Errata and Restrictions
GT-64240-B-0

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

Stepping Summary and Identification

Stepping	Marking	Errata Fixed	Errata NOT Fixed
0 (first silicon)	GT-64240-B-0	Baseline, original silicon	Errata 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61 Restrictions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Errata Revision History

Rev #	Date	Devices Covered	Errata Described
0.1	Feb 8, 2001	GT-64240-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42
0.2	Unreleased	GT-64240-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42
0.3	Apr 11.2001	GT-64240-B-0	1, 2, 3*, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18*, 19*, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52 * Revised errata #3, #18, and #19.
.-	July 10, 2001	GT-64240-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50, 51, 52 Removed errata #49. Redundant to errata #36.
0.5	September 13, 2001	GT-64240-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61
A	October 11, 2001	GT-64240-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24*, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61 *Amended errata #24.

Restriction Revision History

Rev #	Date	Devices Covered	Restriction Described
0.3	Apr 11, 2001	GT-64240-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
.-	July 10, 2001	GT-64240-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
0.5	September 13, 2001	GT-64240-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
A	October 11, 2001	GT-64240-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Errata List

FEr #1: PCI Slave does not Support 64-bit addressing (DAC cycles).

TYPE: Errata

Description
The GT–64240 slave **does not** support DAC cycles on both PCI0 and PCI1. The GT–64240 cannot be mapped in addresses that are above 4Gbyte (0xFFFFFFFF).

The GT–64240 master **does** support DAC cycles. This means that external agents mapped above the 4Gbyte range are accessible by the GT-64240.

Workaround
None.

Fix
This errata will be fixed in the next revision of the GT–64240.

FEr #2: PCI Vital Product Data (VPD) not supported.

TYPE: Errata

Description
Appendix I of the PCI 2.2 spec defines Vital Product Data (VPD) structure that a device on the PCI bus can support. The GT–64240 does not support the VPD protocol.

Workaround
None.

Fix
This errata will be fixed in the next revision of the GT–64240.

FEr #3: Internal registers are not accessible from PCI while I_2O is enabled.

TYPE: Errata

Description
When using I_2O and the I_2O registers are mapped to the first 4Kbyte of SDRAM Bank 0 (SCS[0]) space (i.e., bit[3] in the PCI Address Decode Control register at offset 0xd3c or 0xdbc, is set to '0'), the GT–64240 will not respond (i.e., will not assert DEVSEL) to a PCI access targeted to the following Internal register sets:

- CPU registers located at offsets 0x0xx, 0x1xx, 0x2xx, 0x3xx.
- SDRAM and device registers located at offset 0x4xx.
- IDMA registers located at offsets 0x8xx, 0x9xx.

Accesses to all other internal locations are not effected.

Workaround
- When I_2O registers are mapped to the first 4Kbytes of SDRAM SCS [0] space, avoid PCI accesses to the CPU, SDRAM, and IDMA registers.
- Map the I_2O register space to part of the internal space by setting the PCI Address Decode Control register's MsgAcc bit[3] (at offset 0xd3c or 0xdbc) to Map the I_2O register space to part of the internal space by setting bit[3] of the PCI address decode control register (at offset 0xd3c or 0xdbc) to '1'.

Fix
This errata will be fixed in the next revision of the GT–64240.

FEr #4: Simultaneous PCI-to-PCI I/O transactions from both PCI buses may cause a deadlock.

TYPE: Errata

Description
In cases when PCI I/O transactions are simultaneously initiated at both PCI interfaces and are both targeted to the other PCI interface, a deadlock may occur and both the PCI buses may hang.

Workaround
The system must guarantee that the PCI-to-PCI I/O transactions are never simultaneously initiated.

Fix
This errata will be fixed in the next revision of the GT–64240.

FEr #5: PERR* asserted on non-data phase cycles.

TYPE: Errata

Description
When an external master drives erroneous PAR while IRDY is not asserted, the GT–64240 asserts PERR*, although parity checking is not needed.

Workaround
1. Disable PERR* generation by setting the PCI status and Command register's PErrEn bit [6] to '0'.
2. Make sure that the masters in the system behave normally and don't generate a "late" IRDY signal.

Fix
This errata will be fixed in the next revision of the GT–64240.

FEr #6: Lost PCI delayed read.

TYPE: Errata

Description
If two consecutive delayed reads are followed by a PCI-to-PCI (P2P) configuration type 1 transaction, in some cases, the second delayed read may be lost.

Workaround
1. Make sure the delayed reads are completed before the PCI-to-PCI configuration.
2. Do not generate P2P configurations during normal system operation (typically configurations are only on system initialization).

Fix
This errata will be fixed in the next revision of the GT–64240.

FEr #7: PCI aggressive prefetch not supported.

TYPE: Errata

Description
The GT–64240 PCI Slaves do not support aggressive prefetch. Do not enable this feature.

NOTE: For further information about this feature, see Section 8.8.2, PCI read operation in the datasheet

Workaround
None.

Fix
This errata will be fixed in the next revision of the GT–64240.

FEr #8: PCI-to-PCI SAC to DAC conversions are not supported.

TYPE: Errata

Description
SAC (Single Address Cycle) to DAC (Dual Address Cycle) can occur when a transaction received on one PCI is targeted at the other PCI and goes through the remapping process. In the GT–64240 datasheet rev. 1.1, see Section 3.8.3, PCI Address Remapping and PCI address remapping registers.

However, the GT–64240 does not support these SAC to DAC conversions. Therefore, the PCI P2P Mem0 Base Address Remap (High portion) and PCI P2P Mem1 Base Address Remap (High portion) registers must be set to ‘0’

Workaround
None.

Fix
This errata will be fixed in the next revision of the GT–64240.

FEr #9: GT–64240 may return incorrect data following a target abort.

TYPE: Errata

Description
If a PCI 32-bit access to the GT–64240 causes the PCI Slave to generate a Target Abort (see Section 8.9, PCI Target Termination) and, following that, another PCI 32-bit read access is initiated to the GT–64240, the data returned by the GT–64240 PCI Slave on the second access may be erroneous.

Workaround
If a target abort occurs, follow the aborted access with a dummy transaction (clearing the erroneous data from the internal Slave FIFOs), before issuing the second read cycle.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #10: PCI write protect interrupt asserted on read.

TYPE: Errata

Description
When a PCI read transaction is initiated to a write protected memory region, the PCI Interrupt Cause register's SWrProt bit [19] in (offset 0x1d58 and 0x1dd8) is erroneously asserted.

Workaround
Do not assign the write protect attribute to memory regions.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #11: PCI Power Management (PMG) is not fully compliant with the PCI 2.2 spec

TYPE: Errata

Description
The PCI 2.2 spec and the PCI Power Management Interface Spec define four different PCI Function Power Management States referred to as D0, D1, D2, and D3. When in any state other than D0, the GT–64240 slave must only respond to a configuration transaction. However, the GT–64240 PCI slave does respond to transactions other than configuration.

Workaround
When not in D0 state, the system must turn off the PCI status and Command register's MemEn bit [1] and IOEn bit [0].

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #12: PCI master illegal memory write and invalidate.

TYPE: Errata

Description
GT–64240 PCI master might initiate an illegal “memory write and invalidate” command, with a length that is not a multiple of a cache line size, under the following conditions:

- The GT–64240 PCI master receives two consecutive write transactions, with continuous addresses. The GT–64240 has the option to combine these two accesses to a single access on the PCI bus. In the GT–64240 datasheet rev. 1.1, see Section 8.1.1, Expansion ROM and 8.1.2 Synchronization Barrier for more details.
- Both transactions are a multiple size of 8 bytes (8,16,24...).
- In the first transaction, all enabled bytes were asserted.
- In the second transaction, not all bytes are enabled - a partial write.

Workaround
- Disable “memory write and invalidate” by setting the PCI Status and Command register’s MemWrInv bit [4] in (Offset 0x4 or 0x84 in configuration space) to ‘0’.
- Disable ”combine on write” by setting the PCI Command Register™s MWrCom bit [4] in the (Offset 0xc00 and 0xc80) to ‘0’.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #13: PCI master may retry a transaction with a different command attribute.

TYPE: Errata

Description
When the GT–64240 PCI master receives two consecutive read transactions, with continuous addresses to be generated on the PCI bus, it has the option to combine these two accesses to a single access on the PCI bus. In the GT–64240 datasheet rev. 1.1, see Section 8.1, PCI Master Operation for more details.

If the GT–64240 PCI master receives a retry termination on a “memory read” command, it may retry the transaction with a “memory read multiple” command. This illegal behavior may only occur on a combining transaction.

Workaround
- Disable “read multiple” by setting the PCI command register’s MRdMul bit [9] (Offset 0xc00 and 0xc80) to ‘0’.
- Disable “combine on read” by setting the PCI command register’s MRdCom bit [5] (Offset 0xc00 and 0xc80) to ‘0’.

Fix

This errata will be fixed in the next revision of the GT-64240.

FEr #14: PCI arbiter pins are not PCI compliant.

TYPE: Errata

Description
The GT-64240 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.

Workaround
None.

Fix
There are currently no plans to fix this errata.

FEr #15: 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.

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 currently no plans to fix this errata.

FEr #16: PCI Error Address registers do not lock.

TYPE: Errata

Description
On the event of error condition on the PCI bus (e.g. data parity error), the address, command, and data of the erroneous transaction are latched in the PCI Error Address and Data registers.

In case of multiple errors, only the address, data, and command of the first erroneous transaction is latched and the value in these registers is supposed to be locked until software reads the PCI Error Address register. However, a read from a register in offset 12'bxxxx.x100.00xx (e.g. 0x0440) causes the register to be unlocked, enabling new erroneous transactions to be locked.

Workaround
None.

Fix

This errata will be fixed in the next revision of the GT-64240.

FEr #17: Erroneous interrupt in MultiGT mode.

TYPE: Errata

Description
In MultiGT mode, when there are multiple slaves on the CPU bus and the CPU accesses another slave on the bus, the GT–64240 does not respond to this transaction as expected. The GT–64240 wrongly generates a mismatch error in the CPU Error Cause registers AddrOut bit [0] (offset 0x0140).

Workaround
When in MultiGT mode, mask the AddrOut bit [0] in the CPU Error Mask register (offset 0x0148).

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #18: Wrong behavior of RspSwap* signal.

TYPE: Errata

Description
When the CPU is granted the bus ownership, the RspSwap* control signal is erroneously tri-stated.

It is required that the GT–64240 configuration prevent the RspSwap* signal from floating for a long period (a few tens of cycles).

Workaround
Add 4.7KOhm pull-up resistor to the RspSwap* signal.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #19: Incorrect behavior of PAck* signal pin.

TYPE: Errata

Description
The R7000 non-pedant reads (split reads) protocol requires that the system controller asserts PAck* for one cycle in response to a CPU PReq* assertion.

The GT–64240 asserts this signal for one cycle, as expected, but stops driving it (tri-state) on the following cycle. This results in the signal remaining at a ‘0’ value for several cycles. This, in turn, may cause unpredictable CPU behavior.

Workaround

- Use external HW that gets the GT–64240 PAck* as an input to implement the right behavior of PAck*. This HW must assert (drive to ‘0’) the PAck* for one cycle, only.
- Configure the GT–64240 to **not support** split read transactions. Set the CPU Configuration register’s SplitRd bit [13] (offset 0x000) to ‘0’

Fix

This errata will be fixed in the next revision of the GT-64240.

FEr #20: PCI PME pin is not an open drain output.

TYPE: Errata

Description
When the MPP is configured to behave as PME (PCI Power Management Event) pin, it doesn't behave as an open drain pin. Instead of floating the pin during the inactive cycles, it drives it high.

NOTE: The PCI specification defines MPE as an open drain output, to enable wired OR of all PCI devices PME pins. If the GT–64240 is the only device that drives this pin, a pure output pin is fine.

Workaround

Use external HW to convert the GT–64240 PME output to an open drain output to the PCI bus.

Fix

This errata will be fixed in the next revision of the GT-64240.

FEr #21: Ethernet port doesn't stop at a NULL descriptor.

TYPE: Errata

Description
The ethernet port doesn't stop when it gets to NULL descriptor. Instead, the port tries to fetch the next descriptor from the NULL address, which causes failure. In the GT–64240 datasheet rev. 1.1, 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 currently no plans to fix this errata.

FEr #22: Wrong select cause in the comm unit interrupt mask register.

TYPE: Errata

Description
In case of a Comm unit error event (e.g. address mismatch), the address of the erroneous transaction is latched in the Comm Unit Error Address register. The select cause field of the Comm Unit Interrupt Mask register must point to the interrupt event to which the Error Address register corresponds.

In case of multiple error events, it might point to the wrong interrupt event. In this case, the software cannot use the Error Address register.

NOTE: This register is typically used for software debug. In a working system, such errors should not happen. Thus, there is no need this register anyway.

Workaround
None.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #23: Possible deadlock in the SDMA.

TYPE: Errata

Description
The GT–64240 has two SDMAs that move data and descriptors between its two MPSC channels and memory. An attempt to write to an SDMA register while the SDMA is in operation (i.e. moving data) might cause the SDMA, and the unit that initiated a write access, to hang and potentially cause a system deadlock.

Workaround
Make sure that the SDMA is idle before writing to its internal registers by using the SDMA interrupt mechanism, or poll the descriptors status in memory.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #24: Comm Unit parity error.

TYPE: Errata

Description
The GT–64240 supports propagation of data integrity errors between its different interfaces.

Erroneously, the GT–64240 Comm unit constantly drives parity error (i.e., with each data portion that it delivers), regardless of the actual status of the data. In cases that parity error propagation is enabled at the receiving unit, this may cause an erroneous parity error indication to propagate in the system.

Workaround

If the system uses the GT–64240 Comm unit, the user must disable parity errors propagation in all the units that interface the Comm unit (i.e. reset the following bits: PErrProp bit [22] in CPU Configuration register, ErrProp bit [9] in the SDRAM ECC Control register, PErrProp bit [19] in the PCI Command register). Also, set SysADCValid bit [19] in the CPU Configuration register to ‘0’.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #25: Ethernet CRC Error after collision.

TYPE: Errata

Description
When all of the following conditions occur, the GT–64240 Ethernet ports might erroneously re-transmit a “collisioned” packet with a wrong CRC:

- The port is configured to Half-Duplex.
- The port is in RMII mode.
- A collision occurs.

Workaround
Do not operate the GT–64240 Ethernet port at RMII, Half-Duplex mode.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #26: IDMA EOT wrong operation.

TYPE: Errata

Description
When asserting the End Of Transfer (EOT) signal, the IDMA must stop upon the completion of the current Burst Limit transfer.

Erroneously, the IDMA continues to write data to the destination until the total byte count is reached. The content of this additional redundant data is unpredictable.

Workaround
Do not use the EOT mode with the current GT–64240 revision.

Fix

This errata will be fixed in the next revision of the GT-64240.

FEr #27: IDMA wrong descriptor closing in Big Endian mode.

TYPE: Errata

Description
The GT–64240 supports both Big and Little endian conventions. In the GT–64240 datasheet rev. 1.1, see Section 10.7, Big and Little Endian Support for details.

When in Big Endian mode, the IDMA erroneously writes the descriptors back to the memory in a Little Endian orientation (bytes are swapped).

This incorrect behavior has two instances:

- When in “descriptor compatibility mode”, the end result is that the “remained BC” field has its bytes swapped. In the GT–64240 datasheet rev. 1.1, see Section 10.2, IDMA Descriptors for details.
- When in “new descriptor mode”, the end result is that the byte should have contained the ownership bit and will be overwritten by the least significant byte of the byte count.

NOTE: Practically, if the End of Transfer (EOT) signal is not in use, the system will operate properly, since the byte count written in the close descriptor operation will have a value of zero, which implies that the ownership is really returned to the CPU (ownership bit value is ‘0’).

Workaround

- In “descriptor compatibility mode”, the software must swap the upper two bytes to get the correct data of the remaining byte count.
- In “new descriptor mode”, EOT mode must not be used.

Fix

This errata will be fixed in the next revision of the GT-64240.

FEr #28: IDMA early fetch next descriptor.

TYPE: Errata

Description
If the software initiates fetch of a new descriptor (FetchND) right after channel activation, and before the first data burst is transferred to the destination, the channel might get stuck.

Workaround
Activate FetchND bit only after ensuring that at least one burst limit is completed. Software can identify this condition, by polling the channel byte count register, which is updated after each Burst Limit transfer.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #29: IDMA fetch next descriptor in Demand mode.

TYPE: Errata

Description
When configured to operate in Demand mode, and when a fetch new descriptor (FetchND) instruction is issued, the DMA engine is expected to stop the current data transfer and fetch the next descriptor.

In reality, the fetch of the new descriptor due to the FetchND instruction only occurs after the DMAReq* is asserted. More over, this DMAReq* assertion first causes another Burst Limit byte from the current descriptor to be transferred, before the DMA fetches the next descriptor.

Workaround
When working in Demand mode, make sure to assert DMAReq* one more time after FetchND is issued. If the extra Burst Limit transfer (that is described above) must be prevented, initiate the FetchND before the last DMAReq* of the current descriptor.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #30: IDMA abort and IDMA pause instructions are not supported.

TYPE: Errata

Description
The GT–64240's IDMA channels cannot be aborted (Channel Control (low) register's bit [20]) or paused (Channel Control (low) register's bit [12]) during normal operation. An attempt to do this might cause the channel to hang.

Workaround
None.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #31: IDMA wrong activity status indication.

TYPE: Errata

Description
The GT–64240's IDMA channels report their activity status in the Channel Control (Low) register's ChanAct bit [14]. However, the ChanAct bit may erroneously indicate that the channel is no longer active when the fetch of the data into the channel's FIFO is completed, but the transfer of the data to the destination is not yet completed.

Workaround
None.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #32: IDMA maximum transfer size limitations.

TYPE: Errata

Description
If the source or the destination address of an IDMA transfer crosses a 1 Mbyte boundary for the second time, the channel hangs.

Any transfer below 1Mbyte has no restrictions and will be always executed correctly.

Example:

- If the transfer initial source address is 0x3ff00, the channel hangs when the transfer source address reaches the value of 0x500000.

Workaround
None.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #33: 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 currently no plans to fix this errata.

FEr #34: Timer/Counter interrupt cause register bits are not set when masked.

TYPE: Errata

Description
When a timer/counter reaches its terminal count, it must set the associated terminal count bit in the Timer/Counter Interrupt cause register at offset 0x868 or 0x968. However, if this cause register bit is masked (i.e. the corresponding bit in the Timer/Counter Interrupt Mask register, at offset 0x86c or offset 0x96c, is set), this operation does not occur.

Workaround
If used by the system, do not mask the timer/counter interrupts.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #35: Timer/Counter TCTcnt signal is asserted for more than one clock cycle.

TYPE: Errata

Description
When the Timer/Counter reaches its terminal count (counts down to '0'), it generates the TCTcnt signal. Erroneously, the TCTcnt signal is asserted for more that one clock cycle.

One of the immediate outcomes is that a concatenation of two Timer/Counters. This forms an extended (64-bit) Timer/Counter by connecting the TCTcnt of the first to the TCEn of the second and will not work properly.

Workaround
To implement a 64-bit counter, an external logic must be used to generate a one clock cycle signal on every assertion of the TCTcnt signal.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #36: Timer/Counter interrupt might be erroneously set by software.

TYPE: Errata

Description
Writing a value of '1' to a Timer/Counter Interrupt Cause register bit, erroneously sets the bit (although it should have had no effect).

Workaround

- If only one timer/counter is in use, a '0' value can be written to all register bits.
- If multiple timers/counters are in use, the timers/counters interrupt handler should:
  - Pause all timers (clear their enable bits)
  - Read again the Timers/Counters Interrupt Cause register (to make sure there are no new interrupts)
  - Clear the Timers/Counters Interrupt Cause register (by writing 0 to all bits)
  - Reactivate the counters

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #37: The internal PCI arbiter GNT* signal does not meet 66MHz PCI AC specification.

TYPE: Errata

Description
The GT–64240 internal PCI arbiter generates a GNT* signal with a maximum output delay of 11.5 ns, that does not meet the 66 MHz PCI 2.2 specifications

Workaround

- The system must be designed to accommodate the GNT* output delay limitations
- Disable the internal PCI arbiter (i.e., set the PCI Arbiter Control register's EN bit [31] at offset 0x1d00 or 0x1d80 to '0') and use an external arbiter.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #38: WrHigh parameter is violated at the end of a burst.

TYPE: Errata

Description
The Device controller WrHigh parameter defines the number of clock cycles the GT–64240 must keep BAdr and data valid after Wr* de-assertion, during a write access.

Erroneously, the GT–64240 does not meet WrHigh parameter on the last data of a write access. Instead, it drives the address and the data for only one cycle after Wr* deassertion

Workaround
None.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #39: Redundant access to a 32-bit device.

TYPE: Errata

Description
When accessing a 32-bit device, the GT-64240'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.

NOTE: The redundant write access (to offset 0x0) has all the Wr* signals de-asserted.

Workaround
None.

Fix
There are currently no plans to fix this errata.

FEr #40: Unified Memory Architecture (UMA) is not supported.

TYPE: Errata

Description
The GT–64240 does not support Unified Memory Access (UMA). In the GT–64240 datasheet rev. 1.1, see Section 5.14, Unified Memory Architecture Support in the datasheet for details.

Workaround
None.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #41: First JTAG cycle not driven.

TYPE: Errata

Description
The GT–64240 supports JTAG interface.

When the JTAG state machine enters the shift_dr state, the JTDO signal must be driven on the next falling edge of the JTAG clock (TCK).

Erroneously, on the first clock cycle, the GT-64240 floats the JTDO rather than driving it to its intended value.

Workaround
The system signal must ignore the first JTAG bit shifted out on the JTDO signal.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #42: PCI AC timing violations.

TYPE: Errata

Description
The GT–64240 violates several PCI 2.2 AC specifications for 66Mhz.

The following table below represents the GT-64240 preliminary PCI AC timing:

Signal Name	max output delay	min output delay	Setup	Hold	units
Frame*	7.6	2	3.7	0	ns
Irdy*	7.4	2	3.2	1	ns
Trdy*	6.9	2	4.4	0	ns
Devsel*	7.3	2	4.1	0	ns
Stop*	6.7	2	3.7	0.2	ns
Idsel			3.3	0	ns
AD[63:0]	8.3	2	4.8	0.7	ns
CBE_[7:0]	7.6	2	4.5	0.8	ns
Req*	7.5	2			ns
Gnt*			5	0	ns
Req64*	7.1	2	3.5	0	ns
Ack64*	7.1	2	3.5	0	ns
Par0	7.5	2	4.5	0	ns
par64	7.5	2	3	0.5	ns
Perr*	7.1	2	3.5	0	ns
Serr*	6.5	2			ns

Workaround
None.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #43: 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 currently no plans to fix this errata.

FEr #44: False CPU sync barrier requests on PCI0.

TYPE: Errata

Description
The CPU interface must issue a sync barrier request to the PCI0 interface on each CPU read access to its virtual registers, I/O space, or the configuration register (0xcfc).

In reality, the CPU interface also issues a false sync barrier request on every CPU read access to the PCI interface internal register that has address bits [11:4] equal to 8'hcf.

Workaround
To prevent a false sync barrier request, set bit 28 (ConfSBDis) in the CPU configuration register (0x000) to “1” (disable).

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #45: No Sync barrier requests on PCI1 when reading the configuration register (0xc7c).

TYPE: Errata

Description
A CPU read access from the configuration register 0xc7c does not issue a sync barrier request to the PCI1 interface.

Workaround
To issue a sync barrier to PCI1, the CPU must issue a read from the PCI1 virtual register 0x0c8 or from its I/O space.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #46: Wrong reset value of PCI Power Management register.

TYPE: Errata

Description
The PME bit value (bit [31] of register 0x40) is set to the wrong value ('1' instead of '0') when 'VPD' (pins AD[22:21]) or 'MSI' (pins AD[24:23]) are sampled '1' (supported) at reset.

Workaround
The software must positively guarantee the correct value of PME bit either by ignoring its wrong value or by overwriting it with the correct value.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #47: Wrong data is read from some SDMA internal registers.

TYPE: Errata

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

NOTE: A write to these registers will perform correctly.

Workaround
None.

Fix
There are currently no plans to fix this errata.

FEr #48: CrossBar arbiter cannot guarantee more than 50% bandwidth for a specific unit.

TYPE: Errata

Description:
Every target unit within the GT–64240 (e.g. SDRAM interface unit) has a dedicated sixteen slice "pizza" arbiter. This arbiter allocates bandwidth to the requesting units, in a round robin manner, based on the pizza arbiter programming.

However, to guarantee that the bandwidth is allocated as required, never assign adjacent pizza slices for a single requester. In essence, this restriction implies that the maximum bandwidth guaranteed for a specific requester is bounded to 50% (derived from the maximum of eight non-adjacent slices out of the total sixteen).

NOTE: Despite of the above, when the actual total requested bandwidth is less than the target's available bandwidth, a requesting unit will be given more than 50% of the total available bandwidth, even if programmed to have only eight pizza slices.

Workaround:
None.

Fix:
This errata will be fixed in the next revision of the GT-64240.

FEr #50: The first PCI transaction following Hot Swap set is ignored.

TYPE: Errata

Description
When the PCI Mode register's HotSwap bit [12] is set either following reset or via CPU programming, the first transaction (Frame*) over the PCI is ignored by the GT-64240.

NOTE: The first transaction can be aimed either at the GT-64240 or some other slave.

Workaround
It's highly probable that the very first transaction over the PCI following Hot Swap setting is not aimed at the GT–64240. To guarantee the GT–64240 response, perform a dummy transaction (read or write) after setting the Hot Swap bit.

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #51: Incorrect initial value of registers 0x380, 0x390, 0x3a0, and 0x3b0.

TYPE: Errata

Description
In the datasheet, bits 15:12 are reserved and have zero value.

In reality, their value is '1' and writable.

If a program changes only bits 11:0, eventually bits 15:12 will stay in their initial value, i.e., 0xf, and the region will never be opened.

Workaround
Configure this register with a 4 byte size transaction, where nibbles 3-7 have zero value (32'h00000???, ? = user defined).

Fix
This errata will be fixed in the next revision of the GT-64240.

FEr #52: Using 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 deasserted for at least one cycle during the time RTS is deasserted. This can be done 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
This errata will be fixed in the next revision of the GT-64240.

FEr #53: Abort on a CPU owned descriptor.

TYPE: Errata

Description
When the Ethernet port is transmitting a packet consisting of two descriptors and the last one is owned by the CPU, the port aborts the transmission and then hangs. This occurs when the DMA transfer is faster than the descriptor's ownership update by the CPU, in a looped descriptor chains where the last descriptor points to the first.

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

Fix
There are currently no plans to fix this errata.

FEr #54: 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.
- The first descriptor is marked as a one with no padding.
- The last descriptor is marked as a one that needs padding.

Workaround
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).

Fix
There are currently no plans to fix this errata.

FEr #55: Good packet discard in the Ethernet port.

TYPE: Errata

Description
After the Ethernet port receives two bad frames (MAC address is not in the Hash table) followed by a third, good packet, the third packet is discarded.

Workaround
None.

Fix
There are currently no plans to fix this errata.

FEr #56: Wrong indication of control character at the end of the buffer in UART Mode.

TYPE: Errata

Description
The UART Control Character Register (CHR5-8) holds eight programmable control characters.

Normally, when the UART recognizes a control character, it sets its corresponding RCRn bits [23:16] in the UART Event Status register (CHR10). If a control character is found at the end of the buffer, the descriptor is closed with bit [14] (C) set.

Sometimes, even if the last character in the buffer is not a valid control character, bit 14 (C) of the descriptor is set

Workaround
Ignore bit [14] in the descriptor in UART mode.

Fix
This errata will be fixed in the next revision of the GT-64260.

FEr #57: TxEndHigh and TxEndLow Interrupts are immediately asserted after reset.

TYPE: Errata

Description
After reset, the TxEndHigh and TxEndLow maskable interrupts are immediately asserted. This takes place even before the Ethernet Tx DMA has been started.

Normally, these interrupts indicate that the Ethernet Tx DMA has stopped processing the relevant priority queue after a stop command or after it has reached the end of the descriptor chain.

Workaround
1. Reset TxEndHigh & TxEndLow bits (7:6) in the Ethernet Interrupt Cause registers immediately following reset.
2. If these interrupt bits are not needed after reset assertion, mask these bits in the Interrupt Mask Register.

Fix
This errata will be fixed in the next revision of the GT-64260.

FEr #58: Redundant override of a not-owned descriptor at the end of the descriptor chain.

TYPE: Errata

Description

When the transmit SDMA controller fetches a not-owned descriptor with it's F (First field) bit set after the last descriptor of a frame, the transmission is stopped. The transmit SDMA controller must clear the TxD bit and return to IDLE state.

However, before clearing the TxD bit, the transmit SDMA controller performs an unexpected close descriptor operation (by writing the descriptor with the same value that was fetched). It also generates a redundant SdmaTxBuf interrupt.

If the CPU sets the Ownership bit (or any other bit in that descriptor) after the fetch, but prior to the SDMA write, it is overridden with the old (wrong) Not-Owned value.

The result is that there is no transmission of the packet associated with that descriptor.

NOTE: When the transmit SDMA controller encounters a Not-Owned descriptor with it's F bit reset, after the last descriptor of a frame, it DOES NOT perform the unnecessary close descriptor operation.

Workaround

1. Instead of a Not-Owned descriptor, use a null pointer as the end of the Tx chain.
2. Reset the F (First) bit of the End of Chain Not-Owned descriptor and ignore the TX RESOURCE ERROR interrupt it generates.
3. Use the Shadow field value as an unnecessary close descriptor operation indicator. The Read of the Shadow field in the SdmaTxBuf interrupt service routine, after normal close buffer operation, returns a value of '0'. However, if the Shadow field is initialized to a value other than '0', and the unnecessary close descriptor operation overrides the CPU write, the read of the Shadow field in this SdmaTxBuf interrupt service routine returns a non-zero value. This is demonstrated in the following algorithm:

```
/* for each tx descriptor */
      /* initialize byteCount and Shadow to any value except zero c
      pCurrentTxDescriptor->bytecnt = 0x1234;
      pCurrentTxDescriptor->shadow = pCurrentTxDescriptor->bytecnt;
      /* In prepareDescriptor routine */
      /* Set Shadow to byteCount value when preparing a descriptor */
   currentTxDesc->shadow = currentTxDesc->bytecnt;
      currentTxDesc->cmd_sts.owner == 1;
      /* In the Interrupt Service Routine */
         case SDMA_TX_BUF_RETURN:
      /* if the shadow field is 0 -->> we had a good transmitted packet */
      /* else -->> work around is needed, Tx SDMA is in idle */
         if(currentTxDesc->shadow!= 0)
         {
         GT_REG_READ(sdmaTxCommandRegister[portNumber],(UINT32*)&lSdmaCommand);
      /* if tx not in idle -->> return ERROR */
     if(lSdmaCommand & TX_DEMAND)
     {
     return ERROR;
   }
       /* currentTxDesc is not incremented to the next descriptor */
       /* Prepare the currentTxDesc again */
```

```
            prepareDescriptor(&currentTxDesc);
         /* Give a tx demand to send the packets */
            lSdmaCommand = lSdmaCommand | TX_DEMAND;
      GT_REG_WRITE(sdmaTxCommandRegister[portNumber],*((UINT32*)&lSdmaCommand));
         return REDO;
          }
         /* enable recognition of the work around in the next cycle */
            currentTxDesc->shadow = 0x1234;
         /* increment currentTxDesc to the next descriptor */
            while(currentTxDesc->cmd_sts.last == 0)
            {
            currentTxDesc = (TX_DESC*)(currentTxDesc->next_desc_ptr);
      }
```

Fix
This errata will be fixed in the next revision of the GT-64260.

FEr #59: Close of an Extra buffer in SDMA.

TYPE: Errata

Description
When several MPSCs are running in a high bit rate, the SDMA might consumes an extra buffer.

This extra buffer will be closed with the F and L bits and only contain one byte. This byte actually belongs to the previous buffer

Workaround

- Restrict the maximum bit rate of the MPSC to 40Mbps.
- Since a single byte frame is extremely rare in the system, treat a single byte frame as the last byte of the previous frame.

Fix
This errata will be fixed in the next revision of the GT-64260.

FEr #60: No partial writes to MSB of MPP registers.

TYPE: Errata

Description
Cannot perform a partial (byte) write to the most significant byte (bits [31:24]) of the registers with offsets 0xf00c and 0xf004.

Workaround
When writing to these two registers, use a Read-Modify-Write procedure as follows:

1. Read the register.
2. Write the whole register [31:0] with the updated MSB.

Fix
There are currently no plans to fix this errata.

FEr #61: Loss of GPP interrupts.

TYPE: Errata

Description

When used as interrupt input pins, the General Purpose Pins (GPP) function as edge triggered interrupts. This means that any toggle from 0->1 (or 1->0, depending on the specific configuration) sets an interrupt in the “cause” register.

However, it is possible 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

When an interrupt is detected, the interrupt handler must clear the cause register and then poll 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 currently no plans to fix this errata.

Restrictions List

Res #1: PCI arbiter setting.

TYPE: Restriction

Description

- When enabling priority arbitration (setting the PCI Arbiter Control register's PAEn bit [2] to '0'), set bit [0] of the PCI Arbiter Control register to '1'. This setting avoids arbiter starvation conditions occurring.
- The PCI Arbiter cannot park on a low priority agent. This means that it is forbidden to have a combination of both the PCI Arbiter Control register's P bits [13:7] and PD bits [20:14] set to '0'.

Fix
There are currently no plans to correct this restriction.

Res #2: IDMA Burst Limit smaller than 8 is not supported.

TYPE: Restriction

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

Fix
There are currently no plans to correct this restriction.

Res #3: IDMA Byte Count must be greater than or equal to the burst limit.

TYPE: Restriction

Description
An IDMA transaction byte count value must be equal to or greater than the transaction burst limit.

Fix
This restriction will be fixed in the next revision of the GT-64240.

Res #4: 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 currently no plans to correct this restriction.

Res #5: Burst access limitations to a 32-bit device.

TYPE: Restriction

Description
An access to a 32-bit wide device on the GT–64240 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', '011', or '111'.

Fix
There are currently no plans to correct this restriction.

Res #6: ALE2Wr minimum value.

TYPE: Restriction

Description
Three (3) is the minimum value allowed to be programmed to the Device Bank Parameters register's Ale2Wr bits [13:11].

Fix
This restriction will be fixed in the next revision of the GT-64240. The minimum value allowed will be '1' cycle.

Res #7: Device controller does not support non consecutive byte enables.

TYPE: Restriction

Description
The GT–64240'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 currently no plans to correct this restriction.

Res #8: 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 currently no plans to correct this restriction.

Res #9: 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–64240'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 #10: I_2O registers' mapping to SDRAM bank0 base address.

TYPE: Restriction

Description
When mapping I_2O registers to the low 4Kbyte of SDRAM bank0 (SCS[0]), the bank0 base address must be configured to its default value (0x00000).

Fix
This restriction will be fixed in the next revision of the GT-64240.

Res #11: Wrong swapping during PCI-to-PCI memory write transactions.

TYPE: Restriction

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

1. The PCI P2P Swap Control register is configured to have different swap cases between P2P Mem0 and P2P Mem1 (register M0Sw bits[2:0] is different than M1Sw bits [6:4]).
2. The Master Swap Enable bit [21] is set in the PCI command register.
3. The transaction on the PCI is a burst write that crosses the maximum burst alignment.
4. The transaction is to P2P Mem1. Or, the transaction is to P2P Mem0 and P2P Mem1 bar is enabled.

Workaround
Here are two options:

- Set the PCI command register's MSwapEn bit [21] to '0'.
- Configure P2P Mem0 bar swap control to be with the same value as the P2P Mem1 bar swap control (1d54[2:0] = 1d54[6:4]).

Fix
There are currently no plans to correct this restriction.

Res #12: Wrong Req64 during PCI-to-PCI memory transactions.

TYPE: Restriction

Description

The GT–64240 PCI slave performs an incorrect Req64 during PCI-to-PCI memory transactions under the following conditions:

- The PCI P2P Swap Control register is configured to have different force Req64 to P2P Mem0 and P2P Mem1.
- The MReq64 bit [15] is set to ‘1’ in the PCI Command register.
- The transaction on the PCI is a write burst that crosses the max burst alignment.
- The transaction is to P2P Mem1. Or, the transaction is to P2P Mem0 and P2P Mem1 bar is enabled.

Workaround

There are two solutions:

- 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].

Fix

There are currently no plans to correct this restriction.

Galileo / Marvell
2350 Zanker Road
San Jose, CA 95131
Phone: 1 408 367-1400
Fax: 1 (408) 367-1401
www.galileoT.com

www.marvell.com

Marvell Semiconductor, Inc.
645 Almanor Avenue, Sunnyvale, CA 94085
Phone: (866) 674-7253, Fax: (408) 328-0122
E-mail: datacom@marvell.com

Galileo Technology, Inc.
2350 Zanker Road, San Jose, CA 95131
Phone: (408) 367-1400, Fax: (408) 367-1401
E-mail: info@galileoT.com