Errata and Restrictions
GT-64260-B-0

This document outlines all of the known errata that exist in the GT–64260-B-0.

Stepping Summary and Identification

Stepping	Marking	Errata and Restrictions Fixed	Errata and Restrictions NOT Fixed
0 (first silicon)	GT-64260-B-0	Baseline, original silicon	Errata 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65 Restrictions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25

Errata Revision History

Rev #	Date	Devices Covered	Errata Described
0.1	FEB 8, 2001	GT–64260-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
0.2	Unreleased.	GT–64260-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 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 Removed errata #17 (mips only). *Revised errata #40 and #42.
0.3	APR 10, 2001	GT–64260-B-0	1, 2, 3*, 4, 5*, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40*, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55 Removed errata #41. See restriction #12. *Revised errata #3, #5, #40.
0.35	APR 16, 2001	GT–64260-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55* *Revised errata #55.
0.4	JUNE 6, 2001	GT–64260-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 55 Removed errata #51. Redundant to errata #34.
0.5	SEPT 13, 2001	GT–64260-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65
A	OCT 11, 2001	GT–64260-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65

Restriction Revision History

0.1	FEB 21, 2001	GT-64260-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18
0.2	Unreleased	GT-64260-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11*, 12, 13, 14, 15, 16, 17, 18 Revised restriction #11.
0.35	APR 16, 2001	GT-64260-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 Revised restrictions #8, #13, #14, #15, #16, #17.
0.4	JUNE 6, 2001	GT-64260-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23 Restriction #20 deleted. See errata #53.
0.5	SEPT 13, 2001	GT-64260-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25
A	OCT 11, 2001	GT-64260-B-0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25

Errata List

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

TYPE: Errata

Description
The GT–64260-B-0 slave doesn't support DAC cycles on both PCI_0 and PCI_1. The GT–64260-B-0 cannot be mapped in addresses that are above 4Gbyte (0xFFFFFFFF).

The GT–64260-B-0 master does support DAC cycles. This means that external agents that are mapped above the 4Gbyte range are accessible by the GT-64260.

Workaround
None.

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

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

TYPE: Errata

Description
Appendix I of PCI 2.2 spec defines Vital Product Data (VPD) structure that a device on the PCI bus can support. The GT–64260-B-0 does not support the VPD protocol.

Workaround
None.

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

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–64260-B-0 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 I2O 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 '1'.

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

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 the 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 PCI-to-PCI I/O transactions are never initiated simultaneously.

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

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–64260-B-0 asserts PERR*, although parity checking is not needed.

Workaround
- Disable PERR* generation by setting the PCI status and Command register's PErrEn bit [6] to '0'.
- Make sure that the Masters in the system behave normally and do not generate such a "late" IRDY signal.

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

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

- Make sure the delayed reads are completed before the PCI-to-PCI configuration type 1 transaction.
- 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-64260.

FEr #7: PCI aggressive prefetch not supported.

TYPE: Errata

Description
The GT–64260-B-0 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-64260.

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 (see Section 3.8.3, PCI Address Remapping and the PCI address remapping registers).

However, the GT–64260-B-0 does not support these SAC to DAC conversions. Therefore, the PCI P2P Mem0 Base Address Remap (High) and PCI P2P Mem1 Base Address Remap (High) registers must be set to ‘0’

Workaround
None.

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

FEr #9: GT-64260 may return incorrect data following a target abort.

TYPE: Errata

Description
If a PCI 32-bit access to the GT–64260-B-0 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–64260-B-0, the data returned by the GT–64260-B-0 PCI Slave on the second access may be erroneous.

Workaround
If a Target Abort occurs, follow the aborted access with a dummy transaction (which clears 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-64260.

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] (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-64260.

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 a state other than D0, the GT–64260-B-0 slave must only respond to a configuration transaction. However, the GT–64260-B-0 PCI slave does respond to a transaction 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-64260.

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

TYPE: Errata

Description
The GT–64260-B-0 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–64260-B-0 PCI master receives two consecutive write transactions, with continuous addresses. The GT–64260-B-0 has the option to combine these two accesses to a single access on the PCI bus. In the GT–64260 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-64260.

FEr #13: PCI Master may retry a transaction with a different command attributes.

TYPE: Errata

Description
When the GT–64260-B-0 PCI master receives two consecutive read transactions, with continuous addresses to be generated on the PCI bus, the master has the option to combine these two accesses to a single access on the PCI bus. In the GT–64260 datasheet rev. 1.1, see Sections 8.1.1 and 8.1.2 of the datasheet for more details.

If the GT–64260-B-0 PCI master receives a retry termination on a “memory read” command, it might 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-64260.

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

TYPE: Errata

Description
The GT–64260-B-0 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 access, and PCI-to-PCI (P2P) configuration access, the first PCI access out of the two is driven with a wrong address on the PCI bus.

NOTE: The 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 an error condition on the PCI bus (i.e., 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 are latched, and the value in these registers was supposed to be locked, until the 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-64260.

FEr #18: PCI PME pin is not an open drain output.

TYPE: Errata

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

NOTE: The PCI specification defines MPE as an open drain output, to enable a wired OR of all the PCI devices PMEpin. If the GT-64260 is the only device that drives this pin, a pure output pin is fine.

Workaround
Use external HW to convert the GT–64260-B-0 PME output to an open drain output to the PCI bus.

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

FEr #19: Ethernet port doesn’t stop at a NULL descriptor.

TYPE: Errata

Description
The ethernet port doesn't stop when it gets to NULL descriptor. The port tries to fetch the next descriptor from the NULL address. This results in a failure. In the GT–64260 datasheet rev. 1.1, see Section 14.3, Operational Description.

Workaround
At the last descriptor, make sure 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 #20: 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 don’t occur, thus there is no need for this register.

Workaround
None.

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

FEr #21: Possible deadlock in the SDMA.

TYPE: Errata

Description
The GT–64260-B-0 has four 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-64260.

FEr #22: Comm Unit parity error.

TYPE: Errata

Description
The GT–64260-B-0 supports propagation of data integrity errors between its different interfaces.

Erroneously, the GT–64260-B-0 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–64260-B-0 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).

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

FEr #23: Ethernet CRC error after collision.

TYPE: Errata

Description
When all of the following conditions occur, the GT–64260-B-0 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–64260-B-0 Ethernet port at RMII, Half-Duplex mode.

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

FEr #24: 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–64260-B-0 revision.

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

FEr #25: IDMA wrong descriptor closing in Big Endian mode.

TYPE: Errata

Description
The GT–64260-B-0 supports both Big and Little Endian conventions. In the GT–64260 datasheet rev. 1.1, see Section 10.8, Big and Little Endian Support.

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

The above wrong 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–64260-B-0 datasheet rev. 1.1, see Section 10.2, IDMA Descriptors for details.

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-64260.

FEr #26: IDMA early fetch next descriptor.

TYPE: Errata

Description
If 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 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-64260.

FEr #27: 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 DTL 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-64260.

FEr #28: IDMA abort and IDMA pause instructions are not supported.

TYPE: Errata

Description
The GT–64260-B-0'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-64260.

FEr #29: IDMA wrong activity status indication.

TYPE: Errata

Description
The GT–64260-B-0'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-64260.

FEr #30: IDMA maximum transfer size limitations.

TYPE: Errata

Description
If the source or the destination address of an IDMA transfer crosses a 1Mbyte 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-64260.

FEr #31: IDMA descriptor fetch parity error.

TYPE: Errata

Description
When operating in chain mode and the 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 the 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 #32: 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, timer/counter interrupts must not be masked

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

FEr #33: 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 than one clock cycle.

One of the immediate outcomes is that a concatenation of two timer/counters, to form an extended (64-bit) timer/counter, by connecting the TCTcnt of the first to the TCEn of the second, will not work properly.

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

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

FEr #34: 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 could 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-64260.

FEr #35: The internal PCI arbiter GNT* signal does not meet 66MHz PCI AC specification.

TYPE: Errata

Description
The GT–64260-B-0 internal PCI arbiter generates a GNT* signal with a maximum output delay of 11.5 ns, that does not meet the 66MHz 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-64260.

FEr #36: 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–64260-B-0 must keep BAdr and data valid after Wr* de-assertion, during a write access.

Erroneously, the GT–64260-B-0 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-64260.

FEr #37: Redundant access to a 32-bit device.

TYPE: Errata

Description
When accessing a 32-bit device, the GT-64260'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 #38: Unified Memory Architecture (UMA) is not supported.

TYPE: Errata

Description
The GT-64260 does not support Unified Memory Access (UMA). In the GT–64260 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-64260.

FEr #39: First JTAG cycle not driven.

TYPE: Errata

Description
The GT–64260-B-0 supports JTAG interface.

When the JTAG state machine enters the shift_dr state, the Tdo signal must be driven on the next falling edge of the JTAG clock (TCK). Erroneously, on the first clock cycle, the GT-64260 floats the Tdo rather than driving it to its intended value.

Workaround
The System 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-64260.

FEr #40: The GT-64260 CPU arbiter does not support IBM 750CX, and MPC7450 in 60X mode.

TYPE: Errata

Description
The GT–64260-B-0 internal arbiter uses the ABB* and DBB* input signals that are not driven by the IBM 750CX and the Motorola MPC7450. The problem with the MPC7450 exists **only** in 60X bus mode.

Workaround

- In a single-CPU system that does not require cache coherency (i.e., no snoop transactions will be generated by GT-64260) and does not issue GT-64260 CPU master transactions, set the GT–64260-B-0 to the internal arbiter mode (by driving ‘1’ on the AD[8] signal of the GT–64260-B-0 during reset). In addition, pull **up** the GT–64260’s ABB* and DBB* input signals.
- Set the GT–64260-B-0 to external arbiter mode and use an external arbiter.

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

FEr #42: CPU internal arbiter fails in a multi-master MPC7400/MPC7410 60X applications.

TYPE: Errata

Description
The GT–64260-B-0 CPU internal arbiter does not support multi-master MPC7400 or multi-master MPC7410 system configurations.

NOTE: If activated to drive data transactions on the bus, the GT–64260-B-0 CPU master is also considered as a 60X master.

This errata does not apply to systems using a single CPU and a single, snooper only GT–64260-B-0.

Workaround

- Add an external logic to implement new signals:
  LOGIC_DBG0* = (DBG0* | ~DBB*) that should be connected to the CPU0 DBG* input
  and
  LOGIC_DBG1* = (DBG1* | ~DBB*) that should be connected to the CPU1 DBG* input
- Use an external arbiter.

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

FEr #43: PCI AC timing violations.

TYPE: Errata

Description
The GT–64260-B-0 violates several PCI 2.2 AC specifications for 66Mhz.

The table below represents the GT-64260 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-64260.

FEr #44: GT-64260 never drives BG0*/GT_BR* output.

TYPE: Errata

Description
When configured to work with an external 60x arbiter, the GT–64260-B-0 BG0*/GT_BR* output is never driven. In essence, an external 60x bus arbiter (specifically suggested as a workaround for FErr#40 and FErr#42), never sees a bus request asserted by the GT–64260-B-0.

NOTE: This errata affects only systems where the GT–64260-B-0 is active as a master on the 60x bus (e.g., to generate snoop transactions when cache coherency is used).

Workaround
The external 60x bus arbiter must implement some scheme where it issues a periodic bus grant (GT_BG*) to the GT–64260-B-0. The implemented scheme allows the GT–64260-B-0 to potentially perform master accesses, without knowing if the bus is actually used. Also, the external arbiter must insert one dead cycle between the negation of any BGx and the assertion of the GT_BG*.

It is recommended to assert the GT_BG* for only one cycle. This allows the GT–64260-B-0 to issue a pending transaction, but minimizes bus dead cycles if the GT–64260-B-0 does not have a pending transaction.

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

FEr #45: 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 #46: 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, it 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 the CPU configuration register's ConfSBDis bit [28] in (0x000) to '1' (disable).

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

FEr #47: 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-64260.

FEr #48: 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 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-64260.

FEr #49: Wrong 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 currently no plans to fix this errata.

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

TYPE: Errata

Description:
Every target unit within the GT–64260-B-0 (i.e., 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 the above, in cases where the actual total requested bandwidth is less than the target's available bandwidth, a requesting unit is 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-64260.

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-64260_rts);
     IF (count[]<2) THEN
     count[] = count[]+1;
     ELSE
     count[] = count[];
     END IF;
     IF (count[]>1) THEN
     GT-64260_cts = external_cts;
     ELSE
     GT-64260_cts = GT-64260_rts | external_cts;
     END IF;
    END;
```

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

FEr #53: 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-64260.

NOTE: The first transaction can be aimed either at the GT-64260 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–64260-B-0. To guarantee the GT–64260-B-0 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- 64260.

FEr #54: Incorrect initial value of registers 0x380, 0x390, 0x3a0, and 0x3b0.

TYPE: Errata

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

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-64260.

FEr #55: PPC self ARTRY failure.

TYPE: Errata

Description
Self ARTRY is defined as the case when a PowerPC master (i.e., a CPU), asserts an ARTRY* to its own initiated transaction.

Self ARTRY is always a busy ARTRY (where the processor that initiated the transaction will not attempt to issue any write back of modified data as a response) Self ARTRY* cycles (according to Motorola) will only occur in the following cases:

- For the 7400 and 7410 processors:
  As a result of the following instructions: SYNC, TLBSYNC, TLBIE, and ICBI.
- For the 7450 (V'ger):
  As a result of the 'stwcx' instruction that lost its reservation.

When operating in MPX mode, the GT–64260-B-0 will not support Self ARTRY. If it occurs, the system may hang.

Workaround
Software Workarounds

1. Don't use SYNC, TLBSYNC, TLBIE or ICBI instructions when working with 7400 and/or 7410 systems. Don't use 'stwcx' instruction when working with the 7450 (V'ger).
2. Replace the problematic instructions as below:

   'tlbie', 'icbi', 'stwcx'
   These instructions are not compiler generated so they can be pinpointed in the source code before being compiled. The user should wrap 'tlbie', 'icbi', and 'stwcx' instructions as follows:

   the instruction 'tlbie r3' would become -

   ```
   #define GSYNC_PRE nop;nop;isync;eieio;isync;
   #define GSYNC_POST nop;nop;isync;eieio;isync;nop;nop;
   GSYNC_PRE
   tlbie r3;
   GSYNC_POST
   ```

   'tlbsync' –
   This instruction is only used in a multi CPU architecture which is not supported for MPX mode with Discovery, and therefore irrelevant.

   'sync' –
   Should be replaced with 'isync' instruction

Hardware workaround

In non-cache coherent systems (GT–64260-B-0's Master is disabled) or in cache coherent systems, where the GT–64260-B-0's master is activated only for address snooping, use an external logic to disable CPU address streaming.

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

FEr #56: The GT-64260 might hang due to unqualified sampling of ARTRY* signal negation.

TYPE: Errata

Description
The ARTRY* negation by the master (CPU) is done by tri-stating the ARTRY* for half a cycle and only then negating it. Erroneously, the GT–64260-B-0 samples the signal during this specific cycle and may enter an unstable state due to a setup problem.

This problem only affects systems where ARTRY* is active. This will not occur in the following systems:

- • Systems with single CPU, and
- • No cache coherency, and
- • Without Self-ARTRY

Workaround
There are three possible solutions to this problem:

- In a single CPU systems, where cache coherency is not required and Self-ARTRY* is avoidable, connect the GT–64260-B-0 ARTRY* through a pull up to Vdd.

NOTE: The CPU ARTRY* pin should not be connected to the GT-64260.

- In a single CPU systems using 60X mode, where cache coherency is not required and Self-ARTRY* is unavoidable, connect the GT–64260-B-0 ARTRY* through a pull-up to Vdd instead of implementing the external logic.

NOTE: Do not use this solution on systems using an MPC7450 processor with ‘stwcx’ instruction in their code. The CPU ARTRY* pin must not be connected to the GT–64260-B-0.

The third solution is for all other systems that are different than the systems described in the previous solutions. One of the following external logic suggestions must be implemented.

- 60x bus mode (in AACK* delay):

NOTE: This work around is applicable in the following conditions:

a) The GT-64260 is configured to 60x bus mode. Either by the Reset sampled pins (AD[7:6] == ‘00’) or through the Serial ROM Initialization (Register 0x120, bits [7:4] == 0x4).
b) The GT-64260 is configured (by default) to delay the AACK* assertion at least two cycles after TS* assertion (AACK* delay mode). (Register 0x000, bit [11] == ‘1’)
c) An external logic on the 60x bus negates the ARTRY* at the de-assertion cycle. Figure 1 describes the recommended external logic.

Figure 1: External Logic Block Diagram

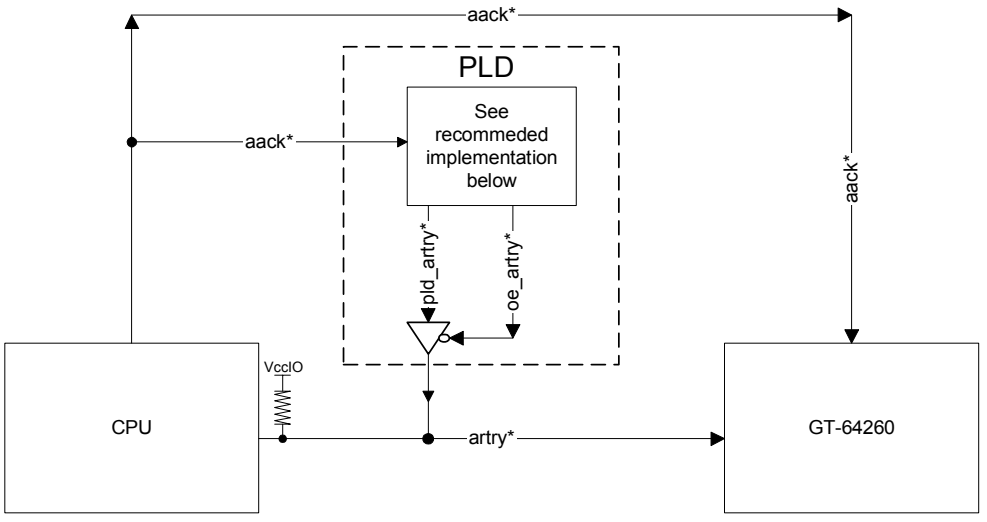
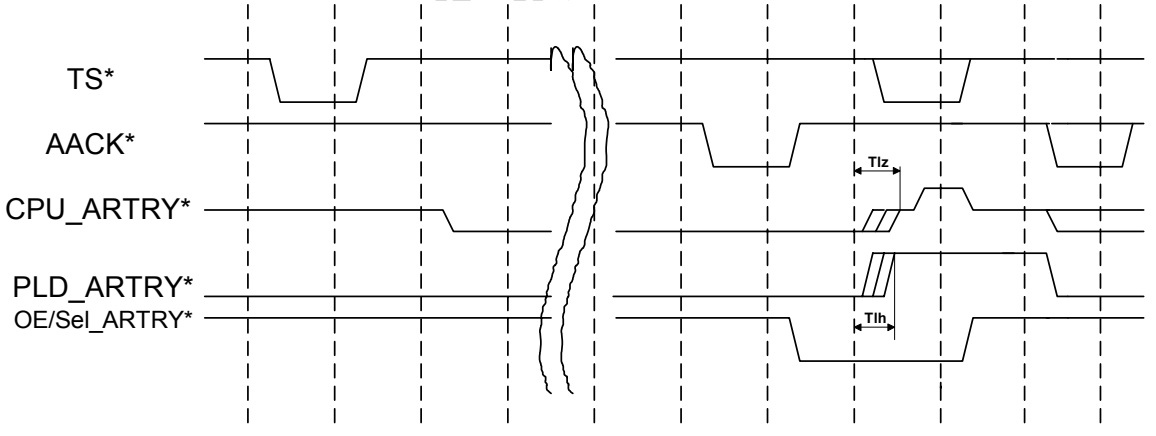


Figure 2 describes the signal timing for the above work around. To prevent contention on the ARTRY* signal, the system implementation must guarantee that Tlh(min)>Tlz(max), while Tlz is the CPU clock to high-Z and Tlh is the clock to out of the PLD.

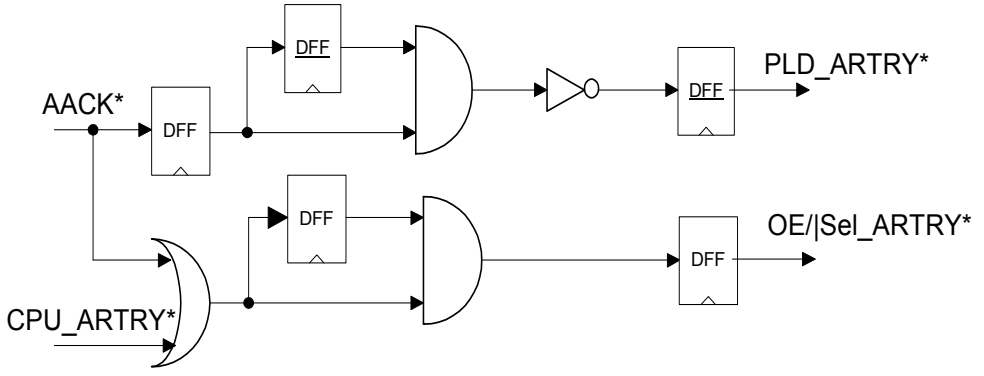
Figure 2: External Logic Timing Diagram



When the Tlh(min)<Tlz(max), the contention can be prevented by replacing the tri-state buffer with an external quick switch. In this configuration the Oe_ARTRY* is used as the select for the quick switch device (Sel_ARTRY*).

Figure 3 describes the recommended external logic implementation with a tri-state buffer and a quick switch device.

Figure 3: Recommended External Logic Implementation



- 60x (without AACK* delay) and MPX bus mode:

  In all systems that none of the above solutions works around the problem, implement an external logic that will force the negation of the ARTRY* during the de-assertion cycle by the CPU. Figure 4 describes the recommended external logic.

Figure 4: External Logic Block Diagram

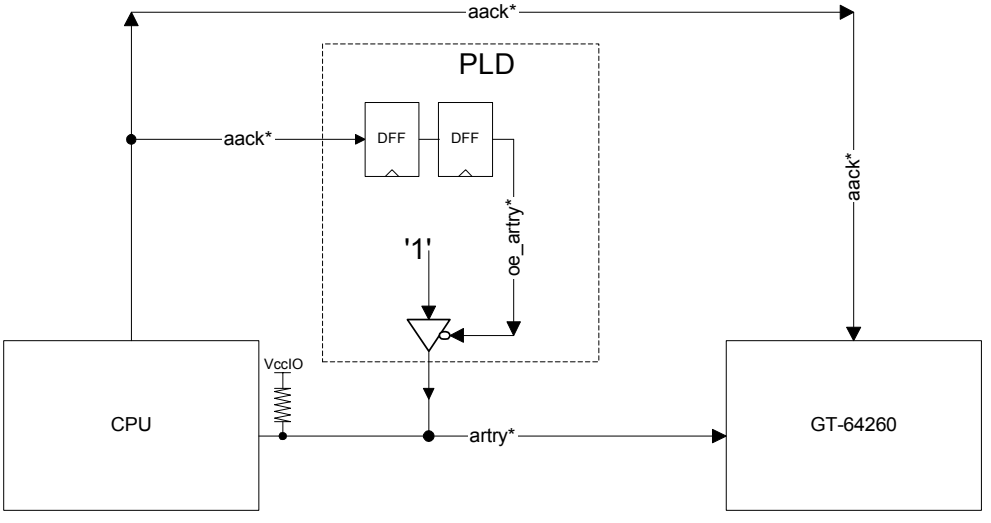
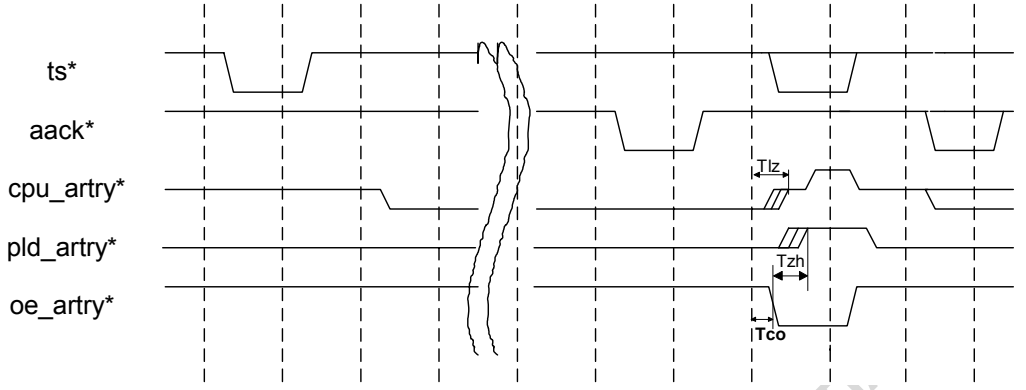


Figure 5 describes the signal timing for the work around.

Figure 5: External Logic Timing Diagram



NOTE: The timing of the output enable to out of the tri-state buffer (Tzh) must meet the ARTRY* the GT–64260-B-0's setup time. The calculation of the maximum clock frequency is calculated as follow:

Freq = 1/(Tco + Tzh + Tsu + T(clock skew, line delay, jitter etc.))

To prevent contention on ARTRY* signal make sure that Tlz(max)<Tzh(min).

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

FEr #57: 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 #58: 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 #59: 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 #60: 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 #61: 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 #62: 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 #63: 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 #64: 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 #65: 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: CPU Master Control register's CleanBlock bit value.

TYPE: Restriction

Description
When interfacing CPU devices that do not support the CleanBlock command, configure the CPU Master Control register's CleanBlock bit [12] to '0'

Fix
There are currently no plans to correct this restriction.

Res #2: 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 #3: An IDMA burst limit less than 8 bytes is not supported.

TYPE: Restriction

Description
The GT–64260-B-0 IDMA must be configured to burst limit equal to or greater than 8 bytes.

Fix
There are currently no plans to correct this restriction.

Res #4: An IDMA byte count must be equal to or greater than 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 corrected in the next revision of the GT- 64260.

Res #5: IDMA addressing restrictions.

TYPE: Restriction

Description
The IDMA channel's source address and next pointer address must never be mapped to an "Access protected" or un-mapped address regions.

The IDMA channel's destination address must never be mapped to an "Access protected", "Write protected", or 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 #6: Burst access limitations to a 32-bit device.

TYPE: Restriction

Description
An access to a 32-bit wide device on the GT–64260-B-0 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 #7: 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 corrected in the next revision of the GT-64260.

Res #8: Device Controller does not support non-consecutive byte enables.

TYPE: Restriction

Description
The GT–64260-B-0'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# = '1111'
- BE# = '0111' • BE# = '1110' • BE# = '1100'
- BE# = '1000' • BE# = '1011' • BE# = '1101'

Fix
There are currently no plans to correct this restriction.

Res #9: 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 following fields of the SDRAM Timing Parameters register (offset 0x4b4):

- CL bits [1:0]
- Trp bits [3:2]
- Trcd bits [5:4]

Fix
There are currently no plans to correct this restriction.

Res #10: 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–64260-B-0'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 #11: The GT–64260-B-0 CPU arbiter supports up to 16 outstanding transactions.

TYPE: Restriction

Description
The internal CPU arbiter supports a maximum of 16 outstanding transactions on the CPU bus.

Configure the system to avoid deeper pipeline situations.

Fix
There are currently no plans to correct this restriction.

Res #12: Wrong CPU sync barrier to a cache coherent region.

TYPE: Restriction

Description
Do not use CPU sync barrier instructions to cache coherent memory regions.

NOTE: If sync barriers are used, synchronization is not guaranteed.

Fix
There are currently no plans to correct this restriction.

Res #13: PCI non-prefetchable reads from SDRAM in PowerPC 60X system.

TYPE: Restriction

Description
SDRAM cache coherent regions must not be marked as PCI non-prefetchable. Any attempt to mark these regions as PCI non-prefetchable may cause the system to hang.

Fix
This restriction will be corrected in the next revision of the GT- 64260.

Res #14: PCI non-prefetchable reads from the 60X bus.

TYPE: Restriction

Description

60X bus regions must not be marked as PCI non-prefetchable. Any attempt to mark these regions as non-prefetchable may cause the system to hang.

Fix
This restriction will be corrected in the next revision of the GT- 64260.

Res #15: Communication ports read from the 60X bus.

TYPE: Restriction

Description
Communication ports (MPSCs or Ethernet) must not perform read accesses to the 60X bus. An attempt by the communication ports to perform this type of read access may cause the system to hang.

NOTE: SDMAs descriptors and Rx data must not be placed on devices on the 60x bus.

Fix
There are currently no plans to correct this restriction.

Res #16: Communication ports read from SDRAM coherent regions.

TYPE: Restriction

Description
Communication Ports (MPSCs or Ethernet) must not perform read accesses to SDRAM cache coherent regions. An attempt by the communication ports to perform this type of read access may cause the system to hang.

NOTE: SDMA descriptors and Rx data must not be placed in cache coherent region in SDRAM.

Fix
There are currently no plans to correct this restriction.

Res #17: PCI read from the GT–64260-B-0's CPU interface internal registers.

TYPE: restriction

Description
If the GT–64260-B-0 performs read accesses on the 60X bus, a simultaneous PCI read from the CPU interface internal registers is forbidden.

Fix
This restriction will be corrected in the next revision of the GT- 64260.

Res #18: DRAM interface read buffers allocation.

TYPE: Restriction

Description
If cache coherency is used in 60X bus mode, the SDRAM unit must be configured to return read data to the CPU from buffer #1.For this configuration, set the SDRAM Configuration register's RdBuff bit [26] to '1'.

Fix
There are currently no plans to correct this restriction.

Res #19: AACK delay in 60x bus mode.

TYPE: Restriction

Description
In 60x bus mode, with cache coherency or multi-CPU configuration, the CPU Configuration register's AACK Delay bit [11] must be set to '1'. For this setting, AACK* is asserted two cycles after TS*.

Fix
There are currently no plans to correct this restriction.

Res #20: 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 corrected in the next revision of the GT-64260.

Res #21: PowerPC self ARTRY in 60x bus mode.

TYPE: Restriction

Description
In 60x bus mode, when self ARTRY cycles are generated by the CPU, the CPU Configuration register's AACK Delay bit [11] must be set to the "aackdelay" mode.

NOTE: Refer to **FEr#55** for more details on self retry cycles.

Fix
There are currently no plans to correct this restriction.

Res #22: PowerPC Master does not support data transactions in multi-Slave system (MultiGT mode).

TYPE: Restriction

Description
In multi-Slave systems (MultiGT mode), the GT–64260-B-0 internal master supports only snoop transactions.

Using the master for regular data transactions is not allowed.

Fix
This restriction will be corrected in the next revision of the GT-64260.

Res #23: Wrong swapping during PCI to PCI memory write transactions.

TYPE: Restriction

Description
Under the following conditions, the GT–64260-B-0 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 #24: Wrong Req64 during PCI-to-PCI memory transactions.

TYPE: Restriction

Description
The GT–64260-B-0 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.

Disclaimer

Preliminary or Advanced Information: This document provides preliminary or advanced information about the product described. All specifications described herein are based on design goals only. **Do not use for final design.** Visit Marvell's web site at www.marvell.com or call 1-866-674-7253 for the latest information on Marvell products.

DISCLAIMER

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of Marvell. Marvell retains the right to make changes to this document at any time, without notice. Marvell makes no warranty of any kind, expressed or implied, with regard to any information contained in this document, including, but not limited to, the implied warranties of merchantability or fitness for any particular purpose. Further, Marvell does not warrant the accuracy or completeness of the information, text, graphics, or other items contained within this document. Marvell makes no commitment either to update or to keep current the information contained in this document. Marvell products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use Marvell products in these types of equipment or applications. The user should contact Marvell to obtain the latest specifications before finalizing a product design.

Marvell assumes no responsibility, either for use of this product or for any infringements of patents and trademarks, or other rights of third parties resulting from its use. No license is granted under any patents, patent rights, or trademarks of Marvell. These products may include one or more optional functions. The user has the choice of implementing any particular optional function. Should the user choose to implement any of these optional functions, it is possible that the use could be subject to third party intellectual property rights. Marvell recommends that the user investigate whether third party intellectual property rights are relevant to the intended use of this product and obtain licenses as appropriate under relevant intellectual property rights.

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