FUNCTIONAL ERRATA AND RESTRICTIONS

GT–64240A-B-0

This document outlines all of the known errata and restrictions that exist in the GT–64240A-B-0.

NOTE: With Rev. B of this document, a new errata and restrictions numbering system was introduced. This new system organizes the errata and restrictions by subject area. To determine the number for errata that pre-dated Rev. B, see the Errata Numbering Correlation table on page 3.

Stepping Summary and Identification

Stepping	Marking	Errata Fixed	Errata NOT Fixed
0 (first silicon)	GT–64240A-B-0	Baseline, original silicon	Errata FEr #COMM-1, FEr #COMM-2, FEr #COMM-3, FEr #COMM-4, FEr #COMM-5, FEr #COMM-6, FEr #COMM-7, FEr #COMM-8, FEr #COMM-9, FEr #CPU-1, FEr #DMA-1, FEr #MEM-1, FEr #MEM-2, FEr #MEM-3, FEr #MISC-1, FEr #MISC-2, FEr #MISC-3, FEr #MISC-4, FEr #MISC-5, FEr #PCI-1, FEr #PCI-2, FEr #PCI-3, FEr #PCI-4, FEr #PCI-5, FEr #PCI-6FEr #PCI-7, FEr #PCI-8 Restrictions Res #COMM-1, Res #COMM-2, Res #CPU-1, Res #CPU-2, Res #DMA-1, Res #DMA-2, Res #MEM-1, Res #MEM-2, Res #MEM-3, Res #MEM-4, Res #MISC-1, Res #MISC-2, Res #MISC-3, Res #PCI-1,, Res #PCI-2

Errata Revision History

Rev #	Date	Device Covered	Errata Described
0.1	October 4, 2001	GT–64240A-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
A	October 15, 2001	GT–64240A-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
B	December 19, 2001	GT–64240A-B-0	FEr #COMM-1, FEr #COMM-2, FEr #COMM-3, FEr #COMM-4, FEr #COMM-5, FEr #COMM-6, FEr #CPU-1, FEr #DMA-1, FEr #MEM-1, FEr #MEM-2, FEr #MISC-1, FEr #MISC-2, FEr #MISC-3, FEr #PCI-1, FEr #PCI-2, FEr #PCI-3, FEr #PCI-4, FEr #PCI-5, FEr #PCI-6
C	February 28, 2002	GT–64240A-B-0	FEr #COMM-1, FEr #COMM-2, FEr #COMM-3, FEr #COMM-4, FEr #COMM-5, FEr #COMM-6FEr #COMM-7, FEr #COMM-8, FEr #COMM-9, FEr #CPU-1, FEr #DMA-1, FEr #MEM-1, FEr #MEM-2,FEr #MEM-3, FEr #MISC-1, FEr #MISC-2, FEr #MISC-3, FEr #MISC-4, FEr #MISC-5, FEr #PCI-1, FEr #PCI-2, FEr #PCI-3, FEr #PCI-4, FEr #PCI-5, FEr #PCI-6, FEr #PCI-7, FEr #PCI-8

Restriction Revision History

Rev #	Date	Device Covered	Restriction Described
0.1	October 4, 2001	GT–64240A-B-0	None.
A	October 15, 2001	GT–64240A-B-0	None.
B	December 19, 2001	GT–64240A-B-0	Res #COMM-1, Res #COMM-2, Res #DMA-1, Res #DMA-2, Res #MEM-1, Res #MEM-2, Res #MEM-3, Res #MEM-4, Res #MISC-1, Res #MISC-2, Res #MISC-3, Res #PCI-1,
C	February 28, 2002	GT–64240A-B-0	Res #COMM-1, Res #COMM-2, , Res #CPU-1,, Res #CPU-2, Res #DMA-1, Res #DMA-2, Res #MEM-1, Res #MEM-2, Res #MEM-3, Res #MEM-4, Res #MISC-1, Res #MISC-2, Res #MISC-3, Res #PCI-1, Res #PCI-2

Document Change History

Rev #	Date	Document Changes
A	October 15, 2001	First revision.
B	December 19, 2001	• FEr#1: *I2C serial ROM in Big Endian mode.* changed to a restriction, Res #MISC-1 No further changes to this item. • Deleted FEr #COMM-1: Paddin*g multiple descriptors in a single Ethernet packet*. Previously numbered FEr #2. Does not apply to GT–64240A. • FEr #COMM-1: *Abort in Ethernet ports while the Tx FIFO is full.* Includes new workaround. • FEr#4: *I_2O queue ports are inaccessible from the PCI through internal space*. Changed to Res #MISC-2. No further changes to this item. • Deleted FEr#5: *Aggressive prefetch is not supported one or two data phases before burst size alignment*. Does not apply to GT–64240A. • FEr#7: *Abort on a CPU owned descriptor.* changed to a restriction, Res #COMM-1 No further changes to this item. • Deleted FEr#8: *A PCI write to the Vital Product Data (VPD) region drives the wrong parity on the device bus.* Does not apply to GT–64240A. • FEr#10: *Write to the most significant byte of the MPP interface registers*. Changed to Res #MISC-3. No further changes to this item. • Deleted FEr#11: *Ethernet TX priority arbiter transmitting High priority queue packets.* Does not apply to GT–64240A. • In FEr #MISC-1 change in the "Fix" status. This erratum will be fixed in the next revision of the device. • Change in the "Fix" status of FEr #PCI-2: *Incorrect swapping during PCI-to-PCI memory transactions.*, previously FEr #6. This erratum will be fixed in the next revision of the device. • Change in the "Fix" status of FEr #PCI-3: *Wrong REQ64 during PCI-to-PCI memory transactions.*, previously FEr #13. This erratum will be fixed in the next revision of the device.
C	February 28, 2002	Added the following errata: • FEr #COMM-7 • FEr #COMM-8 • FEr #COMM-9 • FEr #MEM-3 • FEr #MISC-4 • FEr #MISC-5 • FEr #PCI-7 • FEr #PCI-8 Added the following restrictions: • Res #CPU-1 • Res #CPU-2 • Res #PCI-2

Errata Numbering Correlation

Old Number	New Number	Page	Description
Errata			
FEr #3	FEr #COMM-1	page 5	Abort in Ethernet ports while the Tx FIFO is full.
FEr #6	FEr #PCI-1	page 17	A write from the PCI to a nonexistent internal space register is not supported.
FEr #9	FEr #PCI-2	page 17	Incorrect swapping during PCI-to-PCI memory transactions.
FEr #12	FEr #COMM-2	page 5	Late collision in the Ethernet ports.
FEr #13	FEr #PCI-3	page 18	Wrong REQ64 during PCI-to-PCI memory transactions.
FEr #14	FEr #MISC-1	page 13	Incorrect data is read from some internal registers.
FEr #15	FEr #CPU-1	page 11	Disabling and enabling PErrProp bit [22] in the CPU Configuration register.
FEr #16	FEr #PCI-4	page 18	PCI arbiter pins are not PCI compliant.
FEr #17	FEr #PCI-5	page 18	Erroneous behavior on simultaneous PCI configuration accesses from CPU and PCI.
FEr #18	FEr #COMM-3	page 5	Ethernet port doesn’t stop at a NULL descriptor.
FEr #19	FEr #DMA-1	page 11	IDMA descriptor fetch parity error.
FEr #120	FEr #MEM-1	page 12	Redundant access to a 32-bit device.
FEr #21	FEr #PCI-6	page 19	PCI master may get stuck when disabled during operation.
FEr #22	FEr #COMM-4	page 6	Incorrect data is read from some SDMA internal registers.
FEr #23	FEr #COMM-5	page 6	Padding of two consecutive short packets.
FEr #24	FEr #MISC-2	page 14	Loss of GPP interrupts.
FEr #25	FEr #MISC-3	page 15	GPP Level Interrupts in Level Sensitive Mode.
FEr #26	FEr #COMM-6	page 6	MPSC Clock source limitation from BRG.
FEr #27	FEr #MEM-2	page 12	SDRAM Refresh-Active time (Trc) violation.
Restrictions			
RES #1	Res #DMA-1	page 23	IDMA Burst Limit less than 8 bytes is not supported.
RES #2	Res #DMA-2	page 23	IDMA addressing restrictions.
RES #3	Res #MEM-1	page 23	Burst access limitations to a 32-bit device.
RES #4	Res #MEM-2	page 24	Device controller does not support non consecutive byte enables.
RES #5	Res #MEM-3	page 24	SDRAM timing parameters must have the same value.
RES #6	Res #PCI-1	page 25	PCI master operation mode during DMA transfer.
RES #7	Res #MISC-1	page 25	I^{2}C serial ROM data must be written in Little Endian byte order.
RES #8	Res #MISC-2	page 25	I $_2$O queue ports are inaccessible from the PCI through internal space.
RES #9	Res #COMM-1	page 22	Abort on a CPU owned descriptor.
RES #0	Res #MISC-3	page 25	Write to the most significant byte of the MPP interface registers.

Old Number	New Number	Page	Description
RES #11	Res #COMM-2	page 22	Internal Loopback in Ethernet ports.

Errata List

FEr #COMM-1 Abort in Ethernet ports while the Tx FIFO is full.

TYPE: Errata

Description
When aborting the Ethernet Tx DMA while the Tx FIFO is full, the first packet of the new chain following the abort is not sent.

Workaround
After aborting the transmitter (Tx) operation and before sending any real packets, send a dummy packet.

Fix
There are no plans to fix this erratum.

FEr #COMM-2 Late collision in the Ethernet ports.

TYPE: Errata

Description
Collisions occurring after 120 bytes are erroneously considered as regular collisions.

Workaround
None.

Fix
There are no plans to fix this erratum.

FEr #COMM-3 Ethernet port does not stop at a NULL descriptor.

TYPE: Errata

Description
The ethernet port does not stop when it gets a NULL descriptor. Instead, the port tries to fetch the next descriptor from the NULL address, which causes failure. In the GT–64240A datasheet Rev. A, see Section 13.3, Operational Description for more details.

Workaround

At the last descriptor, confirm that the ownership bit is set to ‘0’, indicating that the descriptor is owned by the CPU. This forces the Ethernet controller to stop at the last descriptor, since it’s owned by the CPU.

Fix
There are no plans to fix this erratum.

FEr #COMM-4 Incorrect data is read from some SDMA internal registers.

TYPE: Errata

Description
When reading the following SDMA registers' Channel0 First Tx Descriptor Pointer (SFTDP0), offset 0x4c14, and Channel1 First Tx Descriptor Pointer (SFTDP1), offset 0x6c14, the returned data is erroneous.

NOTE: A write to these registers will perform correctly.

Workaround
None.

Fix
There are no plans to fix this erratum.

FEr #COMM-5 Padding of two consecutive short packets.

TYPE: Errata

Description
With two, consecutive packets, in which the second packet requires padding, no padding takes place when:

- A short frame (<64 Bytes) spans more than one Tx descriptor. In this case, the packet is not sent at all
- The first descriptor is marked as a descriptor with no padding.
- The last descriptor is marked as a descriptor with padding.

Workaround
Perform one of the following:

- For the case of a short frame (<64 Bytes) that spans more than one Tx descriptor, send a dummy packet after the multi-descriptor single packet.
- All the frame descriptors must be set to the same padding value (bit 18 of the Command/Status word in the Tx descriptor must have the same value in all descriptors – 0 - no padding, 1 - padding).
- When working with short packets, don't use multiple descriptors.

Fix
There are no plans to fix this erratum.

FEr #COMM-6 MPSC Clock source limitation from BRG.

TYPE: Errata

Description
When SClkx and TSClkx are used as source clocks to BRGx, the MPSCx interface cannot use the BRGx as Rx and/or Tx source clocks.

Workaround
When using BRGx as a clock source to MPSCx, use BClkIn (from the MPP interface) or TClk clock sources to the BRGx.

Fix
There are no plans to fix this erratum.

FEr #COMM-7 Incorrect byte swapping with MPSC in transparent mode.

TYPE: Errata

Description
When using the Multi Protocol Serial Controller (MPSC) in Transparent operation mode, it may occur that the first two bytes of a Tx frame are swapped (i.e., the second byte is transmitted first, followed by the first byte).

Workaround
When using the MPSC in Transparent mode, and setting it to a direct serial interface, ensure that CTS is de-asserted for at least one cycle during the time RTS is de-asserted. The workaround can be implemented by adding external logic to control CTS with the following equations:

```
BEGIN
count[].(clk, clrn) = (TxClock, GLOBAL(_reset) & GT-64240_rts);
IF (count[]<2) THEN
count[] = count[]+1;
ELSE
count[] = count[];
END IF;
IF (count[]>1) THEN
GT-64240_cts = external_cts;
ELSE
GT-64240_cts = GT-64240_rts | external_cts;
END IF;
END;
```

Fix
There are no plans to fix this erratum.

FEr #COMM-8 WDNMI* and WDE* incorrectly documented as open drain signals.

TYPE: Errata

Description
In the GT–64240A datasheet, the WDNMI* (Non Maskable Interrupt) and WDE* (Watch Dog Expired) active low signals, which are outputs from the MPP pins, are documented as open drain signals.

In reality, these signals are not open drain. The intention in having them open drain was to allow multiple devices to drive these signals to the CPU without the need for external logic, such as ANDing multiple resources.

Workaround
Add an ‘AND’ gate on the board and connect all the WDNMI* signals from all the devices to it. Its output should be connected to the CPU (as the last WDNMI* signal).

This also applies to the WDE* signals.

Fix
There are no plans to fix this erratum.

FEr #COMM-9 Address decoding error in Communication Unit.

TYPE: Errata

Description

When using the Multi Protocol Serial Controller (MPSC) or the Ethernet ports, the data transfers are via chained lists of descriptors, which are placed in a defined target interface (SDRAM, Device, or PCI).

When the MPSC or Ethernet port fetches a descriptor (either a transmit or receive descriptor), the address is first compared with the CPU Interface Address Decoding registers to select the target interface (SDRAM, Device, PCI bus).

NOTE: This address decoding process is similar to the CPU address decoding process. (See section 4.1 CPU Address Decoding in the datasheet).

If the address does not match any of the address windows, an Address Decoding Error Interrupt is generated. In addition, the MPSC/Ethernet port must halt.

The errata is that the MPSC or Ethernet port might get stuck and never halt.

Workaround

For the MPSC

Abort the transmit and/or the receive operation.

When aborting this operation, remember that the MPSC port includes two engines:

- The SDMA engine is responsible for the descriptors (fetching and closing).
- The Port engine is responsible for the data transmission and reception.

The abort sequence must abort both engines.

The interrupt handler of the driver must handle all the interrupts defined in the Communication unit Interrupt Cause register (offset 0xf310) bits [18:16] and [26:24] (Address Decoding Error Interrupts) in the following way:

1. Abort to the SDMA:
   a) For MPSC0: In Register SDCM0, offset 0x4008, set bit [15] for abort receive and bit [31] for abort transmit. Later, the CPU must poll bit [15] and bit [31] until they are reset to ‘0’. This indicates the completion of the abort sequence.
   b) For MPSC1: In Register SDCM1, offset 0x6008, set bit [15] for abort receive and bit [31] for abort transmit. Later, the CPU must poll bit [15] and bit [31] until they are reset to ‘0’. This indicates the completion of the abort sequence.
2. Abort to the Port:
   a) For MPSC0: In Channel Registers CHR2, offset 0x8010, set bit [7] for abort transmission and bit [23] for abort reception. Later, the CPU must poll bit [7] and bit [23] until they are reset to ‘0’. This indicates the completion of the abort sequence.
   b) For MPSC1: In Channel Registers CHR2, offset 0x9010, set bit [7] for abort transmission and bit [23] for abort reception. Later, the CPU must poll bit [7] and bit [23] until they are reset to ‘0’. This indicates the completion of the abort sequence.
3. Repeat step 1 one more time.

To return to normal operation:

1. Again, configure the descriptor pointers of the transmit and receive chains:
   MPSC0 offset (0x4810, 0x4C10, 0x4C14)
   MPSC1 offset (0x6810, 0x6C10, 0x6C14)
2. Again, enable the SDMA:
   a) For MPSC0: In register SDCM0, at offset 0x4008, set bit [7] for enable DMA receive and bit [23] for transmit demand.
   b) For MPSC1: In register SDCM1, at offset 0x6008, set bit [7] for enable DMA receive and bit [23] for transmit demand.
3. Move the MPSC to EH mode:
   a) For MPSC0: In Channel registers CHR2, offset 0x8010, set bit [31] for Enter Hunt state.
   b) For MPSC1: In Channel registers CHR2, offset 0x9010, set bit [31] for Enter Hunt state.

For the Ethernet

Abort the transmit and/or the receive operation.

The interrupt handler of the driver must handle all the interrupts defined in the Communication Unit Interrupt Cause register (offset 0xf310 - bits [2:0] for Ethernet0, bits [6:4] for Ethernet1, and bits [10:8] for Ethernet2) in the following way:

- Abort the DMA by writing to the Ethernet SDMA Command register (ExSDCMR) the value of '0x80008000', abort transmit- bit 31 (AT), abort receive- bit 15 (AR).
  For Ethernet0 - E0SDCMR - offset 0x2448
  For Ethernet1 - E1SDCMR - offset 0x2848
  For Ethernet2 - E2SDCMR - offset 0x2c48

NOTE: The Ethernet Ports include four queues for receive and two queues for transmit. When the abort is completed, the needed queues should be programmed again.

To return to normal operation after the abort:

1. Program the First Receive Descriptor Pointers, Current Receive Descriptor Pointers, and Current Transmit Descriptor Pointers.

- First Receive Descriptor Pointers:

Ethernet Port	Queue	Offset
Ethernet0	Queue 0, E0FRDP0	0x2480
	Queue 1, E0FRDP1	0x2484
	Queue 2, E0FRDP2	0x2488
	Queue 3, E0FRDP3	0x248C
Ethernet1	Queue 0, E1FRDP0	0x2880
	Queue 1, E1FRDP1	0x2884
	Queue 2, E1FRDP2	0x2888
	Queue 3, E1FRDP3	0x288C

Ethernet Port	Queue	Offset
Ethernet2	Queue 0, E2FRDP0	0x2C80
	Queue 1, E2FRDP1	0x2C84
	Queue 2, E2FRDP2	0x2C88
	Queue 3, E2FRDP3	0x2C8C

- Current Receive Descriptor Pointers:

Ethernet Port	Queue	Offset
Ethernet0	Queue 0, E0CRDP0	0x24A0
	Queue 1, E0CRDP1	0x24A4
	Queue 2, E0CRDP2	0x24A8
	Queue 3, E0CRDP3	0x24AC
Ethernet1	Queue 0, E1CRDP0	0x28A0
	Queue 1, E1CRDP1	0x28A4
	Queue 2, E1CRDP2	0x28A8
	Queue 3, E1CRDP3	0x28AC
Ethernet2	Queue 0, E2CRDP0	0x2C80
	Queue 1, E2CRDP1	0x2C84
	Queue 2, E2CRDP2	0x2C88
	Queue 3, E2CRDP3	0x2C8C

- Current Transmit Descriptor Pointers:

Ethernet Port	Queue	Offset
Ethernet0	Queue 0, E0CTDP0	0x24E0
	Queue 1, E0CTDP1	0x24E4
Ethernet1	Queue 0, E1CTDP0	0x28E0
	Queue 1, E1CTDP1	0x28E4
Ethernet2	Queue 0, E2CTDP0	0x2CE0
	Queue 1, E2CTDP1	0x2CE4

2. Next, enable the DMA by setting the Ethernet SDMA Command register's (ExSDCMR) ERD bits [7]. Then, enable transmit by setting set the TxDH bit [23] (for high priority Queue) or TxDL bit [24] (for low priority Queue).

Ethernet Port	Queue	Offset
Ethernet0	Queue 0, E0SDCMR	0x2448
Ethernet1	Queue 0, E1SDCMR	0x2848
Ethernet2	Queue 0, E2SDCMR	0x2C48

Fix
There are no current plans to fix this erratum.

FEr #CPU-1 Disabling and enabling PErrProp bit [22] in the CPU Configuration register.

Type: Errata

Description
When programming the CPU Configuration register's PErrProp bit [22], it should cause:

- The parity bits corruption, in case of a propagating error.
- SysCmd[5] in the same conditions, in case of a propagating error.

Yet, SysCmd[5] always reflects the propagating data status (error or good) regardless of the PErrProp setting.

Workaround
If no errors are to be reported:

1. Disable SysADCValid bit [19].
2. Disconnect SysCmd[5] from the GT–64240A and pull it down.

NOTE: If parity indication is required, only disconnect SysCmd[5] from the GT–64240A and pull it down.

Fix
This erratum will be fixed in the next revision of the device.

FEr #DMA-1 IDMA descriptor fetch parity error.

TYPE: Errata

Description
When operating in chain mode and close descriptor is enabled, and if parity error is detected during IDMA descriptor fetch, an interrupt is generated and the DMA engine halts. However, the descriptor is not closed and the ownership is never returned to CPU.

Workaround
When a descriptor fetch parity error interrupt occurs, the interrupt handler must access the descriptor in memory and mark it as owned by the CPU.

Fix
There are no plans to fix this erratum.

FEr #MEM-1 Redundant access to a 32-bit device.

TYPE: Errata

Description
When accessing a 32-bit device, the GT–64240A's Device Controller always performs a burst comprised of an even number of accesses, always starting at a 64-bit aligned address. As an example:

A single 32-bit read from an address at offset 0x4 appears on the device bus as a burst of two accesses. The first access is a redundant read from the address at offset 0x0. The second access is a read from the original address with an offset of 0x4.

A single 32-bit write to an address at offset 0x4 appears on the device bus as a burst of two write accesses. The first access is a redundant write to the address at offset 0x0. The second access is a write to the original address with an offset of 0x4.

For accesses to offset 0x0, A single 32-bit read from the address appears on the device bus as a burst of two accesses.The first access is a read from the intended address at offset 0x0. The second access is a redundant read from the address with an offset of 0x4.

A single 32-bit write to an address at offset 0x0 appears on the device bus as a burst of two write accesses. The first access is a write to the intended address at offset 0x0. The second access is a redundant write to the address with an offset of 0x4.

NOTE: Redundant write accesses have all the Wr* signals de-asserted.

Redundant accesses do not occur during burst transactions.

Workaround
If redundant read accesses must be avoided, use external logic.

AD[2] can be used by the external logic to identify redundant read accesses. Bit [2] of the transaction address is driven on AD[2] during the device access.

Fix
There are no plans to fix this erratum.

FEr #MEM-2 SDRAM Refresh-Active time (Trc) violation description.

TYPE: Errata

Description
The SDRAM devices define Trc time as the time from refresh cycle to activate.

On a staggered refresh, the GT–64240A counts nine (9) Tclk cycles from the first refreshed SCS* (SCS[0]*). For SDRAM devices that require more than six (6) Tclk cycles for Trc, this delay can cause a violation of Trc.

Workaround
- When there is a risk of Trc violation, use non-staggered refresh mode (set the SDRAM Configuration registers StagRef bit [16] to '1') and use the SDRAM device with a Trc setting of maximum nine (9) Tclk cycles.
- Use the SDRAM device with Trc less than six (6) Tclk cycles.

Fix
This erratum will be fixed in the next revision of the device.

FEr #MEM-3 Read after Write to write through a snoop region.

TYPE: Errata

Description

This problem may occur with a system in which a write through a snoop region is defined from one of the PCI interfaces.

In this scenario, a write to a snoop region immediately followed by a read to the same region might replace their order. This is a problem in cases in which two accesses are to the same address and the accesses are from one of the PCI buses.

NOTE: An access from the other direction to such region works correctly.

Workaround

There are two solutions in cases a read must follow immediately after a write:

- If in the system a write back region is also defined, insert a dummy read from a writeback region before the read from the write address.
- If no writeback region is defined, repeat the write five times and then issue the read from that address. If additional writes are needed to this region, replace the repeating writes with the needed write accesses.

Fix

This erratum will be fixed in the next revision of the device.

FEr #MISC-1 Incorrect data is read from some internal registers.

TYPE: Errata

Description

The GT–64240A returns wrong data on reads from:

- All MPSC channels
- All BRG registers
- All WatchDog registers
- SDMA interrupt registers.

NOTE: Write operations to the above registers are performed as specified in the datasheet. The following table specifies all the registers included in the errata:

Register Group	Register offsets
MPSCs Signals Routing Registers	0xb400 – 0xb408
MPSCs Interrupts Registers	0xb804, 0xb80c, 0xb884, 0xb88c
MPSC0 Registers	0x8000 – 0x8034
MPSC1 Registers	0x9000 – 0x9034
SDMA Interrupts Register	0xb800, 0xb880
BRG Registers	0xb200 - 0xb20c, 0xb834, 0xb8b4
Watchdog Registers	0xb410, 0xb414

Workaround

- BRG: The BGR operation does not require any reads from the BRG registers.
- WatchDog: The WatchDog's normal operation does not require any reads from the WatchDog registers. When working in polling mode, the user can recognize the WDNMI*or WDE assertion by connecting it to one of the GPP pins.
- MPSC: In the MPSC initialization do not wait for the port to enter the hunt mode. Instead, add a delay of 100 system cycles.

As a result of this erratum the user can only map single interrupts from each unit (MPSC and SDMA) to the Main Interrupt Cause register. Following are several examples of how to implement basic operations over the MPSC channels, regardless of the existing constrains.

- UART implementation when working in polling mode.
  The software can poll the SDMA Descriptors in the memory (Command/Status word Owner Bit) to check if the descriptor has been closed by the GT–64240A SDMA.
- UART implementation when working in interrupt mode.
  Rx and Tx interrupts must be used. The Rx interrupt signals the CPU that a character has been received and the Tx interrupt signals that the port is idle. To implement the Rx interrupt, use the SDMA Cause register's SdmaxRxBuf bit as the interrupt source for the SDMA bit in the Main Interrupt Cause (High) register. This causes the interrupt to be asserted on each received character. To implement the Tx interrupt, use the MPSC Cause and Mask register's MpscxTEIDL bit [12] as the interrupt source for the MPSC bit in the Main Interrupt Cause (High) register. This causes the interrupt to be asserted whenever the port is idle.

NOTE: Since the MPSC registers can not be read, when the Tx interrupt occurs and while there is no data to transmit, the Tx ISR must signal in a global parameter that the port is idle.

- HDLC and Transparent implementation.
  The HDLC and Transparent implementations are the same as the UART implementation when working in interrupt mode.

NOTE: If the software requires more error information from the MPSC or SDMA cause registers, it can unmask the single corresponding bit in the cause registers and read the Main Interrupt Cause (High) register.

Fix

This erratum will be fixed in the next revision of the device.

FEr #MISC-2 Loss of GPP interrupts.

TYPE: Errata

Description

When the General Purpose Pins (GPP) are used as interrupts and configured to edge sensitive mode, any toggle from 0->1 (or 1->0, depending on the specific configuration in the level register) sets an interrupt in the "cause" register.

However, it is possible to override the interrupt indication and cause the interrupt to be lost if, at the same time, there is an attempt to write to the cause register to clear a pending interrupt cause bit. The write to the cause bit may overrides the interrupt indication and result in its loss.

Workaround

There are two solutions.

- The first solution is to use the GPP for interrupts in the Level Sensitive Mode.
- The second solution is to use the GPP for interrupts in the Edge Sensitive Mode

If the GPP are used for interrupts in the Edge Sensitive Mode, the interrupt handler must clear the cause register and then check the GPP value register to make sure that no other interrupt is asserted at the same time.

NOTE: Confirm that no system starvation occurs in case multiple interrupts are activated.

The following macro shows an ISR implementation example:

```
if (GPP interrupt active)
{
    clear the GPP cause register;  /* use the GPP value register to realize the source of the interrupts */
    While ((GPP value [31:0] AND GPP Interrupt Mask[31:0]) != 0x0)
    /*polling on GPP value register after masking with GPP Interrupt Mask register */
     {

        if (GPP value [1] !=0)
            GPP1_int_handler;

        If (GPP value [2] !=0)
            GPP2_int_handler;
                    .
                    .
                    .
                    .
                    .
        clear the GPP cause register;
        eieio ; /* CPU data pipe barrier */
    }
}
```

Fix
There are no plans to fix this erratum.

FEr #MISC-3 GPP Level Interrupts in Level Sensitive Mode.

TYPE: Errata

Description
When using the GPP interrupts in Level Sensitive Mode (the Comm Unit Arbiter Control register's GPPInt bit [10] is set to '1'), the GPP Interrupt Cause register must pass (mirror image) the GPP Value Register, including the value of the GPP input pins. Thus, when a GPP pin is set, the appropriate bit in the cause register must also be set also and when a bit in the cause register is set the appropriate GPP pin is also set.

What actually happens is that after the cause register is set, it will not be cleared when clearing the GPP Input Pin. Instead, the cause register itself must be cleared too.

Workaround
The ISR must clear the interrupt directly on the originating device and then clear the GPP

NOTE: From the interrupt output point of view, this is a similar mechanism as the edge sensitive mode.

Fix
This erratum will be fixed in the next revision of the device.

FEr #MISC-4 Compliance with PCI spec bridge ordering rules.

TYPE: Errata

Description

The PCI specification defines several transaction ordering rules for bridge devices.

One of the ordering rules requires that before a read transaction is completed on its originating bus, the transaction must retrieve from the bridge device any posted writes that were originated on the other side.

For example, if the CPU reads from a device located on the PCI bus, the CPU read must not be completed on the CPU bus until all of the PCI write data is written to memory.

The GT–64240A does not comply with this PCI bridge ordering rule. If proper ordering is required, it can only be maintained through the use other mechanisms, such as sync barrier or messaging unit.

Workaround

If a system requires that PCI ordering must be maintained, use one of the following GT–64240A mechanisms:

- Sync barrier.
- Synchronization read from the external PCI device.
- Have the IDMA flush the GT–64240A write buffers.

NOTE: Full details of the workaround implementation are provided in the *AN-84: PCI Ordering Implementation*.

Fix

This erratum will be fixed in the next revision of the device.

FEr #MISC-5 I^{2}C access to internal space during serial ROM initialization.

TYPE: Errata

Description

When the internal space base address is configured to 0xF1000000 at reset (by AD24 sampled HIGH at reset), the I^{2}C unit does not sample this setting and it will use address 0x14000000 to access the internal space.

This is only applicable when the serial ROM initialization is enabled and before the software changes the Internal Space Decode register's IntDecode bits [11:0] (from CPU, I2C, or PCI). After the software changes the internal space base, the I^{2}C unit will use the IntDecode's value.

Workaround

When the serial ROM initialization is enabled at reset, the serial ROM must access the internal register at address 0x14000000 + offset. The serial ROM address does not depend on the internal space base address sampled at reset.

If the internal base register is written from the serial ROM, all of the following accesses to the internal registers must be to the corresponding address. For example, if the I^{2}C interface includes the following lines:

```
0x14000068
0x00000F10
```

All of the accesses following this line must use the 0XF1000000 as the internal register base address.For example:

```
0xF1000000
0x4281A8FF
```

Fix

There are no current plans to fix this erratum.

FEr #PCI-1 A write from the PCI to a nonexistent internal space register is not supported.

TYPE: Errata

Description
Writes from the PCI to a nonexistent register in the GT–64240A might result in a PCI hang.

Workaround
When writing from the PCI to internal space, only write to an existing register.

Fix
There are no plans to fix this erratum.

FEr #PCI-2 Incorrect swapping during PCI-to-PCI memory transactions.

TYPE: Errata

Description
Under the following conditions, the GT–64240A PCI slave incorrectly performs data swapping during PCI-to-PCI memory transactions.

- PCI_0/1 P2P Swap Control register's M0Sw bits [2:0] is configured to have a different swap than M1Sw bits [6:4].
- The PCI Command register's MSwapEn bit [21] is set.
- The transaction on the PCI interface is an aggressive prefetch read or a burst write that crosses the maximum burst alignment. Use the PCI Access Control Base (Low) register to enable the aggressive prefetch read, PrefetchEn [12], and the maximum burst size, MBurst bits[21:20].
- The transaction is sent to P2P Mem1or the transaction is sent to P2P Mem0 with P2P Mem1 BAR enabled.

Workaround
There are two options:

- Set the PCI Command register's MSwapEn bit [21] to '0'.
- Configure P2P Mem0 BAR swap control to be equal to P2P mem1 bar swap control. Set thePCI_0/1 P2P Swap Control register's M0Sw bits [2:0] and M1Sw bits [6:4] to the same value. If it is a DAC transaction, use DM0Sw bits [10:8] and DM1Sw bits [14:12].

Fix
This erratum will be fixed in the next revision of the device.

FEr #PCI-3 Wrong REQ64 during PCI-to-PCI memory transactions.

TYPE: Errata

Description
Under the following conditions, the GT–64240A PCI slave incorrectly performs REQ64 during PCI-to-PCI memory transactions:

- The P2P Swap Control register’s M0Req64 bit [3] is set differently than M1Req64 bit [7].
- The PCI command register’s MReq64 bit [15] is enabled.
- The transaction on the PCI is an aggressive prefetch read or write burst that crosses the max burst alignment.
- The transaction is to P2P Mem1or the transaction is to P2P Mem0 with the P2P Mem1 BAR enabled.

Workaround
Here are two options:

- Set the PCI Command register’s MReq64 bit [15] to '0'.
- Configure P2P Mem0 BAR to force M0Req64 bit [3] to be equal to P2P Mem1 BAR force M1Req64 bit [7]. If it is a DAC transaction, use DM0Req64 bit [11] and DM1Req64 bit [15].

Fix
This erratum will be fixed in the next revision of the device.

FEr #PCI-4 PCI arbiter pins are not PCI compliant.

TYPE: Errata

Description
The GT–64240A can be configured to implement a PCI arbiter. The PCI arbiter interface is implemented on the Multi Purpose Pins (MPPs). However, these pins are not PCI 2.2 compliant I/O pads. These pins are regular LVTTL pins.

Workaround
None.

Fix
There are no plans to fix this erratum.

FEr #PCI-5 Erroneous behavior on simultaneous PCI configuration accesses from CPU and PCI.

TYPE: Errata

Description
In case of a simultaneous CPU-to-PCI configuration accesses and PCI-to-PCI (P2P) configuration accesses, the first PCI access out of the two is driven with a wrong address on the PCI bus.

NOTE: Configuration cycles are typically used only during system initialization. Also, it is unlikely that there is more than one host in the system generating PCI configuration cycles. Therefore, there is a low probability of encountering such simultaneous configuration cycles.

Workaround
Use semaphores to avoid simultaneous CPU-to-PCI and PCI-to-PCI configuration accesses.

Fix
There are no plans to fix this erratum.

FEr #PCI-6 PCI master may get stuck when disabled during operation.

TYPE: Errata

Description
When the PCI master is operating in combined read mode, and if disabled during operation (by negating its enable bit [2] of the PCI Status and Command register), the PCI master may get stuck.

Workaround
Combined read (MRdCom bit [5] offset 0xc00) must be disabled before disabling the PCI master and only enabled after reactivation (enabling) of the PCI master.

Fix
There are no plans to fix this erratum.

FEr #PCI-7 PCI sync barrier livelock condition.

TYPE: Errata

Description
Livelock may occur in the following cases:

- Sync barrier is used with the PCI Sync Barrier Virtual register while the PCI Command register’s SBdis bit [13] is ‘1’. The second access to the virtual register might be retried forever.
- When the SBdis value is '0' and after the first sync barrier access read completed on the PCI bus is targeted to an agent other then the GT–64240A.

Workaround
Set the SBdis bit to '0'.

After the sync barrier is completed, access the GT–64240A from the PCI bus before any other read to another agent is completed.

Fix
This erratum will be fixed in the next revision of the device.

FEr #PCI-8 When the device is set to Big Endian mode, the PCI VPD read drives the wrong data on the PCI.

TYPE: Errata

Description
When there is a VPD read from the GT–64240A PCI slave and the device is set to Big Endian mode, the wrong data is driven on the PCI bus when the VPD data is read from register offset 0x4C.

In case of a 32-bit device, the data on the PCI will be the mirror data of a 4 byte offset address. If the PCI is set to Little Endian mode, the data will also be byte swapped.

For example, the data in address 0x1F800000 (byte swapped if the PCI is set to Little Endian mode) instead of the data in address 0x1F800004.

In case of an 8- or 16-bit wide device, the data on the PCI is unknown and there is no workaround.

When the device is set to Little Endian mode, the data is correct.

The following table provides an example of how the data would look. This example is valid for 8-, 16- and 32-bit devices.

However, there is a workaround only for a 32-bit device.

PCI Endianess	Device Endianess	Swap Required	Data Returned from the Device	Data on the PCI	Remarks
Little	Little	No swap	00112233	00112233	
Little	Big	Byte swap	00112233	8, 16 bit device - random 32 bit device - mirror address and swapped	
Big	Little	Byte swap and Word swap	00112233	00112233	Configure internal swap to byte swap
Big	Big	Word swap	00112233	8, 16 bit device - unknown 32 bit device - mirror address and swapped	

Workaround
For 32-bit device, program the VPD data to the mirroring 4 byte offset address

For example, don't program the data to address 0x1F800000. Instead, program it to address 0x1F800004 and vice versa.

The following table provides an example of both how the data would look for a 32-bit device and the offered workaround.

PCI Endianess	Device Endianess	Needed Swap	Returned data from the device	Data on the PCI	Remarks
Little	Little	No swap	00112233	00112233	
Little	Big	Byte swap	00112233	00112233	
Big	Little	Byte swap and word swap	00112233	00112233	Configure internal swap to byte swap
Big	Big	Word swap	00112233	00112233	Configure internal swap to no swap

The following table provides an example of how the data would look for an 8- or 16-bit device. There is no workaround here.

PCI Endianess	Device Endianess	Needed Swap	Returned data from the device	Data on the PCI	Remarks
Little	Little	No swap	00112233	00112233	
Little	Big	Byte swap	00112233	Unknown	
Big	Little	Byte swap and word swap	00112233	00112233	Configure internal swap to byte swap
Big	Big	Word swap	00112233	Unknown	

The internal swap setting is held in the PCI Command register's SIntSwap bits [26:24].

Fix
This erratum will be fixed in the next revision of the device.

Restrictions List

Res #COMM-1 Abort on a CPU owned descriptor.

TYPE: Restriction

Description
When the Ethernet port is transmitting a packet of multiple descriptors of which the last one is owned by the CPU, the port aborts the transmission and then hangs.

To send a packet, all the descriptors must be DMA owned.

Fix
There are no plans to correct this restriction.

Res #COMM-2 Internal Loopback in Ethernet ports.

TYPE: Restriction
When working with the Ethernet port in Internal Loopback mode, (the Port Configuration register’s LPBK bits [9:8] are set to ‘01’), the link must be up.

NOTE: The FLP bit (Port Configuration Extended Register [11]) has no effect in this case.

Fix
There are no current plans to correct this restriction.

Res #CPU-1 Split read transactions while using PMC-Sierra MIPS RM7k CPUs.

TYPE: Restriction

Description
The PMC-Sierra RM7k CPU can work in a split read mode through it's initialization sequence.

If set to ‘1’, multiple reads are enabled. A ‘0’ setting means that multiple reads are disabled.

If the CPU is configured to a split read mode, the CPU drives TcWord[1:0] only on cache hits.

If the CPU is configured to a non-split read mode, the CPU drives TcWord[1:0] on every read transaction except cache misses.

In both cases the GT–64240A drives TcWord[1:0] on every read data phase except cache hits. Thus, in non-split read mode, a contention may occur on the TcWord[1:0] during non-cached read data phases. It will not lead to a logic problem because during these time periods this bus is not probed. Still, this occurrence may cause electrical problems.

To avoid the contention, configure Modebit 26 in the CPU to ‘1’. If split read is not allowed in the system, set the CPU Configuration register’s SplitRd bit [13] to ’0’. This will prevent the GT–64240A to respond with Pack* on a Prqst* by the CPU, eliminating the CPU's split read capability.

NOTE: If using a SandCraft CPU, no special actions are required.

The only systems affected by this restriction are those using a PMC-Sierra RM7k CPU with the split read feature disabled.

Fix
There are no plans to fix this restriction.

Res #CPU-2 Burst to internal register not reported to the error registers

TYPE: Restriction

Description
In the CPU Error Cause register (0x140), TTErr bit [2] must be set when the CPU attempts to burst (read or write) to an internal register, to send an error indication.

However, the GT–64240A does not report the error. Therefore, it will never set this bit nor assert an interrupt.

Fix
There are no plans to correct this restriction.

Res #DMA-1 IDMA Burst Limit less than 8 bytes is not supported.

TYPE: Restriction

Description
The GT–64240A IDMA must be configured to burst limit equal or greater than 8 bytes.

Fix
There are no plans to correct this restriction.

Res #DMA-2 IDMA addressing restrictions.

TYPE: Restriction

Description
IDMA channel’s source address and next pointer address must never be mapped to an “Access protected” or to an un-mapped address region.

IDMA channel’s destination address must never be mapped to an “Access protected” or “Write protected” or to an un-mapped address regions.

Any violation to the above restriction causes the IDMA channel to hang.

Fix
There are no plans to correct this restriction.

Res #MEM-1 Burst access limitations to a 32-bit device.

TYPE: Restriction

Description
An access to a 32-bit wide device on the GT–64240A device bus, must not cross a 32 byte boundary.

Ensure that the following bits are set for this restriction:

- The PCI Access Control register’s Mburst bits [21:20] must be set to ‘00’.
- The DMA Channel Control register’s IDMAs BurstLimit bits [8:6] must be set either ‘000’, ‘001’, or ‘011’.

Fix
There are no plans to correct this restriction.

Res #MEM-2 Device controller does not support non-consecutive byte enables.

TYPE: Restriction

Description
The GT–64240A's Device Controller does not support a non-consecutive Byte Enable (BE) accesses. In short, only the following byte enable combinations are supported in a 32-bit access:

- BE# = '0001'
- BE# = '0011'
- BE# = '0111'
- BE# = '1111'
- BE# = '1110'
- BE# = '1100'
- BE# = '1000'
- BE# = '1011'
- BE# = '1101'

Fix
There are no plans to correct this restriction.

Res #MEM-3 SDRAM timing parameters must have the same value.

TYPE: Restriction

Description
The SDRAM timing parameters CAS latency, RAS to CAS, and RAS Precharge must be set to the same value (2 or 3).

These timing parameters are configured in the CL, Trcd, and Trp fields of the SDRAM Timing Parameters register at offset 0x4b4.

Fix
There are no plans to correct this restriction.

Res #MEM-4 SDClkOut mode not supported.

TYPE: Restriction

Description
The GT–64240A can be configured to work with different clocking schemes.

It is best to use the SDClkin and SDRAM clock skewing option in the clocking scheme. SDClkOut mode has not been fully verified in silicon and therefore should not be used.

Fix
There are no current plans to fix this restriction.

Res #MISC-1 I^2C serial ROM data must be written in Little Endian byte order.

TYPE: Restriction

Description

The data written through the I^2C port must always be in Little Endian byte order. For example if the data is “11223344” it will be written this way to the register although in Big Endian mode it should be “44332211”.

Always keep the data to be written through the I^2C port in Little Endian byte order.

Fix

There are no plans to correct this restriction.

Res #MISC-2 I_2O queue ports are inaccessible from the PCI through internal space.

TYPE: Restriction

Description

The I_2O inbound/outbound queue ports are only accessible through the lower 4KB of Bank0 (SCS0).

Only access the I_2O inbound/outbound queue ports through the low 4 KB of Bank0 (SCS0). Set the PCI_0/1 Address Decode Control registers’ MsgAcc bit [3] to '0'.

Fix

There are no plans to correct this restriction.

Res #MISC-3 Write to the most significant byte of the MPP interface registers.

TYPE: Restriction

Description

When writing to the MPP interface register’s most significant byte, read the register and write the whole register [31:0] with the updated MSB (Read-Modify-Write).

Fix

There are no plans to correct this restriction.

Res #PCI-1 PCI master operation mode during DMA transfer.

TYPE: Restriction

Description

When the IDMA channel source address is mapped (either through address decode or PCI override) to the PCI, the GT–64240A’s PCI Command register’s MRdTrig bit [7] must be set to ‘0’ (“read store & forward”).

Fix

There are currently no plans to correct this restriction.

Res #PCI-2 I^{2}C Serial ROM writes to the PCI Configuration Address and Configuration Data registers.

TYPE: Restriction

Description

With the serial ROM initialization is enabled, an access to the PCI0 interface Configuration Address and Data registers must be performed as an access to the other PCI1 interface Configuration Address and Data registers offset. This means:

- PCI0 Configuration Address register (offset 0xcf8) must be accessed as PCI1 Configuration Address register (offset 0xc78). The access will be physically targeted to PCI_0 interface internal register offset 0xcf8.
- PCI0 Configuration Data register (offset 0xcfc) must be accessed as PCI1 Configuration Data register (offset 0xc7c). The access will be physically target to PCI0 interface internal register offset 0xcfc.
- PCI1 Configuration Address register (offset 0xc78) must be accessed as PCI0 Configuration Address register (offset 0xcf8). The access will be physically target to PCI1 interface internal register offset 0xc78.
- PCI1 Configuration Data register (offset 0xc7c) must be accessed as PCI_0 Configuration Data register (offset 0xcfc). The access will be physically target to PCI1 interface internal register offset 0xc7c.

For example:

```
/* PCI0 Master enable */
0x14000c78,
0x80000004,
0x14000c7c,
0x02b00004,
/* PCI1 Master enable */
0x14000cf8,
0x80000004,
0x14000cfc,
0x02b00004,
```

Fix

There are no plans to correct this restriction.

Marvell Technology
700 First Ave.
Sunnyvale, CA 94089
Phone: (408) 222-2500, Fax: (408) 752-9028
E-mail: salescom@marvell.com