] G01, G02, F01, F03, G04, F02, E01, E03, E04, E02 [43:34] D01, D03, D02, C01, C02, B01, A03, C04, B03, C05 [33:24] A04, D05, C07, A06, D07, B06, C08, A07, D08, B07 [23:14] C09, A08, B08, A09, C10, B09, D10, A10, C11, B10 [13:0] D12, A11, C12, B11, A12, C13, B12, C14, A13, D13, B13, C15, A14, D15			
SysADC[7:0] B14, C16, A15, B15, A16, C17, B16, D17	I/O	System Address/Data Check	An 8-bit bus containing parity for the SysAD bus. **NOTE:** SysADC is valid on data cycles only.
SysCmd[8:0] A17, C18, B17, A18, B18, C19, A19, D18, B19	I/O	System Command/Data Identifier Bus	A 9-bit bus for command and data identifier transmission between the CPU and GT–64120A.

Table 3: CPU Interface Pin Assignments (Continued)

Pin Name/ Ball #	I/O	Full Name	Description
Interrupt* C20	I/O	Interrupt	An “OR” of all the internal interrupt sources on the GT–64120A. **NOTE:** This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Table 56, on page 143 for more information.
Rst* N25	I	Reset	Resets all units of the GT–64120A to their initial state. This includes a general chip reset as well as PCI_0 and PCI_1 units. This signal must be asserted for at least 10 TClk cycles. When in the reset state, all PCI output pins are put into tristate and all open drain signals are floated.

Table 4: CPU Secondary Cache Support Pin Assignments

Pin Name/ Ball #	I/O	Full Name	Description
ScTCE* A20	I	Secondary Cache Tag RAM Chip Enabled	Chip enable for secondary cache tag RAM. This input pin must be pulled HIGH through a 4.7 KOhm resistor if it is not used.
ScDOE* D20	O	Secondary Cache Data RAM Output Enable	Output Enable for Secondary Cache Data RAM. **NOTE:** This output pin must be left unconnected if it is not used.
ScWord[1:0] C21, A21	O	Secondary Cache Word Index	Determines correct double-word of Secondary cache index. **NOTE:** This output pin must be left unconnected if it is not used.
Hit B20	I	Secondary Cache Tag Match	Asserted by tag RAM on secondary cache tag match. **NOTE:** This input pin must be pulled LOW through a 4.7 KOhm resistor if it is not used.

Table 5: PCI Bus 0 Pin Assignments

Pin Name/ Ball #	I/O	Full Name	Description
VREF0 U26	I	PCI_0 Voltage Reference	This pin must be connected directly to the 3.3V or the 5V power plane. The power plane used depends on which voltage level PCI_0 supports. VREF0 and VREF1 use independent voltage levels.
PClk0 P26	I	PCI_0 Clock	This pin provides the timing for the PCI transactions. The PCI clock range is between 0 and 66MHz. The PClk1 cycle must be higher than TClk cycle by at least 1ns. **NOTE:** See Section 22.1 “TClk/PClk Restrictions” on page 261 for more information. See the PCI AC timing parameters (Section 22. “AC Timing” on page 255) when designing PCI systems that run faster than 33MHz.
PAD0[31:0]	I/O	PCI_0 Address/Data	32-bit multiplexed PCI 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, PAD0[31:0] contains data.
[31:22] T24, P25, R23, R24, U24, T26, R25, R26, V25, V26 [21:12] U25, V24, W25, V23, W26, W24, AD26, AC25, AC24, AC26 [11:0] AD23, AF24, AE26, AD25, AE23, AC22, AF23, AD22, AD20, AE22, AF22, AD21			
CBE0[3:0]* T25, Y24, AB25, AE24	I/O	PCI_0 Command/Byte Enable	During the address phase of the transaction, CBE0[3:0]* provides the PCI bus command. During the data phase, these lines provide the byte enables.
Par0 AB23	I/O	PCI_0 Parity	Calculated by the GT–64120A as an even parity bit for the PAD0[31:0] and CBE0[3:0]* lines.
Frame0* Y26	I/O	PCI_0 Frame	Asserted by the GT–64120A to indicate the beginning and duration of a master transaction. Frame0* asserts to indicate the beginning of the cycle. While Frame0* is 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–64120A when it acts as a target.
IRdy0* Y25	I/O	PCI_0 Initiator Ready	Indicates the bus master’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.

Table 5: PCI Bus 0 Pin Assignments (Continued)

Pin Name/ Ball #	I/O	Full Name	Description
TRdy0* AA26	I/O	PCI_0 Target Ready	Indicates 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* Y23	I/O	PCI_0 Stop	Indicates that the current target is requesting the bus master to stop the current transaction. As a master, the GT–64120A responds to the assertion of Stop0* by disconnecting, retrying or aborting. As a target, the GT–64120A asserts Stop0* to retry or disconnect.
Lock0* AA25	I	PCI_0 Lock	Indicates an atomic operation that may require multiple transactions to complete. When the GT–64120A is a PCI target, Lock0* is sampled on the rising edge of the PCIk when Frame0* is asserted. If Lock0* is sampled asserted, the GT–64120A enters into a locked state and remains in this state until Lock0* is sampled deasserted on the following rising edge of PCIk, when Frame0* is sampled asserted.
IdSel0	I	PCI_0 Initialization Device Select	Asserted to act as a chip select during PCI configuration read and write transactions.
DevSel0* U23	I/O	PCI_0 Device Select	Asserted by the target of the current access. When the GT–64120A 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 the five bus cycles, the GT–64120A aborts the cycle. As a target, when the GT–64120A recognizes its transaction, it asserts DevSel0* in a medium speed (two cycles after the assertion of Frame0*).
Req0* AC20	O	PCI_0 Bus Request	Asserted by the GT–64120A to indicate to the PCI bus arbiter that it requires use of the PCI bus.
Gnt0* AF21	I	PCI_0 Bus Grant	Indicates to the GT–64120A that access to the PCI bus is granted.
PErr0* AB26	I/O	PCI_0 Parity Error	Asserted when a data parity error is detected. This pin features a sustained tri-state output.

Table 5: PCI Bus 0 Pin Assignments (Continued)

Pin Name/ Ball #	I/O	Full Name	Description
SErr0* AB24	O	PCI_0 System Error	Asserted when a serious system error (not necessarily a PCI error) is detected. The GT–64120A asserts the SErr0* two cycles after the failing address. This pin features an open-drain output.
Int0* AE21	O	PCI_0 Interrupt Request	Asserted by the GT–64120A when one of the unmasked internal interrupt sources is asserted. This pin features an open-drain output.

Table 6: PCI Bus 1 Pin Assignments

Pin Name/ Ball #	I/O	Full Name	Description
VREF1 L24	I	PCI_1 Voltage Reference	This pin must be connected directly to the 3.3V or the 5V power plane depending on which voltage level PCI_1 supports. VREF0 and VREF1 use independent voltage levels.
PClk1 N26	I	PCI_1 Clock	This pin provides the timing for the PCI-related bus transaction. The PCI clock range is between 0 and 66MHz. On double PCI configuration, this clock frequency can be independent of both TClk and PClk0. On 64-bit PCI configuration, it must be tied to PClk0. The PClk1 cycle must be higher than TClk cycle by at least 1ns. **NOTE:** See Section 22.1 “TClk/PClk Restrictions” on page 261 for more information. See the PCI AC timing parameters (Section 22. “AC Timing” on page 255) when designing PCI systems that run faster than 33MHz.
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 are PAD0[63:32] or the most significant 32 bits of data for PCI_0 transactions.
PAD1: [31:22] N24, M25, P24, N23, L26, M24, L25, M26, K24, J25 [21:12] K23, K26, J24, H26, H25, J26, D26, E23, D25, F24 [11:0] C26, D24, C25, E24, C23, B23, A24, B24, C22, D22, B22, A23 PAD0: [31:22] T24, P25, R23, R24, U24, T26, R25, R26, V25, V26 [21:12] U25, V24, W25, V23, W26, W24, AD26, AC25, AC24, AC26 [11:0] AD23, AF24, AE26, AD25, AE23, AC22, AF23, AD22, AD20, AE22, AF22, AD21			

Table 6: PCI Bus 1 Pin Assignments (Continued)

Pin Name/ Ball #	I/O	Full Name	Description
CBE1[3:0]*/ CBE0[7:4]*	I/O	PCI_1 Bus Command/Byte Enable	During the address phase of the transaction, CBE1[3:0]* provides the PCI bus command. During the data phase, these lines provide the byte enables.
		PCI_0 Bus (64-bit) Byte Enable	If PCI_0 is configured for 64 bit, these pins are CBE0[7:4]* and are byte enables for the most significant 32 bits of PCI_0 data phases.
CBE1: M23, G25, E26, A25 CBE0: T25, Y24, AB25, AE24			
Par1/Par64 E25	I/O	PCI_1 Parity	Par1 is calculated by the GT–64120A as an even parity bit for the PAD1[31:0] and CBE1[3:0]* lines during address/data phases.
		PCI_0 (64-bit) Parity	If PCI_0 is configured for 64 bit, this pin is Par64 and functions as Parity for Upper WORD of data an even parity bit for PAD0[63:32] and CBE0[7:4]*.
Frame1*/ Req64* H23	I/O	PCI_1 Frame	This pin is asserted by the GT–64120A to indicate the beginning and duration of a master transaction. Frame1* asserts to indicate the beginning of the cycle. While Frame1* is asserted, data transfer continues. Frame1* deasserts to indicate that the next data phase is the final data phase transaction. Frame1* is monitored by the GT–64120A when it acts as a target.
		PCI_0 (64-bit) Request 64	If PCI_0 is configured for 64-bit, this pin is Req64* and functions as a request for a 64-bit transaction. Req64* has the same timing as Frame0*. **NOTE:**
		NOTE: This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.	
IRdy1* G26	I/O	PCI_1 Initiator Ready	This pin indicates the bus master’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.
TRdy1* H24	I/O	PCI_1 Target Ready	This pin indicates 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 IRdy* are asserted together.

Table 6: PCI Bus 1 Pin Assignments (Continued)

Pin Name/ Ball #	I/O	Full Name	Description
Stop1* G23	I/O	PCI_1 Stop	This pin is asserted by the target requesting the bus master to stop the current transaction. As a master, the GT–64120A PCI device1 responds to the assertion of Stop1* by disconnecting, retrying or aborting. As a target, the GT–64120A asserts Stop1* to retry or disconnect.
IdSel1 K25	I	PCI_1 Initialization Device Select	This pin indicates a chip select for PCI_1 during PCI configuration read and write transactions.
DevSel1*/ Ack64* F25	I/O	PCI_1 Device Select	This pin is asserted by the target of the current access. When PCI_1 is bus master, it expects the target to assert DevSel1* within five bus cycles, confirming the access. If the target does not assert DevSel1* within the required bus cycles, the GT–64120A aborts the cycle. As a target, when the GT–64120A PCI device1 recognizes its transaction, it asserts DevSel1* in a medium speed (two cycles after the assertion of Frame1*).
		PCI_0 (64-bit) Acknowledge 64	If PCI_0 is configured for 64-bit then this signal functions as Ack64*. When actively driven by PCI target, it indicates the target is willing to accept 64-bit wide data. Ack64* has the same timing as Devsel0*.
Req1* B21	O	PCI_1 Bus Request	Asserted by the GT–64120A to indicate to the PCI bus arbiter that it requires use of the PCI bus.
Gnt1* A22	I	PCI_1 Bus Grant	Indicates to the GT–64120A that access to the PCI bus is granted.
PErr1* F26	I/O	PCI_1 Parity Error	Asserted when a data parity error is detected. This pin features a sustained tri-state output.
SErr1* G24	O	PCI_1 System Error	Asserted when a serious system error (not necessarily a PCI error) is detected. The GT–64120A asserts the SErr1* two cycles after the failing address. This pin features an open drain output.

Table 7: SDRAM and Devices Pin Assignments

Pin Name/ Ball #	I/O	Full Name	Description
DWr* AE11	O	SDRAM Write	LOW when the GT–64120A writes to the SDRAM.
DAdr[0]/ BAdr[0] AD19	I/O	SDRAM Address 0	In an access to a SDRAM bank, this pin functions as a SDRAM address bit.
		Burst Address 0	In write and read accesses from devices this pin functions as a burst address bit. See Section 13.2 “Devices” on page 148 for information on how to connect this address bit to different devices.
		NOTE: This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.	
DAdr[1]/ BAdr[1] AF20	I/O	SDRAM Address [1]	In an access to a SDRAM bank, this pin functions as a SDRAM address bit.
		Burst Address [1]	In write and read accesses from devices this pin functions as a burst address bit. See Section 13.2 “Devices” on page 148 for information on how to connect this address bit to different devices.
		NOTE: This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.	
DAdr[2]/ BAdr[2] AC19	I/O	SDRAM Address [2]	In an access to a SDRAM bank, this pin functions as a SDRAM address bit.
		Burst Address [2]	In write and read accesses from devices this pin functions as a burst address bit. See Section 13.2 “Devices” on page 148 for information on how to connect this address bit to different devices.
		NOTE: This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.	
DAdr[7:3]/ Wr[4:0]* AF18, AE19, AF19, AD18, AE20	I/O	SDRAM Address [7:3]	In SDRAM accesses, these pins function as SDRAM addresses.
		Device Byte Write [4:0]	In device writes, they function as byte write enable indications to the bank bytes [4:0].
		NOTE: This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.	

Table 7: SDRAM and Devices Pin Assignments (Continued)

Pin Name/ Ball #	I/O	Full Name	Description
DAdr[10:8]/ Wr[7:5]* AC17, AE18, AD17	I/O	SDRAM Address [10:8]	In SDRAM accesses, these pins function as SDRAM addresses.
		Device Byte Write [7:5]	In device writes, they function as byte write enable indications to the bank bytes [7:5].
		NOTE: This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.	
BankSel[0] AF17	I/O	SDRAM Bank Select [0]	In SDRAM accesses, this pin functions as bank select bit[0]. **NOTE:** This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.
SRAS* AE15	O	SDRAM Row Address Strobe	Asserts to indicate an active row address is on the DAdr lines.
SCAS* AD12	O	SDRAM Column Address Strobe	Asserts to indicate an active column address is on the DAdr lines.
SCS[3:0]* AD13, AF14, AC14, AE14	O	SDRAM Chip Selects	SDRAM chip selects for up to four banks.
SDQM[7:0]* AD10, AF11, AE12, AF12, AD11, AE13, AC12, AF13	O	SDRAM Byte Enables	For AD bus to Write up to 64 bits of data to an SDRAM bank.

Table 8: Local Address and Data Bus Pin Assignments

Pin Name/ Ball #	I/O	Full Name	Description
AD[63:42]	I/O	Data[63:42]	In both SDRAM and device accesses, these pins function as part of the data to be read/written from/to the bank.
[63:54] M02, M01, L03, N02, M04, N01, M03, P02, P04, N03 [53:42] R02, P03, R01, T02, R03, T01, R04, U02, T03, U01, U04, V02			
AD[41]/ DevRW* U03	I/O	Data [41]	In SDRAM access and in device data phase, it is part of the data to be read/written from/to the bank.
		Device Read-Write	In device address phase, it indicates if the access is a read (‘1’) or a write (‘0’). Latching is done via ALE.

Table 8: Local Address and Data Bus Pin Assignments (Continued)

Pin Name/ Ball #	I/O	Full Name	Description
AD[40]/ BootCS* V01	I/O	Data [40]	In SDRAM access and in device data phase, it is part of the data to be read/written from/to the bank.
		Boot Chip Select	In device address phase, it is the boot device chip select. The Chip Select needs to be qualified with the CSTiming* signal. Latching is done via ALE.
AD[39:36]/ CS[3:0]* W02, W01, V03, Y02	I/O	Data [39:36]	In SDRAM access and in device data phase, it is part of the data to be read/written from/to the bank.
		Chip Select [3:0]	In device address phase, these pins function as device chip selects. The Chip Selects need to be qualified with the CSTiming* signal. Latching is done via ALE.
AD[35:32]/ DMAAck[3:0]* W04, Y01, W03, AA02	I/O	Data [35:32]	In SDRAM access and in device data phase, it is part of the data to be read/written from/to the bank.
		DMA Acknowledge[3:0]	In device address phase, these pins function as DMA Acknowledges. They need to be qualified with the CSTiming* signal. Latching is done via ALE.
AD[31:0]	I/O	Address/ Data[31:0]	In SDRAM access, it is part of the data to be read/written from/to the bank. In device access, it is a multiplexed address and data bus. Function as address bus during address phase (latching is done via ALE), and as data bus during data phase.
[31:22] AC01, AB03, AD02, AC03, AD01, AF02, AE03, AF03, AE04, AD04 [21:12] AF04, AE05, AC05, AD05, AF05, AE06, AC07, AD06, AF06, AE07 [11:0] AF07, AD07, AE08, AC09, AF08, AD08, AE09, AF09, AE10, AD09, AF10, AC10			
ADP[0]/ EOT[0]/ DAdr[12] AB04	I/O	AD Bus ECC [0]	In SDRAM accesses, these bits serve as ECC bit [0], generated by GT–64120A for writes, and read from ECC bank upon data reads.
		End of DMA Transfer [0]	In DMA transfers, EOT[0] serve as End Of Transfer indications for DMA channel 0.
		SDRAM Address [12]	Can be configured to function as DAdr[12] for 256Mbit SDRAM configuration. See Section 5.1.2.4 “Multiplexing DAdr[12]” on page 63.
ADP[3:1]/ EOT[3:1]/DWr* AB01, AA03, AC02, AB04	I/O	AD Bus ECC [3:1]	In SDRAM accesses, these bits serve as ECC bits [3:1], generated by GT–64120A for writes, and read from ECC bank upon data reads.
		End of DMA Transfer [3:1]	In DMA transfers, EOT[3:1] serve as End Of Transfer indications for the DMA channels 1-3.
		SDRAM Write	ADP[3] can be configured to function as DWr* on RESET. See Section 12. “Reset Configuration” on page 143 for more information.

Table 8: Local Address and Data Bus Pin Assignments (Continued)

Pin Name/ Ball #	I/O	Full Name	Description
ADP[4]/ BankSel[1] AB02	I/O	AD Bus ECC [4]	In SDRAM accesses, these bit serves as ECC bit [4], generated by GT–64120A for writes, and read from ECC bank upon data reads.
		SDRAM Bank Select [1]	ADP[4] can be configured to function as SDRAM bank select bit[1]. See Section 5.1.2.2 “Duplicating DAdr[11] and BankSel[1] on DMAReq[2:1]*” on page 62.
ADP[5]/ DAdr[11] Y03	I/O	AD Bus ECC [5]	In SDRAM accesses, these bit serves as ECC bit[5], generated by GT–64120A for writes, and read from ECC bank upon data reads.
		SDRAM Address [11]	ADP[5] can be configured to function as SDRAM address bit[11].
ADP[7:6]/ SRAS*, SCAS* Y04, AA01,	I/O	AD Bus ECC [7:6]	In SDRAM accesses, these bits serve as ECC bits [7:6], generated by GT–64120A for writes, and read from ECC bank upon data reads.
		SDRAM Row and Column Address Strobs	ADP[7:6] can be configure to function as SRAS* and SCAS* on RESET. See Section 12. “Reset Configuration” on page 143 for more information.
CSTiming* AF16	I/O	Chip Select Timing	Active for the number of cycles that the device currently being accessed was programmed to in the respective device control register. Used to qualify the CS[3:0]*, BootCS and the DMAAck[3:0]* signals. **NOTE:** This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.
ALE AD15	O	Address Latch Enable	Used to latch the Address, BootCS*, CS[3:0]*, DevRW* and DMAAck[3:0]* from the AD bus.
Ready*/EOT[1] AC15	I	Ready	A device cycle extender. **NOTE:** When inactive during a device access, an access will extend until Ready* is asserted.
		End of Transfer[1]	Ready* can be programmed to function as EOT[1]. See Section 5.1.2.3 “Duplicating DMA End of Transfer Pins” on page 63.
		NOTE: This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.	

Table 8: Local Address and Data Bus Pin Assignments (Continued)

Pin Name/ Ball #	I/O	Full Name	Description
BypsOE*/ MGNT*/DWr* AD14	O	Bypass Output Enable	Controls the output enable of bypass latches/buffers/switches. Use the bypass when a 64-bit read transaction is executed from the CPU will be transferred directly.
		SDRAM Write	BypsOE*/MGNT* can be programmed to function as DWr*. See- Section 5.1.2.3 “Duplicating DMA End of Transfer Pins” on page 63.
		Memory (AD) bus Grant	Asserted in response to MREQ* in case UMA was activated at RESET.

Table 9: DMA Pin Assignments

Pin Name/ Ball #	I/O	Full Name	Description
DMAReq[3]*/ SCAS*/ EOT[0]/ TREQ*/ DAdr[12] AE16	I/O	DMA Request[3]	DMA requests by external devices to channel 3.
		SDRAM Column Address Strob	DMAReq[3]* can be programmed to function as SCAS*.
		End of Transfer[0]	DMAReq[3]* can be programmed to function as EOT[0].
		UMA Total Request	For UMA operation, DMAReq[3]* can be programed to indicate that there is a pending internal request in DRAM and Device interface that requires GT–64120A ownership of AD bus. See Section 5.6.6 “Total Request” on page 79.
		SDRAM Address[12]	DMAReq[3]* can be programmed to function as DAdr[12] for 256Mbit SDRAM. See Section 5.1.2.4 “Multiplexing DAdr[12]” on page 63.
		NOTE: This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.	
DMAReq[2]*/ DAdr[11] AD16	I/0	DMA Request[2]	DMA requests by external devices to channel 2.
		SDRAM Address[11]	DMAReq[2]* can be programmed to function as DAdr[11]. See Section 5.1.2.3 “Duplicating DMA End of Transfer Pins” on page 63.
		NOTE: This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.	

Table 9: DMA Pin Assignments (Continued)

Pin Name/ Ball #	I/O	Full Name	Description
DMAReq[1]*/ BankSel[1] AE17	I/0	DMA Request[1]	DMA requests by external devices to channel 1.
		Bank Select[1]	DMAReq[1]* can be programmed to function as SDRAM Bank-Sel[1]. See Section 5.1.2.2 “Duplicating DAdr[11] and BankSel[1] on DMAReq[2:1]*” on page 62.
		NOTE: This pin is sampled on RESET to configure the GT–64120A prior to boot-up. See Section 12. “Reset Configuration” on page 143 for more information.	
DMAReq[0]*/ MREQ*/ SRAS* AF15	I/O	DMA Request [0]	DMA request indication by an external device.
		Memory Bus Request	DMAReq[0] can be configured to function as memory bus (AD) request by a device to support UMA.
		SDRAM Row Address Strob	This pin can be programmed to function as SRAS*. See Section 5.1.2.1 “Duplicating SDRAM Control Lines” on page 62.

Table 10: JTAG Interface Pin Assignments

Pin Name/ Ball #	I/O	Full Name	Description
JTCLK L01	I	JTAG Clock	Clock for test logic. JTMS and JTDI are received on the rising edge, JTDO is driven from the falling edge. This signal determines the shifting rate. This input pin must be pulled LOW through a 4.7 KOhm resistor if it is not used.
JTMS L02	I	JTAG Mode Select	A broadcast signal which controls test logic operation. This input pin must be pulled HIGH through a 4.7 KOhm resistor if it is not used.
JTDO K03	O	JTAG Data Out	Serial data output. Tri-state changes on negative change of JTCLK. This output pin must be left UNCONNECTED if it is not used.
JTDI K04	I	JTAG Data In	Serial data input This input pin must be pulled HIGH through a 4.7 KOhm resistor if it is not used.

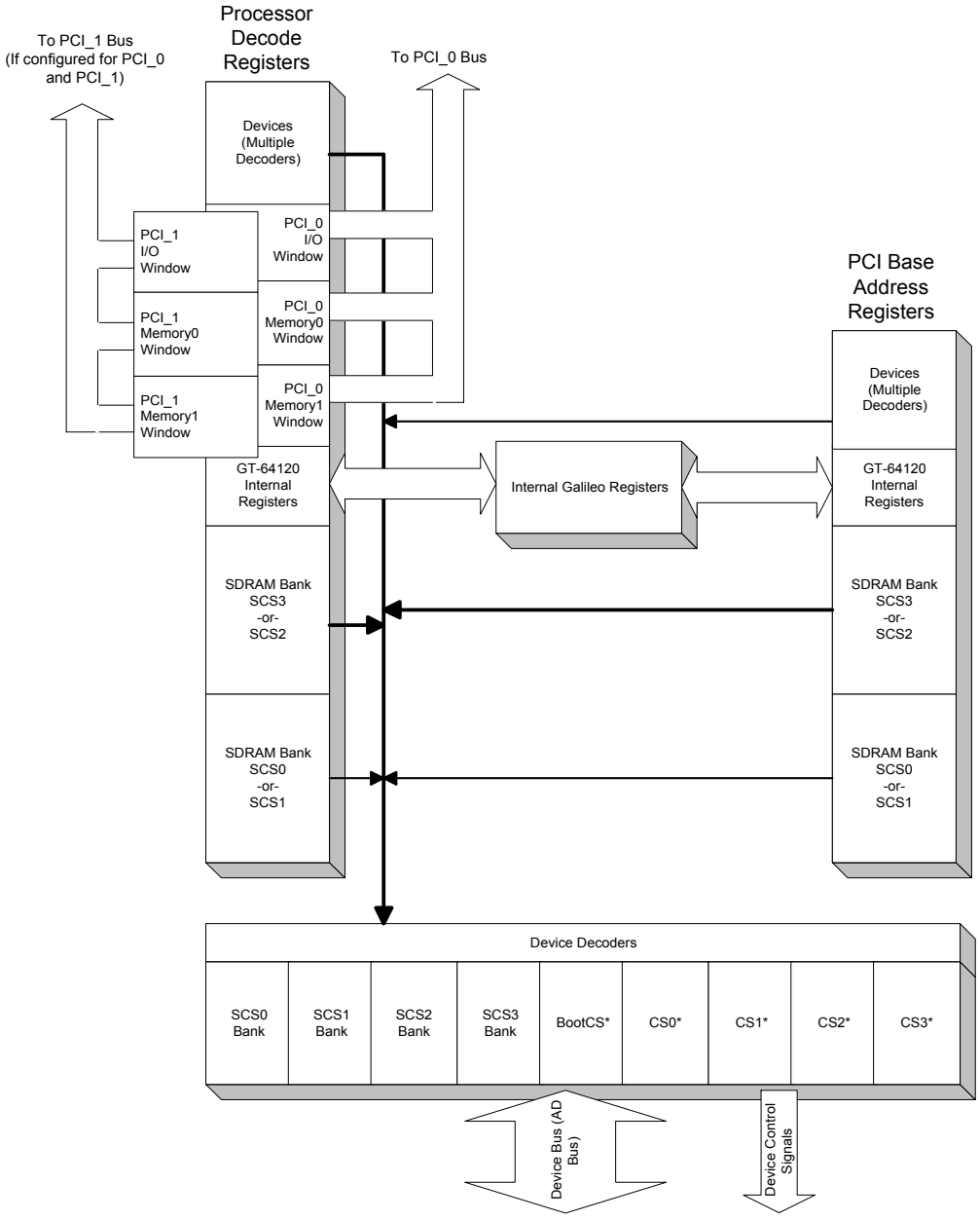
Table 11: PLL Pin Assignment

Pin Name/ Ball #	I/O	Full Name	Description
ByPsPLL AC08	I	Bypass PLL	Reserved for Galileo Technology usage.

3. ADDRESS SPACE DECODING

The GT–64120A 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/0 and PCI_1 Memory0/1 will only exist if the GT–64120A is configured for both PCI_0 and PCI_1.

Each group can have a minimum of 2 Mbytes and a maximum of 256 Mbytes of address space. The individual devices in the device groups (e.g. SCS[0]*) are further sub decoded to 1 Mbyte resolution. Table 12 shows the CPU decode and device sub-decode associations, Table 13 shows the same process for PCI.

Table 12: 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 GT–64120A internal registers.

Table 13: PCI_0 Base Address Register and Device Decoder Mappings

PCI Base Address Register (BAR) Decoder[1]	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 GT–64120A internal registers.
Internal Registers (I/O) - BAR 5 at 0x24	None. Decodes PCI_0 I/O accesses to GT–64120A 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 14: PCI_1 Base Address Register and Device Decoder Mappings

PCI Base Address Register (BAR) Decoder[1]	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*

Table 14: PCI_1 Base Address Register and Device Decoder Mappings (Continued)

PCI Base Address Register (BAR) Decoder[1]	Associated Device Sub-Decoders
Internal Registers (Memory) - BAR 4 at 0xa0	None. Decodes PCI_1 memory accesses to GT–64120A internal registers.
Internal Registers (I/O) - BAR 5 at 0xa4	None. Decodes PCI_1 I/O accesses to GT–64120A 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 35:28 of the SysAD address are compared against bits 14:7 in the various CPU Low decode registers. These values much match exactly. This effectively sets a 256 Mbyte “page” for the resource group.
2. Bits 27:21 of the SysAD address are compared against bits 6:0 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 27:21 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.

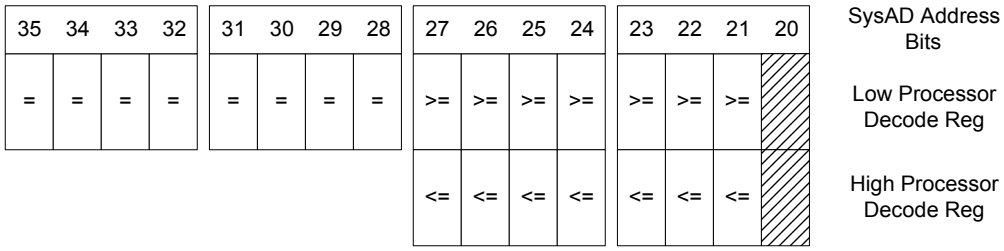
Once a CPU resource group has been decoded, it must be subdecoded to determine which physical device should be accessed within that group. This decoding is controlled by the device Low and High decode registers. The comparison works as follows:

1. Bits 27:20 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 27:20 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.

Examples of the CPU-side decode process are shown in Figure 3 and Figure 4.

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

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 64Mbytes in length (0xA.4000.0000 to 0xA.43FF.FFFF):

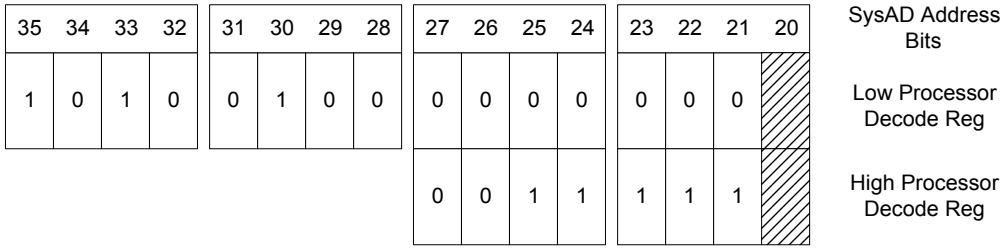
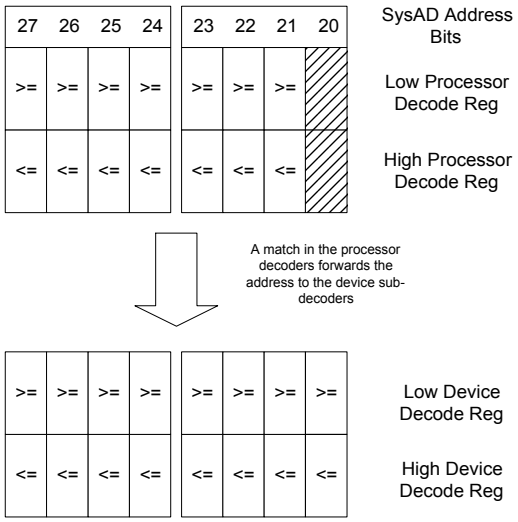
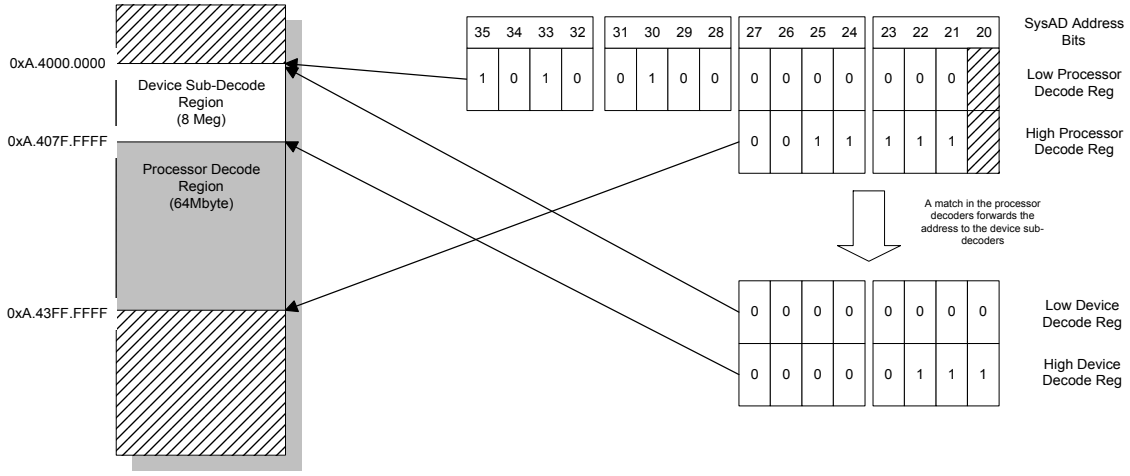


Figure 4: Device Sub-Decode Function and Example



Example: Using the previous CPU decode example (0xA.4000.0000 to 0xA.43FF.FFFF), place a device sub-decode in the first 8 Meg:

0xA.4000.0000 to 0xA.407F.FFFF

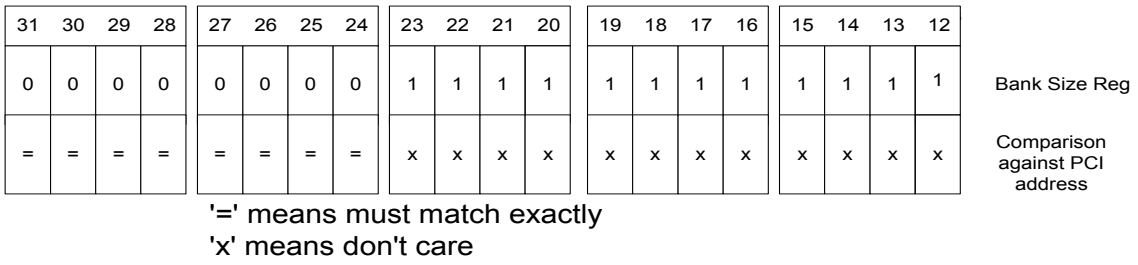


3.1.2 PCI Side Decoding Process

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.)

The size of the “window” in PCI space for each Base Address register is set by the Bank Size registers for each Base Address register. The bank size sets which address bits are significant for the comparison between the active PCI address and the values in the Base Address registers (see Figure 5).

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



The comparison works as follows:

1. Bits 31:N of the PCI address are compared against bits 31:N in the various Base Address registers (BAR). These values much match exactly. The value of ‘N’ is set by the least significant bit with a ‘0’ in the Bank Size registers (for example, ‘N’ would be equal to 24 in the example shown in Figure 5, above.)
2. If all of the above is true, the resource group is selected and a subdecode is performed to determine the specific resource.

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 27:20 (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 27:20 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 27:20 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.

NOTE: The coupling of the SysAD, PCI, and device memory maps requires special attention for designers of PC Plug and Play adapters.

3.2 Disabling the Device Decoders

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

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

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

When a DMA channel is activated, the DMA controller uses the 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.

NOTE: 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–64120A

- 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–64120A issues a bus error (over SysCmd[5]), if the access was a read access, and 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–64120A BARs result in no response at all from the GT–64120A.

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

NOTE: Address space decoders must never be programmed to overlap. Address space decoders programmed 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 15. The default PCI map and BAR sizing information is shown in Table 16 and Table 17.

Table 15: 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 GT–64120A internal registers	Internal Registers
NOTE: If I_2O is enabled, the internal space also contains the I_2O registers at offsets: 0x0.1400.1C00 to 0x0.1400.1CFF			
0x0.1000.0000 to 0x0.11FF.FFFF 32 Megabytes	PCI0 I/0	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 48 Megabytes[1]	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*

Table 15: CPU and Device Decoder Default Address Mapping (Continued)

CPU Decode Range and Size	Resource Group	Device Decode Range and Size	Device Selected
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*
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/0	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

1. By default, 0x0.1E00.0000 to 0x0.1EFF.FFFF is allocated to CS[2:0] but is not accessible since the device decoders are not programmed to respond to a hit from this region.

Table 16: 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 **NOTE:** If I_2O is enabled, the first 4Kbyte of SCS[1:0]* space are directed to the I_2O internal registers instead of the SDRAM.	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 16: 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
0x0.1400.0000 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
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 17: 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*

Table 17: 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
0x0.0100.0000 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*
0x0.1F00.0000 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–64120A supports address remapping on both the CPU interface and the PCI interface.

NOTE: Although the DMA 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 such resource has a Remap register associated with it. These registers are listed in Section 20.4 “CPU Address Decode” on page 182.

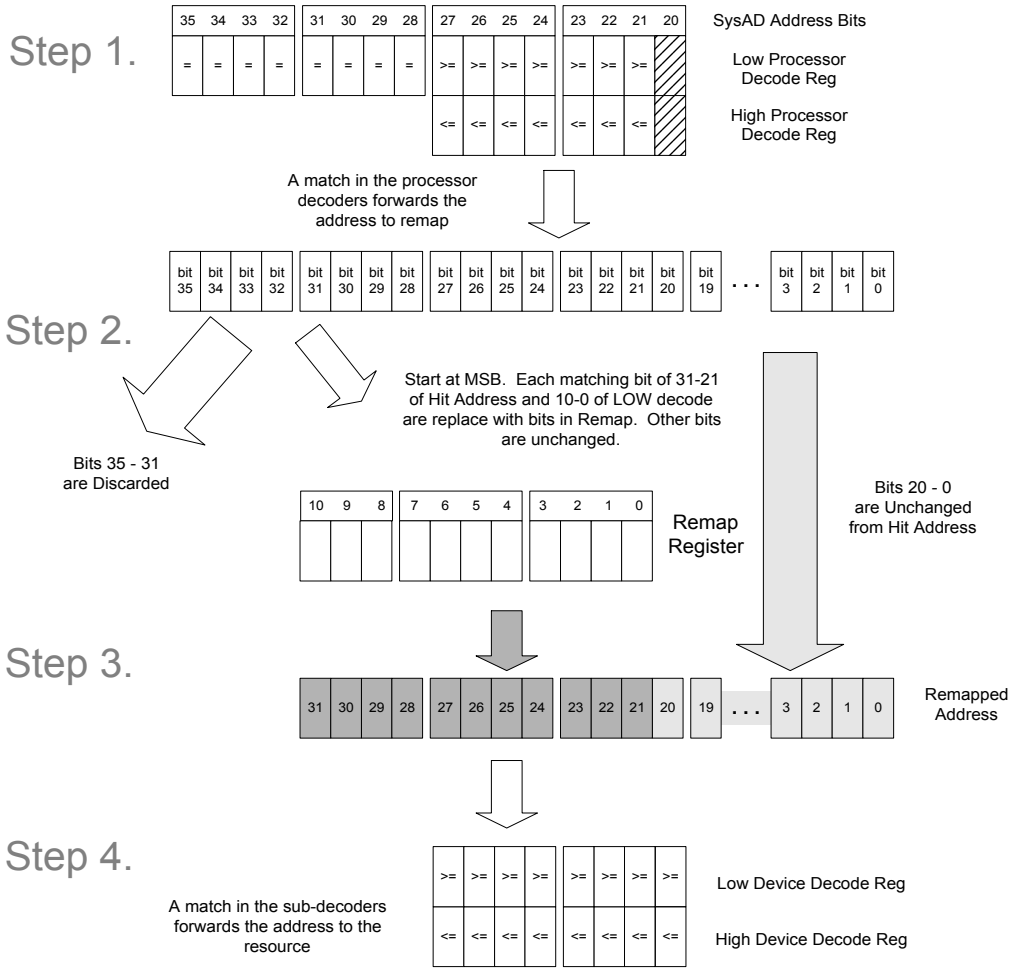
The offsets for these registers are 0x0d0 - 0x100 and each MAP register is 11 bits wide.

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 MSB to 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 [27:20] 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 3 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 20.16 "PCI Internal" on page 210. 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 18.

Table 18: 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

Notice that 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 0x001111. 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 CPU PCI Override

In default, 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 PCI window is 2Gbyte, 1Gbyte or the default.

When PCI override bits are set to 01, if bits[31:30] of SysAD matches bits [10:9] of PCI Mem0 Low decode address register, the transaction is directed to PCI Mem0. This effectively sets a 1Gbyte window to PCI. 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, if bit[31] of SysAD matches bit [10] of PCI Mem0 Low decode address register, the transaction is directed to PCI Mem0. This effectively sets a 2Gbyte window to PCI. 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).

NOTES: Setting PCI override bits to 11 is forbidden.

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

3.8 DMA PCI Override

In default, the DMA controller uses CPU interface address decoding. However, the DMA 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–64120A SysAD bus interface allows the CPU to gain access to the GT–64120A'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–64120A is a slave-only interface. The GT–64120A will never master the SysAD bus.

4.1 CPU Interface Signals[1]

The CPU interface incorporates the following signals:

Table 19: CPU Interface Signals

Signal	Description
Master Address/Data	Transfers multiplexed address/data.
SysCmd[8:0] - Master Port Command	Transfers information about the access (read/write, size) and the data (good/bad, last word).
SysADC[7:0] - Master Data Check	An 8-bit bus containing parity for the SysAD bus. SysADC is valid on data cycles only.
ValidOut*	Indicates that the local master is driving valid address/data/command on the SysAD bus.
ValidIn*	Indicates that the GT–64120A is driving valid data/command on the SysAD bus.
WrRdy*	Indicates that the GT–64120A is capable of accepting a write transaction up to eight 32-bit words in length.
Release*	Indicates to the GT–64120A 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*	An "OR" of all the internal interrupt sources on the GT–64120A.

The SysAD bus is synchronous with respect to TClk 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.

1. There is no RdRdy* signal output from the GT-64120. This signal should be tied LOW on the CPU as the GT-64120 is always ready to accept a read command.

4.2 SysAD, SysADC, and SysCmd Buses

The SysAD and SysCmd bus protocol implemented by the GT–64120A is completely compatible with the 64-bit Orion bus protocol used by the IDT R4xxx, R5000, and R7000 processors. The GT–64120A extends this protocol to support bursts less than 4 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–64120A 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 20: 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. Must be 0 for address cycles.
SysCmd[6]	0 = Read transaction (during address cycles) 1 = Write transaction (during address cycles) Must be 0 for data cycles.
SysCmd[5]	Indicates error status for data cycles. Must be 0 for address cycles.
SysCmd[4]	0 = Check the Data & Check-bits (during data cycles) 1 = Do not check Data (during data cycles) Must be 1 for address cycles
SysCmd[3:0]	Encoded to indicate size of the transfer during address cycles. Reserved during Data cycles.

Table 21: 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	Write 5 bytes.
0	0	1	0	1	1	1	0	1	Wr6Byte	Write 6 bytes.
0	0	1	0	1	1	1	1	0	Wr7Byte	Write 7 bytes.
0	0	1	0	1	1	1	1	1	WrDWord	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.
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 22: Read Response SysCmd[8:0] Encodings (driven by GT–64120A)

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 GT–64120A.

Table 23: 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

Table 23: CPU Write SysCmd[8:0] Encodings (driven by local master) (Continued)

SysCmd[8:0] Encoding[1]									Command Mnemonic	Command Description
8	7	6	5	4	3	2	1	0		
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 GT–64120A.

4.2.1 SysAD Read Protocol

SysAD reads occur in three phases:

Table 24: 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–64120A drives data on the SysAD bus and a read response on SysCmd.
Last-burst data	The GT–64120A 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–64120A together with SysAD data for every data transfer.

The address phase for all transactions begins with the assertion of ValidOut* to the GT–64120A. Valid address and command information must be present on SysAD and SysCmd during this phase. Release* must also be asserted to the GT–64120A to indicate that the local master is releasing mastership of the SysAD/SysCmd/SysADC buses to the GT–64120A 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–64120A will:

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

The GT–64120A 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–64120A floats the SysAD, SysADC, and SysCmd buses, returning ownership to the CPU.

Figure 7: Double Word Read Through Master With Parity Check Bits

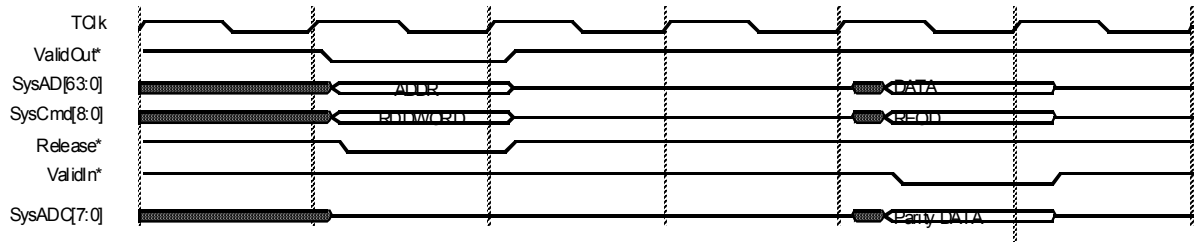
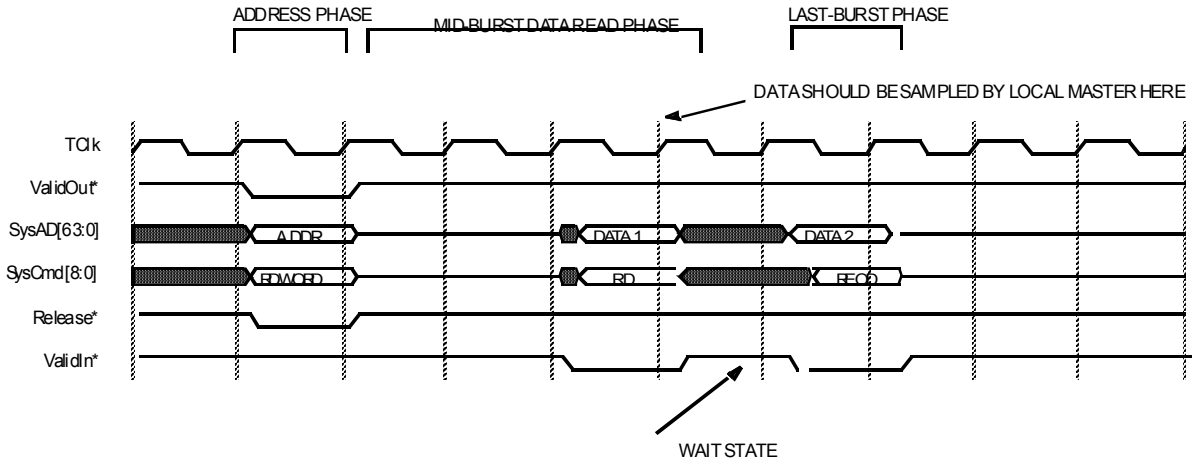


Figure 8: Four Word Burst Read through SysAD



4.2.2 SysAD Write Protocol

CPU writes occur in three phases:

Table 25: SysAD Write Phases

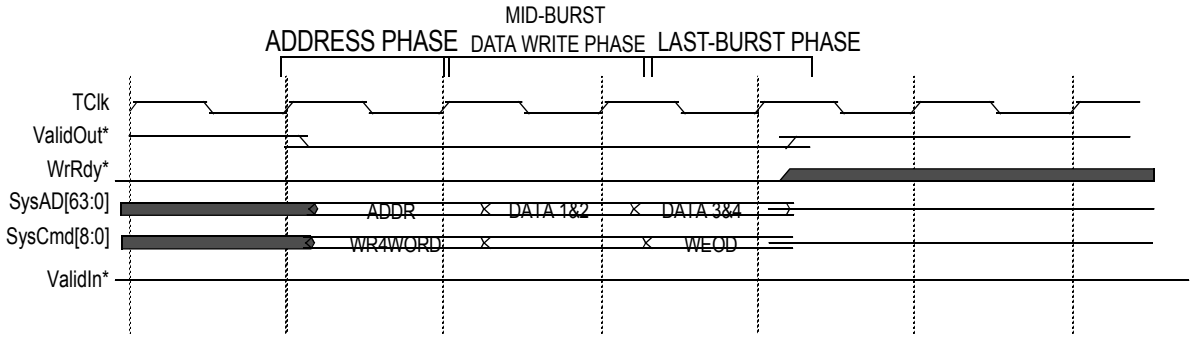
Phase	Description
Address	Address information is driven on the SysAD bus and command information is driven on SysCmd.
Mid-burst write data	The Local Master drives data on the SysAD bus, possibly Parity data on SysADC[7:0], and a write command (mnemonic = WD) on SysCmd.
Last-burst write data	The Local 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–64120A. 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–64120A 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–64120A 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 55. 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–64120A'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–64120A 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 MIPS Write Modes and Write Patterns Supported

The GT–64120A 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 MIPS 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.

4.5 CPU Interface Endianess

The GT–64120A provides the capability to swap the endianess 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 CPU endianess is programmed on RESET by sampling the Interrupt* pin, see Section 12. "Reset Configuration" on page 143. The setting of this pin also programs the Endianess bit in the CPU Configuration register at 0x000. When accessing the internal registers, the endianess of the data will be determined by the setting of Interrupt*.

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

The setting of the PCI Command register's MByteSwap and MWordSwap bits (see Table 194, on page 210) determines how data transactions from the CPU to, or from, the PCI are handled, along with the setting of the Endianess bit in the CPU Interface Configuration register (see Table 87, on page 180). Both MByteSwap and Endianess 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–64120A 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 Multiple GT–64120A Support

Up to four GT–64120A 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–64120A devices by sampling the value of DAdr[10] on RESET, see Section 12. “Reset Configuration” on page 143. If this pin is sampled HIGH, multiple GT–64120A’s 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–64120A.

If multiple GT–64120A devices are enabled, the values sampled on Ready* and CSTiming* determine the ID of the particular GT–64120A, as shown in Table 26. This sampled values also program the MultiGTAct bits [1:0] in the Multi-GTID Register, 0x120. These bits are programmable after RESET.

Table 26: Pin Strapping the GT–64120A ID

Pin	Configuration Function
Ready*, CSTiming*	Multi-GT–64120A Address ID
00- 01- 10- 11-	GT responds to SysAD[26,25]=00 GT responds to SysAD[26,25]= 01 GT responds to SysAD[26,25]=10 GT responds to SysAD[26,25]= 11 **NOTE:** Boot GT–64120A must be programmed to 11

4.7.1 Hardware Connections

When Multiple-GT–64120A devices are enabled, WrRdy*, ValidIn*, and ScDOE* have slightly different functionality.

*4.7.1.1 WrRdy**

When Multiple-GT–64120A devices are enabled, WrRdy* is an open-source output requiring a 4.7 KOhm pull-down resistor. All WrRdy* outputs from the GT–64120A devices must be tied together to drive the CPU WrRdy* input.

WrRdy* is driven LOW for one cycle before floating the output.

*4.7.1.2 ValidIn**

When Multiple-GT–64120A devices are enabled, ValidIn* is an open-drain output requiring a 4.7 KOhm pull-up resistor. All ValidIn* outputs from the GT–64120A devices must be tied together to drive the CPU ValidIn* input.

ValidIn* is driven HIGH for one cycle before floating the output.

*4.7.1.3 ScDOE**

When Multiple-GT–64120A devices are enabled, ScDOE* is an open-source output requiring a 4.7 KOhm pull-down resistor. All ScDOE* outputs from the GT–64120A devices must be tied together to drive the CPU and Secondary cache inputs.

ScDOE* is driven LOW for one cycle before floating the outputs.

4.7.2 Multi-GT Mode Enabled

When the Multi-GT mode 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*.

NOTE: Since 0x0.1FC0.0000 implies SysAD[26:25] == 3, the GT–64120A holding the boot device should be strapped to ID = 3.

- When the Multi-GT mode bit is CLEARED, the CPU Interface resumes normal address decoding.

NOTE: As long as Multi-GT mode bit is SET, there is no access to PCI, SDRAM and Devices, and DMA internal registers. There is access only to the CPU interface internal registers and to boot ROM.

4.7.3 Initializing a Multiple GT–64120A System

The following is the recommended procedure for initializing a system with two GT–64120A devices attached to the same CPU.

NOTE: For the following description, assume that the two GT–64120A devices are called GT-1 and GT-2, respectively. Both devices have DAdr[10] pulled to VCC (enabling Multi-GT mode). GT-1 has Ready* and CSTiming* tied to 11 (boot GT–64120A). GT-2 has Ready* and CSTiming* tied to 00. GT-1 has the bootrom.

Table 27: Initializing a Multiple GT–64120A 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, [11:0]=offset]. Registers on GT-2 are accessible via address [SysAD[26:25]=00, [11:0]=offset].

Table 27: Initializing a Multiple GT–64120A System

Initialization Steps	Description
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 Multi-GT mode. Further, the address mapping of register and memory space in the GT–64120A and on their interfaces must be unique. In other words, the four PCI address ranges, two SDRAM ranges, I/O space, and internal GT–64120A 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.	

Now both GT–64120A devices resume NORMAL operation with USUAL address decoding.

NOTE: In Multi-GT mode, the GT–64120A 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–64120A devices in the system, ValidIn* is not returned to the CPU and the system will hang.

4.7.4 Multi-GT Restrictions

1. Due to System Interface loading, maximum operating frequency will decrease as the number of GT–64120A devices increase.
2. When in multi-GT configuration, the GT–64120A only supports R4000 writes.

4.8 CPU Interface Restrictions

1. The GT–64120A does not support access of more than 4 bytes to internal space.
2. Block read/write to 8 or 16 bit wide devices are not allowed. It means that data placed in such devices must not be cacheable.

5. MEMORY CONTROLLER

The GT–64120A 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–64120A 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–64120A.)

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

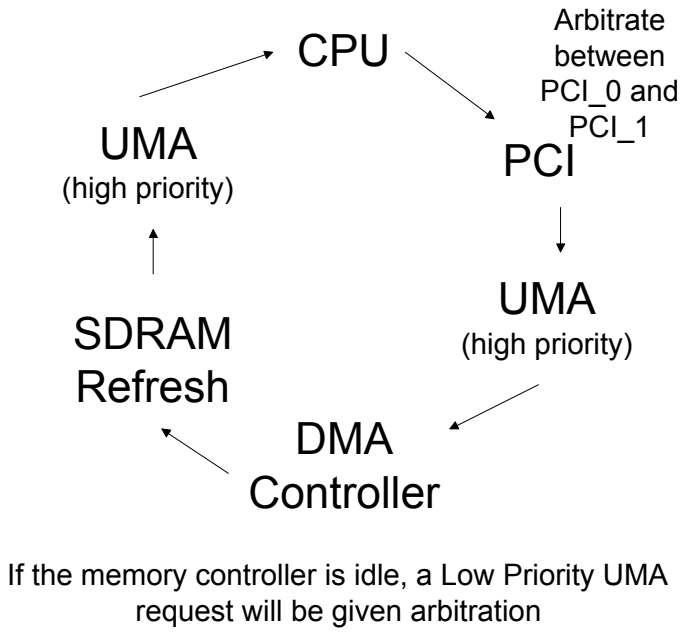
NOTE: A device may not master transactions via the GT–64120A's memory controller.

The GT–64120A'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–64120A implements a round robin arbitration scheme for requests of the memory controller, as shown in Figure 10.

Figure 10: Memory Controller Arbitration



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 5.1.5 “SDRAM Address Decode Register (0x47c)” on page 65 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–64120A.

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–64120A.

5.1.1.1 Refresh Rates

The GT–64120A implements standard SCAS before SRAS refreshing.

Refresh rates for each bank can be programmed to occur at different frequencies according to the RefIntCnt, a 14-bit value SDRAM configuration register. For example, the default value of RefIntCnt is 0x200. If TClk is 100 MHz, than 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–64120A 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 75.

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 11.

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 12.

Figure 11: Non-Staggered Refresh Waveform

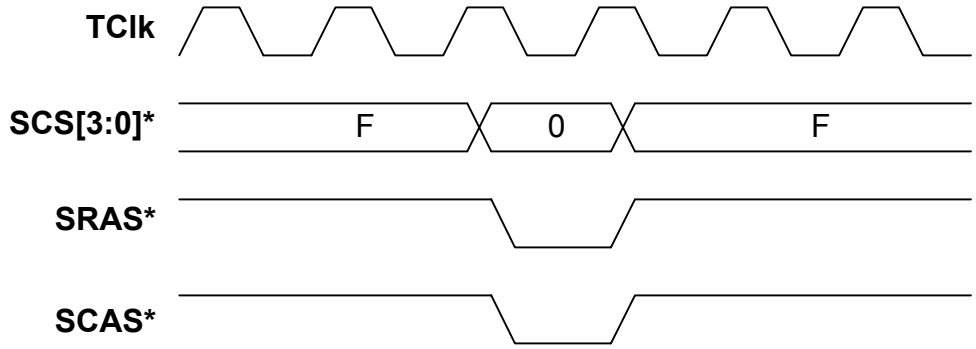
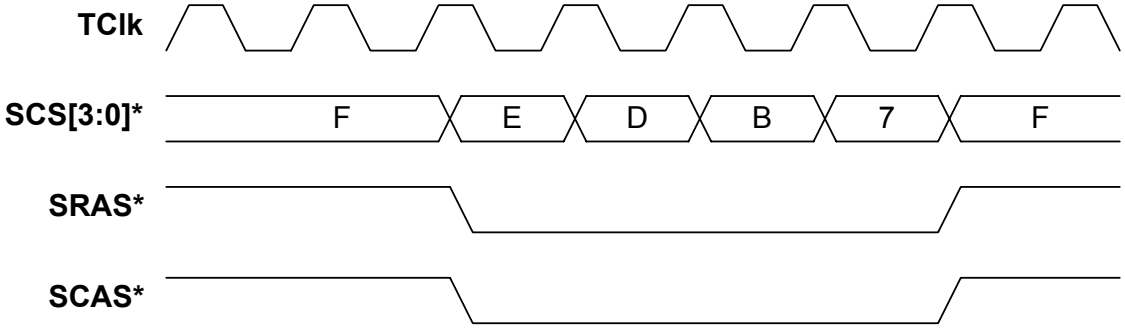


Figure 12: Staggered Refresh Waveform



5.1.1.3 Read Modify Write Enable/ECC

The GT–64120A supports Error Checking and Correction of 64-bit wide SDRAMs.

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.

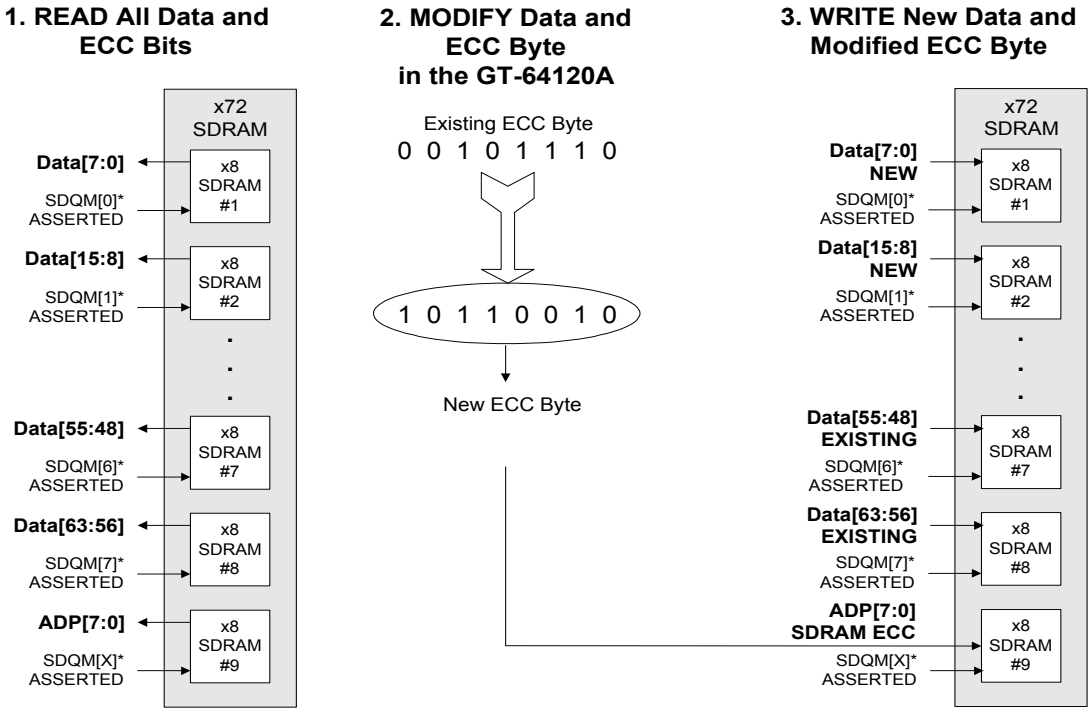
Bit 15 enables or disables read modify write protocol to SDRAM. If ECC is enabled in any of the DRAM banks, this bit must be set to 1 to enable read modify write.

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–64120A.
3. Write the new data and new ECC byte.

Figure 13 illustrates the procedure that the GT–64120A uses to generate ECC in a partial write to SDRAM.

Figure 13: 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 DMAReq0*/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 BypsOE* 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 Duplicating DMA End of Transfer Pins

The DMA controllers use the End of Transfer pins to give an external device the ability to terminate a current DMA transfer. See Section 9.2.15 “End Of Transfer Enable, EOTE, bit 18” on page 130 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, EOT[0].

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 28: 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–64120A SDRAM controller can be configured to interface registered SDRAM DIMMs.

Setting bit 23 to 1means 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 Both Bank

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 Mode Register

Each SDRAM has its own Mode Register. The Mode Register is used to define the specific mode of operation for the SDRAM. This definition includes the selection of a burst length, SCAS latency, burst type, 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–64120A has the flexibility to allow the CPU or a PCI Master to update the SDRAM's Mode Register at any time during operation.

The parameters that the GT–64120A can change are the CAS latency and the burst length. In order to change these parameters in the SDRAM's Mode Register:

1. The corresponding SDRAM Bank Parameters Register (0x44c - 0x458) must be updated 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–64120A 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.[1]

The GT–64120A automatically initializes the SDRAM on boot up to Sub-block burst ordering as required for MIPS CPUs block reads.

NOTE: The GT–64120A can be set 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 which gives the software the best chance of interleaving and enhancing overall system performance.

The row and column address translation is different for 16 Mbit, 64/128 Mbit or 256 Mbit SDRAMs as well as 32-bit and 64-bit SDRAM banks. The address decoding depends on the setting of AddrDecode at 0x47c. See

1. The GT–64120A 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.

Table 29 through Table 33 for the SRAS* and SCAS* address translation from the SysAD interface and PCI.

Table 29: 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 30: SysAD/PCI Address Decoding for 64-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	5, 22-12	5, “0”, 24-23, 11-6, 4-3
001	6, 22-12	6, “0”, 24-23, 11-7, 5-3
010 (UMA)	11, 22-12	11, “0”, 24-23, 10-3
011	13, 22-14, 12-11	13, “0”, 24-23, 10-3
100	21, 22, 20-11	21, “0”, 24-23, 10-3
101	22, 21-11	22, “0”, 24-23, 10-3
110	23, 22-12	23, “0”, 24, 11-3 (only for x4 & x8)
111	24, 22-12	24, “0”, 23, 11-3 (only for x4)

Table 31: SysAD/PCI Address Decoding for 32-bit SDRAM, 64 Mbit

AddrDecode, 0x47c	SysAD/PCI Bits used for SRAS* on BankSel[1:0], DAdr[11:0]	SysAD/PCI Bits used for SCAS* on BankSel[1:0], DAdr[11:0]
000	Illegal setting for 64, 128Mbit and 256Mbit SDRAM	
001	6, 5, 23-12	6, 5, “00”, 25-24, 11-7, 4-2

Table 31: SysAD/PCI Address Decoding for 32-bit SDRAM, 64 Mbit (Continued)

AddrDecode, 0x47c	SysAD/PCI Bits used for SRAS* on BankSel[1:0], DAdr[11:0]	SysAD/PCI Bits used for SCAS* on BankSel[1:0], DAdr[11:0]
010	12, 11, 23-13, 10	12, 11, “00”, 25-24, 9-2
011	13, 12, 23-14, 11-10	13, 12, “00”, 25-24, 9-2
100	21, 20, 23-22, 19-10	21, 20, “00”, 25-24, 9-2
101	23, 22, 21-10	23, 22, “00”, 25-24, 9-2
110	24, 23, 21-10	24, 23, “00”, 25, 22, 9-2 (only for x4 & x8)
111	25, 24, 21-10	25, 24, “00”, 26, 22, 9-2 (Only for x4)

Table 32: SysAD/PCI Address Decoding for 64-bit SDRAM, 64/128 Mbit

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

Table 33: SysAD/PCI Address Decoding for 64-bit SDRAM, 256 Mbit

AddrDecode, 0x47c	SysAD/PCI Bits used for SRAS* on BankSel[1:0], DAdr[12:0]	SysAD/PCI Bits used for SCAS* on BankSel[1:0], DAdr[12:0]
000	Illegal setting for 64, 128Mbit and 256Mbit SDRAM	
001	7, 6, 25-13	7, 6, 29-28, “0”, 27-26, 12-8, 5-3
010	12, 11, 25-13	12, 11, 29-28, “0”, 27-26, 10-3

Table 33: SysAD/PCI Address Decoding for 64-bit SDRAM, 256 Mbit (Continued)

AddrDecode, 0x47c	SysAD/PCI Bits used for SRAS* on BankSel[1:0], DAdr[12:0]	SysAD/PCI Bits used for SCAS* on BankSel[1:0], DAdr[12:0]
011	14, 13, 25-15, 12-11	14, 13, 29-28, “0”, 27-26, 10-3
100	22, 21, 25-23, 20-11	22, 21, 29-28, “0”, 27-26, 10-3
101	24, 23, 25, 22-11	24, 23, 29-28, “0”, 27-26, 10-3
110	25, 24, 23-11	25, 24, 29-28, “0”, 27-26, 10-3
111	26, 25, 27, 22-11	26, 25, 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–64120A. 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–64120A 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.

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–64120A 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–64120A 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–64120A supports a maximum of 16M addresses, 14 address bits for SRAS and 10 address bits for SCAS (DAdr[11] is ignored).

With 128 MBit SDRAMs, the GT–64120A supports a maximum of 32M addresses, 14 address bits for SRAS and 11 address bits for SCAS.

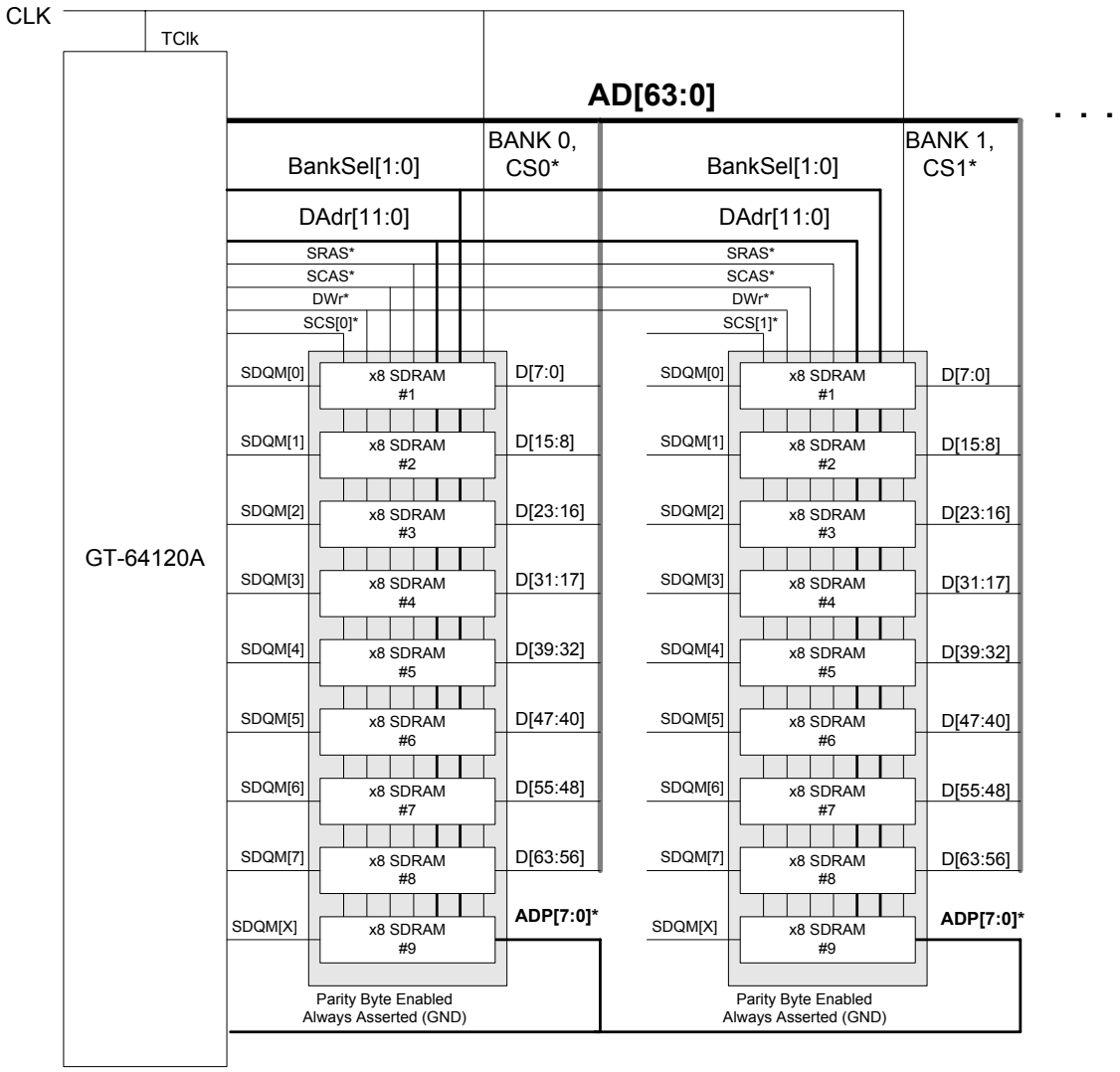
5.2.3 256 Mbit SDRAMs

For 256 MBit SDRAMs, DAdr[12:0] and BankSel[1:0] are outputs of the GT–64120A and must be directly connected 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 MBit SDRAMs, the GT–64120A supports a maximum of 128M addresses, 15 address bits for SRAS and 12 address bits for SCAS.

Figure 14: 64/128 Mbit/64-bit SDRAM Connection to Memory Bus Using x8 Devices



5.3 Programmable SDRAM Parameters

The SDRAM controller of the GT–64120A 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 Mode Register" on page 64 to update the SDRAM's Mode Register.

5.3.0.1 SCAS Latency*

SCAS* Latency is the number of TClks from the assertion of SCAS* to the sampling of the first read data. This parameter can be programmed to be either 2 or 3 TClks. Selecting this parameter depends on TClk frequency and the speed grade of the SDRAM.

NOTE: Check your SDRAM data sheet for the most optimal setting.

5.3.0.2 Flow-through

Bit 2 specifies the number of times that the data is sampled by the GT–64120A 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.

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).

5.3.0.3 SRAS Precharge*

Bit 3 specifies the SRAS precharge time.

Setting this bit to 0 means the SRAS to precharge time is two TClk cycle.

Setting this bit to 1 means the SRAS to precharge time is three TClk cycles.

This parameter specifies the number of TClks following a precharge cycle that a new SRAS* transaction may occur.

5.3.0.4 64-bit Interleaving

Bit 5 specifies how many banks 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.

Setting this bit to 0 means the GT–64120A interleaves between two banks.

Setting this bit to 1 means the GT–64120A interleaves between four banks.

5.3.0.5 Bank Width

Bit 6 specifies the data width of the particular bank.

Setting this bit to 0 means the bank width is 32 bits.

Setting this bit to 1 means the bank width is 64-bit.

5.3.0.6 Bank Location

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.

Setting this bit to 0 means the 32-bit SDRAM is controlled on the even bank, AD[31:0].

Setting this bit to 1 means the 32-bit SDRAM is controlled on the odd bank, AD[63:32].

5.3.0.7 ECC Support

Bit 8 enables or disables ECC on a 64-bit wide SDRAM bank.

Setting this bit to 1 means ECC is enabled, indicating a 72-bit DIMM.

5.3.0.8 64-bit Bypass Mode for CPU Reads

Bit 9 enables or disables 64-bit SDRAM Read bypass.

NOTE: This option is only for SDRAM banks configured as 64-bit (the option is not available for devices).

There is an optional bypass mode for CPU reads where a clock cycle of latency can be eliminated when the CPU executes read cycles from 64-bit SDRAM. Using the bypass mode requires the addition of bus switches to enable the flow of data to the CPU directly. Instead of passing response data from the SDRAM to the GT–64120A 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 TClk cycles to 8.

Setting this bit to 0 means bypass is disabled. All reads from 64-bit SDRAM flow through the GT–64120A before being presented to the CPU on the SysAD lines.

Setting this bit to 1 means the BypsOE* output signal is active on any reads from a 64-bit SDRAM (64/32/16/8 bit reads) and controls the bus switches which directly connect the CPU's SysAD lines to the SDRAM data lines.

NOTE: The bypass is used for partial reads as well. Reads from 64-bit SDRAM are no longer sampled by the GT–64120A prior to presenting the data to the CPU.

64-bit bypass can only be enabled if the bank is set for 64-bit (bit 6 = 1).

No writes are ever transferred via the bypass switches.

5.3.0.9 SRAS to SCAS**

Bit 10 specifies the number of TClks that the GT–64120A inserts between the assertion of SRAS* with a valid row address to the assertion of SCAS* with a valid column address.

Setting this bit to 0 means the SRAS to SCAS latency is 2 Tclk cycle. This is the default setting.

Setting this bit to 1 means the SRAS to SCAS latency is 3 TClk cycles.

5.3.0.1016/64/128/256 MBit SDRAM Configuration

Bits [14,11] specify when the particular bank supports 16, 64/128, or 256 MBit SDRAMs.

Setting these bits to 00 means the bank supports 16-MBit SDRAMs.

Setting these bits to 01 means the bank supports 64/128-Mbit Swims.

Setting these bits to 11 means the bank supports 256-Mbit SDRAMs.

NOTE: The value of 10 is a reserved setting and should not be used.

5.3.0.11Burst Length

Bit 13 specifies the data burst length supported for the particular SDRAM bank.

Setting this bit to 0 means the bank supports eight data bursts.

Setting this bit to 1 means the banks supports four data bursts.

NOTE: The data can be either 32 or 64 bit, depending on the setting of bit 6, see Section 5.3.0.5 “Bank Width” on page 70.

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–64120A’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 34 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 34: CPU SDRAM Performance on Reads

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

1. See Section 5.3.0.8 “64-bit Bypass Mode for CPU Reads” on page 71 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 35: 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–64120A.
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–64120A.

There are five different sync. modes that the GT–64120A can be placed in depending on the PClk/TClk ratio. These sync. modes are:

Table 36: GT–64120A Sync. Modes

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

1. If TClk and PClk are synchronized, see Table 306, on page 262.

Table 37: SDRAM Performance Summary PCI Read Accesses

TClk/PClk Frequency (MHz)	Sync. Mode	Event	# of TClk	# of PClk
83/33	0	A B C	9 12 26	3 5 10
	5	A B C	10 12 26	3 5 10

Table 37: SDRAM Performance Summary PCI Read Accesses (Continued)

TClk/PClk Frequency (MHz)	Sync. Mode	Event	# of TClk	# of PClk
83/50	0	A B C	8 8 20	4 5 12
	1	A B C	8 8 20	4 5 12
83/66	0	A B C	5 6 16	4 5 12
	1	A B C	6 6 16	4 5 12
100/33	0	A B C	14 12 30	5 4 10
	5	A B C	14 12 30	5 4 10
	6 or 7	A B C	12 8 24	4 3 8
100/50	0	A B C	11 9 24	6 5 12
	1	A B C	11 9 24	6 5 12
	2	A B C	9 5 18	5 3 9
100/66	0	A B C	9 8 21	6 6 14
	1	A B C	9 8 21	6 6 14

5.5 SDRAM Interleaving

The GT–64120A 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 DMA controllers.

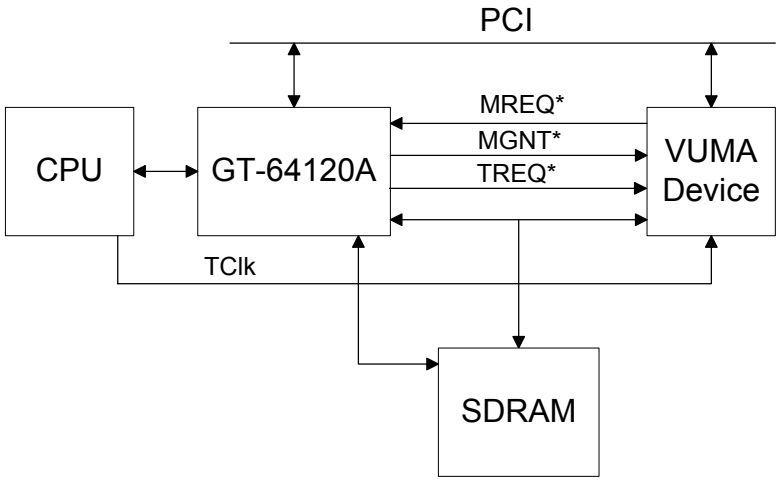
At the end of every SDRAM memory transaction, the GT–64120A will precharge the bank.

5.6 Unified Memory Architecture (UMA) Support

The GT–64120A supports UMA. This feature allows an external master device to share the same physical SDRAM memory is controlled by the GT–64120A. This feature works according to the VESA Unified Memory Architecture (VUMA) specification[1].

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 15.

Figure 15: VUMA Device and GT–64120A sharing SDRAM



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

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 BypsOE* 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–64120A and must be an output for the VUMA device. This signal is used by the VUMA device to make a request to the GT–64120A to access the shared SDRAM. MREQ* should be driven by the VUMA device on a rising edge of TClk. MREQ* is sampled on a rising edge of TClk by the GT–64120A.

MGNT* is an output of the GT–64120A and must be an input to the VUMA device. This signal is used by the GT–64120A to inform the VUMA device that it can access the shared SDRAM. MGNT* is driven by the GT–64120A on a rising edge of TClk. MGNT* must be sampled by the VUMA on a rising edge of TClk.

The AC Timing parameters are shown Figure 22 on page 5–86.

Table 38: UMA AC Timing Parameters

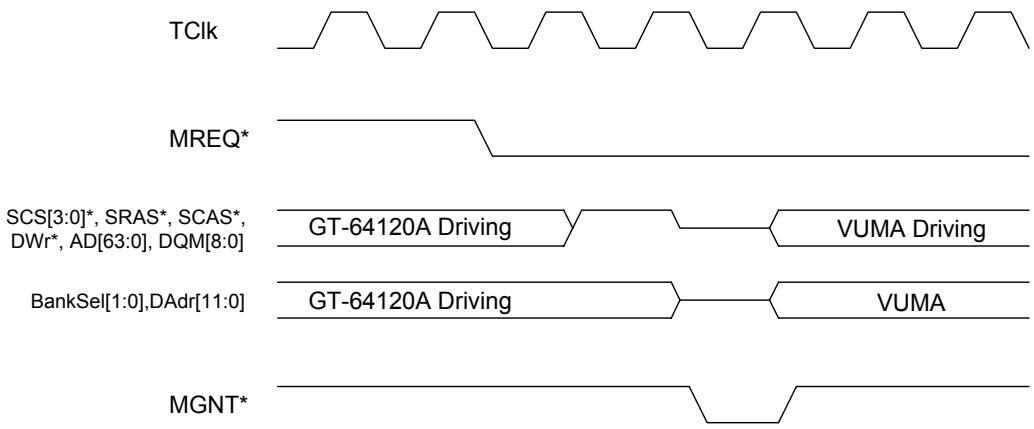
Signals	Description	Min.	Max.	Unit	Loading
MREQ*	Setup.	3		ns	
MREQ*	Hold.	1		ns	
MGNT*	Output Delay From TClk rising.	2	10	ns	30 pF

5.6.2 SDRAM Pins

Once MGNT* is asserted by the GT–64120A and the VUMA 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–64120A regains access to SDRAM. During this period, the VUMA device must drive these signals to access SDRAM.

When the GT–64120A 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 16.

Figure 16: Handing the Bus Over



5.6.3 Address Decoding

The GT–64120A 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 65 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

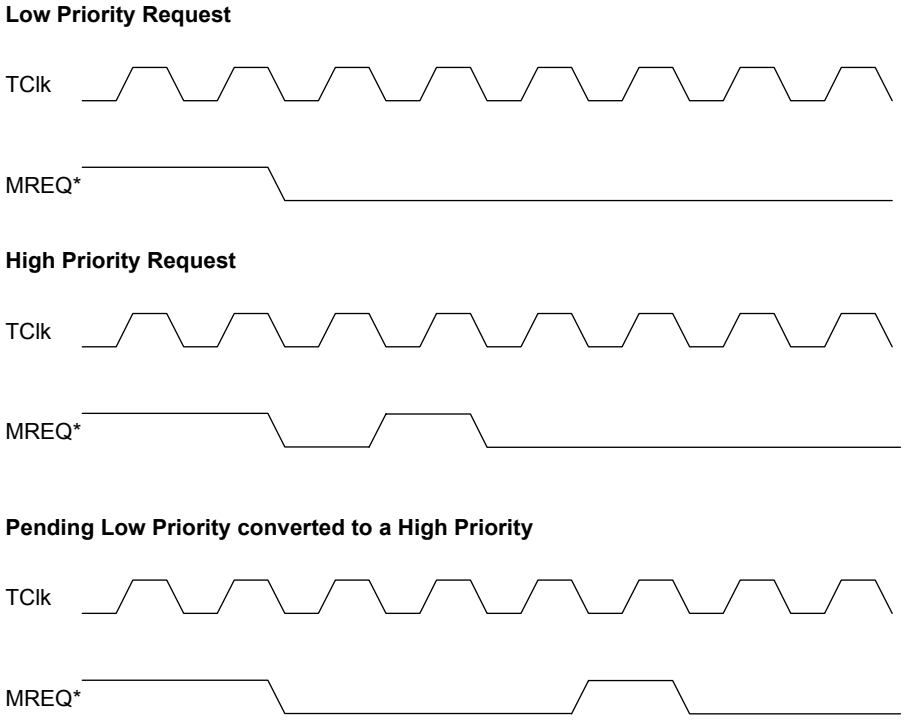
As shown in Figure 15 on page 5–75, the VUMA device arbitrates with the GT–64120A for access to SDRAM through MREQ* and MGNT*, which should be synchronous to TClk.

The GT–64120A is always the default owner of the SDRAM and access to this memory is only allowed to the VUMA upon demand.

The GT–64120A has the right to take the VUMA device off of the bus by de-asserting MGNT*.

VUMA devices 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 17: MREQ* Requests from the VUMA Device[1]



5.6.4.1 VUMA 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 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 deassert 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 reassertion, the VUMA device must keep MREQ* asserted until the GT–64120A gives the VUMA access to SDRAM via MGNT*.
2. The VUMA 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 samples MGNT* as asserted, it gains and retains access to SDRAM until MREQ* is de-asserted.
4. The GT–64120A always asserts MGNT* for one clock cycle only, and immediately requests back own-

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

ership of the bus.

5. The VUMA retains ownership of SDRAM indefinitely. The standard calls for the VUMA device to keep ownership for no longer than 60 TClks before it must release the bus. This is not a requirement for the GT–64120A 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 de-asserts MREQ* to transfer ownership back to the GT–64120A either on its own, or because of a preemption requires, MREQ* should be de-asserted for at least 2 TClks before asserting it again to raise a request.

5.6.5 Latencies, Low and High Priority

If a VUMA places a low priority request for access to SDRAM, there is no set time specified by the GT–64120A to assert MGNT*. Once there are no pending transactions to the memory controller, MGNT* is asserted.

If a VUMA places a high priority request for an access to SDRAM, the GT–64120A has a maximum of 35 TClks before it asserts MGNT*.

5.6.6 Total Request

The GT–64120A immediately requests back ownership of the bus after MGNT* assertion.

This may cause some difficulty to implement a fair arbitration mechanism on the SDRAM bus. 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–64120A that requires SDRAM bus ownership.

5.6.7 Disable Refresh

In some applications, the GT–64120A 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 refresh cycles of the GT–64120A (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–64120A 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–64120A must de-assert its request of the memory.

The internal register read transaction is held off until the GT–64120A obtains mastership of the memory bus.

5.7 Device Controller

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

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

The five individual chip selects are typically broken up into four individual device banks plus one chip select for a boot device (non-volatile memory). Each device bank 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 373. 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 bank's read timing parameters. ALEtoWr, WrActive, and WrHigh are set for each bank's write timing parameters. There are certain restrictions to setting these timing parameters. See Section 5.9 "Memory Controller Restrictions" on page 87 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 84.

5.7.1 TurnOff, bits [2:0]

TurnOff is the number of TClk cycles that the GT–64120A does not drive the memory bus after a read from a device. This prevents contentions on the memory bus after a read cycle for a slow device.

This parameter is measured from the number of cycles between the deassertion 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, bits [6:3]

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–64120A.

NOTE: This parameter can be extended by the Ready* pin. See Section 5.7.10 "Ready* Support" on page 84.

5.7.3 AccToNext, bits [10:7]

AccToNext defined as 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

1. 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–64120A has no affect on system performance or architecture.

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 84.

5.7.4 ALEtoWr, bits[13:11]

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, bits[16:14]

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 84.

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

5.7.6 WrHigh, bits[19:17]

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.

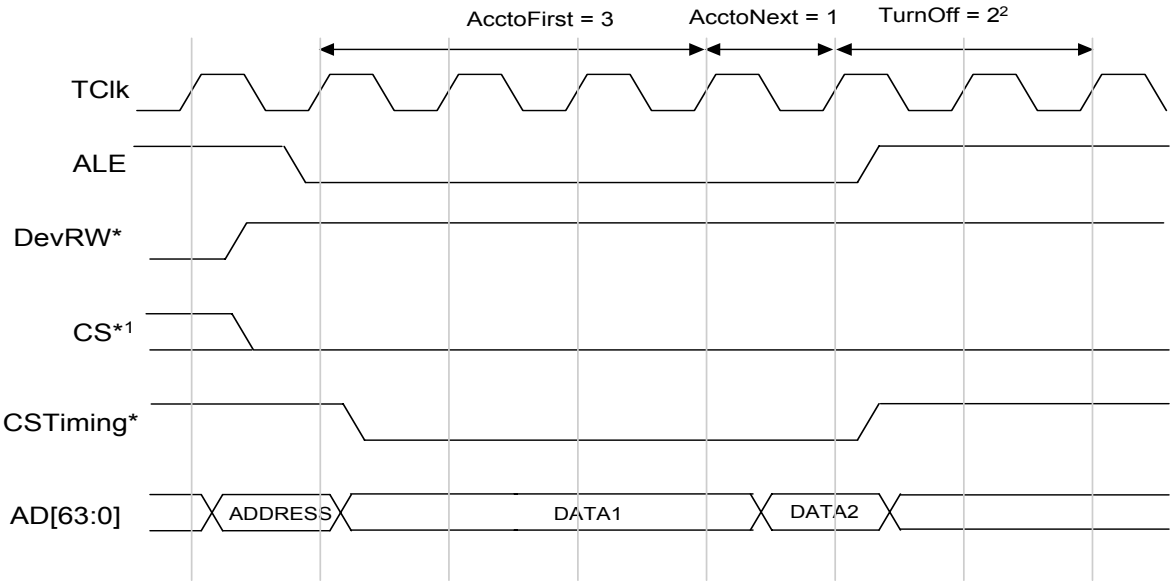
5.7.7 Device Bank Width and Location

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.

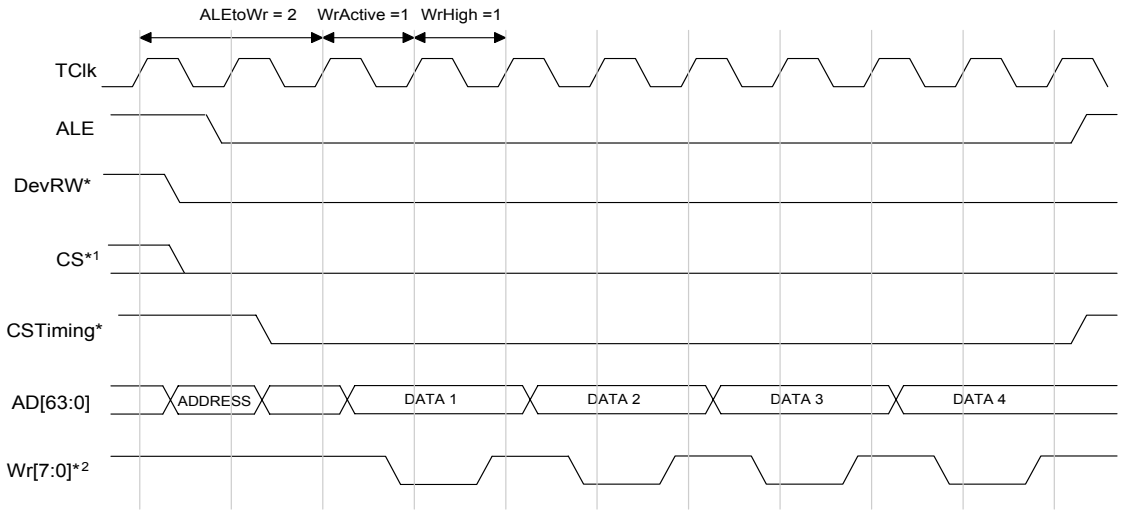
In case of a 64-bit device, DevLoc has no meaning.

Figure 18: Waveform Showing Device Read 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. GT-64120A may start a new AD cycle after TurnOff.

Figure 19: 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 (BAdr[2:0]) that is different from the latched address on the multiplexed AD bus.

NOTE: BAdr[2:0] are the same pins as the least significant SDRAM address lines, DAdr[2:0]. See Section 2. "Pin Information" on page 17 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–64120A 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.

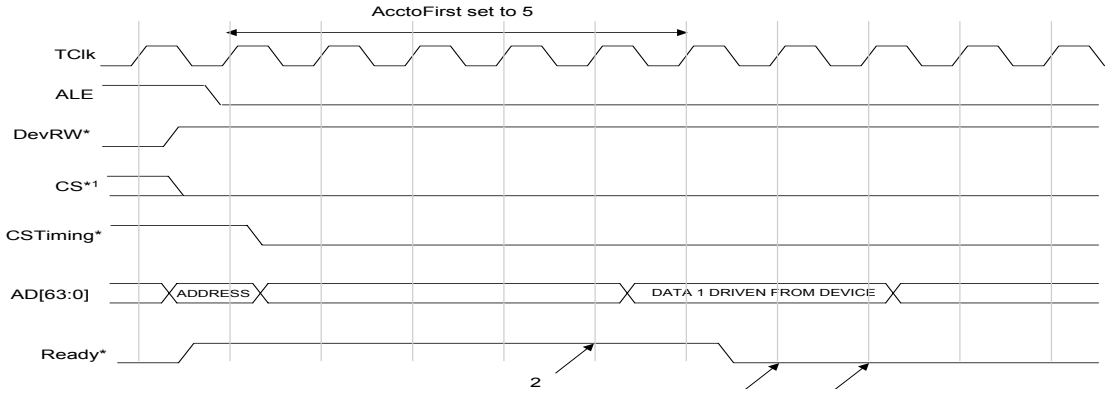
5.7.10 Ready* Support

The Ready* pin is sampled on three occasions: one clock before the data is sampled to the GT–64120A during both AccToFirst (see Figure 20) and AccToNext (see Figure 21) phases of read cycles and 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–64120A.

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.

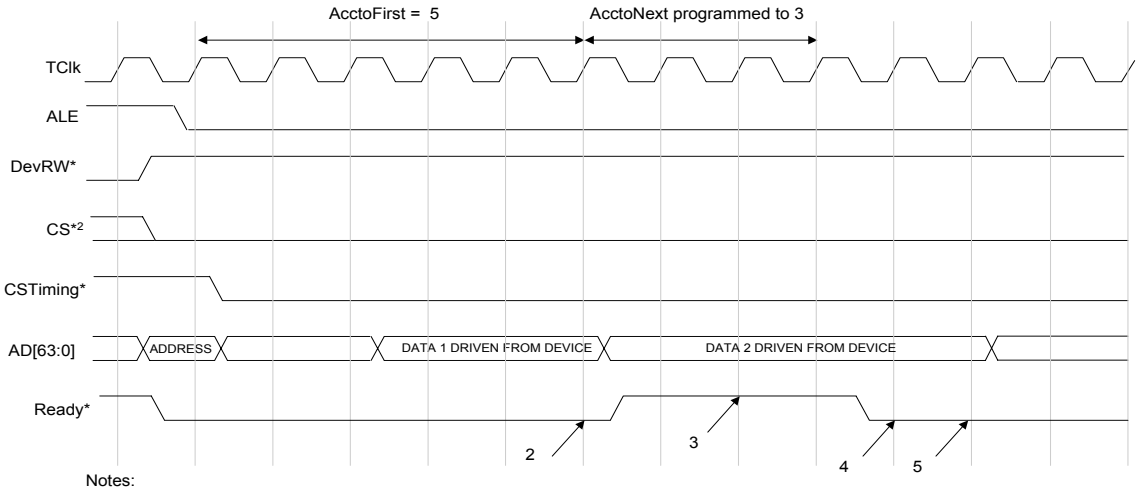
NOTE: While Ready* is de-asserted, SDRAM refresh requests are not serviced. Therefore, it is important to confirm that Ready* is not de-asserted indefinitely.

Figure 20: Ready* Extending AccToFirst on Read Cycle



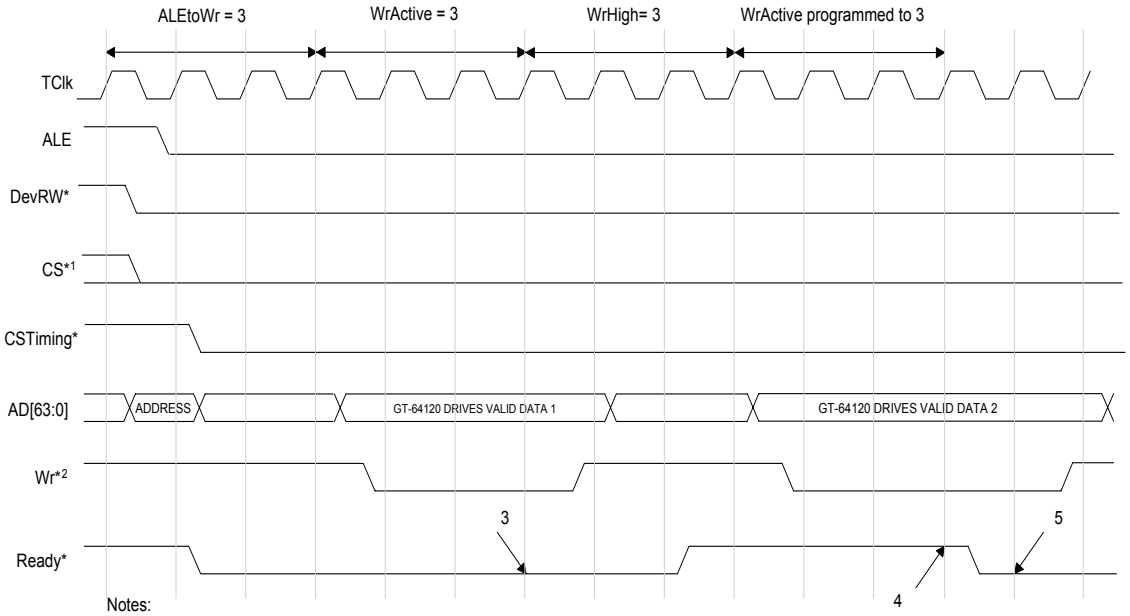
Notes:
1. CS* is driven off the same rising TClk* as ALE. Throughout consecutive transactions to the same device, CS* will remain asserted. This is why CS* must ALWAYS be qualified with CSTiming*.
2. Ready* is sampled as deasserted one clock before data should be sampled according to AcctoFirst. AD[63:0] is NOT sampled by the GT-64120A on the following rising TClk.
3. Ready* is asserted on some following rising TClk.
4. AD[63:0] (DATA 1) is sampled on the following rising TClk of the rising TClk that Ready* was asserted. Effectively, AccToFirst is 7 in this example (instead of the programmed5).

Figure 21: 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 deasserted 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 22: Extending WrActive Parameter on Write 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. Wr[7:0]* are asserted and deasserted from the falling edge of TClk.
3. Ready* is asserted on last rising TClk of WrActive, therefore, the GT-64120A assumes the device is written to correctly and continues to the next write cycle.
4. Ready* is deasserted on the last rising TClk of WrActive, therefore, the GT-64120A does NOT continue to the next write cycle effectly extending the WrActive parameter. Wr* remains asserted.
5. Ready* is asserted on the next rising TClk, therefore, the GT-64120A 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 TClk denoted by 5, not 4.

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 12. “Reset Configuration” on page 143.

If ECC is not enabled AND the ADP lines are not programmed to function as other pins on RESET, ADP[3:0] functions as EOT[3:0], see Section 9.2.15 “End Of Transfer Enable, EOTE, bit 18” on page 130. Further, ADP[4] will be used as BankSel[1] and ADP[5] will be used as DAdr[11]. ADP[7:6] are unused.

Table 39 summarizes the functionality of the ADP lines depending on system configuration.

Table 39: 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 wide device, 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–64120A 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. Access to SDRAM during SDRAM initialization time after reset results in unpredictable behavior.
8. When SDRAM CAS latency is 3, RAS precharge time (bit 3 of SDRAM parameter register) must be programmed to 1.
9. When using 64/256Mbit SDRAM, address decode 0 is not allowed.
10. In order for memory controller to return data of an internal register read, it must have control over the AD bus.
11. When using aggressive prefetch, the upper address allowed to be read from the PCI is last DRAM address minus 0x8.
12. Table 40 describes the limitation of a 32 bit device in the GT–64120A.

Table 40: 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	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. **NOTE:** If Ready* is used, the GT–64120A must sample it active an even number of times. i.e., once for address 0 and once for address 4 (see example 1 above).
Workaround	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.

6. DATA INTEGRITY

GT–64120A 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–64120A 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 41. For example, data bit 12 has the check value of 01100001, and check bit 5 has the check value of 00100000.

Table 41: ECC Code Matrix

Check Bit	Data Bit	ECC Code Bits 7	6	5	4	3	2	1	0	Number of 1s in
	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 41: ECC Code Matrix (Continued)

Check Bit	Data Bit	ECC Code Bits 7	6	5	4	3	2	1	0	Number of 1s in
	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 41: ECC Code Matrix (Continued)

Check Bit	Data Bit	ECC Code Bits 7	6	5	4	3	2	1	0	Number of 1s in
	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–64120A 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 41, 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–64120A 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–64120A runs a read modify write sequence.

For a read transaction from a 64-bit wide SDRAM, if ECC is enabled, the GT–64120A 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 41 shows that this syndrome corresponds to check bit 3. GT–64120A 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 41. GT–64120A 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–64120A reports an error but will not change the data.

NOTE: ECC is not supported in bypass mode (data must go through GT–64120A in order to be checked and corrected).

6.1.2 ECC Error Report

If the GT–64120A 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 Mem 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–64120A 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–64120A 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–64120A detects bad parity and PErrEn bit in Status and Command Configuration register is set, it asserts PERR*.

As a target, the GT–64120A 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–64120A detects bad parity and PErrEn bit in Status and Command Configuration register is set, it asserts PERR*.

The GT–64120A asserts SERR* if one of the following conditions 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.

6.3 Parity Support for Devices

There is no dedicated logic in the GT–64120A 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–64120A 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–64120A samples the incoming data driven on SysAD and the parity driven on SysADC. In case of parity error, the GT–64120A 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–64120A samples the SysADC (CPU parity) lines when the CPU performs a write transaction to PCI or SDRAM. Parity checking is performed by the GT–64120A.

If a parity error occurs, the GT–64120A latches the address, data, and parity lines, see Section 6.6 “Register Information” on page 96. The GT–64120A 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–64120A 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 96). 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–64120A samples the ADP (SDRAM ECC) lines when the CPU performs a read transaction from the SDRAM. An ECC check is performed by the GT–64120A SDRAM interface.

In case there is a 2-bit ECC error, the GT–64120A 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–64120A can be programmed to interrupt the CPU on single bit ECC errors, see Section 6.6 “Register Information” on page 96. 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–64120A ECC error report registers, see Section 6.6 “Register Information” on page 96.

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–64120A SDRAM

When a PCI master writes data to the GT–64120A’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–64120A 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–64120A's SDRAM, the GT–64120A samples the ADP (SDRAM ECC) lines. An ECC check is performed by the GT–64120A.

In case there is a 2-bit ECC error, the GT–64120A 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 96, 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–64120A 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–64120A can be programmed to interrupt the CPU on single-bit ECC errors, see Section 6.6 "Register Information" on page 96. 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–64120A ECC error report registers.

6.6 Register Information

Table 42 describes the registers used to implement parity and ECC in the GT–64120A.

Table 42: 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).

Table 42: Registers for Implementing Parity and ECC (Continued)

Register	Offset	Description
ECC Error Address	0x490	• 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–64120A 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.

NOTE: CPU Error Addresses and ECC Error Addresses contain an address in little endian mode. If the CPU is configured to big endian, CPU reads from internal registers result in byte swaps. This means that the read addres is swapped.

7. PCI INTERFACES

The GT–64120A 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 I_2O specification is also included.

7.1 Reset Configuration

At reset, the GT–64120A can be configured to have one 64-bit PCI interface or two 32-bit PCI interfaces, see Section 12. “Reset Configuration” on page 143. 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–64120A 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–64120A needs to decode the target address to determine which address space is being accessed.

If the GT–64120A is configured as one 64-bit PCI interface, then PCI_0 registers are used for address comparison and remapping.

If the GT–64120A is configured with two 32-bit PCI interfaces, then 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–64120A 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–64120A does not issue Dual Address cycles (DAC) or Lock cycles on the PCI.

The PCI posted write buffer in the GT–64120A 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 through the CPU can have the data byte order swapped as the data moves through the GT–64120A. Byte swapping is turned on via the MByteSwap bit in the PCI Internal Command register (0xc00.)

When the GT–64120A 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–64120A 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).

If the GT–64120A is configured to have two 32-bit PCI interfaces, then each master interface includes its own FIFO of eight entries, each 64-bits wide. 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.

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–64120A 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–64120A 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–64120A 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–64120A'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.

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–64120A 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–64120A 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.

Figure 23: PCI Master FIFOs in Single 64-bit Mode

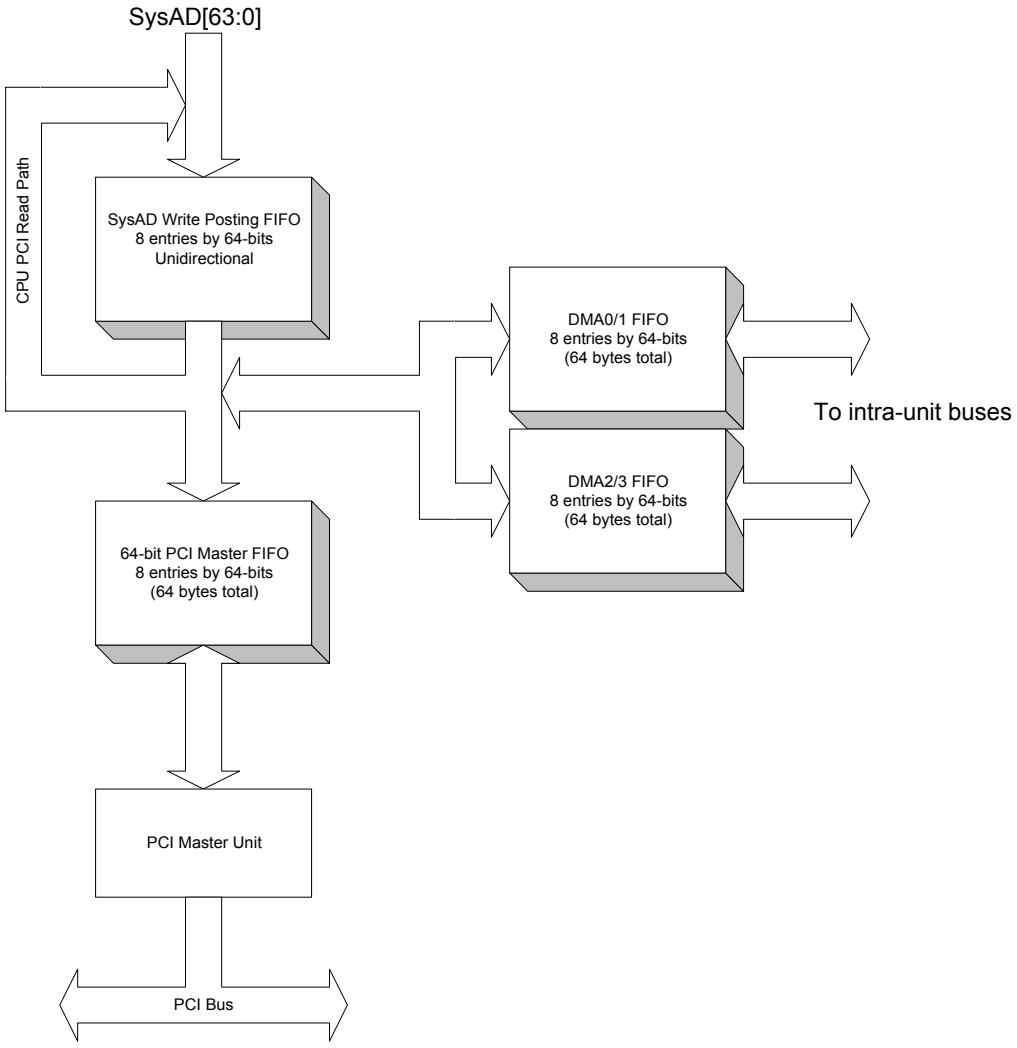
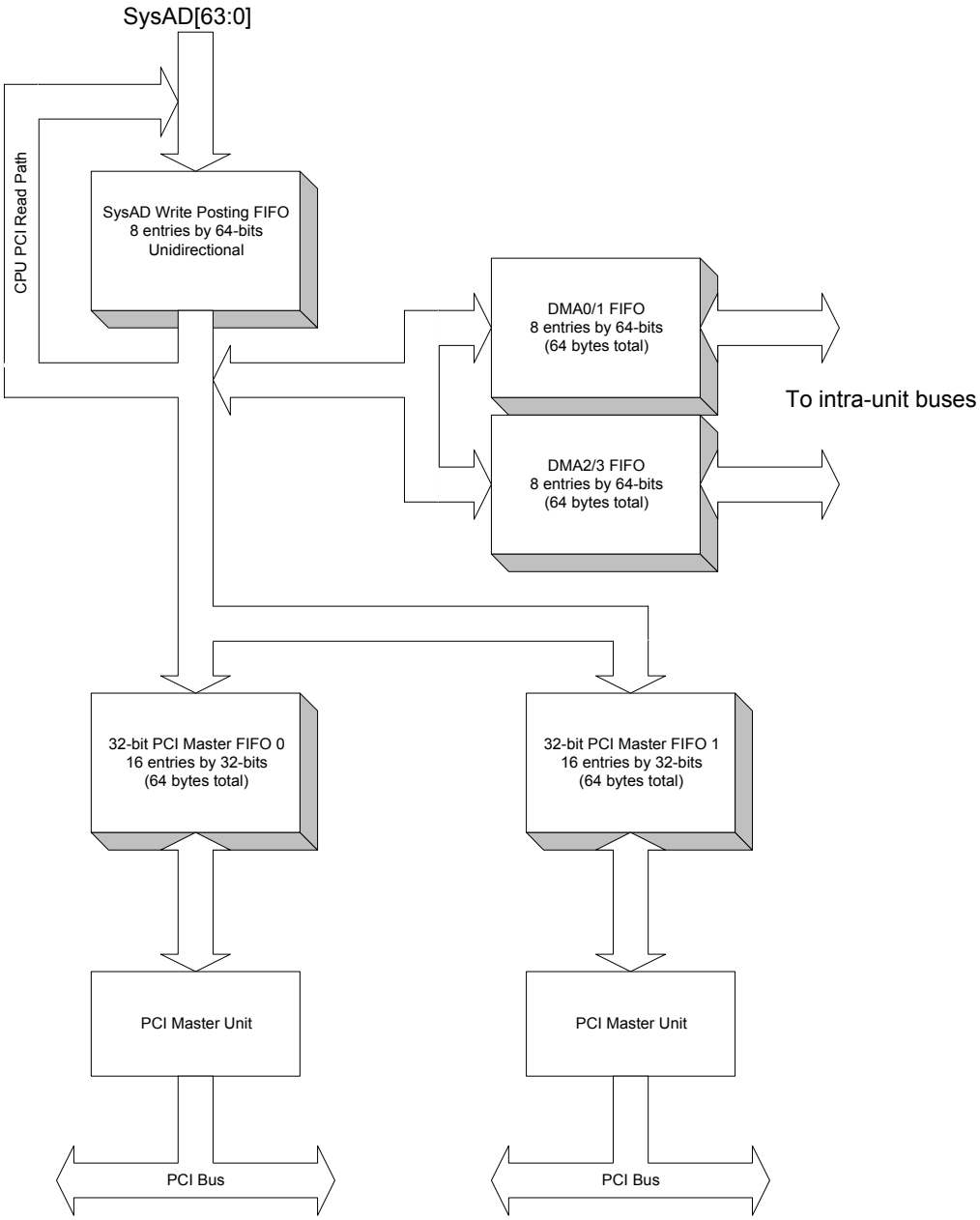


Figure 24: PCI Master FIFOs in Dual 32-bit Mode



7.3 PCI Target Interface

The GT–64120A 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–64120A 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–64120A 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.

When the GT–64120A 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–64120A 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–64120A’s PCI target interface deasserts TRDY until one of the FIFOs is empty again and might even issue a DISCONNECT to the PCI bus if reached timeout, see Section 12. “Reset Configuration” on page 143.

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–64120A’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 deasserts for two PCI clocks when the FIFO switch occurs.

7.3.2 Controlling Burst Length

The Memory Interface of the GT–64120A is capable of maximum burst of eight data beats 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.

Figure 25: PCI Target Interface "Ping-Pong" FIFOs

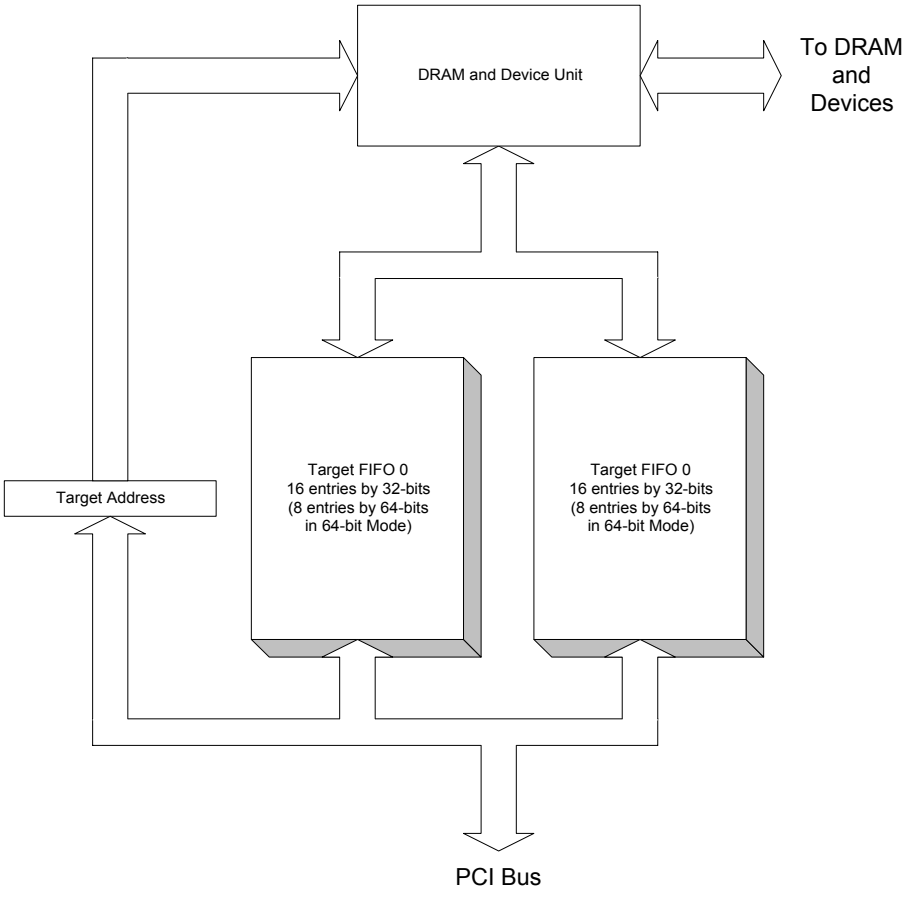
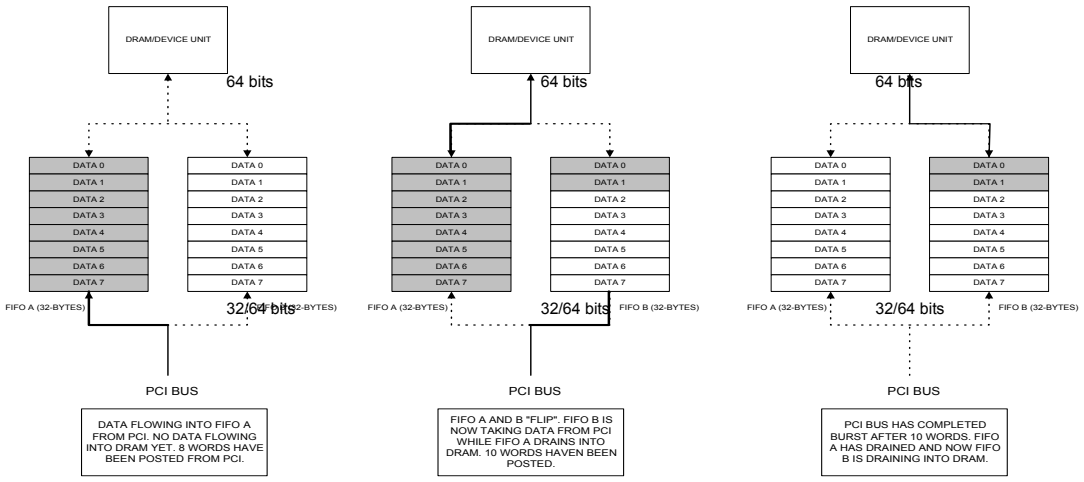


Figure 26: PCI Target Interface FIFOs Operational Example



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

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–64120A disconnects after the first TRDY.

7.3.4 PCI Target Address Space Decode and Byte Swapping

The GT–64120A 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 endianess of the data (e.g. little-endian to big-endian).

The GT–64120A 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–64120A 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 4 on page 3–37.

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 Tweaking the Performance of the Target Interface

The GT–64120A 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–64120A 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]

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.

1. The PCI specification also states that a master may not enforce target rules. In other words, even if the GT–64120A takes longer than 16 clocks to return the first data, the master must wait.

7.4 PCI Synchronization Barriers

The GT–64120A considers some cycles to be “synchronization barrier” cycles. In such cycles, the GT–64120A 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–64120A.

The GT–64120A provides the CPU with a simpler way to perform synchronization with PCI buffers[1]. 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–64120A 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–64120A’s internal configuration registers.

The GT–64120A 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–64120A 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.

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

The GT–64120A 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.[1] Table 43 shows DevNum to IdSel mapping.

Table 43: 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

The CPU accesses the GT–64120A internal configuration registers when the fields DevNum and BusNum in the Configuration Address register are equal to 0. The GT–64120A Configuration registers are also accessed from the PCI bus when the GT–64120A is a target responding to PCI configuration read and write cycles.

The CPU accesses the GT–64120A internal PCI_1 configuration registers through PCI_0 Configuration Address and Data registers (0xcf8,0xcfc). For example, in order to access GT–64120A 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–64120A PCI configuration space and an access to an external PCI device configuration space. Both are accessed using an access to the GT–64120A 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 endianess.

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 SysAD bus

1. “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.

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 RegNum.

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

7.6 Target Configuration and Plug and Play

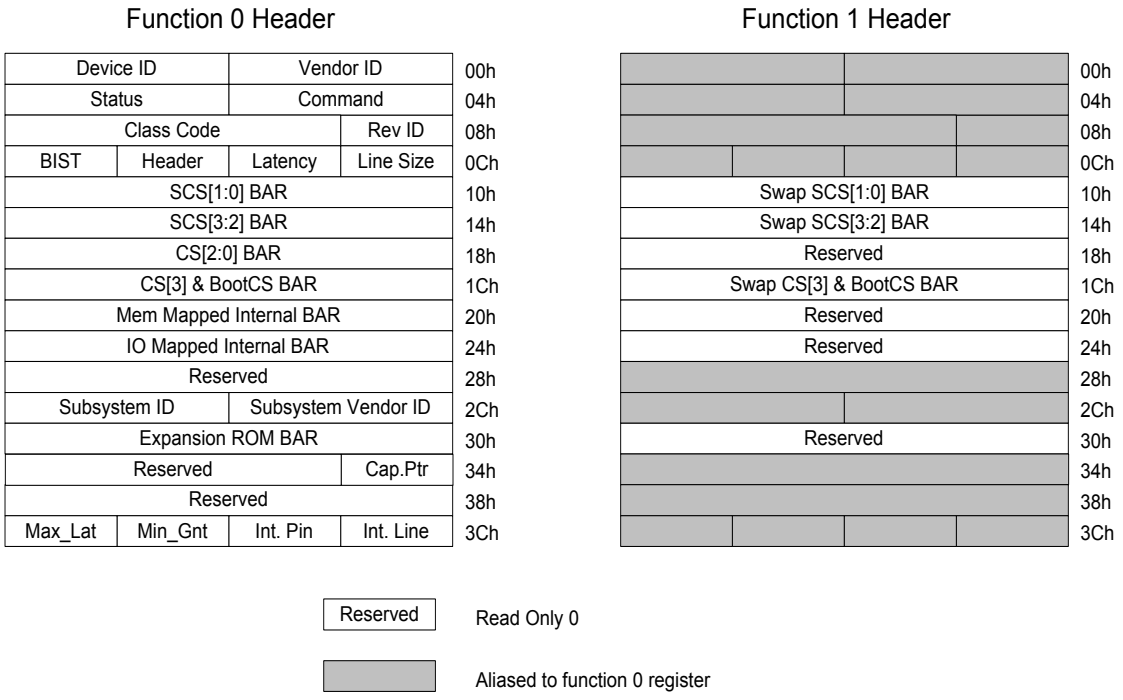
The GT–64120A includes all of the required plug and play PCI configuration registers. These registers, as well as the GT–64120A's internal registers are accessible from both the CPU and the PCI bus.

The GT–64120A 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 27.

Figure 27: 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–64120A 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 Auto-load of Configuration Registers at RESET" on page 110.

7.6.2 PCI Autoload of Configuration Registers at RESET

Thirteen PCI registers per PCI interface can be automatically loaded after Rst*. The autoload mode is enabled by asserting the DAdr[0] pin LOW on Rst*. Any PCI transactions targeted for the GT–64120A must be retried while the loading of the PCI configuration registers is in process.

The PCI register values are loaded from the ROM controlled by BootCS*. These register values are shown in Table 44 for PCI_0 and in Table 45 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 must be programmed to support burst read cycles.

Table 44: PCI_0 Registers Loaded at RESET

Register	Offset	Boot Device Address
Command	0xc00	0x1fffff80
Timeout & Retry Counters	0xc04	0x1fffff84
SCS[1:0]* Bank Size	0xc08	0x1fffff88
SCS[3:2]* Bank Size	0xc0c	0x1fffff8c
CS[2:0]* Bank Size	0xc10	0x1fffff90
CS[3]* & Boot CS* Bank Size	0xc14	0x1ffffff94
Conf_en	0xc3c	0x1ffffff98
Prefetch/Max_burst	0xc40	0x1ffffff9c
Device and Vendor ID	0x000	0x1fffffa0
Class Code and Revision ID	0x008	0x1fffffa4
Cache_line/Latency	0x00c	0x1fffffa8
Subsystem Device and Vendor ID	0x02c	0x1fffffac
Interrupt Pin and Line	0x03c	0x1fffffb0

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

Table 45: PCI_1 Registers Loaded at RESET

Register	Offset	Boot Device Address
Timeout & Retry Counters	0xc80	0x1fffffc0
Counter	0xc84	0x1fffffc4
SCS[1:0]* Bank Size	0xc88	0x1fffffc8
SCS[3:2]* Bank Size	0xc8c	0x1fffffcc
CS[2:0]* Bank Size	0xc90	0x1fffffd0
CS[3]* & Boot CS* Bank Size	0xc94	0x1fffffd4
Conf_en	0xcbc	0x1fffffd8
Prefetch/Max_burst	0xcc0	0x1fffffdc
Device and Vendor ID	0x080	0x1fffffe0
Class Code and Revision ID	0x088	0x1fffffe4
Cache_line/Latency	0x08c	0x1fffffe8
Subsystem Device and Vendor ID	0x0ac	0x1fffffec
Interrupt Pin and Line	0x0bc	0x1ffffff0

7.6.3 Expansion ROM Functionality

The GT–64120A 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 12. “Reset Configuration” on page 143.

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–64120A’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–64120A 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.

NOTE: See the PCI AC timing parameters (Section 22. “AC Timing” on page 255) when designing PCI systems that run faster than 33MHz.

Latency through the GT–64120A 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 5.4.2 “PCI Read Performance from SDRAM” on page 72.

NOTE: PCI clock frequency must be smaller than TClk frequency by at least 1MHz.

7.8 64-bit PCI Configuration

GT–64120A is configured to work as a 64-bit PCI device if DAdr[2] and FRAME1*/REQ64* pins are sampled LOW on reset, see Section 12. “Reset Configuration” on page 143.

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

When the GT–64120A 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–64120A 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–64120A PCI target retries any transaction targeted to the GT–64120A space.

NOTE: The GT–64120A 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–64120A 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–64120A 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

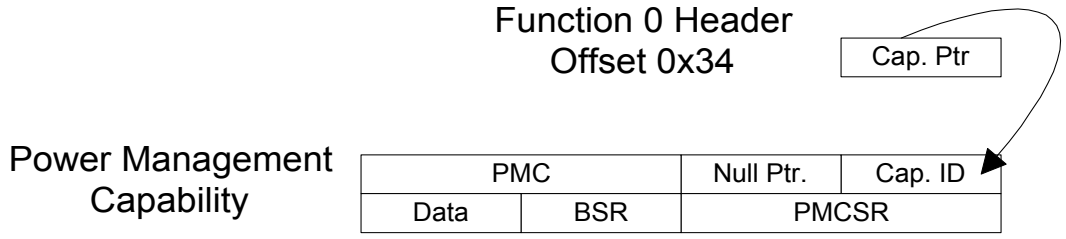
GT–64120A is CompactPCI Hot Swap Capable. It supports the following hot swap device requirements:

- All PCI outputs floats when RST# is asserted.
- All GT–64120A PCI state machines are kept in their idle state while RST# is asserted.
- GT–64120A PCI interface does not leave it's idle state until PCI bus is in an IDLE state. If reset is deasserted in the middle of a PCI transaction, the GT–64120A PCI interface stays in it's idle state until PCI bus is back in idle).
- The GT–64120A has no assumptions on clock behavior prior to it's setup to the rising edge of RST#.
- The GT–64120A is tolerant of the 1V precharge voltage during insertion.
- The GT–64120A can be powered from Early VCC.

7.12 PCI Power Management Support

GT–64120A is PCI Power Management compliant. It contains the required configuration registers as shown in Figure 28.

Figure 28: Power Management Registers



The GT–64120A 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 Section 11. “Interrupt Controller” on page 141.

NOTE: The GT–64120A does not support it’s 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 Master Interface Restrictions

1. Latency count, as specified in LatTimer (PCI Configuration Register 0x00c), must not be programmed to less than 6.

7.13.2 Slave Interface Restrictions

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 programed 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, MaxBurst must be set to 1 (i.e. 64 bytes).
6. PCI read access to 8- or 16-bit wide devices is not supported, unless FRAME* is asserted fro a single cycle.
7. When interfacing 32-bit SDRAM or device, or 64-bit SDRAM configured to a burst length of four, the Max burst must be programmed to 0 (maximum burst of four 64-bit words).

7.13.3 Master and Slave Interface Restrictions

1. PClk frequency must be smaller than TClk frequency by at least 1MHz.
2. Although the PCI specification only supports little endian data, the GT–64120A does support big endian on a 32-bit wide PCI bus, via the PCI Command register’s byte and word swap bits.

NOTE: The GT–64120A does not fully support big endian transfers on a 64-bit PCI bus. Contact Galileo Technology for full details on this issue.

8. INTELLIGENT I/O (I_2O) STANDARD SUPPORT

The GT–64120A includes hardware support for the Intelligent I/O (I_2O) Standard.

This support includes all of the registers required for implementing the I_2O Messaging Unit as defined in the I_2O Specification. This Messaging Unit (MU) is compatible with that found in Intel's i960Rx processors. However, the combination of MIPS processors and the GT–64120A will deliver as much as 20 times the integer performance of the i960RP.

The I_2O hardware support found in the GT–64120A also provides designers of non-I_2O 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 I_2O "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 I_2O support is in the I_2O specification documentation.

The I_2O 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–64120A and a MIPS processor, for example.) This same message passing mechanism is used to pass messages between peers in a system.

I_2O 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 I_2O 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 I_2O specification documentation also defines a simpler mechanism for implementing message passing through doorbell and message registers. The GT–64120A also includes this support.

8.2 I_2O Registers

From the PCI side, the registers used to implement I_2O support resides in the first 128 bytes of the memory region defined by SCS[1:0] Base Address register in PCI function 0 of PCI interface 0[1]. The I_2O registers are accessible from the CPU side through an offset from the CPU Internal Space Base register.

1. There is no I_2O support on the PCI_1 interface.

8.2.1 I_2O Register Map

Table 46: I_2O Register Map

I_2O Register	PCI SIDE[1]	CPU Side[2]
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 I_2O Support

The I_2O registers are only visible from the PCI side when I_2O support is enabled via the pin strapping option shown in the RESET Configuration section.

When I_2O support is disabled, the locations from BAR0+0x0 to BAR0+0x7F appear as SDRAM.

8.4 Register Map Compatibility with the i960Rx Family

The register map of the GT–64120A is compatible with the I_2O Specification. However, some of the registers found in Intel's i960Rx processors are not implemented. This information is shown in Table 47.

Table 47: Register Differences Between GT–64120A and i960Rx

Register Name	GT–64120A	i960Rx	Comment
APIC Register Select	Not implemented	Implemented	No APIC support required for GT–64120A, not a part of the I_2O spec.
APIC Window Select	Not implemented	Implemented	No APIC support required for GT–64120A, not a part of the I_2O spec.
Index Registers	Not implemented	Implemented	Index registers are not used for I_2O message passing so this is not a compatibility issue. Index registers are implemented as normal SDRAM in the GT–64120A.

8.5 Message Registers

The GT–64120A 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–64120A
- Outbound messages are sent by the GT–64120A'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–64120A 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–64120A'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 I_2O message passing mechanism, and are also the most powerful part of the MU built into the GT–64120A. 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 The 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 48: Circular Queue Starting Addresses

Queue	Starting Address
Inbound Free	QBAR
Inbound Post	QBAR + Queue Size

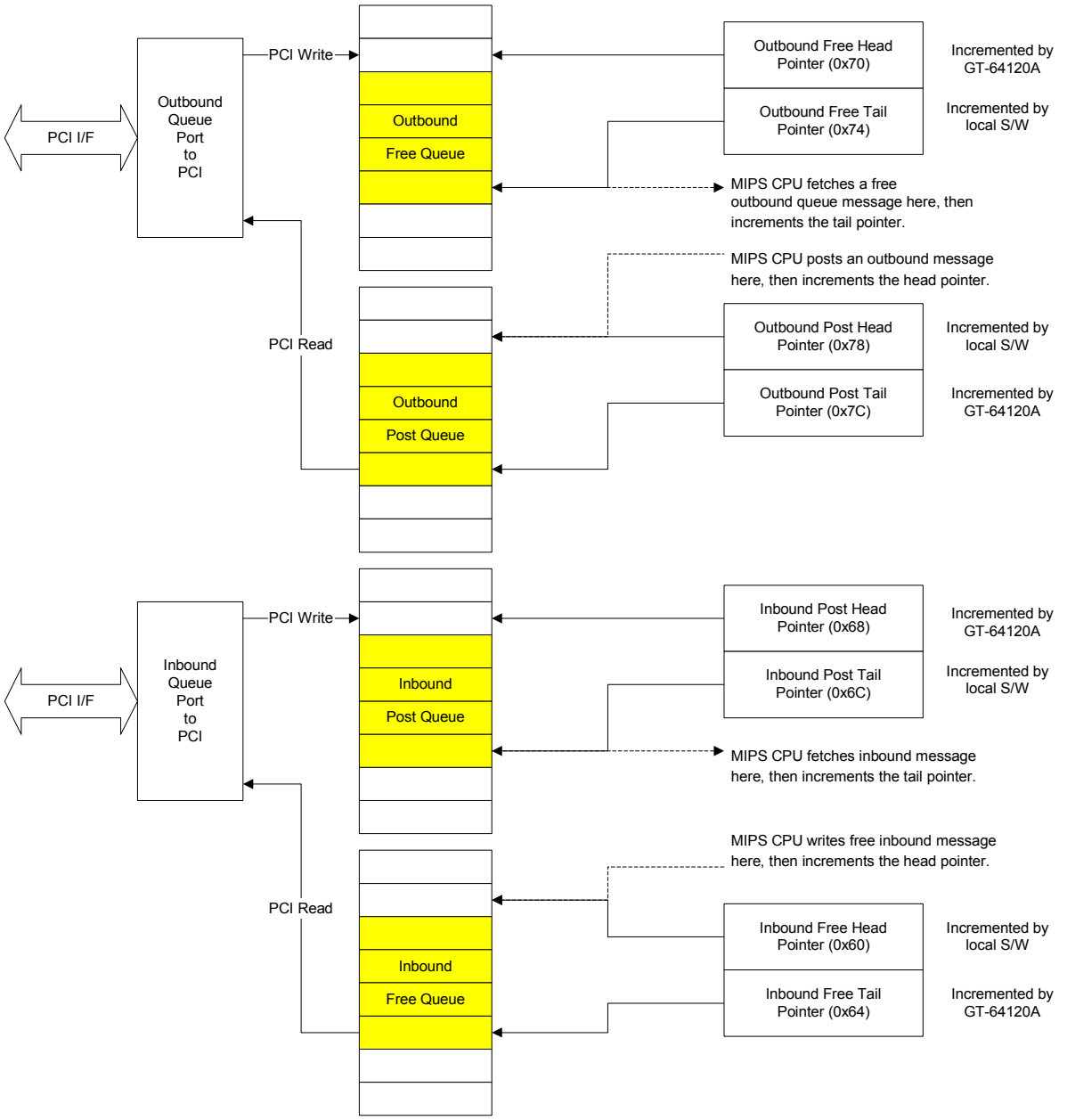
Table 48: Circular Queue Starting Addresses (Continued)

Queue	Starting Address
Outbound Post	QBAR + 2*Queue Size
Outbound Free	QBAR + 3*Queue Size

Each queue has a head pointer and a tail pointer kept in GT–64120A 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 I_2O queue will result in disconnect after 1st data transfer.

Figure 29: I_2O 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 I_2O/PCI address space, decoded by BAR0.

NOTE: The Inbound and Outbound Queue Port virtual registers are not read/write physical registers within the GT–64120A. These virtual registers are reading and writing pointers into the circular queues (located in SDRAM) that are controlled by the GT–64120A. Refer to Figure 29.

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–64120A 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–64120A 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 CPUs 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–64120A 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–64120A; 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–64120A 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 in maintained in software by the MIPS CPU. The head pointer is maintained automatically by the GT–64120A 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–64120A 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 49: I_2O 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–64120A	MIPS CPU
Inbound Free		No	No	MIPS CPU	GT–64120A
Outbound Post	Outbound Queue Port	Yes, when queue is not empty	No	MIPS CPU	GT–64120A
Outbound Free		No	Yes, when queue is full	GT–64120A	MIPS CPU

8.8 Index Registers

Index registers are not supported in the GT–64120A.

9. DMA CONTROLLERS

The GT–64120A has four independent DMA controllers. The DMA 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 DMA 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 DMA 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 DMA 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 DMA controller can be programmed to move up to 64KBytes of data per transaction. The burst length of each transfer of DMA can be set from 1 to 64 bytes. Accesses can be non-aligned both in the source and the destination.

The GT–64120A 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. These 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 DMA 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 Section 9.2.3 “SrcDir, bits[3:2]” on page 127.

9.1.3 Destination Address Register

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 Section 9.2.4 “DestDir, bits[5:4]” on page 127.

9.1.4 Pointer to The Next Record Register

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 Section 9.2.6 “ChainMod, bit 9” on page 127.

9.2 DMA Channel Control Register (0x840 - 0x84c)

Each DMA channel has its own unique control register where certain DMA modes can be programmed. Following are the bit descriptions for each field in the control register.

9.2.1 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.

Setting this bit to 0 means all DMA transfers are internal. The transfer utilizes a DMA FIFO.

Setting this bit to 1 means DMA transfer is in Fly-By mode. The data is transferred from source to destination externally, without using a DMA FIFO.

NOTE: 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.

9.2.2 R/W bit

NOTE: 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 a write.

Setting this bit to 0 means a Fly-By READ from SDRAM.

Setting this bit to 1means a Fly-By WRITE to SDRAM.

9.2.3 SrcDir, bits[3:2]

The SrcDir field contains information about how the source address for the DMA transfer is handled. If set to 00 (the default), these bits automatically increment the source address.

Setting these bits to 01 means the source address automatically decrements.

Setting these bits to 10 means the source address remains constant throughout the DMA burst.

A setting of 11 is reserved.

9.2.4 DestDir, bits[5:4]

The DestDir field contains information about how the destination address for the DMA transfer is handled.

Setting these bits to 00 (the default) means the destination address automatically increments.

Setting these bits to 01 means the destination address automatically decrements.

Setting these bits to 10 means the destination address remains constant throughout the DMA burst.

A setting of 11 is reserved.

9.2.5 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 modulo-2 steps (i.e. 1, 2, 4, 8,..., 64).

9.2.6 ChainMod, bit 9

ChainMod contains whether this channel is set in chained mode or not.

Setting this bit to 0 (the default) enables chained mode for the channel. At the completion of a DMA transaction, the Pointer to Next Record register provides the address of a new DMA Record. If Pointer to Next Record register contains a value of 0 (NULL), this is the last record in the chain.

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.

NOTE: The channel must be enabled by setting ChanEn to 1 (see Section 9.2.9 "ChanEn, bit 12" on page 128) and setting FetNexRec (see Section 9.2.10 "FetNexRec, bit 13" on page 128) to 1. Setting these two bits must be done during the same writes. See Figure 30 on page 9–132 for a diagram of chained mode DMA.

Setting ChainMod to 1 means the chained mode is disabled and the Pointer to Next Record Register is not loaded at the completion of the DMA transaction.

NOTE: In non-chained mode the Byte Count, Source, and Destination Registers must be initialized prior to enabling the channel.

9.2.7 IntMode, bit 10

IntMode controls when this channel asserts the DMAComp (DMA Complete) Interrupt.

Setting this bit to 0 (the default) means the channel asserts the DMAComp Interrupt every time the DMA byte count reaches 0.

Setting this bit to 1, with the channel set to chained mode, means the DMAComp Interrupt is only asserted when both the Pointer to Next Record register has a NULL value and Byte Count is 0.

NOTE: If chained mode is disabled, the setting of IntMode is irrelevant and DMAComp Interrupt will be asserted every time the Byte Count reaches 0.

9.2.8 TransMod, bit 11

TransMod indicates whether the channel is set to operate in demand mode or block mode.

Setting this bit to 0 (the default) means the channel operates in demand mode when DMA accesses are initiated by externally asserting one of the four DMAReq[3:0]* pins or through the timer/counter terminal count.

Setting this bit to 1 means the channel operates in block mode where DMA accesses are initiated by setting ChanEn, Bit 12. In Block mode, the DMAReq* lines are not sampled.

9.2.9 ChanEn, bit 12

ChanEn enables or disables the DMA channel.

Setting this bit to 0 means the channel is disabled.

Setting this bit to 1 means the DMA is initiated based on the current setting loaded in the channel record (i.e., Byte Count, Source Address and Destination Address).

NOTE: The DMA channel is enabled or disabled via ChanEn if the channel is in Demand or Block Mode.

9.2.10 FetNexRec, bit 13

FetNexRec is a field which is only meaningful when chained mode is enabled for the channel.

Setting this bit to 1 forces a fetch of the next record based on the value in the Pointer to Next Record Register.

This bit can be set even if the current DMA has not yet completed.

This bit is reset to 0 (the default) after completing the of the new record fetch.

The CPU must not assert FetNexRec if the descriptor equals Null.

9.2.11 DMAActSt, bit 14 (Read Only)

DMAActSt is a read only field that can be polled to see the DMA activity status of the channel.

Setting this bit to 0 means the channel is not active.

Setting this bit to 1 means the channel is active.

In non-chain mode, this bit is deasserted when Byte_Count reaches zero. In chain-mode, this bit is deasserted when pointer to next record is NULL and Byte_Count reaches zero.

This bit is reset if the CPU sets chanEn to 0 during DMA transfer.

9.2.12 SDA, Source/Destination Alignment, bit 15

The SDA bit determines whether address alignment is done for source or destination.

Setting this bit to 0 means the alignment is towards the source and after the DMA's first read all reads will be to 64-bit word aligned address.

Setting this bit to 1 means the alignment is towards the destination and after the first write following writes will be with all Byte Enables asserted.

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.

NOTE: If both the DMA Source and Destination addresses are aligned, the meaning of this bit is irrelevant.

9.2.13 Mask DMA Requests, MDREQ, bit 16

Some slower devices require extra time in order to deassert a DMA request signal. This bit can be used to provide this extra time.

Setting this bit to 1 means the DMA request for the channel is masked for three cycles starting from DMA arbitration cycle.

Setting this bit to 0 means there is no masking. This allows devices more time to deassert DMAReq* when DMAAck* qualified with CSTiming* is asserted without the possibility that another premature DMA request entered.

9.2.14 Close Descriptor Enable, CDE, bit 17

A DMA transfer may end (be stopped 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).

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.

Setting this bit to 1 means that prior to fetching a new descriptor the current one is closed with remaining byte count.

Setting this bit to 0 means the descriptor is not closed. The descriptor is closed only if it is open (i.e., data has been transferred from the source to the destination).

9.2.15 End Of Transfer Enable, EOTE, bit 18

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 this next DMA. If the DMA channel is in non-chain mode, then the current DMA transfer is stopped without further action.

Setting this bit to 1enables ending DMA transfers on channel via the EOT pin.

Setting this bit to 0 means that End Of Transfer (EOT) input is ignored for the channel.

9.2.16 End Of Transfer Interrupt Enable, EOTIE, bit 19

EOTIE enables or disables interrupts due to End Of Transfer (EOT) signal activation.

Setting this bit to 1 means the interrupts are enabled, otherwise they are disabled. The interrupt is set immediately after the channel empties it FIFO. If the channel is idle, the interrupt is set immediately. If the CDE bit is set, then the interrupt is set immediately after the channel closes its current record (in the case that there is an open descriptor).

9.2.17 Abort DMA, ABR, bit 20

It is possible that the CPU may need to abort a DMA transfer and reprogram the DMA. This bit flushes internal indications which do not get flushed by a mere channel disable.

Setting this bit to 1 means the CPU aborts the DMA.

NOTE: Be sure to also set the En/Dis bit to 0 in order for the DMA transfer to abort.

The ABR bit must be used together with En/Dis if CDE and/or EOT are enabled and the CPU wants to abort a transfer for reprogramming.

9.2.18 SLP (bits [22:21]), DLP (bits [24:23]), RLP (bits [26:25])

SLP, DLP, and RLP bits are used to redefine address space of source, destination, or record address location. They enable overriding the local address space with PCIMem0 or PCIMem1 address space.

Table 50: 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.

Table 50: Location of Source Address, SLP (Continued)

SLP ([22:21])	Function
10	Source address is in PCI_1 memory space.
11	Reserved.

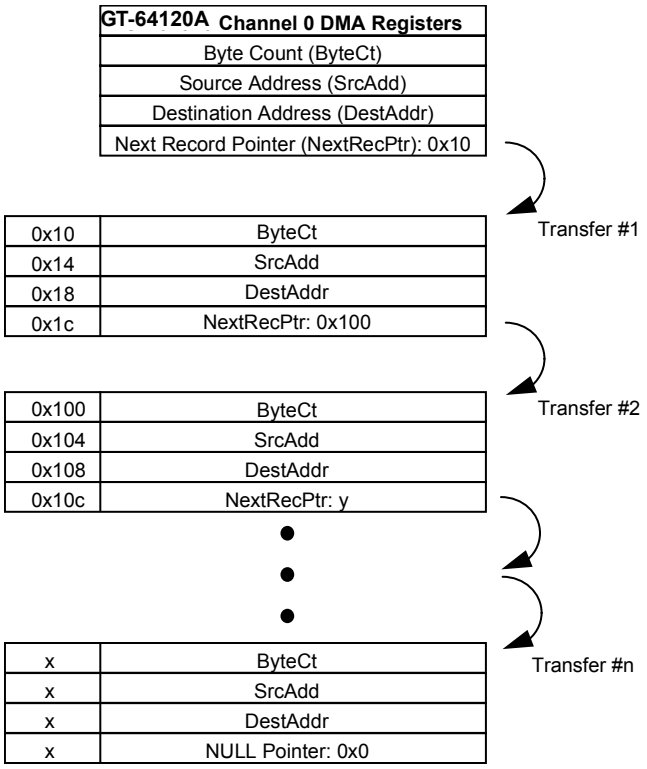
Table 51: Location of Destination Address, DLP

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 52: 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 30: 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–64120A's DMA controllers. The following definitions are used throughout this section:

Table 53: DMA Controller Design Information Terms and Definitions

Term	Definition
Device	A device located on the memory bus mapped to one of the GT–64120A'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–64120A is accessing a Device.

In this mode the transfer initiator (usually a Device) asserts DMAReq* to signal the GT–64120A that a new DMA transfer should begin. As an acknowledgment response, the GT–64120A 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 then 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–64120A 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–64120A acknowledges the last written data from the DMA unit.

9.7.1.4 Transferring data examples/recommendations

Table 54: 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–64120A 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 deassert 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–64120A is currently accessing the device for read.
SDRAM or PCI	Device	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–64120A is currently accessing the device for write. **NOTE:** In this situation, the GT–64120A 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 deassertion 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. • 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 a DMA controller is actively working on. The main issue is to synchronize between when the GT–64120A 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-64120A and assert the FetNexRec
        bit
      }
      else      {
       read the Pointer to Next Record GT-64120A. If it's equal to NULL {
   Mark (by using a flag or something) that in the next DMA chain
   complete interrupt you'll need  to -
                    [[[[{
          Update the NRP register in the GT-64010 and Write the Fetch Next Record
}]]]]
      }
 }
```

9.7.5 Fly-by DMA

9.7.5.1 Background

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–64120A. 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.

9.7.5.2 FlyBy in the GT–64120A

During fly-by, the GT–64120A 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.

9.7.5.3 How to program the GT–64120A for Fly-By

The DMA control register has two bits for Fly-by indications:

Table 55: FlyBy 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.

9.7.5.4 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.

9.7.5.5 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 Section 9.2.11 “DMAActSt, bit 14 (Read Only)” on page 129. 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 that 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 Section 9.2.3 “SrcDir, bits[3:2]” on page 127), 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 39), it results in unpredicted behavior and the DMA channel might need to be stopped by clearing the activate bit of the channel control register.
9. When Accessing 8- or 16-bit devices, the burst limit is 8 bytes.
10. When accessing 32-bit SDRAM or devices, or when accessing 64-bit SDRAM configured to burst length of four, the burst limit is restricted to 32 bytes.
11. Fetches of new descriptors always result in a burst of 16 bytes, regardless of burst limit value.

12. DMA address decoding is performed up to addrss bit 31. This means that if two first level decoding address windows differ only in bits [35:32], a DMA access is a hit in both windows. This may result in access to the wrong device.
13. 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.
14. If the PCI master accessing the GT–64120A 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-programed to 1(e.g. 3 cycles).

10. TIMER/COUNTERS

There are three 24-bit wide and one 32-bit wide timer/counters on the GT–64120A. Each one can be selected to operate as a timer or as a counter.

In Counter mode, the counter counts down to terminal count, will stop 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.

NOTE: The count frequency for the timer/counter is equal to TCLk frequency.

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 0xffffff.

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–64120A.

11. INTERRUPT CONTROLLER

The GT–64120A includes an interrupt controller that routes internal interrupt requests to both the CPU and the PCI bus. The interrupt controller ORs all internal interrupt sources and asserts an interrupt to the CPU or to the PCI when one or more internal interrupts are asserted.

11.1 Interrupt Cause Registers

There are two interrupt cause registers that can be checked to detect the occurrence of certain events.

One cause register consists of the interrupts asserted for events caused by the GT–64120A and by PCI_0. This register is located at offset 0xc18.

The other cause register (also known as the high cause register) consists of the interrupts asserted for events caused by PCI_1. This register is located at offset 0xc1c.

NOTE: If the device is configured for PCI_0 only, the high cause register is reserved and is not used.

The Interrupt* pin must be connected to one of the CPU input interrupt lines directly. The Int0* pin is used for the devices on PCI_0. If a source asserts an interrupt, its respective bit in the Cause register is set. All of the bits from the cause register(s) are OR-ed together and the output is on the Interrupt* line for the CPU and on Int0* for PCI devices.

Interrupt* signals the CPU to read the interrupt cause register and run a particular service routine depending on the interrupt bit that was set. Int0* serves the same purpose but signals PCI_0 devices. There is not a separate interrupt for PCI_1 but Interrupt* and Int0* can asserted due to events which occur on PCI_1.

The interrupt is acknowledged by the CPU or by the PCI bus by resetting its bit in the cause register (writing zero to the specific bit and one to all other bits). An exception to this is the CPUInt ([25:21] and PCIInt ([29:26])) which are used by the PCI to generate interrupt to the CPU and vice versa. These are SET by writing 0 from the interrupt originating side and CLEARED by writing 0 from the interrupt destination side.

NOTE: See Section 20.17 “Interrupts” on page 221 for more information about the events which causes each interrupt cause register bit to assert.

11.2 Interrupt Mask Registers

There are two interrupt mask registers for both the Interrupt* pin and the Int0* pin (4 mask registers total). These registers allow interrupt bits to be masked off from asserting an interrupt to the CPU or to PCI.

For Interrupt*, the mask register for the main cause register is located at 0xc1c and at 0xc9c for the high cause register.

For Int0*, the mask register for the main cause register is located at 0xc24 and at 0xc38 for the high cause register.

A 0 in a mask register bit masks the interrupt from asserting an interrupt. A 1 in a mask register bit allows this interrupt bit to be included in the OR-ing of the cause register bits for Interrupt* or Int0* output pin.

11.3 Interrupt Summaries

IntSum in the Interrupt Cause register is the logical OR of bits[29:1], regardless of the Mask registers' values. This is used for polling if any interrupt occurred within the GT–64120A. Therefore, bit[0] of all of the mask registers is 0. If both PCI_0 and PCI_1 are used, IntSum is the logical OR of both the cause register bits as well as the HIGH cause register bits.

CPU IntSum in the main cause register is the logical OR of the interrupt cause bits, masked by the CPU interrupt mask bits.

PCIIntSum in the main cause register is the logical OR of the interrupt cause bits, masked by the CPU interrupt mask bits.

11.4 Interrupt Select Registers

There are two interrupt select registers used to optimize interrupt routines when the GT–64120A is configured for both PCI_0 and PCI_1. One select register exists for the CPU interrupt (Interrupt*) at 0xc70 and another register exists for the PCI_0 interrupt (Int0*) at 0xc74. If the device is configured for PCI_0 only, both of these registers are reserved and are not used.

Instead of checking BOTH the main cause register and the high cause register when interrupted, the CPU or PCI device has the option to read the appropriate select register. The select register will contain the cause register bits of either the main or the high cause register, depending on which register has interrupt bits set. Bit 30 of the select register indicates which cause register (main or high) is the source of the interrupt. Bit 29:1 contains the aliased interrupt bits of the appropriate cause register.

If both the main and the high cause registers have interrupt bits set, bit 31 will be set to 1. Bit 30 will indicate the select register is aliassing the main cause register (set to 0).

12. RESET CONFIGURATION

The GT–64120A 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–64120A 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 1 ms before any CPU transactions are generated.

Table 56: Reset Configuration

Pin	Configuration Function
DAdr[2], Frame1*/Req64*:	PCI Bus Configuration
00- 01- 10- 11-	Only PCI_0 is used as 64-bit. Only PCI_0 is used as 32-bit, PCI_1 is NOT used. Reserved Both PCI_0 and PCI_1 are used as 32-bit.
Interrupt*:	CPU Data Endianess
0- 1-	Big endian Little endian
Ready*, CSTiming*	Multi-GT–64120A Address ID
00- 01- 10- 11-	GT responds to SysAD[26,25]= 00 GT responds to SysAD[26,25]= 01 GT responds to SysAD[26,25]= 10 GT responds to SysAD[26,25]= 11 **NOTE:** Boot GT–64120A should be programmed to 11.
BankSel[0]:	PCI Class Code Default Select
0- 1-	Memory Controller (0x580) Host Bridge (0x600)
DAdr[10]:	Multiple GT–64120A Support
0- 1-	Not supported Supported
DAdr[9]:	66 MHz Capable
0- 1-	Disable Enable
DAdr[8]:	I_2O Support
0- 1-	Enable Disable

Table 56: Reset Configuration (Continued)

Pin	Configuration Function
DAdr[7]:	UMA Support
0- 1-	Enable - DMAReq[0]*/MREQ* functions as MREQ*. Disable - DMAReq[0]*/MREQ* functions as DMAReq[0]*.
DAdr[6]:	Programming Conditional PCI Retry
0- 1-	Enable Disable
DAdr[5]:	Expansion ROM Enable
0- 1-	Enable Disable
DAdr[4:3]:	Device Boot Bus Width (Controlled by BootCS*) AND CS[3]* Device Width.
00- 01- 10- 11-	8 bits 16 bits 32 bits 64 bits
DAdr[0]:	Autoload
0- 1-	Enable Disable
SDQM[1]:	PCI_1 Power Management
0- 1-	Disable Enable
SDQM[0]:	PCI_0 Power Management
0- 1-	Disable Enable
DMAReq[3]*	Duplicate ALE
0- 1-	Do not Duplicate ALE. Duplicate ALE output on ADP[1] (no ECC in system).
DMAReq[1]*	Duplicate SDRAM Control Signals
0- 1-	Do not Duplicate SRAS*, SCAS* and DWr*. 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.
ByPsPL	Bypass PLL
	It should be pulled down to enable the PLL.

Table 56: Reset Configuration (Continued)

Pin	Configuration Function
DAdr[1]	Reserved
	Must pull LOW during reset.
DMAReq[2]*	Reserved
	Must pull LOW during reset.

NOTE: There must be no devices driving input pins which sample VCC or GND via pull-up/down resistors (i.e. Ready*) during RESET.

13. 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.

13.1 SDRAM

The GT–64120A supports both 64-bit and 32-bit SDRAM.

Table 57: 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* 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]*	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 ~~TClk~~	Clock Input

1. SRAS*, SCAS*, and DWr* can be duplicated on ADP[7], ADP[6] and ADP[3] if programmed on RESET.

Table 58: 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.

13.2 Devices

The GT–64120A supports 64-, 32-, 16- and 8-bit devices.

Table 59: 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]
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 60: 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]

Table 60: 32-bit Devices (Continued)

Connection	Connect...	To...
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]*	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 61: 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 62: 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.	

14. JTAG Application Notes

The GT–64120A supports test mode operation through it's JTAG interface.

The GT–64120A JTAG interface does not include JTRST* pin. When activating test mode, JTMS must be set to 1 for at least five JTCK clock cycles, in order for the JTAG state machine to return it's idle state. When running in test mode, the Rst* pin must be set to 1.

To place GT–64120A in the functional mode, JTAG interface must be disabled. Disable JTAG by configuring the following pins:

- JTCK must be pulled LOW through a 4.7 KOhm resistor.
- JTMS must be pulsed HIGH through a 4.7 KOhm resistor.
- JTDI must be pulled HIGH through a 4.7 KOhm resistor.
- JTDO must be left UNCONNECTED.

15. BIG AND LITTLE ENDIAN

15.1 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 Web site.

There are three bits in the GT–64120A which control byte swapping. 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 bits are given the same value as sampled at Rst* via pullup or pulldown on Interrupt* pin. All bits can also be programmed after reset is de-asserted.

If all bits are set to 1, the GT–64120A assumes Little-endian data format and NO byte swapping is done within the device.

Additionally, there are three WORD-SWAP bits in GT–64120A 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–64120A 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.

The nomenclature for this section is shown in Table 63.

Table 63: 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.

15.1.1 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 there is byte swapping of data transfers to/from the GT–64120A internal registers (including Configuration Data register, 0xcfc). No byte swapping takes place during data transfers in which the source/target is external.

15.1.2 Bits 0 and 16 of the PCI Internal Command register

Bit 0 of the PCI Internal Command register (0xc00) controls byte swapping of GT–64120A PCI master interface. Bit 16 controls byte swapping of GT–64120A PCI target interface. These bits affect the following:

- Setting these bits to 1 means there is no byte swapping within the PCI Interface unit of any data transfer.
- Setting these bits to 0 means there is no byte swapping of data transfers to/from PCI Interface unit's internal registers. Byte swapping takes place during data transfers in which the source/target is external

15.1.3 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 there is no word swapping of data transfers to/from PCI Interface unit's internal registers. Word swapping takes place during data transfers in which the source/target is external.

NOTE: Only the 32-bit PCI interface supports word swapping.

15.2 Configuring a System for Big and Little Endian

Table 64 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 64: 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

Table 64: Configuring for Big and Little Endian (Continued)

Resource	Swap Bits (CPU bit : PCI byte swap bit : PCI word swap bit)			
	110	001	010	101
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

16. Using the GT–64120A Without the CPU Interface

Table 65 lists the pins that must be strapped when the GT–64120A 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 65: 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 12. “Reset Configuration” on page 143.
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.

17. USING THE GT–64120A IN DIFFERENT PCI CONFIGURATIONS

The PCI interface of the GT–64120A 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 66 lists what must be done with the pins when the GT–64120A is used without any PCI interface.

NOTE: Rst* must always be connected. Most pins should be strapped HIGH or LOW through a resistor. Galileo Technology recommends using 4.7 KOhm resistors.

Table 66: No PCI Interface

Pin	Pin Usage
VREF0	VREF0
PClk0	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.
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.

Table 66: 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 67 lists what must be done with the pins when the GT–64120A is used with PCI_0 as a 32-bit PCI interface only (i.e., no PCI_1).

NOTE: When the GT–64120A is configured to a single 32-bit PCI_0 interface, the GT–64120A 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 67: 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 67: 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 or tied directly to PClk0.
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 68 lists what must be done with the pins when the GT–64120A is used with PCI_0 as a 32-bit PCI interface and PCI_1 as a 32-bit PCI interface.

Table 68: PCI_0 as 32-bit PCI and PCI_1 as 32-bit PCI

Pin	Pin Usage
VREF0	VREF0
PClk0	PClk0
DevSel0*	DevSel0*
Stop0*	Stop0*
Par0	Par0

Table 68: PCI_0 as 32-bit PCI and PCI_1 as 32-bit PCI (Continued)

Pin	Pin Usage
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
PClk1	PClk1
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*
PAD1[31:0]/PAD0[63:32]	PAD1[31:0]
CBE1[3:0]*/CBE0[7:4]*	CBE1[3:0]*

Table 69 lists what must be done with the pins when the GT–64120A is used with PCI_0 as a 64-bit PCI interface only (i.e. no PCI_1).

Table 69: 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.
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.

Table 69: PCI_0 as 64-bit PCI Only (Continued)

Pin	Pin Usage
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]*

NOTE: The combined capacitance of PClk0 and PClk1 exceeds the 10pF maximum capacitance limitation of the PCI Specification.

18. PLL Power Filter Circuit

The GT–64120A has an on-chip PLL to improve its AC timing.

To garauntee the stability of the PLL operation, it is critical to insulate the PLL power supply from external signal noise.

18.1 PLL Power Supply

The GT–64120A uses two dedicated power supply pins for the PLL:

- N04 - AVCC2.5PLL - Supplies the 2.5V DC for the analog and digital parts of the PLL.
- V04 - AGNDPLL - Supplies the GND for the analog and digital parts of the PLL.

The GT–64120A DC specification requires that the PLL GND and the PLL VCC must be supplied with a nominal value of 2.5V DC, with a tolerance of up to 5%. The recommended filtering circuit ensures that the PLL DC specifications are met.

The following sections outline two circuits depending on if the 2.5V supply source is available or un-available on board.

18.2 PLL Power Filter With a 2.5V Power Supply Available On Board

Figure 31 shows a recommended circuit for the GT–64120A PLL filter.

The circuit's purpose is to prevent the interference of the differential and common modes, usually present in PCBs containing several devices, reaching the PLL power supply traces and, subsequently, disturbing its normal operation.

It is assumed that the 2.5V DC source, necessary to bias the PLL, is available on board.

The user must:

- Use dedicated traces to supply the AGNDPLL and the AVCC2.5PLL directly from the systems power supply to the filtering circuit.
- Confirm that the PLL supply balls (N04, V04) are isolated from other VCC and GND pins of the GT–64120A.

Figure 31: PLL Power Filter Circuit With Common On-board 2.5V Supply

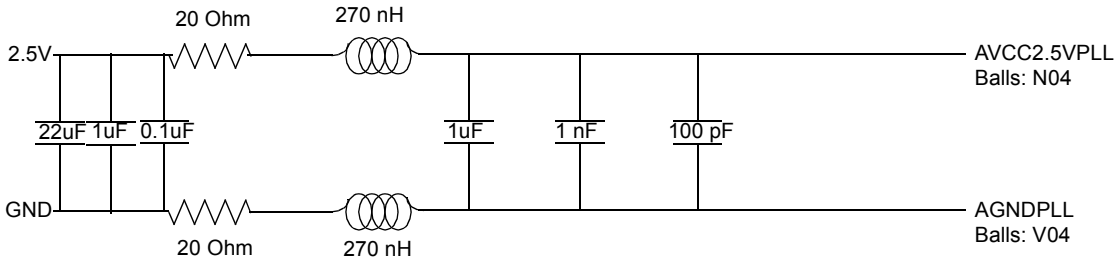
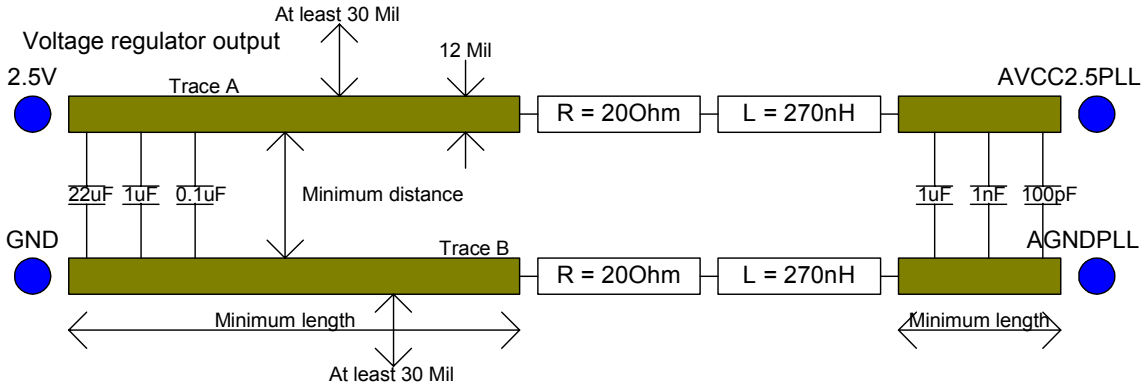


Figure 32: PLL Layout Guideline for a PCI Add-on Card



NOTE: In Figure 32, Traces A and B must be parallel and the same length. Also, Galileo Technology recommends to route the traces on the component side, or print side, and, if possible, leave the area clean in all layers where the traces are routed.

18.3 PLL Power Filter With No Power Supply Available On-board (Backplane Layout)

Figure 34 and Figure 35 shows a recommended circuit for the GT–64120A PLL filter when a 2.5V power supply **is not** readily available on board.

For example, for a 5/3.3V DC board supply, the industry standard LP2951 in an SMT packaging can be used to produce the 2.5V DC for the PLL, with the 240 Ohm resistors connected to the output pin and an adjust pin as indicated.

The user must:

- Use dedicated traces to supply the AGNDPLL and the AVCC2.5PLL directly from the systems power supply to the filtering circuit.
- Confirm that the PLL supply balls (N04, V04) are isolated from other VCC and GND pins of the GT–64120A.

Figure 33: PLL Power Filter Circuit With Dedicated 3.3/5V(LP2951**)/12V(LM317) Supply**

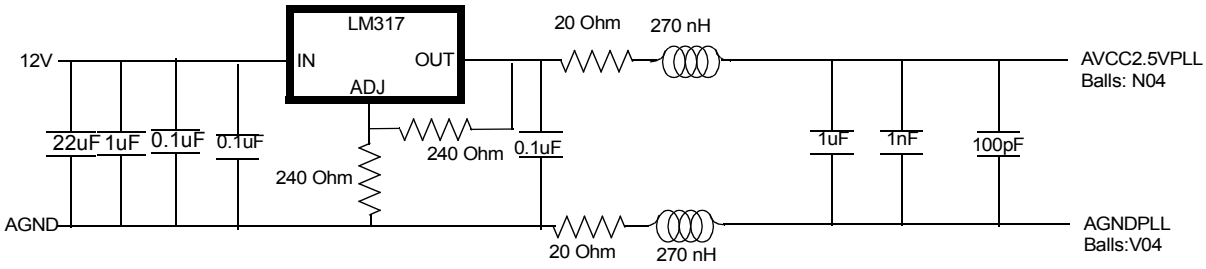


Figure 34: PLL Layout Guideline for Backplane Layout with 12V Power Supply PLL Layout

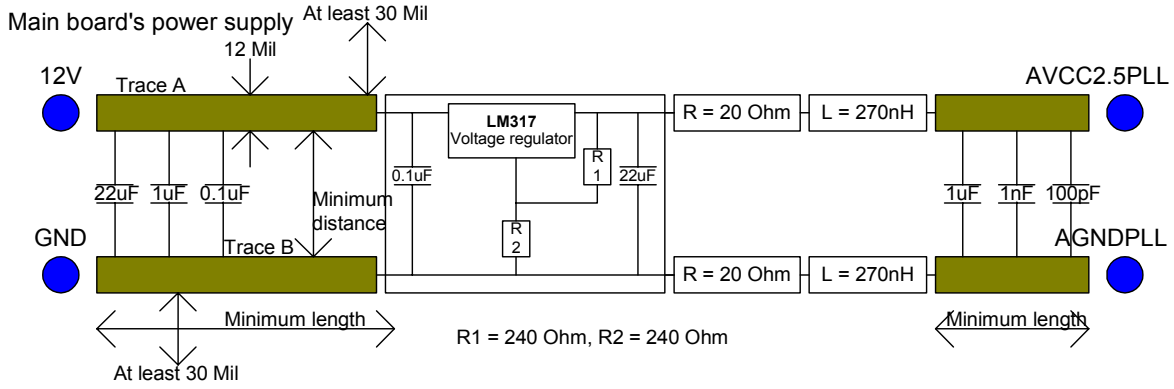
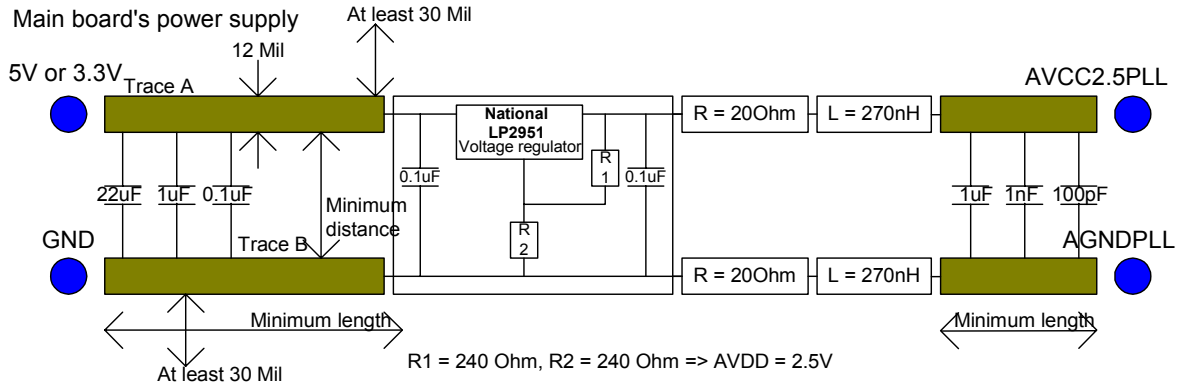


Figure 35: PLL Layout Guideline for Backplane Layout with 5/3.3V Power Supply



NOTE: In Figure 34 and Figure 35, Traces A and B must be parallel and the same length. Also, Galileo Technology recommends to route the traces on the component side, or print side, and, if possible, leave the area clean in all layers where the traces are routed.

Also, read the National LP2951 datasheet before using it in this layout.

18.4 PLL Characteristics

The PLL minimum pull-in time PLUS locking time is 1 ms. This means that the reset pin must be kept asserted for at least 1 ms after TClk is on.

18.5 PLL Power Filter Layout Considerations

For the two dedicated traces going from the supply source to the filtering circuit, the following must be garaunteed:

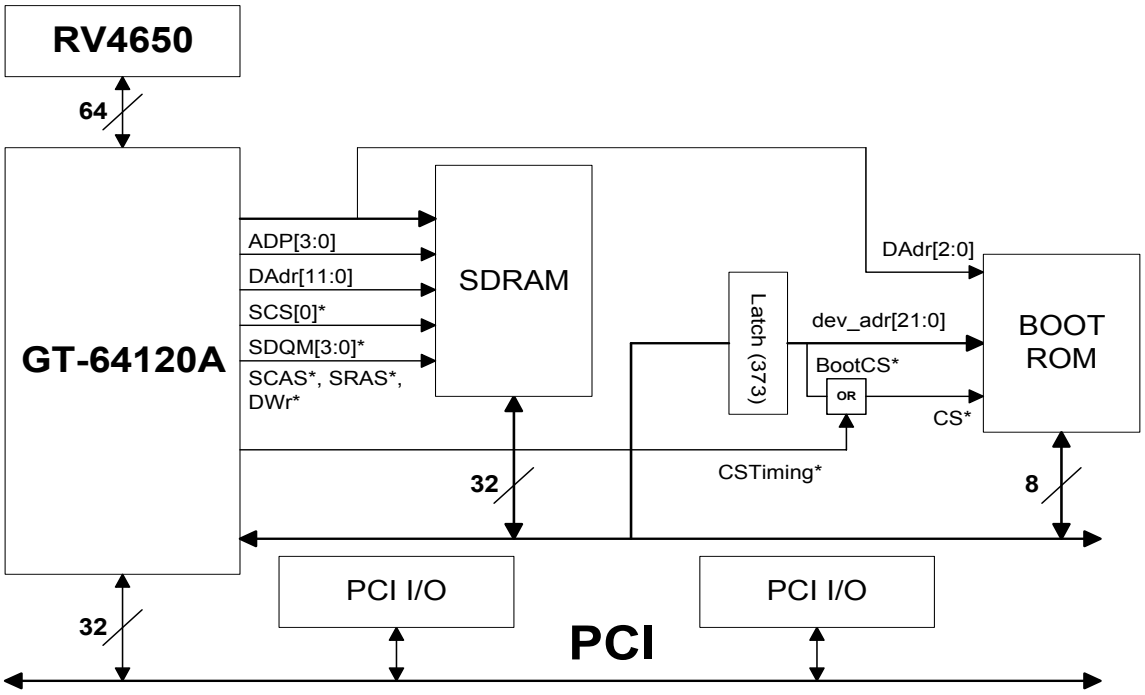
- Provide each trace with a minimum width of 20 mil.
- Route the traces in parallel, with minimal spacing.
- Give each trace an equal and minimal length.
- Route the traces in noise-free areas and as far as possible from high current traces.
- Place the 0.1nF capacitor as close as possible to the PLL DC supply pins.
- Place the capacitors in the shown order, with the smallest capacitor closest to the PLL's DC Supply Pins.

19. SYSTEM CONFIGURATIONS

19.1 Minimal System Configuration

- Low Cost RV4650 CPU
- 32-bit SDRAM
- 8-bit Boot EPROM
- 32-bit PCI

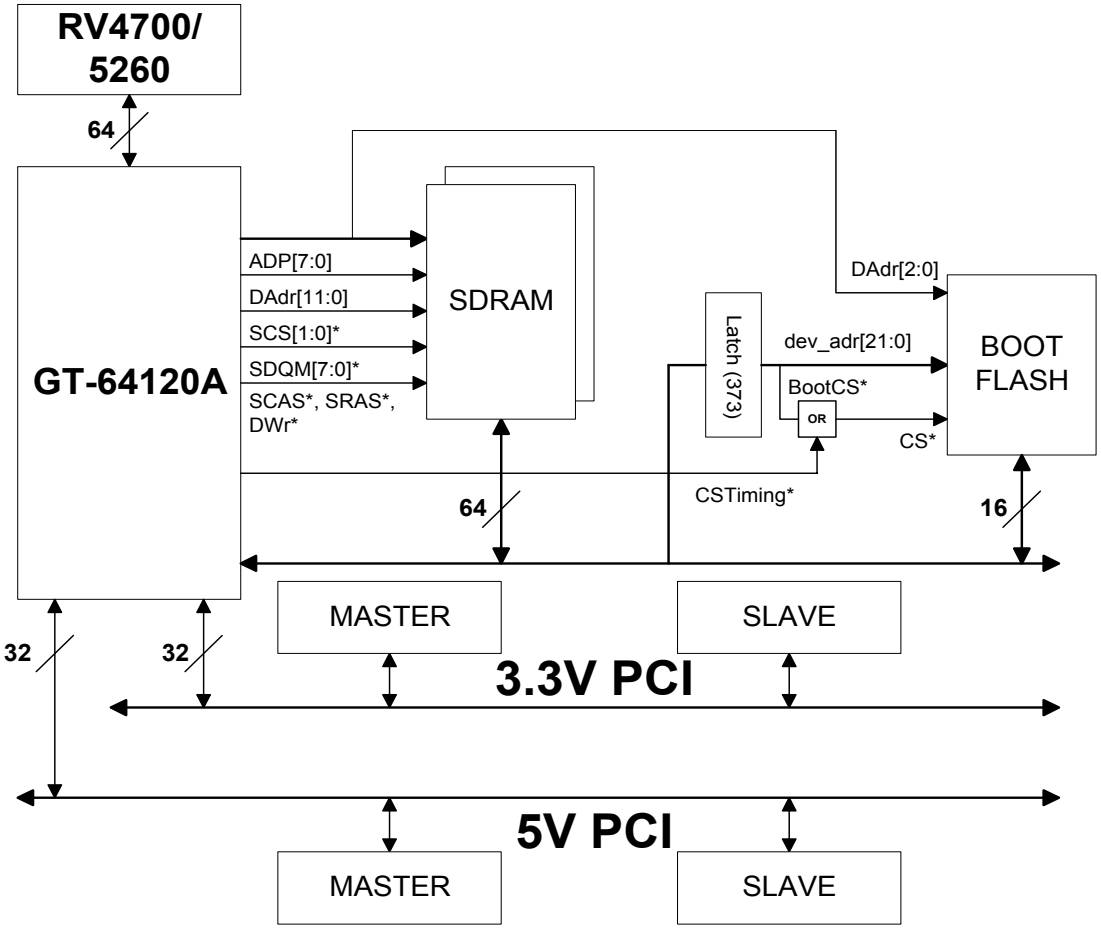
Figure 36: Minimal System Configuration



19.2 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).

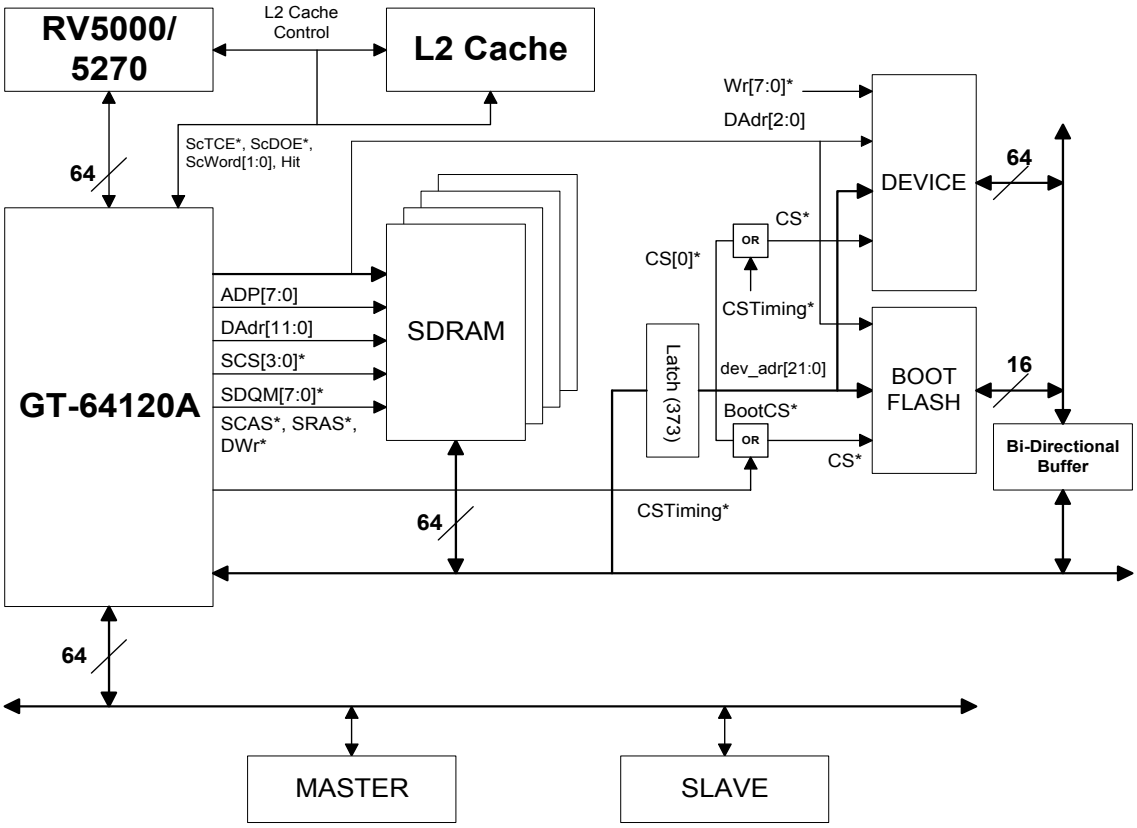
Figure 37: Typical System Configuration



19.3 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 38: High Performance System



20. REGISTERS TABLES

The GT–64120A's internal registers can be 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 should 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, which 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
- Timer/Counter

The following internal registers are CBE* sensitive:

- PCI Internal Registers
- PCI Configuration Registers
- Interrupt Registers

20.1 Access to On-Chip PCI Configuration Space Registers

An access from the CPU to one of the GT–64120A 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.

20.2 Register Maps

Table 70: CPU Configuration Register Map

Description	Offset	Page Number
CPU Interface Configuration	0x000	page 180
Multi-GT Register	0x120	page 182

Table 71: CPU Address Decode Register Map

Description	Offset	Page Number
SCS[1:0]* Low Decode Address	0x008	page 182
SCS[1:0]* High Decode Address	0x010	page 182
SCS[3:2]* Low Decode Address	0x018	page 182
SCS[3:2]* High Decode Address	0x020	page 182
CS[2:0]* Low Decode Address	0x028	page 183
CS[2:0]* High Decode Address	0x030	page 183
CS[3]* & Boot CS* Low Decode Address	0x038	page 183
CS[3]* & Boot CS* High Decode Address	0x040	page 183
PCI_0 I/O Low Decode Address	0x048	page 183
PCI_0 I/O High Decode Address	0x050	page 184
PCI_0 Memory 0 Low Decode Address	0x058	page 184
PCI_0 Memory 0 High Decode Address	0x060	page 184
PCI_0 Memory 1 Low Decode Address	0x080	page 184
PCI_0 Memory 1 High Decode Address	0x088	page 186
PCI_1 I/O Low Decode Address	0x090	page 185
PCI_1 I/O High Decode Address	0x098	page 185
PCI_1 Memory 0 Low Decode Address	0x0a0	page 185
PCI_1 Memory 0 High Decode Address	0x0a8	page 185
PCI_1 Memory 1 Low Decode Address	0x0b0	page 185
PCI_1 Memory 1 High Decode Address	0x0b8	page 186
Internal Space Decode	0x068	page 186
SCS[1:0]* Address Remap	0x0d0	page 186
SCS[3:2]* Address Remap	0x0d8	page 186

Table 71: CPU Address Decode Register Map (Continued)

Description	Offset	Page Number
CS[2:0]* Remap	0x0e0	page 186
CS[3]* & Boot CS* Remap	0x0e8	page 187
PCI_0 I/O Remap	0x0f0	page 187
PCI_0 Memory 0 Remap	0x0f8	page 187
PCI_0 Memory 1 Remap	0x100	page 187
PCI_1 I/O Remap	0x108	page 187
PCI_1 Memory 0 Remap	0x110	page 188
PCI_1 Memory 1 Remap	0x118	page 188

Table 72: CPU Error Report Register Map

Description	Offset	Page Number
CPU Error Address (Low)	0x070	page 188
CPU Error Address (High)	0x078	page 188
CPU Error Data (Low)	0x128	page 188
CPU Error Data (High)	0x130	page 189
CPU Error Parity	0x138	page 189

Table 73: CPU Sync Barrier Register Map

Description	Offset	Page Number
PCI_0 Sync Barrier Virtual Register	0x0c0	page 189
PCI_1 Sync Barrier Virtual Register	0x0c8	page 189

Table 74: SDRAM and Device Address Decode Register Map

Description	Offset	Page Number
SCS[0]* Low Decode Address	0x400	page 190
SCS[0]* High Decode Address	0x404	page 190
SCS[1]* Low Decode Address	0x408	page 190
SCS[1]* High Decode Address	0x40c	page 190
SCS[2]* Low Decode Address	0x410	page 190

Table 74: SDRAM and Device Address Decode Register Map (Continued)

Description	Offset	Page Number
SCS[2]* High Decode Address	0x414	page 191
SCS[3]* Low Decode Address	0x418	page 191
SCS[3]* High Decode Address	0x41c	page 191
CS[0]* Low Decode Address	0x420	page 191
CS[0]* High Decode Address	0x424	page 191
CS[1]* Low Decode Address	0x428	page 192
CS[1]* High Decode Address	0x42c	page 192
CS[2]* Low Decode Address	0x430	page 192
CS[2]* High Decode Address	0x434	page 192
CS[3]* Low Decode Address	0x438	page 192
CS[3]* High Decode Address	0x43c	page 193
Boot CS* Low Decode Address	0x440	page 193
Boot CS* High Decode Address	0x444	page 193
Address Decode Error	0x470	page 193

Table 75: SDRAM Configuration Register Map

Description	Offset	Page Number
SDRAM Configuration	0x448	page 193
SDRAM Operation Mode	0x474	page 195
SDRAM Burst Mode	0x478	page 195
SDRAM Address Decode	0x47c	page 195

Table 76: SDRAM Parameters Register Map

Description	Offset	Page Number
SDRAM Bank0 Parameters	0x44c	page 196
SDRAM Bank1 Parameters	0x450	page 197
SDRAM Bank2 Parameters	0x454	page 197
SDRAM Bank3 Parameters	0x458	page 197

Table 77: ECC Register Map

Description	Offset	Page Number
ECC Error Address	0x490	page 199
ECC Error Data (High)	0x480	page 198
ECC Error Data (Low)	0x484	page 198
ECC from Memory	0x488	page 198
ECC Calculated	0x48c	page 198

Table 78: Device Parameters Register Map

Description	Offset	Page Number
Device Bank0 Parameters	0x45c	page 199
Device Bank1 Parameters	0x460	page 200
Device Bank2 Parameters	0x464	page 200
Device Bank3 Parameters	0x468	page 200
Device Boot Bank Parameters	0x46c	page 200

Table 79: DMA Record Register Map

Description	Offset	Page Number
Channel 0 DMA Byte Count	0x800	page 201
Channel 1 DMA Byte Count	0x804	page 201
Channel 2 DMA Byte Count	0x808	page 201
Channel 3 DMA Byte Count	0x80c	page 201
Channel 0 DMA Source Address	0x810	page 201
Channel 1 DMA Source Address	0x814	page 201
Channel 2 DMA Source Address	0x818	page 202
Channel 3 DMA Source Address	0x81c	page 202
Channel 0 DMA Destination Address	0x820	page 202
Channel 1 DMA Destination Address	0x824	page 202
Channel 2 DMA Destination Address	0x828	page 202
Channel 3 DMA Destination Address	0x82c	page 202
Channel 0 Next Record Pointer	0x830	page 202
Channel 1 Next Record Pointer	0x834	page 203

Table 79: DMA Record Register Map (Continued)

Description	Offset	Page Number
Channel 2 Next Record Pointer	0x838	page 203
Channel 3 Next Record Pointer	0x83c	page 203
Channel 0 Current Descriptor Pointer	0x870	page 203
Channel 1 Current Descriptor Pointer	0x874	page 203
Channel 2 Current Descriptor Pointer	0x878	page 203
Channel 3 Current Descriptor Pointer	0x87c	page 204
Channel 0 Control	0x840	page 204
Channel 1 Control	0x844	page 207
Channel 2 Control	0x848	page 207
Channel 3 Control	0x84c	page 207

Table 80: DMA Arbiter Register Map

Description	Offset	Page Number
Arbiter Control	0x860	page 207

Table 81: Timer/Counter Register Map

Description	Offset	Page Number
Timer /Counter 0	0x850	page 208
Timer /Counter 1	0x854	page 208
Timer /Counter 2	0x858	page 208
Timer /Counter 3	0x85c	page 209
Timer /Counter Control	0x864	page 209

Table 82: PCI Internal Register Map

Description	Offset	Page Number
PCI_0 Command	0xc00	page 210
PCI_1 Command	0xc80	page 211
PCI_0 Time Out & Retry	0xc04	page 211
PCI_1 Time Out & Retry	0xc84	page 212

Table 82: PCI Internal Register Map (Continued)

Description	Offset	Page Number
PCI_0 SCS[1:0]* Bank Size	0xc08	page 212
PCI_1 SCS[1:0]* Bank Size	0xc88	page 212
PCI_0 SCS[3:2]* Bank Size	0xc0c	page 213
PCI_1 SCS[3:2]* Bank Size	0xc8c	page 213
PCI_0 CS[2:0]* Bank Size	0xc10	page 213
PCI_1 CS[2:0]* Bank Size	0xc90	page 213
PCI_0 CS[3]* & Boot CS* Bank Size	0xc14	page 214
PCI_1 CS[3]* & Boot CS* Bank Size	0xc94	page 214
PCI_0 Base Address Registers' Enable	0xc3c	page 214
PCI_1 Base Address Registers' Enable	0xcbc	page 215
PCI_0 Prefetch/Max Burst Size	0xc40	page 215
PCI_1 Prefetch/Max Burst Size	0xcc0	page 216
PCI_0 SCS[1:0]* Base Address Remap	0xc48	page 216
PCI_1 SCS[1:0]* Base Address Remap	0xcc8	page 217
PCI_0 SCS[3:2]* Base Address Remap	0xc4c	page 217
PCI_1 SCS[3:2]* Base Address Remap	0xccc	page 218
PCI_0 CS[2:0]* Base Address Remap	0xc50	page 218
PCI_1 CS[2:0]* Base Address Remap	0xcd0	page 219
PCI_0 CS[3]* & Boot CS* Address Remap	0xc54	page 219
PCI_1 CS[3]* & Boot CS* Address Remap	0xcd4	page 219
PCI_0 Swapped SCS[1:0]* Base Address Remap	0xc58	page 217
PCI_1 Swapped SCS[1:0]* Base Address Remap	0xcd8	page 217
PCI_0 Swapped SCS[3:2]* Base Address Remap	0xc5c	page 218
PCI_1 Swapped SCS[3:2]* Base Address Remap	0xcdc	page 218
PCI_0 Swapped CS[3]* & BootCS* Base Address Remap	0xc64	page 219
PCI_1 Swapped CS[3]* & BootCS* Base Address Remap	0xce4	page 220
PCI_0 Configuration Address	0xcf8	page 220
PCI_1 Configuration Address	0xcf0	page 220
PCI_0 Configuration Data Virtual Register	0xcfc	page 220
PCI_1 Configuration Data Virtual Register	0xcf4	page 221

Table 82: PCI Internal Register Map (Continued)

Description	Offset	Page Number
PCI_0 Interrupt Acknowledge Virtual Register	0xc34	page 221
PCI_1 Interrupt Acknowledge Virtual Register	0xc30	page 221

Table 83: Interrupts Register Map

Description	Offset	Page Number
Interrupt Cause Register	0xc18	page 221
High Interrupt Cause Register	0xc98	page 223
CPU Interrupt Mask Register	0xc1c	page 224
CPU High Interrupt Mask Register	0xc9c	page 225
PCI_0 Interrupt Cause Mask Register	0xc24	page 225
PCI_0 High Interrupt Cause Mask Register	0xca4	page 227
PCI_0 SErr0 Mask	0xc28	page 227
PCI_1 SErr1 Mask	0xca8	page 228
CPU Select Cause Register	0xc70	page 223
PCI_0 Interrupt Select Register	0xc74	page 224

Table 84: PCI Configuration Register Map

Description	Offset from PCI_0 or CPU	Offset from PCI_1	Page Number
PCI_0 Device and Vendor ID	0x000	0x080	page 229
PCI_1 Device and Vendor ID	0x080	0x000	page 229
PCI_0 Status and Command	0x004	0x084	page 230
PCI_1 Status and Command	0x084	0x004	page 231
PCI_0 Class Code and Revision ID	0x008	0x088	page 231
PCI_1 Class Code and Revision ID	0x088	0x008	page 232
PCI_0 BIST, Header Type, Latency Timer, Cache Line	0x00c	0x08c	page 232
PCI_1 BIST, Header Type, Latency Timer, Cache Line	0x08c	0x00c	page 232
PCI_0 SCS[1:0]* Base Address	0x010	0x090	page 233
PCI_1 SCS[1:0]* Base Address	0x090	0x010	page 233

Table 84: PCI Configuration Register Map (Continued)

Description	Offset from PCI_0 or CPU	Offset from PCI_1	Page Number
PCI_0 SCS[3:2]* Base Address	0x014	0x094	page 234
PCI_1 SCS[3:2]* Base Address	0x094	0x014	page 234
PCI_0 CS[2:0]* Base Address	0x018	0x098	page 234
PCI_1 CS[2:0]* Base Address	0x098	0x018	page 235
PCI_0 CS[3]* & Boot CS* Base Address	0x01c	0x09c	page 235
PCI_1 CS[3]* & Boot CS* Base Address	0x09c	0x01c	page 235
PCI_0 Internal Registers Memory Mapped Base Address	0x020	0x0a0	page 235
PCI_1 Internal Registers Memory Mapped Base Address	0x0a0	0x020	page 236
PCI_0 Internal Registers I/O Mapped Base Address	0x024	0x0a4	page 236
PCI_1 Internal Registers I/O Mapped Base Address	0x0a4	0x024	page 236
PCI_0 Subsystem ID and Subsystem Vendor ID	0x02c	0x0ac	page 237
PCI_1 Subsystem ID and Subsystem Vendor ID	0x0ac	0x02c	page 237
Expansion ROM Base Address Register	0x030	0x0b0	page 237
PCI_0 Capability List Pointer	0x034	0x0b4	page 237
PCI_1 Capability List Pointer	0x0b4	0x034	page 238
PCI_0 Interrupt Pin and Line	0x03c	0x0bc	page 238
PCI_1 Interrupt Pin and Line	0x0bc	0x03c	page 238
PCI_0 PMC	0x040	0x0c0	page 238
PCI_1 PMC	0x0c0	0x040	page 239
PCI_0 PMCSR	0x044	0x0c4	page 239
PCI_1 PMCSR	0x0c4	0x044	page 240

Table 85: PCI Configuration, Function 1, Register Map

Description	Offset from PCI_0 or CPU	Offset from PCI_1	Page Number
PCI_0 Swapped SCS[1:0]* Base Address	0x110	0x190	page 241
PCI_1 Swapped SCS[1:0]* Base Address	0x190	0x110	page 241
PCI_0 Swapped SCS[3:2]* Base Address	0x114	0x194	page 241
PCI_1 Swapped SCS[3:2]* Base Address	0x194	0x114	page 242
PCI_0 Swapped CS[3]* & Boot CS* Base Address	0x11c	0x19c	page 242
PCI_1 Swapped CS[3]* & Boot CS* Base Address	0x19c	0x11c	page 242

Table 86: I_2O Support Register Map

NOTE: I_2O 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.

Description	Offset	Page Number
Inbound Message Register 0	0x10	page 243
Inbound Message Register 1	0x14	page 243
Outbound Message Register 0	0x18	page 243
Outbound Message Register 1	0x1c	page 244
Inbound Doorbell Register	0x20	page 244
Inbound Interrupt Cause Register	0x24	page 244
Inbound Interrupt Mask Register	0x28	page 245
Outbound Doorbell Register	0x2c	page 245
Outbound Interrupt Cause Register	0x30	page 246
Outbound Interrupt Mask Register	0x34	page 246
Inbound Queue Port Virtual Register	0x40	page 247
Outbound Queue Port Virtual Register	0x44	page 247
Queue Control Register	0x50	page 247
Queue Base Address Register	0x54	page 248
Inbound Free Head Pointer Register	0x60	page 248
Inbound Free Tail Pointer Register	0x64	page 248
Inbound Post Head Pointer Register	0x68	page 249

Table 86: I_2O Support Register Map (Continued)

NOTE: I_2O 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.

Description	Offset	Page Number
Inbound Post Tail Pointer Register	0x6c	page 249
Outbound Free Head Pointer Register	0x70	page 249
Outbound Free Tail Pointer Register	0x74	page 250
Outbound Post Head Pointer Register	0x78	page 250
Outbound Post Tail Pointer Register	0x7c	page 250

20.3 CPU Configuration

Table 87: CPU Interface Configuration, Offset: 0x000

Bits	Field Name	Function	Initial Value
8:0	CacheOpMap	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].	0x0
9	CachePres	Secondary Cache support 0 - GT-64012 not present. Hit input not monitored. 1 - GT-64012 present. Hit input monitored.	0x0
10	Reserved	Reserved	0x0
11	WriteMode	CPU Write mode 0 - Pipelined writes mode 1 - R4000 mode. There must be at least two dead-cycles minimum between consecutive address-phase.	0x0
12	Endianess	Byte Orientation 0 - Big endian 1 - Little endian	Sampled at RESET via the Interrupt* pin.
13	Reserved	Must be 0.	0x0
14	R5KL2 present	Second level cache present. 0 - R5KL2 not present. Hit input not monitored. 1 - R5KL2 present (Hit input monitored)	0x0

Table 87: CPU Interface Configuration, Offset: 0x000 (Continued)

Bits	Field Name	Function	Initial Value
15	External Hit Delay	Register Second Level Cache Hit Signal 0 - Not sampled inside the GT–64120A. 1 - Sampled inside the GT–64120A. **NOTE:** TagRAMs used with L2/L3 cache output the Hit signal registered. If for some reason, the TagRAM in use outputs the Hit signal non-registered, this bit must be set to 1 to maintain timing relationship between the cache and the GT–64120A.	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–64120A support 0 - Not Supported 1 - Supported	Sampled at RESET via the DAdr[10] pin
19	SysADCValid	GT–64120A 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
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

Table 88: 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–64120A responds with activity.	Value sampled at reset on Ready* and CSTiming*.
31:2	Reserved		0x0

20.4 CPU Address Decode

Table 89: 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 90: SCS[1:0]* High Decode Address, Offset: 0x010

Bits	Field Name	Function	Initial Value
6:0	High	SDRAM banks 1 and 0 are accessed when the decoded addresses are between Low and High.	0x07
31:7	Reserved		0x0

Table 91: 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 92: SCS[3:2]* High Decode Address, Offset: 0x020

Bits	Field Name	Function	Initial Value
6:0	High	SDRAM banks 3 and 2 are accessed when the decoded addresses are between Low and High.	0x0f
31:7	Reserved		0x0

Table 93: 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 94: CS[2:0]* High Decode Address, Offset: 0x030

Bits	Field Name	Function	Initial Value
6:0	High	Device banks 2, 1, and 0 will be accessed when the decoded addresses are between Low and High.	0x70
31:7	Reserved		0x0

Table 95: 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 96: CS[3]* & Boot CS* High Decode Address, Offset: 0x040

Bits	Field Name	Function	Initial Value
6:0	High	Device bank 3 and the boot bank are accessed when the decoded addresses are between Low and High.	0x7f
31:7	Reserved		0x0

Table 97: 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 98: PCI_0 I/O High Decode Address, Offset: 0x050

Bits	Field Name	Function	Initial Value
6:0	High	The PCI_0 I/O address space is accessed when the decoded addresses are between Low and High.	0x0f
31:7	Reserved		0x0

Table 99: 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 100: PCI_0 Memory 0 High Decode Address, Offset: 0x060

Bits	Field Name	Function	Initial Value
6:0	High	The PCI_0 memory address space is accessed when the decoded addresses are between Low and High.	0x1f
31:7	Reserved		0x0

Table 101: 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 102: PCI_0 Memory 1 High Decode Address, Offset: 0x088

Bits	Field Name	Function	Initial Value
6:0	High	The PCI_0 memory address space is accessed when the decoded addresses are between Low and High.	0x1f
31:7	Reserved		0x0

Table 103: 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 104: PCI_1 I/O High Decode Address, Offset: 0x098

Bits	Field Name	Function	Initial Value
6:0	High	The PCI_1 I/O address space is accessed when the decoded addresses are between Low and High.	0x0f
31:7	Reserved		0x0

Table 105: 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 106: PCI_1 Memory 0 High Decode Address, Offset: 0x0a8

Bits	Field Name	Function	Initial Value
6:0	High	The PCI_1 memory address space will be accessed when the decoded addresses are between Low and High.	0x1f
31:7	Reserved		0x0

Table 107: 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 108: PCI_1 Memory 1 High Decode Address, Offset: 0x0b8

Bits	Field Name	Function	Initial Value
6:0	High	The PCI_1 memory address space is accessed when the decoded addresses are between Low and High.	0x2f
31:7	Reserved		0x0

Table 109: Internal Space Decode, Offset: 0x068

Bits	Field Name	Function	Initial Value
14:0	IntDecode	Registers inside the GT–64120A are accessed when SysAD bits 35:21 match the value programmed in bits 14:0.	0x00a0
31:15	Reserved		0x0

Table 110: 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 111: 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 112: 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 113: 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 114: 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 115: 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 116: 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 117: 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 118: 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 119: 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

20.5 CPU Errors Report

Table 120: CPU Error Address (Low), Offset: 0x070

Bits	Field Name	Function	Initial Value
31:0	IllegLoAdd	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 121: CPU Error Address (High), Offset: 0x078

Bits	Field Name	Function	Initial Value
3:0	IllegHiAdd	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 122: 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 123: 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 124: 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.	0x0

20.6 CPU Sync Barrier

Table 125: 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 126: 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

20.7 SDRAM and Device Address Decode

Table 127: SCS[0]* Low Decode Address, Offset: 0x400

Bits	Field Name	Function	Initial Value
7:0	Low	SDRAM bank 0 is accessed when the decoded addresses are between Low and High.	0x00
31:8	Reserved		0x0

Table 128: SCS[0]* High Decode Address, Offset: 0x404

Bits	Field Name	Function	Initial Value
7:0	High	SDRAM bank 0 is accessed when the decoded addresses are between Low and High.	0x07
31:8	Reserved		0x0

Table 129: SCS[1]* Low Decode Address, Offset: 0x408

Bits	Field Name	Function	Initial Value
7:0	Low	SDRAM bank 1 is accessed when the decoded addresses are between Low and High.	0x08
31:8	Reserved		0x0

Table 130: SCS[1]* High Decode Address, Offset: 0x40c

Bits	Field Name	Function	Initial Value
7:0	High	SDRAM bank 1 is accessed when the decoded addresses are between Low and High.	0x0f
31:8	Reserved		0x0

Table 131: SCS[2]* Low Decode Address, Offset: 0x410

Bits	Field Name	Function	Initial Value
7:0	Low	SDRAM bank 2 is accessed when the decoded addresses are between Low and High.	0x10
31:8	Reserved		0x0

Table 132: SCS[2]* High Decode Address, Offset: 0x414

Bits	Field Name	Function	Initial Value
7:0	High	SDRAM bank 2 is accessed when the decoded addresses are between Low and High.	0x17
31:8	Reserved		0x0

Table 133: SCS[3]* Low Decode Address, Offset: 0x418

Bits	Field Name	Function	Initial Value
7:0	Low	SDRAM bank 3 is accessed when the decoded addresses are between Low and High.	0x18
31:8	Reserved		0x0

Table 134: SCS[3]* High Decode Address, Offset: 0x41c

Bits	Field Name	Function	Initial Value
7:0	High	SDRAM bank 3 is accessed when the decoded addresses are between Low and High.	0x1f
31:8	Reserved		0x0

Table 135: CS[0]* Low Decode Address, Offset: 0x420

Bits	Field Name	Function	Initial Value
7:0	Low	Device bank 0 is accessed when the decoded addresses are between Low and High.	0xc0
31:8	Reserved		0x0

Table 136: CS[0]* High Decode Address, Offset: 0x424

Bits	Field Name	Function	Initial Value
7:0	High	Device bank 0 is accessed when the decoded addresses are between Low and High.	0xc7
31:8	Reserved		0x0

Table 137: CS[1]* Low Decode Address, Offset: 0x428

Bits	Field Name	Function	Initial Value
7:0	Low	Device bank 1 is accessed when the decoded addresses are between Low and High.	0xc8
31:8	Reserved		0x0

Table 138: CS[1]* High Decode Address, Offset: 0x42c

Bits	Field Name	Function	Initial Value
7:0	High	Device bank 1 is accessed when the decoded addresses are between Low and High.	0xcf
31:8	Reserved		0x0

Table 139: CS[2]* Low Decode Address, Offset: 0x430

Bits	Field Name	Function	Initial Value
7:0	Low	Device bank 2 is accessed when the decoded addresses are between Low and High.	0xd0
31:8	Reserved		0x0

Table 140: CS[2]* High Decode Address, Offset: 0x434

Bits	Field Name	Function	Initial Value
7:0	High	Device bank 2 is accessed when the decoded addresses are between Low and High.	0xdf
31:8	Reserved		0x0

Table 141: CS[3]* Low Decode Address, Offset: 0x438

Bits	Field Name	Function	Initial Value
7:0	Low	Device bank 3 is accessed when the decoded addresses are between Low and High.	0xf0
31:8	Reserved		0x0

Table 142: CS[3]* High Decode Address, Offset: 0x43c

Bits	Field Name	Function	Initial Value
7:0	High	Device bank 3 is accessed when the decoded addresses are between Low and High.	0xfb
31:8	Reserved		0x0

Table 143: Boot CS* Low Decode Address, Offset: 0x440

Bits	Field Name	Function	Initial Value
7:0	Low	Boot bank is accessed when the decoded addresses are between Low and High.	0xfc
31:8	Reserved		0x0

Table 144: Boot CS* High Decode Address, Offset: 0x444

Bits	Field Name	Function	Initial Value
7:0	High	Boot bank is accessed when the decoded addresses are between Low and High.	0xff
31:8	Reserved		0x0

Table 145: 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

20.8 SDRAM Configuration

Table 146: 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

Table 146: SDRAM Configuration, Offset: 0x448 (Continued)

Bits	Field Name	Function	Initial Value
15	RMW	Enable Read Modify Write 0 - Disabled 1 - Enabled	0x0
16	StagRef	Staggered Refresh 0 - Staggered refresh 1- Non-staggered refresh	0x0
17	CPUtoDRAMErr	Propagate CPU write parity error to SDRAM 0 - SDRAM interface unit generate bad ECC (2 bits error) in case of CPU write with bad parity driven on SysADC 1 - SDRAM interface unit always generates correct ECC on write access to SDRAM	0x0
18	ECCInt	ECC error interrupt generation 0 - Only if two (or more) errors are detected 1 - If one ore more errors are detected	0x0
19	DupCntl	Duplicate Control Pins 0 - Do not duplicate 1 - Duplicate control pins • DMAReq[0]* = SRAS* • DMAReq[3]* = SCAS* • BypsOE* = DWr*	0x0
20	DupBA	Duplicate Bank Addresses 0 - Do not duplicate 1 - Duplicate bank addresses • 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 • DMAReq[3]* = EOT[0]	0x0
22	DupEOT1	Duplicate End of Transfer 1 0 - Do not duplicate 1 - Duplicate End of Transfer 1 • Ready* = EOT[1]	0x0
23	RegSDRAM	Registered SDRAM Enable 0 - Disable 1 - Enable	0x0

Table 146: 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 147: 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 148: SDRAM Burst Mode, Offset: 0x478

Bits	Field Name	Function	Initial Value
1:0	Reserved	Must be 0x0.	0x1
2	Burst Order	0 - Linear 1 - Sub-block Must be 1.	0x1
11:3	Reserved	Must be 0x1.	0x1
31:12	Reserved		-

Table 149: 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 65 for more information.	0x2
31:3	Reserved		

20.9 SDRAM Parameters

Table 150: 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	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	0x0
11	SDRAMSize0	16 or 64/128/256 Mbit SDRAM 0 - 16 Mbit SDRAM 1 - 64/128/256 Mbit SDRAM	0x0

Table 150: SDRAM Bank0 Parameters, Offset: 0x44c (Continued)

Bits	Field Name	Function	Initial Value
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 151: 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 152: 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 153: 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

Table 153: SDRAM Bank3 Parameters, Offset: 0x458 (Continued)

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

20.10 ECC

Table 154: ECC Error Data (High), Offset: 0x480

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

Table 155: ECC Error Data (Low), Offset: 0x484

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

Table 156: 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 157: ECC Calculated, Offset: 0x48c

Bits	Field Name	Function	Initial Value
7:0	ECCCalc	Eight bits of ECC code calculated inside the GT–64120A.	0x0
31:8	Reserved		0x0

Table 158: ECC Error Address, 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	ECCAddr	Bits[31:2] of SDRAM address of last read access with ECC error detected	0x0

20.11 Device Parameters

Table 159: Device Bank0 Parameters, Offset: 0x45c

Bits	Field Name	Function	Initial Value
2:0	TurnOff	The number of cycles between the deassertion 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 deassertion and assertion of Wr* signals.	0x7
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

Table 159: Device Bank0 Parameters, Offset: 0x45c (Continued)

Bits	Field Name	Function	Initial Value
23	DevLoc	32/16/8 bit device location 0 - Even bank • AD[31:0], AD[15:0], and AD[7:0] 1 - Odd bank • 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 160: Device Bank1 Parameters, Offset: 0x460

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Device Bank0.	0x386fffff

Table 161: Device Bank2 Parameters, Offset: 0x464

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Device Bank0.	0x386fffff

Table 162: Device Bank3 Parameters, Offset: 0x468

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

Table 163: 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?fffff

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.

20.12 DMA Record

Table 164: 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 165: Channel 1 DMA Byte Count, Offset: 0x804

Bits	Field Name	Function	Initial Value
31:0		Functions as in Channel 0 DMA Byte Count.	0x0

Table 166: Channel 2 DMA Byte Count, Offset: 0x808

Bits	Field Name	Function	Initial Value
31:0		Functions as in Channel 0 DMA Byte Count.	0x0

Table 167: Channel 3 DMA Byte Count, Offset: 0x80c

Bits	Field Name	Function	Initial Value
31:0		Functions as in Channel 0 DMA Byte Count.	0x0

Table 168: Channel 0 DMA Source Address, Offset: 0x810

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address from which the DMA controller reads the data.	0x0

Table 169: Channel 1 DMA Source Address, Offset: 0x814

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address from which the DMA controller reads the data.	0x0

Table 170: Channel 2 DMA Source Address, Offset: 0x818

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address from which the DMA controller reads the data.	0x0

Table 171: Channel 3 DMA Source Address, Offset: 0x81c

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address from which the DMA controller reads the data.	0x0

Table 172: Channel 0 DMA Destination Address, Offset: 0x820

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address to which the DMA controller writes the data.	0x0

Table 173: Channel 1 DMA Destination Address, Offset: 0x824

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address to which the DMA controller writes the data.	0x0

Table 174: Channel 2 DMA Destination Address, Offset: 0x828

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address to which the DMA controller writes the data.	0x0

Table 175: Channel 3 DMA Destination Address, Offset: 0x82c

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address to which the DMA controller writes the data.	0x0

Table 176: 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 177: 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 178: 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 179: 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 180: Current Descriptor Pointer 0, Offset: 0x870

Bits	Field Name	Function	Initial Value
31:0	CDPTR0	Channel 0 Current Address Descriptor Pointer	0x0

Table 181: Current Descriptor Pointer 1, Offset: 0x874

Bits	Field Name	Function	Initial Value
31:0	CDPTR1	Channel 1 Current Address Descriptor Pointer	0x0

Table 182: Current Descriptor Pointer 2, Offset: 0x878

Bits	Field Name	Function	Initial Value
31:0	CDPTR2	Channel 2 Current Address Descriptor Pointer	0x0

Table 183: Current Descriptor Pointer 3, Offset: 0x87c

Bits	Field Name	Function	Initial Value
31:0	CDPTR3	Channel 3 Current Address Descriptor Pointer	0x0

20.13 DMA Channel Control

Table 184: 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 184: Channel 0 Control, Offset: 0x840 (Continued)

Bits	Field Name	Function	Initial Value
9	ChainMod	Chained Mode 0 - Chained mode 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. 1 - Non-Chained mode Only uses the values programmed by the CPU (or PCI) directly into the ByteCt, SrcAdd, and DestAdd registers.	0x0
10	IntMode	Interrupt Mode 0 - Interrupt asserted every time the DMA byte count reaches terminal count. 1 - Interrupt every NULL pointer (in Chained mode).	0x0
11	TransMod	Transfer Mode 0 - Demand 1 - Block	0x0
12	ChanEn	Channel Enable 0 - Disable 1 - Enable	0x0
13	FetNexRec	Fetch Next Record 1 - Forces a fetch of the next record, even if the current DMA has not ended. This bit is reset after fetch is completed and is only meaningful in Chained mode.	0x0
14	DMAActSt	DMA Activity Status Read only. Assertion of this bit is caused by asserting ChanEn (bit 12). 0 - Channel is not active. 1 - Channel is active.	0x0
15	SDA	Source Destination Alignment 0 - Alignment is towards the source address. 1 - Alignment is towards the destination address **NOTE:** No support for destination alignment when burst limit is 1, 2, or 4 bytes.	0x0
16	MDREQ	Mask DMA Requests 0 - Don’t mask 1 - Mask DMA requests for 3 cycles starting DMA arbitration cycle.	0x0

Table 184: Channel 0 Control, Offset: 0x840 (Continued)

Bits	Field Name	Function	Initial Value
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
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

Table 184: Channel 0 Control, Offset: 0x840 (Continued)

Bits	Field Name	Function	Initial Value
28	DMAReqSrc	DMA Request Source 0 - Request taken from DMAReq* pin. 1 - Request taken from timer/counter.	0x0
31:29	Reserved		0x0

Table 185: Channel 1 Control, Offset: 0x844

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Channel 0 Control.	0x0

Table 186: Channel 2 Control, Offset: 0x848

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Channel 0 Control.	0x0

Table 187: Channel 3 Control, Offset: 0x84c

Bits	Field Name	Function	Initial Value
31:0	Various	Fields function as in Channel 0 Control.	0x0

20.14 DMA Arbiter

Table 188: 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

Table 188: Arbiter Control, Offset: 0x860 (Continued)

Bits	Field Name	Function	Initial Value
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

20.15 Timer / Counter

Table 189: Timer/Counter 0, Offset: 0x850

Bits	Field Name	Function	Initial Value
31:0	TC0Value	The counter or timer initial value.	0xffffffff

Table 190: Timer/Counter 1, Offset: 0x854

Bits	Field Name	Function	Initial Value
23:0	TC1Value	The counter or timer initial value.	0xffffff
31:24	Reserved		0x0

Table 191: Timer/Counter 2, Offset: 0x858

Bits	Field Name	Function	Initial Value
23:0	TC2Value	The counter or timer initial value.	0xffffff
31:24	Reserved		0x0

Table 192: Timer/Counter 3, Offset: 0x85c

Bits	Field Name	Function	Initial Value
23:0	TC3Value	The counter or timer initial value.	0xffffff
31:24	Reserved		0x0

Table 193: 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		0x0

20.16 PCI Internal

Table 194: PCI_0 Command, Offset: 0xc00

Bits	Field Name	Function	Initial Value
0	MByteSwap	Master Byte Swap. When set to zero, the GT–64120A 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	Indicates the ratio between TClk and PClk as follows: x00 - Default mode. When the PClk ranges from DC to 66MHz. This mode works in all cases. Use following settings for higher performance. 001 - When PClk is higher than or equal to half the TClk frequency (e.g. when TClk 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 PClk is higher than or equal to half of TCLK frequency (e.g. TClk = 100MHz, PClk = 50MHz). 101 - When PClk is higher than or equal to a third of the TClk frequency but less than half of the TCLK frequency. 11x - When the two clocks are synchronized and PClk is higher than or equal to third of the TCLK frequency but less than half of the TCLK frequency. **NOTE:** Regardless of the selected syncmode, PClk frequency must be smaller than TClk frequency by at least 1MHz	0x0
7:4	Reserved	Read only.	0x0
9:8	Reserved	Must be 0.	0x0
10	MWordSwap*	Master Word Swap When set to 1, the GT–64120A PCI master swaps the words of the incoming and outgoing PCI data (swaps the 2 words of a long-word). **NOTE:** This bit is not supported in a 64-bit PCI configuration.	0x0
11	SWordSwap*	Slave Word Swap When set to 1, the GT–64120A 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. **NOTE:** This bit is not supported in a 64-bit PCI configuration.	0x0

Table 194: PCI_0 Command, Offset: 0xc00 (Continued)

Bits	Field Name	Function	Initial Value
12	SSBWord-Swap*	Slave Swap BAR Word Swap. When set to 1, the GT–64120A 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. **NOTE:** This bit is not supported in a 64-bit PCI configuration.	0x0
15:13	Reserved	Must be 0.	0x0
16	SByteSwap	Slave Byte Swap. When set to zero, the GT–64120A PCI slave 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.
25:17	Reserved	Must be 0.	0x0
26	DRAMtoPCIErr	Propagate SDRAM ECC errors to PCI 0 - PAR is always driven by the GT–64120A with even parity matching the address/data 1 - In case of PCI read from SDRAM which results in ECC error detection, GT–64120A drives PAR with parity NOT matching the data	0x0
31:27	Reserved	Must be 0.	0x0

Table 195: PCI_1 Command, Offset: 0xc80

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for the PCI_0 Command.	

Table 196: PCI_0 Timeout & ReTry, Offset: 0xc04

Bits	Field Name	Function	Initial Value
7:0	Timeout0	Specifies in PCI clock units the number of clocks the GT–64120A, as a slave, holds the PCI bus before the generation of retry termination. Used for the first data transfer.	0x0f
15:8	Timeout1	Specifies in PCI clock units the number of clocks the GT–64120A, as a slave, holds the PCI bus before the generation of disconnect termination. Used for data transfers following the first data.	0x07

Table 196: PCI_0 Timeout & ReTry, Offset: 0xc04 (Continued)

Bits	Field Name	Function	Initial Value
23:16	RetryCtr	Specifies the number of retries of the GT–64120A Master. The GT–64120A 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

Table 197: PCI_1 Timeout & ReTry, Offset: 0xc84

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Time Out and ReTry.	0x070f

Table 198: PCI_0 SCS[1:0] Bank Size, Offset: 0xc08

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	BankSize	Specifies the SCS[1:0]* address mapping in conjunction with the SCS[1:0]* Base Address register. 0 - The corresponding bit in the address and in the base address must be equal in order to have a hit. 1 -The corresponding bit in the address is a “don’t-care”. 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).	0x00fff

Table 199: 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 200: PCI_0 SCS[3:2]* Bank Size, Offset: 0xc0c

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
31:12	BankSize	Specifies the SCS[3:2]* address mapping in conjunction with the SCS[3:2]* Base Address register. 0 -The corresponding bit in the address and in the base address must be equal in order to have a hit. 1 - The corresponding bit in the address is a “don’t-care”. 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).	0x00fff

Table 201: 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 202: PCI_0 CS[2:0]* Bank Size, Offset: 0xc10

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only	0x0
31:12	BankSize	Specifies the CS[2:0]* address mapping in conjunction with the CS[2:0]* Base Address register. 0 - The corresponding bit in the address and in the base address must be equal in order to have a hit. 1 - The corresponding bit in the address is a “don’t-care”. 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).	0x01fff

Table 203: 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 204: 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	Specifies the CS[3]* and Boot CS* address mapping in conjunction with the CS[3]* and Boot CS* Base Address register. 0 - The corresponding bit in the address and in the base address must be equal in order to have a hit. 1 - The corresponding bit in the address is a “don’t-care”. 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).	0x00fff

Table 205: 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 206: 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

Table 206: PCI_0 Base Address Registers' Enable, Offset: 0xc3c (Continued)

Bits	Field Name	Function	Initial Value
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
8	SCS[1:0]En	Controls address matching with SCS[1:0]* base/size. 0 - Enable 1 - Disable	0x0
31:9	Reserved		0x0

Table 207: PCI_1 Base Address Registers' Enable, Offset: 0xcbc
NOTE: Reserved if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
31:0		Same As for PCI_0 Base Address registers' enable.	0x0f

NOTE: The GT–64120A 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 208: PCI_0 Prefetch/Max Burst Size, Offset: 0xc40

Bits	Field Name	Function	Initial Value
0	Dram0MaxBrst	SCS[1:0]* Max Burst Length 0 - Maximum burst of four 64-bit words. 1 - Maximum burst of eight 64-bit words.	0x0
1	Dram1MaxBrst	SCS[3:2]* Max Burst Length 0 - Maximum burst of four 64-bit words. 1 - Maximum burst of eight 64-bit words.	0x0

Table 208: PCI_0 Prefetch/Max Burst Size, Offset: 0xc40 (Continued)

Bits	Field Name	Function	Initial Value
2	Dev0MaxBrst	CS[2:0]* Max Burst Length 0 - Maximum burst of four 64-bit words. 1 - Maximum burst of eight 64-bit words.	0x0
3	Dev1MaxBrst	CS[3]* & Boot CS* Max Burst Length 0 - Maximum burst of four 64-bit words. 1 - Maximum burst of eight 64-bit words.	0x0
4	RdMemPref	Read Memory Prefetch 0 - Regular prefetch 1 - Aggressive prefetch	0x0
5	RdLnPref	Read Line Prefetch 0 - Regular prefetch 1 - Aggressive prefetch	0x0
6	RdMulPref	Read Multiple Prefetch 0 - Regular prefetch 1 - Aggressive prefetch	0x1
31:7	Reserved		0x0

Table 209: 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 210: 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 211: 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 212: 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 DRAM0Remap	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 213: 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 214: 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 215: PCI_1 SCS[3:2]* Base Address Remap, Offset: 0xccc (
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 216: 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 217: PCI_1 Swapped SCS[3:2]* Base Address Remap, Offset: 0xcdc
NOTE: 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 218: 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 219: 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 220: 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 221: 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 222: 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 223: 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 224: 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 225: 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–64120A performs a Configuration or Special cycle on the PCI bus.	0x000

Table 226: 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 227: 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 228: 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 229: 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

20.17 Interrupts

Table 230: Interrupt Cause Register - Offset: 0xc18
NOTE: All bits are cleared by writing a value of 0 by the CPU or PCI, unless stated otherwise.

Bits	Field Name	Function	Initial Value
0	IntSum	Interrupt Summary Logical OR of all the interrupt bits, regardless of the Mask registers' values.	0x0 Read only
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.	0x0
2	DMAOut	Asserts when the DMA accesses an address out of range.	0x0
3	CPUOut	Asserts when the CPU accesses an address out of range.	0x0
4	DMA0Comp	Asserts at completion of DMA Channel 0 transfer.	0x0
5	DMA1Comp	Asserts at completion of DMA Channel 1 transfer.	0x0
6	DMA2Comp	Asserts at completion of DMA Channel 2 transfer.	0x0
7	DMA3Comp	Asserts at completion of DMA Channel 3 transfer.	0x0
8	T0Exp	Asserts when Timer 0 expires.	0x0

Table 230: Interrupt Cause Register - Offset: 0xc18 (Continued)
NOTE: All bits are cleared by writing a value of 0 by the CPU or PCI, unless stated otherwise.

Bits	Field Name	Function	Initial Value
9	T1Exp	Asserts when Timer 1 expires.	0x0
10	T2Exp	Asserts when Timer 2 expires.	0x0
11	T3Exp	Asserts when Timer 3 expires.	0x0
12	MasRdErr0	Asserts when the GT–64120A detects a parity error during a PCI_0 master read operation.	0x0
13	SlvWrErr0	Asserts when the GT–64120A detects a parity error during a PCI_0 slave write operation.	0x0
14	MasWrErr0	Asserts when the GT–64120A detects a parity error during a PCI_0 master write operation.	0x0
15	SlvRdErr0	Asserts when the GT–64120A detects a parity error during a PCI_0 slave read operation.	0x0
16	AddrErr0	Asserts when the GT–64120A detects a parity error on the PCI_0 address lines.	0x0
17	MemErr	Asserts when a memory parity error is detected.	0x0
18	MasAbort0	Asserts upon PCI_0 master abort.	0x0
19	TarAbort0	Asserts upon PCI_0 target abort.	0x0
20	RetryCtr0	Asserts when the PCI_0 retry counter expires.	0x0
21	PMCInt_0	If Power Management is enabled, this bit acts as PMC0 interrupt. Asserts when power state bits in PMCSR0 register are updated from PCI.	0x0
25:22	CPUInt	If Power Management is disabled, this bit acts as CPUInt. These bits are set by the CPU by writing 0 to generate an interrupt on the PCI bus. These bits are cleared when the PCI writes 0.	0x0
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.	0x0
30	CPUIntSum	CPU Interrupt Summary Logical OR of bits[29:26,20:1], masked by bits[29:26,20:1] of the CPU Mask register.	0x0
31	PCIIntSum	Interrupt Summary Logical OR of bits[25:1], masked by bits[25:1] of the PCI_0 Mask register.	0x0

Table 231: High Interrupt Cause Register Offset: 0xc98
NOTE: Reserved if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
11:0		Read only.	0x0
12	MasRdErr1	Asserts when the GT–64120A detects a parity error during a PCI_1 master read operation.	0x0
13	SlvWrErr1	Asserts when the GT–64120A detects a parity error during a PCI_1 slave write operation.	0x0
14	MasWrErr1	Asserts when the GT–64120A detects a parity error during a PCI_1 master write operation.	0x0
15	SlvRdErr1	Asserts when the GT–64120A detects a parity error during a PCI_1 slave read operation.	0x0
16	AddrErr1	Asserts when the GT–64120A detects a parity error on the PCI_1 address lines.	0x0
17	Reserved	Read only	0x0
18	MasAbort1	Asserts upon PCI_1 master abort.	0x0
19	TarAbort1	Asserts upon PCI_1 target abort.	0x0
20	RetryCtr1	Asserts when the PCI_1 retry counter expires.	0x0
21	PMCInt1	If Power Management is enabled, act as PMC_1 interrupt. Otherwise, this bit is reserved. Assert when power state bits in PMCSR1 register are updated from PCI.	0x0
31:22	Reserved	Read only.	0x0

Table 232: CPU Interrupt Select - Offset: 0xc70
NOTE: Reserved if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
0	Reserved	Read only.	0x0
29:1	AliasedBits	Aliased to bits [29:1] of the selected cause register	0x0
30	SelectCause	Selected Cause Register 0 - Low causer register 1 - High causer register	0x0
31	LowAndHigh	Interrupt is both Low and High Cause registers.	0x0

Table 233: PCI_0 Interrupt Select - Offset: 0xc74
NOTE: Reserved if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for CPU Interrupt Select.	0x0

Table 234: CPU Interrupt Mask Register, Offset: 0xc1c

Bits	Field Name	Function	Initial Value
0	Reserved	Read only.	0x0
1	MemOutMask	Masks MemOut interrupt to the CPU.	0x0
2	DMAOutMask	Masks DMAOut interrupt to the CPU.	0x0
3	CPUOutMask	Masks CPUOut interrupt to the CPU.	0x0
4	DMA0CompMask	Masks DMA_0 Comp interrupt to the CPU.	0x0
5	DMA1CompMask	Masks DMA_1 Comp interrupt to the CPU.	0x0
6	DMA2CompMask	Masks DMA_2 Comp interrupt to the CPU.	0x0
7	DMA3CompMask	Masks DMA_3 Comp interrupt to the CPU.	0x0
8	T0ExpMask	Masks T_0 Exp interrupt to the CPU.	0x0
9	T1ExpMask	Masks T_1 Exp interrupt to the CPU.	0x0
10	T2ExpMask	Masks T_2 Exp interrupt to the CPU.	0x0
11	T3ExpMask	Masks T_3 Exp interrupt to the CPU.	0x0
12	MasRdErr0Mask	Masks MasRdErr_0 interrupt to the CPU.	0x0
13	SlvWrErr0Mask	Masks SlvWrErr_0 interrupt to the CPU.	0x0
14	MasWrErr0Mask	Masks MasWrEr_r0 interrupt to the CPU.	0x0
15	SlvRdErr0Mask	Masks SlvRdErr_0 interrupt to the CPU.	0x0
16	AddrErr0Mask	Masks AddrErr_0 interrupt to the CPU.	0x0
17	MemErrMask	Masks MemErr interrupt to the CPU.	0x0
18	MasAbort0Mask	Masks MasAbort_0 interrupt to the CPU.	0x0
19	TarAbort0Mask	Masks TarAbort_0 interrupt to the CPU.	0x0
20	RetryCtr0Mask	Masks RetryCtr_0 interrupt to the CPU.	0x0

Table 234: CPU Interrupt Mask Register, Offset: 0xc1c (Continued)

Bits	Field Name	Function	Initial Value
21	PMC0Mask	If Power Management is enabled, masks PMC_0 interrupt to the CPU. Otherwise, reserved as read only_0.	0x0
25:22	Reserved	Read only.	0x0
29:26	PCIIntMask	Masks PCIInt interrupt to the CPU.	0x0
31:30	Reserved	Read only.	0x0

Table 235: CPU High Interrupt Cause Mask Offset: 0xc9c
NOTE: Reserved if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
12	MasRdErr1 Mask	Masks MasRdErr_1 to the CPU.	0x0
13	SlvWrErr1Mask	Masks SlvWrErr_1 to the CPU.	0x0
14	MasWrErr1 Mask	Masks MasWrErr_1 to the CPU.	0x0
15	SlvRdErr1Mask	Masks SlvRdErr_1 to the CPU.	0x0
16	AddrErr1Mask	Masks AddrErr_1 to the CPU.	0x0
17	Reserved	Reserved	0x0
18	MasAbort1Mask	Masks MasAbort_1 to the CPU.	0x0
19	TarAbort1Mask	Masks TarAbort_1 to the CPU.	0x0
20	RetryCtr1Mask	Masks RetryCtr_1 to the CPU.	0x0
21	PMC1Mask	If Power Management is enabled, masks PMC_1 interrupt to the CPU. If Power Management is disabled, this bit is reserved (read only 0).	0x0
31:22	Reserved	Read only.	0x0

Table 236: PCI_0 Interrupt Cause Mask, Offset: 0xc24

Bits	Field Name	Function	Initial Value
0	Reserved	Read only.	0x0
1	MemOutMask	Masks MemOut interrupt to PCI_0.	0x0
2	DMAOutMask	Masks DMAOut interrupt to PCI_0.	0x0

Table 236: PCI_0 Interrupt Cause Mask, Offset: 0xc24 (Continued)

Bits	Field Name	Function	Initial Value
3	CPUOutMask	Masks CPUOut interrupt to PCI_0.	0x0
4	DMA0CompMask	Masks DMA0Comp interrupt to PCI_0.	0x0
5	DMA1CompMask	Masks DMA1Comp interrupt to PCI_0.	0x0
6	DMA2CompMask	Masks DMA2Comp interrupt to PCI_0.	0x0
7	DMA3CompMask	Masks DMA3Comp interrupt to PCI_0.	0x0
8	T0ExpMask	Masks T0Exp interrupt to PCI_0.	0x0
9	T1ExpMask	Masks T1Exp interrupt to PCI_0.	0x0
10	T2ExpMask	Masks T2Exp interrupt to PCI_0.	0x0
11	T3ExpMask	Masks T3Exp interrupt to PCI_0.	0x0
12	MasRdErr0Mask	Masks MasRdErr_0 interrupt to PCI_0.	0x0
13	SlvWrErr0Mask	Masks SlvWrErr_0 interrupt to PCI_0.	0x0
14	MasWrErr0Mask	Masks MasWrErr_0 interrupt to PCI_0.	0x0
15	SlvRdErr0Mask	Masks SlvRdErr_0 interrupt to PCI_0.	0x0
16	AddrErr0Mask	Masks AddrErr_0 interrupt to PCI_0.	0x0
17	MemErrMask	Masks MemErr interrupt to PCI_0.	0x0
18	MasAbort0Mask	Masks MasAbort_0 interrupt to PCI_0.	0x0
19	TarAbort0Mask	Masks TarAbort_0 interrupt to PCI_0.	0x0
20	RetryCtr0Mask	Masks RetryCtr_0 interrupt to PCI_0.	0x0
21	PMC0Mask	If Power Management is enabled, masks PMC0 interrupt to PCI_0. If Power Management is disabled, masks CPUInt Interrupt to PCI_0.	0x0
25:22	CPUInt	Masks CPUInt interrupt to PCI_0.	0x0
31:26	Reserved	Read only.	0x0

Table 237: PCI_0 High Interrupt Cause Mask Offset: 0xca4
NOTE: Reserved if configured for only PCI_0.

Bits	Field Name	Function	Initial Value
11:0	Reserved	Read only.	0x0
12	MasRdErr1 Mask	Masks MasRdErr_1 to PCI_0.	0x0
13	SlvWrErr1Mask	Masks SlvWrErr_1 to PCI_0.	0x0
14	MasWrErr1Mask	Masks MasWrErr_1 to PCI_0.	0x0
15	SlvRdErr1Mask	Masks SlvRdErr_1 to PCI_0.	0x0
16	AddrErr1Mask	Masks AddrErr_1 to PCI_0.	0x0
17	Reserved	Reserved	0x0
18	MasAbort1Mask	Masks MasAbort_1 to PCI_0.	0x0
19	TarAbort1Mask	Masks TarAbort_1 to PCI_0.	0x0
20	RetryCtr1Mask	Masks RetryCtr_1 to PCI_0.	0x0
21	PMC1Mask	If Power Management is enabled, masks PMC_1 interrupt to PCI_0. If Power Management is disabled, this bit is reserved. Read only 0.	0x0
31:22	Reserved	Read only.	0x0

Table 238: SErr0* Mask, PCI_0 Events Offset: 0xc28

Bits	Field Name	Function	Initial Value
0	AddrErr0	Mask bit. When this bit is set and the GT–64120A detects a parity error on PCI_0 address lines, SErr0* is asserted.	0x0
1	MasWrErr0	Mask bit. When this bit is set and the GT–64120A detects a parity error during a PCI_0 master write operation, SErr0* is asserted.	0x0
2	MasRdErr0	Mask bit. When this bit is set and the GT–64120A detects a parity error during a PCI_0 master read operation, SErr0* is asserted.	0x0
3	MemErr	Mask bit. When this bit is set and a memory parity error has been detected, SErr0* is asserted.	0x0

Table 238: SErr0* Mask, PCI_0 Events Offset: 0xc28 (Continued)

Bits	Field Name	Function	Initial Value
4	MasAbor0t	Mask bit. When this bit is set and the GT–64120A performs a PCI_0 master abort, SErr0* is asserted.	0x0
5	TarAbort0	Mask bit. When this bit is set and the GT–64120A detects a PCI_0 target abort, SErr0* is asserted.	0x0
31:6	Reserved		0x0

Table 239: SErr1* Mask, PCI_1 Events Offset: 0xca8
NOTE: 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–64120A 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–64120A 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–64120A 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–64120A performs a PCI_1 master abort, SErr1* is asserted.	0x0
5	TarAbort1	Mask bit. When this bit is set and the GT–64120A detects a PCI_1 target abort, SErr1* is asserted.	0x0
31:6	Reserved		0x0

20.18 PCI Configuration

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–64120A is configured for only PCI_0 operation.

Table 240: PCI_0 Device and Vendor ID

- Offset from PCI_0 or CPU: 0x000
- Offset from PCI_1: 0x080

Bits	Field Name	Function	Initial Value
15:0	VenID	Provides the manufacturer of the GT–64120A. Read only.	0x11ab
31:16	DevID	Provides the unique GT–64120A PCI device0 ID number. Read Only.	0x4620

Table 241: PCI_1 Device and Vendor ID

- Offset from PCI_0 or CPU: 0x080
- Offset from PCI_1: 0x000

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Device and VenID	0x462011ab

Table 242: PCI_0 Status and Command

- Offset from PCI_0 or CPU: 0x004
- Offset from PCI_1: 0x084

Bits	Field Name	Function	Initial Value
0	IOEn	Controls the GT–64120A's response to I/O accesses. 0 - Disable 1 - Enable	0x0
1	MEMEn	Controls the GT–64120A's response to memory accesses. 0 - Disable 1 - Enable	0x0
2	MasEn	Controls the GT–64120A's ability to act as a master on the PCI bus. 0 - Disable 1 - Enable	0x0
3	Reserved	Read only.	0x0
4	MemWrInv	Controls the GT–64120A'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–64120A'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–64120A'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–64120A PCI interface is capable of running at 66MHz regardless of this bit value. **NOTE:** See the PCI AC timing parameters (Section 22. "AC Timing" on page 255) when designing PCI systems that run faster than 33MHz.	Sampled at RESET via the DAdr[9] pin.
22	Reserved	Read only.	0x0

Table 242: PCI_0 Status and Command (Continued)

- Offset from PCI_0 or CPU: 0x004
- Offset from PCI_1: 0x084

Bits	Field Name	Function	Initial Value
23	TarFastBB	Read only. Indicates that the GT–64120A is capable of accepting fast back-to-back transactions on the PCI bus.	0x1
24	DataParDet	This bit is set by the GT–64120A when it detects a data parity error during master operation.	0x0
26:25	DevSelTim	These pins indicate the GT–64120A'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 28–31, the user cannot set the bit. The user can only clear the bit by writing 1 to it.

Table 243: PCI_1 Status and Command

- Offset from PCI_0 or CPU: 0x084
- Offset from PCI_1: 0x004

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 244: PCI_0 Class Code and Revision ID

- Offset from PCI_0 or CPU: 0x008
- Offset from PCI_1: 0x088

Bits	Field Name	Function	Initial Value
7:0	RevID	Indicates the GT–64120A PCI_0 revision number.	0x10
15:8	Reserved	Read only.	0x0

Table 244: PCI_0 Class Code and Revision ID (Continued)

- Offset from PCI_0 or CPU: 0x008
- Offset from PCI_1: 0x088

Bits	Field Name	Function	Initial Value
23:16	SubClass	Indicates the GT–64120A subclass 0x00 - Host bridge device 0x80 - Memory device	Depends on value sampled at reset on BankSel[0].
31:24	BaseClass	Indicates the GT–64120A base class 0x06 - Bridge device 0x05 - Memory device	Depends on value sampled at reset on BankSel[0].

Table 245: PCI_1 Class Code and Revision ID

- Offset from PCI_0 or CPU: 0x088
- Offset from PCI_1: 0x008

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 class code and revision ID	

Table 246: PCI_0 BIST, Header Type, Latency Timer, Cache Line

- Offset from PCI_0 or CPU: 0x00c
- Offset from PCI_1: 0x08c

Bits	Field Name	Function	Initial Value
7:0	CacheLine	Specifies the GT–64120A's cache line size.	0x00
15:8	LatTimer	Specifies, in units of PCI bus clocks, the value of the GT–64120A'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 247: PCI_1 BIST, Header Type, Latency Timer, Cache Line

- Offset from PCI_0 or CPU: 0x08c
- Offset from PCI_1: 0x00c

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.

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–64120A incorporates three Swapped Base Address registers for SCS[1:0]*, SCS[3:2]* and CS[3]* & 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–64120A 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 248: PCI_0 SCS[1:0]* Base Address

- Offset from PCI_0 or CPU: 0x010
- Offset from PCI_1: 0x090

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 198, on page 212. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x00000

Table 249: PCI_1 SCS[1:0]* Base Address

- Offset from PCI_0 or CPU: 0x090
- Offset from PCI_1: 0x010

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 SCS[1:0]* Base Address	0x08

Table 250: PCI_0 SCS[3:2]* Base Address

- Offset from PCI_0 or CPU: 0x014
- Offset from PCI_1: 0x094

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 200, on page 213. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x01000

Table 251: PCI_1 SCS[3:2]* Base Address

- Offset from PCI_0 or CPU: 0x094
- Offset from PCI_1: 0x014

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 SCS[3:2]* Base Address.	0x01000008

Table 252: PCI_0 CS[2:0]* Base Address

- Offset from PCI_0 or CPU: 0x018
- Offset from PCI_1: 0x098

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 202, on page 213. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x1c000

Table 253: PCI_1 CS[2:0]* Base Address

- Offset from PCI_0 or CPU: 0x098
- Offset from PCI_1: 0x018

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 CS[2:0]* Base Address	ox1c000000

Table 254: PCI_0 CS[3]* and Boot CS* Base Address

- Offset from PCI_0 or CPU: 0x01c
- Offset from PCI_1: 0x09c

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 204, on page 214. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x1f000

Table 255: PCI_1 CS[3]* and Boot CS* Base Address

- Offset from PCI_0 or CPU: 0x09c
- Offset from PCI_1: 0x01c

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 CS[3]* and Boot CS* Base Address.	0x1f000000

Table 256: PCI_0 Internal Registers Memory Mapped Base Address

- Offset from PCI_0 or CPU: 0x020
- Offset from PCI_1: 0x0a0

Bits	Field Name	Function	Initial Value
0	MemSpace	Memory Space Indicator Read only.	0x0
2:1	Type	Type Read only.	0x0

Table 256: PCI_0 Internal Registers Memory Mapped Base Address (Continued)

- Offset from PCI_0 or CPU: 0x020
- Offset from PCI_1: 0x0a0

Bits	Field Name	Function	Initial Value
3	Prefetch	Prefetch Read only.	0x0
11:4	Reserved	Read only.	0x0
31:12	MemMapBase	Defines the address assignment of the GT–64120A's internal registers. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x14000

Table 257: PCI_1 Internal Registers Memory Mapped Base Address

- Offset from PCI_0 or CPU: 0x0a0
- Offset from PCI_1: 0x020

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Internal register's Memory Mapped Base Address.	0x14000000

Table 258: PCI_0 Internal Registers I/O Mapped Base Address

- Offset from PCI_0 or CPU: 0x024
- Offset from PCI_1: 0x0a4

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–64120A's internal registers. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x14000

Table 259: PCI_1 Internal Registers I/O Mapped Base Address

- Offset from PCI_0 or CPU: 0x0a4
- Offset from PCI_1: 0x024

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Internal register's I/O Mapped Base Address.	0x14000001

Table 260: PCI_0 Subsystem Device and Vendor ID

- Offset from PCI_0 or CPU: 0x02c
- Offset from PCI_1: 0x0ac

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 261: PCI_1 Subsystem Device and Vendor ID

- Offset from PCI_0 or CPU: 0x0ac
- Offset from PCI_1: 0x02c

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 subsystem device and vendor ID.	0x0

Table 262: Expansion ROM Base Address

- Offset from PCI_0 or CPU: 0x030
- Offset from PCI_1: 0x0b0

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–64120A. 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 263: PCI_0 Capability List Pointer

- Offset from PCI_0 or CPU: 0x034
- Offset from PCI_1: 0xb4

NOTE: Reserved if Power management is disabled.

Bits	Field Name	Function	Initial Value
7:0	CapPtr	Capability List Pointer. Read only.	0x40
31:8	Reserved		0x0

Table 264: PCI_1 Capability List Pointer

- Offset from PCI_0 or CPU: 0x0b4
- Offset from PCI_1: 0x34

Bits	Field Name	Function	Initial Value
7:0	CapPtr	Capability List Pointer. Read only.	0x40
31:8	Reserved		0x0

Table 265: PCI_0 Interrupt Pin and Line

- Offset from PCI_0 or CPU: 0x03c
- Offset fromPCI_1: 0x0bc

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–64120A. The GT–64120A uses INTA.	0x1
31:16	Reserved	Read only.	0x0

Table 266: PCI_1 Interrupt Pin and Line

- Offset from PCI_0 or CPU: 0x0bc
- Offset from PCI_1: 0x03c

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 interrupt pin and line.	0x00000100

Table 267: PCI_0 PMC

- Offset from PCI_0 or CPU: 0x040
- Offset from PCI_1: 0x0c0

NOTE: Reserved if Power management is disabled.

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

Table 267: PCI_0 PMC (Continued)

- Offset from PCI_0 or CPU: 0x040
- Offset from PCI_1: 0x0c0

NOTE: Reserved if Power management is disabled.

Bits	Field Name	Function	Initial Value
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

Table 268: PCI_1 PMC

- Offset from PCI_0 or CPU: 0x0c0
- Offset from PCI_1: 0x040

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 PMC register.	0x00090001

Table 269: PCI_0 PMCSR

- Offset from PCI_0 or CPU: 0x044
- Offset from PCI_1: 0x0c4

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

Table 269: PCI_0 PMCSR (Continued)

- Offset from PCI_0 or CPU: 0x044
- Offset from PCI_1: 0x0c4

Bits	Field Name	Function	Initial Value
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 270: PCI_1 PMCSR

- Offset from PCI_0 or CPU: 0x0c4
- Offset from PCI_1: 0x044

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 PMCSR register.	0x0

20.18.1Function 1 Configuration Registers

The GT–64120A 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 1must 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.

GT–64120A 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 GT–64120A as a multifunction 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 programing by CPU software or by using the GT–64120A Auto-Load option.

Table 271: Function1 PCI_0 Swapped SCS[1:0]* Base Address

- Offset from PCI_0 or CPU: 0x110
- Offset from PCI_1: 0x190

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 272: Function 1 PCI_1 Swapped SCS[1:0]* Base Address

- Offset from PCI_0 or CPU: 0x190
- Offset from PCI_1: 0x110

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Swapped SCS[1:0]* Base Address	0x08

Table 273: Function 1 PCI_0 Swapped SCS[3:2]* Base Address

- Offset from PCI_0 or CPU: 0x114
- Offset from PCI_1: 0x194

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[3:2]*. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.	0x01000

Table 274: Function 1 PCI_1 Swapped SCS[3:2]* Base Address

- Offset from PCI_0 or CPU: 0x194
- Offset from PCI_1: 0x114

Bits	Field Name	Function	Initial Value
31:0	Various	Same as for PCI_0 Swapped SCS[3:2]* Base Address.	0x01000008

Table 275: Function 1 PCI_0 Swapped CS[3]* & Boot CS* Base Address

- Offset from PCI_0 or CPU: 0x11c
- Offset from PCI_1: 0x19c

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 204, on page 214. Reserved if this BAR is disabled through PCI_0 Base Address registers' enable setting.)	0x1f000

Table 276: Function 1 PCI_1 Swapped CS[3]* & Boot CS* Base Address

- Offset from PCI_0 or CPU: 0x19c
- Offset from PCI_1: 0x11c

Bits	Field Name	Function	Initial Value
31:0	Various	As in PCI_0 Swapped CS[3]* and Boot CS* Base Address.	0x1f000000

NOTE: For more information on these fields, refer to the PCI specification.

20.19 I_2O Support Registers

If I_2O 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 I_2O 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.

Table 277: 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 278: 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 279: 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 280: 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 CPUI to PCI.	0x0

Table 281: 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 deassert the interrupt).	0x0

Table 282: 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. **NOTE:** Unlike Intel's i960RP, bits [8:6] and bit 3 are reserved since GT–64120A does not support Index, APIC, or NMI mechanisms.	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. **NOTE:** 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.	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

Table 282: Inbound Interrupt Cause Register, Offset: 0x24 (Continued)

Bits	Field Name	Function	Initial Value
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

Table 283: 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 Out-FQOvr bit in Inbound Interrupt Cause Register.	0x0
31:6	Reserved		0x0

Table 284: 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. **NOTE:** Unlike Intel's i960RP, there are no reserved bits for PCI interrupts INTA#, INTB#, INTC#, INTD#. Writing 1 to the bit by the PCI clears the bit (and deassert the interrupt).	0x0

Table 285: 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. **NOTE:** 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.	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

Table 286: 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 287: 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 288: 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 289: 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 290: 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 291: Inbound Free Head Pointer Register, Offset: 0x60

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	InFHPtr	Inbound Free Head Pointer Reserved from the PCI side. **NOTE:** This register is maintained by CPU software. It is reserved for PCI accesses.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

Table 292: Inbound Free Tail Pointer Register, Offset: 0x64

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	InFTPtr	Inbound Free Tail Pointer Reserved from the PCI side. **NOTE:** This register is incremented by GT–64120A after a PCI read from Inbound port. It is reserved for PCI accesses.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

Table 293: Inbound Post Head Pointer Register, Offset: 0x68

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	InPHPtr	Inbound Post Head Pointer Reserved from the PCI side. **NOTE:** This register is incremented by GT–64120A after a PCI write to Inbound port. It is reserved for PCI accesses.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

Table 294: Inbound Post Tail Pointer Register, Offset: 0x6c

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	InPTPtr	Inbound Post Tail Pointer Reserved from the PCI side. **NOTE:** This register is maintained by the CPU software. It is reserved for PCI accesses.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

Table 295: Outbound Free Head Pointer Register, Offset: 0x70

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	OutFHPtr	Outbound Free Head Pointer Reserved from the PCI side. **NOTE:** This register is incremented by GT–64120A after PCI write to Outbound port. It is reserved for PCI accesses.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

Table 296: Outbound Free Tail Pointer Register, Offset: 0x74

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	OutFTPtr	Outbound Free Tail Pointer Reserved from the PCI side. **NOTE:** his register is maintained by CPU software. It is reserved for PCI accesses.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

Table 297: Outbound Post Head Pointer Register, Offset: 0x78

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	OutPHPtr	Outbound Post Head Pointer Reserved from the PCI side. **NOTE:** This register is maintained by the CPU software. It is reserved for PCI accesses.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

Table 298: Outbound Post Tail Pointer Register, Offset: 0x7c

Bits	Field Name	Function	Initial Value
1:0	Reserved		0x0
19:2	OutPTPtr	Outbound Post Tail Pointer Reserved from the PCI side. **NOTE:** This register is incremented by GT–64120A after a PCI read from the Outbound port. It is reserved for PCI accesses.	0x0
31:20	QBAR	Queue Base Address Register Read only.	0x0

21. DC CHARACTERISTICS

Table 299: Absolute Maximum Ratings

Symbol	Parameter	Min.	Max.	Unit
V_{CC} 2.5V	Core Supply Voltage	-0.3	3.0	V
V_{icc} 3.3V	I/O Supply Voltage	-0.3	4.0	V
V_i	Input Voltage	-0.3	5.5	V
V_o	Output Voltage	-0.3	5.5	
I_o	Output Current		24	
I_{ik}	Input Protect Diode Current		+-20	mA
I_{ok}	Output Protect Diode Current		+-20	mA
T_c	Operating Case Temperature	0	125	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.

It is also strongly recommended that before designing a system, read *AN-63: Thermal Management for Galileo Technology Products*. This application note describes basic understanding of thermal management of integrated circuits (ICs) and guidelines to ensure optimal operating conditions for Galileo Technology's products.

Table 300: Recommended Operating Conditions

Symbol	Parameter	Min.	Typ.	Max.	Unit
V_{CC} 2.5V	Core Supply Voltage	2.375	2.5	2.625	V
V_i 3.3V	I/O Supply Voltage	3.15	3.3	3.45	V
V_i	Input Voltage	0		5.5	V
v_o	Output Voltage	0		Vcc3.3	V
T_c	Operating Case Temperature (without external heatsink) **NOTE:** T_c is for commercial grade devices.	0		70	C
T_{amb}	Operating Ambient Temperature **NOTE:** T_{amb} is for industrial grade devices.	-40		85	C

Table 301: Ball Capacitance

Symbol	Parameter	Min.	Typ.	Max.	Unit
C_{in}	Input Capacitance		7.2		pF
C_{out}	Output Capacitance		7.2		pF

21.1 DC Electrical Characteristics Over Operating Range

Table 302: DC Electrical Characteristics Over Operating Range

Symbol	Parameter	Test Condition	Min.	Max.	Unit	Loading
V_{ih}	Input HIGH level	Guaranteed Logic HIGH level	2.0	5.5	V	
V_{il}	Input LOW level	Guaranteed Logic LOW level	-0.3	0.8	V	
V_{oh}	Output HIGH Voltage: DWr*, SDQM[7:0], SCAS*, SCS[3:0]*, SRAS*, BankSel[0], DAdr[10:3]/Wr[7:0]*, DAdr[2:0]/BAdr[2:0], MGNT*/ BypsOE*, DMAReq[3:1]*, DMAReq[0]*/MREQ*	IoH = 16 mA	2.4		V	50 pF
V_{oh}	Output HIGH Voltage: ScDOE*, ALE, ADP[7:6], ADP[5]/DAdr[11], ADP[4]/ BankSel[1], ADP[3:0]/ EOT[3:0]	IoH = 12 mA	2.4		V	50 pF
V_{oh}	Output HIGH Voltage: AD[63:42], AD[41]/DevRW*, AD[40]/BootCS*, AD[39:36]/ CS[3:0]*, AD[35:32]/ DMAAck[3:0]*, AD[31:0], SysAD[63:0], SysADC[7:0], SysCmd[8:0]	IoH = 8 mA	2.4		V	50 pF
V_{oh}	Output HIGH Voltage: ValidIn*	IoH = 8 mA	2.4		V	40 pF
V_{oh}	Output HIGH Voltage: WrRdy*, ScWord[1:0], CSTiming*	IoH = 8 mA	2.4		V	30 pF
V_{oh}	Output HIGH Voltage: Interrupt*	IoH = 8 mA	2.4		V	20 pF

Table 302: DC Electrical Characteristics Over Operating Range (Continued)

Symbol	Parameter	Test Condition	Min.	Max.	Unit	Loading
V_{ol}	Output LOW Voltage: DWr*, SDQM[7:0], SCAS*, SCS[3:0]*, SRAS*, BankSel[0], DAdr[10:3]/Wr[7:0]*, DAdr[2:0]/BAdr[2:0], MGNT*/BypsOE*, DMAReq[3:1]*, DMAReq[0]*/MREQ*	IoL = 16 mA		0.4	V	50 pF
V_{ol}	Output LOW Voltage: ScDOE*, ALE, ADP[7:6], ADP[5]/DAdr[11], ADP[4]/BankSel[1], ADP[3:0]/EOT[3:0]	IoL = 12 mA		0.4	V	50 pF
V_{ol}	Output LOW Voltage: AD[63:42], AD[41]/DevRW*, AD[40]/BootCS*, AD[39:36]/CS[3:0]*, AD[35:32]/DMAAck[3:0]*, AD[31:0], SysAD[63:0], SysADC[7:0], SysCmd[8:0]	IoL = 8 mA		0.4	V	50 pF
V_{ol}	Output LOW Voltage: ValidIn*	IoL = 8 mA		0.4	V	40 pF
V_{ol}	Output LOW Voltage: WrRdy*, ScWord[1:0], CSTiming*	IoL = 8 mA		0.4	V	30 pF
V_{ol}	Output LOW Voltage: Interrupt*	IoL = 8 mA		0.4	V	20 pF
I_{ih}	Input HIGH Current			10	uA	
I_{il}	Input LOW Current			10	uA	
I_{ozh}	High Impedance Output Current			10	uA	
I_{ozl}	High Impedance Output Current			10	uA	
V_h	Input Hysteresis		300	350	mV	

Table 302: DC Electrical Characteristics Over Operating Range (Continued)

Symbol	Parameter	Test Condition	Min.	Max.	Unit	Loading
I_{cc}	Operating Current	I/O VCC3.3=3.465V f = 100MHz TClk/66Mhz PClk		190	mA	
		Core VCC2.5=2.625V f = 100MHz TClk/66Mhz PClk		330	mA	
	PAD0[31:0], PAD1[31:0], Frame0*, Frame1*/Req64*, Irdy0*, Irdy1*, Trdy0*, Trdy1*, DevSel0*, DevSel1*/Ack64*, Stop0*, Stop1*, Perr0*, Perr1*, Par0, Par1/Par64, CBE0[3:0]*, CBE1[3:0]*, Gnt0*, Gnt1*, Req0*, Req1*, IdSel0, IdSel1, Lock0*, Serr0*, Serr1*, Int0*	See PCI Specification Rev. 2.1				

21.2 Thermal Data

Table 303 shows the package thermal data for the GT–64120A.

Galileo Technology recommends the use of heatsink for most systems, especially those with little or no airflow. Using a heatsink with the commercial grade device is especially important.

Use an adequate airflow, layout, and other means to meet the recommended operating conditions listed in Table 303.

NOTE: For further information, see *AN-63: Thermal Management for Galileo Technology Products*.

Table 303: Thermal Data for The GT–64120A in BGA 388

Airflow	Definition	Value		
		0 m/s	1 m/s	2 m/s
Θja	Thermal resistance: junction to ambient.	16.3 c/w	13.2 c/w	11.8 c/w
Ψjt	Thermal characterization parameter: junction to case center.	0.28 c/w	0.35 c/w	0.43 c/w
Θjc	Thermal resistance: junction to case (not air-flow dependent)	5.3 C/W		

22. AC TIMING

Table 304: AC Commercial Grade Timing

All Delays, Setup, and Hold times are referred to TClk RISING edge, unless stated otherwise.

		83 Mhz		100 Mhz			
Signals	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Unit**	**Loading**
Clk							
TClk	Pulse Width High	4.8		4		ns	
TClk	Pulse Width Low	4.8		4		ns	
TClk	Clock Period	12	20	10	20	ns	
TClk	Rise Time		TBD		TBD	V/ns	
TClk	Fall Time		TBD		TBD	V/ns	
Rst*	Active	1		1		ms	
CPU Interface							
SysAD[63:0], SysCMD[8:0], ValidOut*, Release*, ScTCE*	Setup	3.5		2.5		ns	
Hit	Setup (MCM 69T618 -5 -6 -7)	3.5		2.5			
SysAD[63:0], SysCMD[8:0], ValidOut*, Release*, ScTCE*, Hit	Hold	1		1		ns	
SysAD[63:0], SysCMD[8:0], SysADC[7:0], ScDOE*, ValidIn*, WrRdy*, ScWord[1:0], Interrupt*	Output Delay	2	7	2	5	ns	50pF

Table 304: 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.		
PCI Interface							
Pclk0, Pclk1	Clock Period	15		15		ns	
All Inputs	Setup	4		3.5		ns	
All Inputs	Hold	0		0		ns	
All Outputs, except Req*	Output Delay	2	7	2	6.5	ns	
Req*	Output Delay	2	8	2	7.5	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[11:0]	Setup	3		3		ns	
DAdr[11:0]	Hold	1		1		ns	
DAdr[11:0]	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 Falling (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	
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, TClk Falling (ALE)	2	7	2	6	ns	30pF
ADP[3]/DWr*	Output Delay (DWr*)	2	7	2	6	ns	50pF

Table 304: 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)							
ADP[5:4]/BA[0:1]	Output Delay (BA[0:1])	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	4		3		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]*/BA[1]*	Setup (DMAReq[1]*)	4.5		3.5		ns	
DMAReq[1]*/BA[1]*	Hold (DMAReq[1]*)	1		1		ns	
DMAReq[1]*/BA[1]*	Output Delay (BA[1]*)	2	7	2	5	ns	50pF
DMAReq[2]*/BA[0]*	Setup (DMAReq[2]*)	4.5		3.5		ns	
DMAReq[2]*/BA[0]*	Hold (DMAReq[2]*)	1		1		ns	
DMAReq[2]*/BA[0]*	Output Delay (BA[0]*)	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	2	6	ns	50pF
MGnt*/ByPsOE*	Output Delay (MGnt*)	2	7	2	6	ns	30pF

Table 304: 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)							
MGnt*/ByPsOE*	Output Delay (Bypass OE*)	2	8.5	2	7.5	ns	30pF
SRAS*, SCAS*, DWr*	Output Delay	2	7	2	5	ns	50pF
ALE	Output Delay, TClk FALLING (ALE)	2	7	2	5.5	ns	30pF
CSTiming*	Output Delay	2	7	2	5.5	ns	30pF

Table 305: AC Industrial Grade Timing

All Delays, Setup, and Hold times are referred to TClk RISING edge, unless stated otherwise.

Signals	Description	83 Mhz		Unit	Loading
		Min.	Max.		
Clk					
TClk	Pulse Width High	4.8		ns	
TClk	Pulse Width Low	4.8		ns	
TClk	Clock Period	12	20	ns	
TClk	Rise Time		TBD	V/ns	
TClk	Fall Time		TBD	V/ns	
Rst*	Active	1		ms	
CPU Interface					
SysAD[63:0], SysCMD[8:0], ValidOut*, Release*, ScTCE*	Setup	3.5		ns	
Hit	Setup (MCM 69T618 -5 -6 -7)	3.5			
SysAD[63:0], SysCMD[8:0], ValidOut*, Release*, ScTCE*, Hit	Hold	1		ns	
SysAD[63:0], SysCMD[8:0], SysADC[7:0], ScDOE*, ValidIn*, WrRdy*, ScWord[1:0], Interrupt*	Output Delay	2	6	ns	50pF

Table 305: AC Industrial Grade Timing (Continued)

All Delays, Setup, and Hold times are referred to TClk RISING edge, unless stated otherwise.

Signals	Description	83 Mhz		Unit	Loading
		Min.	Max.		
PCI Interface					
Pclk0, Pclk1	Clock Period	15		ns	
All Inputs	Setup	4		ns	
All Inputs	Hold	0		ns	
All Outputs, except Req*	Output Delay	2	8	ns	
Req*	Output Delay	2	8.5	ns	
Memory Interface					
AD[63:0]	Setup	3		ns	
AD[63:0]	Hold	1		ns	
AD[63:0], SCS[3:0]*	Output Delay	2	6.5	ns	50pF
DAdr[11:0]	Setup	2.5		ns	
DAdr[11:0]	Hold	1		ns	
DAdr[11:0]	Output Delay	2	7	ns	50pF
DAdr[2:0]	Output Delay (Device Burst)	2	7	ns	50pF
DAdr[10:3]	Output Delay from TClk Falling (Device Write)	2	7	ns	50pF
ADP[7:0]	Output Delay (Parity)	2	7	ns	50pF
ADP[7:0]	Setup (Parity)	3		ns	50pF
ADP[7:0]	Hold (Parity)	1		ns	
ADP[3:0]/EOT[3:0]*	Setup (EOT[3:0]*)	4		ns	
ADP[3:0]/EOT[3:0]*	Hold (EOT[3:0]*)	1		ns	
ADP[1]/ALE	Output Delay (ALE)	2	7	ns	30pF
ADP[1]/ALE	Output Delay, TClk Falling (ALE)	2	7	ns	30pF
ADP[3]/DWr*	Output Delay (DWr*)	2	7	ns	50pF

Table 305: AC Industrial Grade Timing (Continued)

All Delays, Setup, and Hold times are referred to TClk RISING edge, unless stated otherwise.

Signals	Description	83 Mhz		Unit	Loading
		Min.	Max.		
Memory Interface (Continued)					
ADP[7:6]/SRAS*/SCAS*	Output Delay (SRAS* and SCAS*)	2	7	ns	50pF
SDQM[7:0]	Output Delay	2	7		
Ready*	Setup	4		ns	
Ready*	Hold	1		ns	
DMAReq[0]*/MREQ*/SRAS*	Setup (DMAReq[0]/ MREQ*)	4.5		ns	
DMAReq[0]*/MREQ*/SRAS*	Hold (DMAReq[0]/ MREQ*)	1		ns	
DMAReq[0]*/MREQ*/SRAS*	Output Delay (SRAS*)	2	7	ns	50pF
DMAReq[1]*/BA[1]*	Setup (DMAReq[1]*)	4.5		ns	
DMAReq[1]*/BA[1]*	Hold (DMAReq[1]*)	1		ns	
DMAReq[1]*/BA[1]*	Output Delay (BA[1]*)	2	7	ns	50pF
DMAReq[2]*/BA[0]*	Setup (DMAReq[2]*)	4.5		ns	
DMAReq[2]*/BA[0]*	Hold (DMAReq[2]*)	1		ns	
DMAReq[2]*/BA[0]*	Output Delay (BA[0]*)	1.3	7	ns	50pF
DMAReq[3]*/SCAS*/EOT[0]*	Setup (DMAReq[3]*/ EOT[0]*)	4.5		ns	
DMAReq[3]*/SCAS*/EOT[0]*	Hold (DMAReq[3]*/ EOT[0]*)	1		ns	
DMAReq[3]*/SCAS*/EOT[0]*	Output Delay (SCAS*)	2	7	ns	50pF

Table 305: AC Industrial Grade Timing (Continued)

All Delays, Setup, and Hold times are referred to TClk RISING edge, unless stated otherwise.

Signals	Description	83 Mhz		Unit	Loading
		Min.	Max.		
Memory Interface (Continued)					
MGnt*/ByPsOE*	Output Delay (MGnt*)	2	8.5	ns	30pF
MGnt*/ByPsOE*	Output Delay (Bypass OE*)	2	8.5	ns	30pF
SRAS*, SCAS*, DWr*	Output Delay	2	7	ns	50pF
ALE	Output Delay, TClk FALLING (ALE)	2	7	ns	30pF
CSTiming*	Output Delay	2	7	ns	30pF

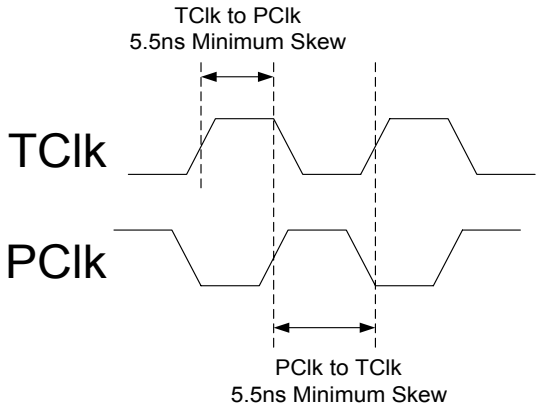
22.1 TClk/PClk Restrictions

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 39. 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 39: TClk = PClk Skew Requirement



In addition to the above restriction, there are few sync modes specific restrictions, summarized in Table 306.

Table 306: TClk/PClk Restrictions

Sync Mode	PClk Frequency Range	Restrictions
0,4	from DC up to TClk	T**pclk** > T**tclk** + 1.5ns
1	from TClk/2 up to TClk	T**pclk** < 2T **tclk** -1ns, unless running with **synchronized** Tpclk = 2Ttclk and minimum skew of 5.5ns between PClk rise and TClk rise is guaranteed. For example, if T **tclk** = 15ns (66MHz), T**pclk** should be smaller than 29ns (unless running with synchronized clocks).
2,3	from TClk/2 up to TClk	**TClk and PClk are synchronized**, and a maximum skew of 2ns between PClk rise and TClk rise is guaranteed.
5	from TClk/3 up to TClk/2	T**pclk** < 3T **tclk** - 1ns, unless running with **synchronized** Tpclk = 3T tclk and minimum skew of 5.5ns between PClk rise and TClk rise is guaranteed. For example, if T **tclk** = 13.3ns (75MHz), T**pclk** should be smaller than 39ns (unless running with synchronized clocks).
6,7	from TClk/3 up to TClk/2	**TClk and PClk are synchronized** , and a maximum skew of 2ns between PClk rise and TClk rise is guaranteed.

The sync mode can be programed by the CPU, the PCI or during autoload. For sync mode information, see Section 20.16 “PCI Internal” on page 210.

22.2 Additional Delay due to Capactive Loading

Some applications may require additional capacitive loading on different output pins of the GT–64120A. For example, when using multiple GT–64120A 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.

22.2.1 Calculating the Maximum Delay due to Loading

The basic equation for calculating the maximum delay is:

```
Tmax = [Atypa + (Btyp * Cr)] * 1.6
```

where:

- Tmax is the maximum delay in nanoseconds.
- Atypa must be calculated by the designer as shown in Section 22.2.1.1 Calculating Atypa on page 263.
- Btyp is a parameter according to the specific output buffer from Table 307.
- Cr is the capacitance required.

22.2.1.1Calculating Atypa

Atyp can be calculated by using the given values in the AC Timing Parameters table. We start with the equation:

```
Tspec = [Atypa + (Btyp * Cds)] * 1.6
```

and solving for Atypa:

```
Atypa = (Tspec/1.6) - (Btyp * Cds)
```

where:

- Tspec is the maximum delay parameter from the AC Timing Parameters Table
- Btyp is a parameter according to the specific output buffer from Table 307.
- C is the capacitance parameter from the AC Timing Parameters Table.

NOTE: 1.6 is the worst case derating factor.

22.2.2 Calculating the Minimum Delay due to Loading

The basic equation for calculating the maximum delay is:

```
Tmin = [Atypb + (Btyp * Cr)] * 0.7
```

where:

- Tmin is the maximum delay in nanoseconds
- Atypb must be calculated by the designer as shown in 22.2.2.1
- Btyp is a parameter according to the specific output buffer from Table 307.
- Cr is the capacitance required

22.2.2.1Calculating Atypb

Atyp can be calculated by using the given values in the AC Timing Parameters table. We start with the equation:

```
Tspec = [Atypb + (Btyp * Cds)] * 0.7
```

and solving for Atypb:

```
Atypb = (Tspec/0.7) - (Btyp * Cds)
```

where:

- Tspec is the maximum delay parameter from the AC Timing Parameters Table 304.
- Btyp is a parameter according to the specific output buffer from Table 307.
- Cds is the capacitance parameter from the AC Timing Parameters Table 304.

NOTE: 0.7 is the worst case derating factor.

22.2.3 Btyp Values

Table 307 lists the Btyp values for the different output buffers of the GT–64120A. See the DC Parameters Section for the corresponding pin and output driver.

Table 307: 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

22.2.4 Example Calculation for Tmax

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 8ns. Looking at Table 307, Btyp for 8mA drivers is = 0.031 (Low to High transition).

Substituting these values into:

```
Atypa = (Tspec/1.6) - (Btyp * Cds)
```

gives Atypa = 3.45.

Substituting Atypa of 3.45, Btyp of 0.031 and Cr of 75pF in:

```
Tmax = [Atypa + (Btyp * Cr)] * 1.6
```

gives Tmax = 9.24. This means that the maximum output delay of AD[0] with a 75pF load is 9.24ns.

23. PINOUT TABLE, 388 PIN BGA

NOTE: The following table is sorted by ball number.

Table 308: GT–64120A Pinout Table

Ball#	Signal Name	Ball#	Signal Name	Ball#	Signal Name
A01–A26		***B01–B26***		***C01–C26***	
A01	GND	B01	SysAD[38]	C01	SysAD[40]
A02	GND	B02	GND	C02	SysAD[39]
A03	SysAD[37]	B03	SysAD[35]	C03	GND
A04	SysAD[33]	B04	Release*	C04	SysAD[36]
A05	ValidIn*	B05	ValidOut*	C05	SysAD[34]
A06	SysAD[30]	B06	SysAD[28]	C06	WrRdy*
A07	SysAD[26]	B07	SysAD[24]	C07	SysAD[31]
A08	SysAD[22]	B08	SysAD[21]	C08	SysAD[27]
A09	SysAD[20]	B09	SysAD[18]	C09	SysAD[23]
A10	SysAD[16]	B10	SysAD[14]	C10	SysAD[19]
A11	SysAD[12]	B11	SysAD[10]	C11	SysAD[15]
A12	SysAD[9]	B12	SysAD[7]	C12	SysAD[11]
A13	SysAD[5]	B13	SysAD[3]	C13	SysAD[8]
A14	SysAD[1]	B14	SysADC[7]	C14	SysAD[6]
A15	SysADC[5]	B15	SysADC[4]	C15	SysAD[2]
A16	SysADC[3]	B16	SysADC[1]	C16	SysADC[6]
A17	SysCmd[8]	B17	SysCmd[6]	C17	SysADC[2]
A18	SysCmd[5]	B18	SysCmd[4]	C18	SysCmd[7]
A19	SysCmd[2]	B19	SysCmd[0]	C19	SysCmd[3]
A20	ScTCE*	B20	Hit	C20	Interrupt*
A21	ScWord[0]	B21	Req1*	C21	ScWord[1]
A22	Gnt1*	B22	PAD1[1]	C22	PAD1[3]
A23	PAD1[0]	B23	PAD1[6]	C23	PAD1[7]
A24	PAD1[5]	B24	PAD1[4]	C24	GND
A25	CBE1[0]*	B25	GND	C25	PAD1[9]
A26	GND	B26	GND	C26	PAD1[11]

Table 308: GT–64120A Pinout Table (Continued)

Ball#	Signal Name	Ball#	Signal Name	Ball#	Signal Name
D01–D26		***E01–E04, E23–E26***		***H01–H04, H23–H26***	
D01	SysAD[43]	E01	SysAD[47]	H01	SysAD[57]
D02	SysAD[41]	E02	SysAD[44]	H02	SysAD[55]
D03	SysAD[42]	E03	SysAD[46]	H03	SysAD[58]
D04	GND	E04	SysAD[45]	H04	VCC2.5
D05	SysAD[32]	E23	PAD1[14]	H23	Frame1*/Req64*
D06	VCC3.3	E24	PAD1[8]	H24	Trdy1*
D07	SysAD[29]	E25	Par1/Par64	H25	PAD1[17]
D08	SysAD[25]	E26	CBE1[1]*	H26	PAD1[18]
D09	GND	***F01–F04, F23–F26***		***J01–J04, J23–J26***	
D10	SysAD[17]	F01	SysAD[51]	J01	SysAD[60]
D11	VCC2.5	F02	SysAD[48]	J02	SysAD[59]
D12	SysAD[13]	F03	SysAD[50]	J03	SysAD[62]
D13	SysAD[4]	F04	VCC3.3	J04	SysAD[56]
D14	VCC2.5	F23	VCC3.3	J23	VCC2.5
D15	SysAD[0]	F24	PAD1[12]	J24	PAD1[19]
D16	VCC3.3	F25	DevSel1*/Ack64*	J25	PAD1[22]
D17	SysADC[0]	F26	Perr1*	J26	PAD1[16]
D18	SysCmd[1]	***G01–G04, G23–G26***		***K01–K04, K23–K26***	
D19	GND	G01	SysAD[53]	K01	SysAD[63]
D20	ScDOE*	G02	SysAD[52]	K02	SysAD[61]
D21	VCC3.3	G03	SysAD[54]	K03	JTDO
D22	PAD1[2]	G04	SysAD[49]	K04	JTDI
D23	GND	G23	Stop1*	K23	PAD1[21]
D24	PAD1[10]	G24	Serr1*	K24	PAD1[23]
D25	PAD1[13]	G25	CBE1[2]*	K25	IdSel1
D26	PAD1[15]	G26	Irdy1*	K26	PAD1[20]

Table 308: GT–64120A Pinout Table (Continued)

Ball#	Signal Name	Ball#	Signal Name	Ball#	Signal Name
L01–L04, L11–L26		***N01–N04, N11–N26***		***R01–R04, R11–R26***	
L01	JTCLK	N01	AD[58]	R01	AD[51]
L02	JTMS	N02	AD[60]	R02	AD[53]
L03	AD[61]	N03	AD[54]	R03	AD[49]
L04	VCC3.3	N04	AVCC2.5PLL	R04	AD[47]
L11	GND	N11	GND	R11	GND
L12	GND	N12	GND	R12	GND
L13	GND	N13	GND	R13	GND
L14	GND	N14	GND	R14	GND
L15	GND	N15	GND	R15	GND
L16	GND	N16	GND	R16	GND
L23	VCC3.3	N23	PAD1[28]	R23	PAD0[29]
L24	VREF1	N24	PAD1[31]	R24	PAD0[28]
L25	PAD1[25]	N25	Rst*	R25	PAD0[25]
L26	PAD1[27]	N26	PClk1	R26	PAD0[24]
M01–M04, M11–M26		***P01–P04, P11–P26***		***T01–T04, T11–T26***	
M01	AD[62]	P01	TClk	T01	AD[48]
M02	AD[63]	P02	AD[56]	T02	AD[50]
M03	AD[57]	P03	AD[52]	T03	AD[45]
M04	AD[59]	P04	AD[55]	T04	VCC3.3
M11	GND	P11	GND	T11	GND
M12	GND	P12	GND	T12	GND
M13	GND	P13	GND	T13	GND
M14	GND	P14	GND	T14	GND
M15	GND	P15	GND	T15	GND
M16	GND	P16	GND	T16	GND
M23	CBE1[3]*	P23	GND	T23	VCC3.3
M24	PAD1[26]	P24	PAD1[29]	T24	PAD0[31]
M25	PAD1[30]	P25	PAD0[30]	T25	CBE0[3]*
M26	PAD1[24]	P26	PClk0	T26	PAD0[26]

Table 308: GT–64120A Pinout Table (Continued)

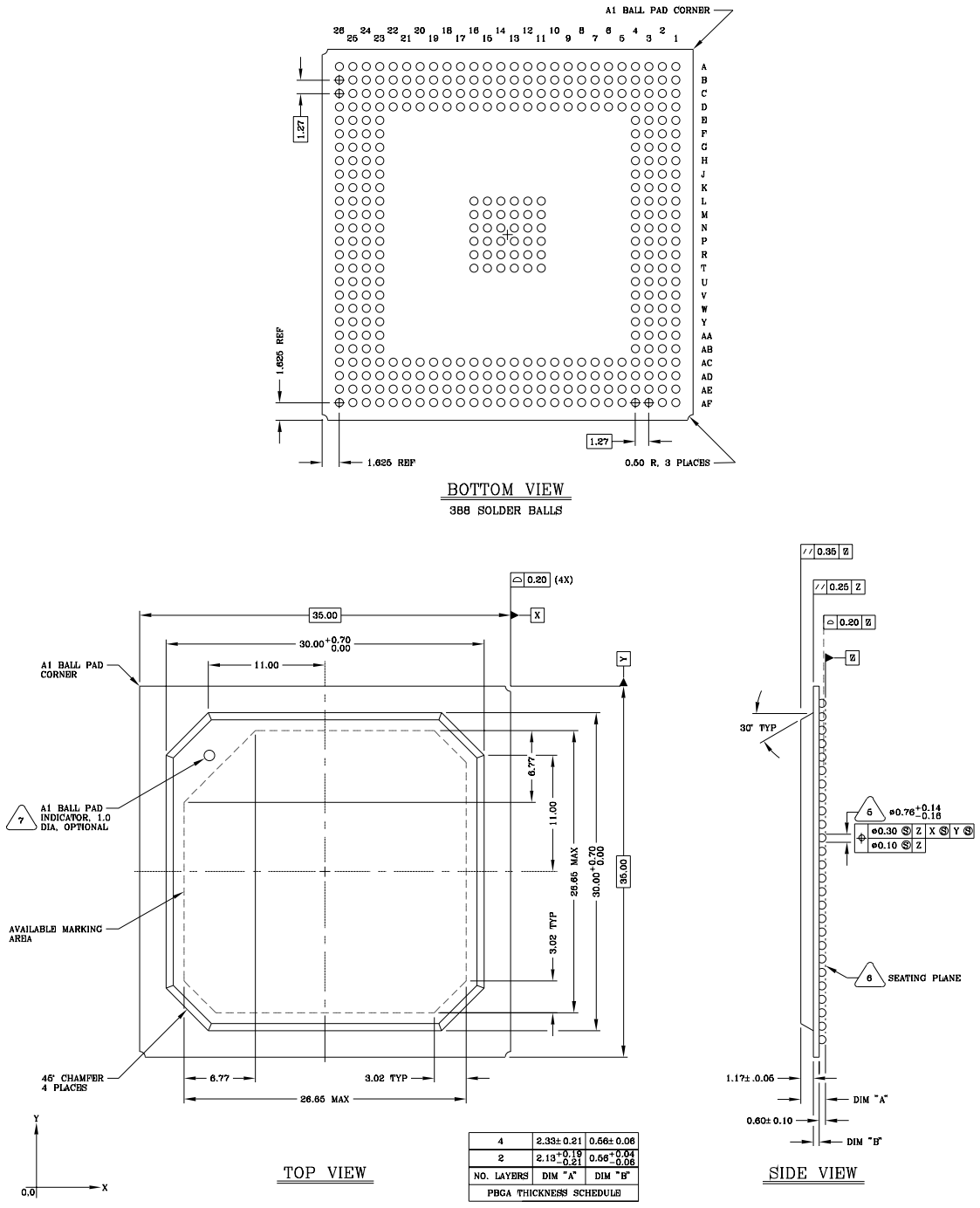
Ball#	Signal Name	Ball#	Signal Name	Ball#	Signal Name
U01–U04, U23–U26		***Y01–Y04, Y23–Y26***		***AC01–AC26***	
U01	AD[44]	Y01	AD[34]/DMAAck*[2]	AC01	AD[31]
U02	AD[46]	Y02	AD[36]/CS*[0]	AC02	ADP[1]/EOT[1]
U03	AD[41]/DevRW*	Y03	ADP[5]/DAdr[11]	AC03	AD[28]
U04	AD[43]	Y04	ADP[7]	AC04	GND
U23	IdSel0	Y23	Stop0*	AC05	AD[19]
U24	PAD0[27]	Y24	CBE0[2]*	AC06	VCC3.3
U25	PAD0[21]	Y25	Irdy0*	AC07	AD[15]
U26	VREF0	Y26	Frame0*	AC08	ByPsPLL
				NOTE: Must be pulled down.	
V01–V04, V23–V26		***AA01–AA04, AA23–AA26***		AC09	AD[8]
V01	AD[40]/BootCS*	AA01	ADP[6]	AC10	AD[0]
V02	AD[42]	AA02	AD[32]/DMAAck*[0]	AC11	VCC3.3
V03	AD[37]/CS*[1]	AA03	ADP[2]/EOT[2]	AC12	SDQM*[1]
V04	AGNDPLL	AA04	VCC2.5	AC13	VCC2.5
V23	PAD0[18]	AA23	VCC3.3	AC14	SCS*[1]
V24	PAD0[20]	AA24	DevSel0*	AC15	Ready*
V25	PAD0[23]	AA25	Lock0*	AC16	VCC3.3
V26	PAD0[22]	AA26	Trdy0*	AC17	DAdr[10]/Wr*[7]
W01–W04, W23–W26		***AB01–AB04, AB23–AB26***		AC18	VCC2.5
W01	AD[38]/CS*[2]	AB01	ADP[3]/EOT[3]	AC19	DAdr[2]/BAdr[2]
W02	AD[39]/CS*[3]	AB02	ADP[4]/BankSel[1]	AC20	Req0*
W03	AD[33]/DMAAck*[1]	AB03	AD[30]	AC21	VCC3.3
W04	AD[35]/DMAAck*[3]	AB04	ADP[0]/EOT[0]	AC22	PAD0[6]
W23	VCC2.5	AB23	Par0	AC23	GND
W24	PAD0[16]	AB24	Serr0*	AC24	PAD0[13]
W25	PAD0[19]	AB25	CBE0[1]*	AC25	PAD0[14]
W26	PAD0[17]	AB26	Perr0*	AC26	PAD0[12]

Table 308: GT–64120A Pinout Table (Continued)

Ball#	Signal Name	Ball#	Signal Name	Ball#	Signal Name
AD01–AD26		***AE01–AE26***		***AF01–AF26***	
AD01	AD[27]	AE01	GND	AF01	GND
AD02	AD[29]	AE02	GND	AF02	AD[26]
AD03	GND	AE03	AD[25]	AF03	AD[24]
AD04	AD[22]	AE04	AD[23]	AF04	AD[21]
AD05	AD[18]	AE05	AD[20]	AF05	AD[17]
AD06	AD[14]	AE06	AD[16]	AF06	AD[13]
AD07	AD[10]	AE07	AD[12]	AF07	AD[11]
AD08	AD[6]	AE08	AD[9]	AF08	AD[7]
AD09	AD[2]	AE09	AD[5]	AF09	AD[4]
AD10	SDQM*[7]	AE10	AD[3]	AF10	AD[1]
AD11	SDQM*[3]	AE11	DWr*	AF11	SDQM*[6]
AD12	SCAS*	AE12	SDQM*[5]	AF12	SDQM*[4]
AD13	SCS*[3]	AE13	SDQM*[2]	AF13	SDQM*[0]
AD14	BypsOE*/MGNT*	AE14	SCS*[0]	AF14	SCS*[2]
AD15	ALE	AE15	SRAS*	AF15	DMAReq[0]*/MREQ*
AD16	DMAReq[2]*	AE16	DMAReq[3]*	AF16	CSTiming*
AD17	DAdr[8]/Wr[5]*	AE17	DMAReq[1]*	AF17	BankSel[0]
AD18	DAdr[4]/Wr[1]*	AE18	DAdr[9]/Wr[6]*	AF18	DAdr[7]/Wr[4]*
AD19	DAdr[0]/BAdr[0]	AE19	DAdr[6]/Wr[3]*	AF19	DAdr[5]/Wr[2]*
AD20	PAD0[3]	AE20	DAdr[3]/Wr[0]*	AF20	DAdr[1]/BAdr[1]
AD21	PAD0[0]	AE21	Int0*	AF21	Gnt0*
AD22	PAD0[4]	AE22	PAD0[2]	AF22	PAD0[1]
AD23	PAD0[11]	AE23	PAD0[7]	AF23	PAD0[5]
AD24	GND	AE24	CBE0[0]*	AF24	PAD0[10]
AD25	PAD0[8]	AE25	GND	AF25	GND
AD26	PAD0[15]	AE26	PAD0[9]	AF26	GND

24. 388 BGA PACKAGE MECHANICAL INFORMATION

Figure 40: 388 BGA Package Mechanical Information



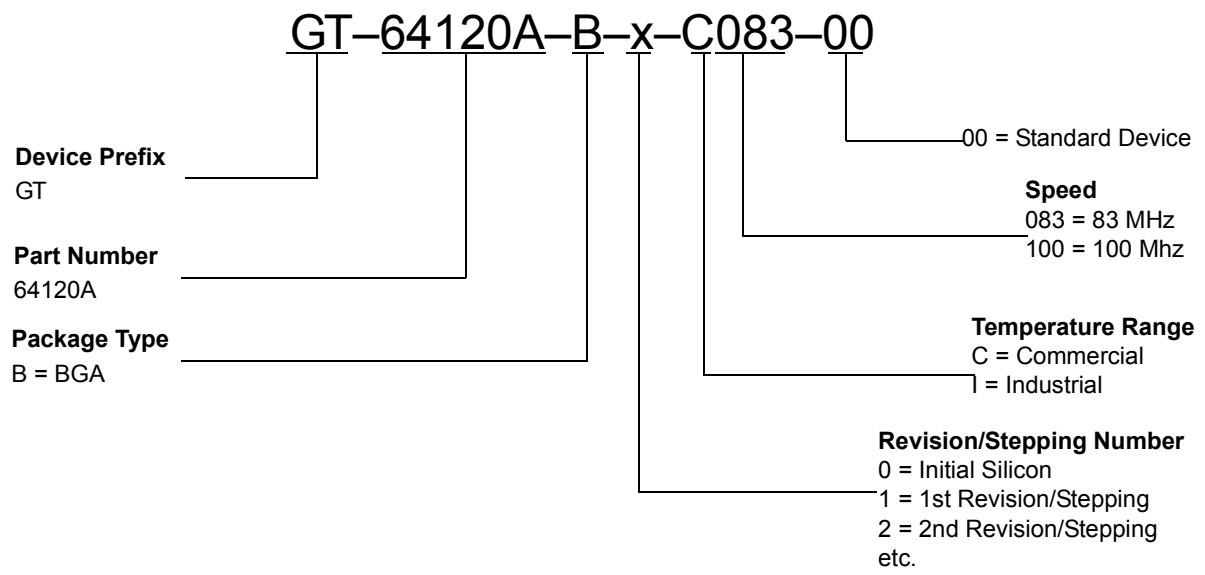
25. FUNCTIONAL WAVEFORMS

Functional waveforms are not included in this datasheet.

Transcribing waveforms from simulator outputs to “real world” documentation is extremely error prone and we do not currently have a tool that can do it effectively. Rather than risk introducing errors, Galileo Technology provides a separate document that includes several dozen functional waveforms. This waveform summary is available at http://www.galileot.com/library/syslib.htm.

26. GT–64120A PART NUMBERING

Figure 41: Sample Part Number



26.1 Standard Part Number

The standard part number for the GT–64120A is: GT–64120A–B–x.

Without the –XYYY–ZZ suffix, this part number indicates that it is the commercial temperature grade, 100MHz version. In other words, GT–64120A-B-x is the same as GT–64120A–B–x–C100–00, although it is not marked with the suffix information.

26.2 Valid Part Numbers

The following part numbers are the only valid part numbers that can be used when ordering the GT–64120A:

- GT–64120A–B–x, Commercial Temperature, 100MHz
- GT–64120A–B–x–C083–00, Commercial Temperature, 83MHz
- GT–64120A–B–x–I083–00, Industrial Temperature, 83MHz

27. Revision History

Table 309: Document History

Document Type	Rev. Number	Date
Product Preview	0.2	12/9/98
First release.		
Product Preview	0.3	JULY 25, 1999

1. New Data Integrity section starting on page 90.
2. End of Transfer Input pins, EOT[3:0], are active HIGH. This change is made throughout the data sheet.
3. Changed DMAReq[2]*/DAddr[11] and DMAReq[1]*/BankSel[1] from input only to I/O in Section 2.1 “Pin Assignment Tables” on page 18.
4. Revised explanation of JTMS pulled HIGH when JTAG is disabled (previously, it was stated to be pulled LOW) in Section 2.1 “Pin Assignment Tables” on page 18.
5. New description for desabling device decoders in Section 3.2 “Disabling the Device Decoders” on page 39.
6. New note in Section 3.3 “DMA Unit Address Decoding” on page 39:

NOTE: DMA address decoding is only up to address bit [31]. Bits [35:32] of CPU address decoding registers are ignored.

7. New note explaining that DMA source and destination addresses are never remapped in Section 3.6 “Address Remapping” on page 43.
8. Changed encoding parameters for command mnemonic Wr2Words to 001011111 in Table 21, on page 50.
9. New CPU restriction explaining that the GT–64120A does not support access of more than 4 bytes to internal space, see Section 4.8 “CPU Interface Restrictions” on page 58.
10. Corrected decode sequence for SysAD/PCI Address Decoding for 64-bit SDRAM, 64/128 Mbit, column 001. The new sequence is:
    6, 7, 27, “0”, 26-25, 12-8, 5-3.
    See Table 32, on page 67.
11. Updated section Section 5.6.8 “Internal Register Reads with UMA Enabled” on page 79 to explain how Internal Register reads work with UMA enabled.
12. Updated memory interface restrictions in Section 5.9 “Memory Controller Restrictions” on page 87.
13. Changed BE to Wr* in 32-bit device limitation examples in Table 40, on page 88.
14. Moved sections on Error Checking and Correction (ECC) calculations, error reports, and restrictions to new Section 6. “Data Integrity” on page 90.
15. New description of how autoload PLD must support burst read cycles in Section 7.6.2 “PCI Autoload of Configuration Registers at RESET” on page 110.
16. New PCI Interface Restrictions in Section 7.13.2 “Slave Interface Restrictions” on page 114.
17. Added DMA restrictions in Section 9.9 “DMA Restrictions” on page 138.
18. New note in Section 12. “Reset Configuration” on page 143 stating that:

NOTE: Rst* must be de-asserted for at least 10 PClk cycles before any CPU transactions are generated.

19. Revised reset configuration information for DAdr[4:3]. If sampled on reset, DAdr[4:3] determines CS[3]* as well as BootCS* initial boot bus width. See Table 56, on page 143.
20. Revised note in Section 9.2.6 “ChainMod, bit 9” on page 127.
21. Added section about using the JTAG application to support the GT–64120A’s test mode operation, see Section 14. “JTAG Application Notes” on page 151.
22. Hit and ScTCE* pins added to Table 65.
23. Revised pin usage information for using the GT–64120A with PCI_0 as a 32-bit PCI interface, see Table 67.

Table 309: Document History (Continued)

Document Type	Rev. Number	Date
Product Preview (Continued)	0.3	JULY 25, 1999
24. New note after Table 69, on page 160: **NOTE:** The combined capacitance of PClk0 and PClk1 exceeds the 10pF maximum capacitance limitation of the PCI Specification. 25. New Phase Locked Loop (PLL) application notes in Section 18. “PLL Power Filter Circuit” on page 163. 26. New description for bit [15] of the CPU Interface Configuration in Table 87. 27. New CPU error report register tables in Section 20.5 “CPU Errors Report” on page 188. 28. Updated register table for register SDRAM Bank 3 Parameters, Table 153, on page 197. 29. New information about maximum burst and prefetch settings for PCI_0, see Table 208. 30. New note for register PCI_0 Status and Command that explains how to clear bits 24 and 28 to 31 by writing 1 to these bits, see Table 242, on page 230. 31. New minimum and maximum specifications for the part’s VCC2.5 recommended operating conditions. See Table 300. • Minimum: 2.375v • Maximum: 2.625v 32. In Table 299, the absolute maximum operating case temperature is now 100c. New BGA thermal data in Table 302, on page 252. 33. Updated AC timing specifications, see Section 22. “AC Timing” on page 255. 34. Revised Tclk/Pclk restrictions in Section 22.1 “TClk/PClk Restrictions” on page 261. 35. New part numbering details in Section 26. “GT–64120A Part Numbering” on page 273.		
Datasheet	1.0	FEB 22, 2000
36. Revision to Table 21, “Address Phase SysCmd[8:0] Encodings (driven by CPU),” on page 50. The command descriptions for Rd4Words and Wr4Words is not supported. 37. New note added to Section 5. “Memory Controller” on page 59 explaining that when the datasheet refers to 64-bit SDRAM, it means 64-bits of data plus eight additional bits for ECC. 38. New restriction added to Section 5.9 Memory Controller Restrictions on page 88. This restriction has also been added as a note in Section 7.3.3 “PCI Target Read Prefetching” on page 105. 39. Corrected the explanation of the setting for 0 (Big-endian mode) in Section 15.1.1 “Bit 12 of the CPU Interface Configuration register” on page 153. When set to 0, there is byte swapping of data transfers to/from the GT–64120A internal registers (including Configuration Data register, 0xcfc). 40. Added note to Section 9.1.4 “Pointer to The Next Record Register” on page 126 stating that the next record pointer must be 16 bytes aligned. This means bits [3:0] must be set to 0. This not also is included in Table 176 to Table 179. 41. Added note to Table 67, “PCI_0 as 32-bit PCI Only,” on page 157 stating that the GT–64120A 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. 42. Revised PLL power supply information in Section 18.1 “PLL Power Supply” on page 163. 43. Corrected initial value for the bits in Table 148, “SDRAM Burst Mode, Offset: 0x478,” on page 195. The new initial value is 0x1. 44. Corrections to Table 151 to Table 153 so they correspond correctly to Table 150, “SDRAM Bank0 Parameters, Offset: 0x44c,” on page 196.		

Table 309: Document History (Continued)

Document Type	Rev. Number	Date
Datasheet (Continued	1.0	FEB 22, 2000
45. Added note to Table 194, "PCI_0 Command, Offset: 0xc00," on page 210 explaining that bits [12:10] are not supported in a 64-bit PCI configuration. 46. Revised AC timing parameters in Table 304, "AC Commercial Grade Timing," on page 255. 47. Revised PClk/TClk restriction in Section 22.1 "TClk/PClk Restrictions" on page 261.		
Datasheet	1.01	SEP 15, 2000
1. Added recommended operating ambient temperature for industrial grade part in Table 300, on page 251. 2. Added operating ambient temperature parameter for the industrial grade part in Table 305, on page 258.		
Datasheet	1.1	JAN 10, 2001
1. Added notes about the effect of I_2O enabled to Table 15, "CPU and Device Decoder Default Address Mapping," on page 40 andTable 16, "PCI Function 0 and Device Decoder Default Address Mapping," on page 41. 2. New PLL section on page 163. 1. New input hysteresis figures in Table 302, "DC Electrical Characteristics Over Operating Range," on page 252. • Min: 300 mV • Max: 350 mV 2. Revisions to the AC timing sections for the commercial and industrial grade parts. See Section 22. "AC Timing" on page 255. 3. Corrected the initial values in the following registers: • Table 249, "PCI_1 SCS[1:0]* Base Address," on page 233. • Table 251, "PCI_1 SCS[3:2]* Base Address," on page 234. • Table 268, "PCI_1 PMC," on page 239. • Table 272, "Function 1 PCI_1 Swapped SCS[1:0]* Base Address," on page 241. • Table 274, "Function 1 PCI_1 Swapped SCS[3:2]* Base Address," on page 242. 4. Revised thermal data in Table 303, "Thermal Data for The GT–64120A in BGA 388," on page 254.		