5.0 DETAILED FUNCTIONAL DESCRIPTION

5.1 PCI Interface

5.1.1 PCI COMMAND SET

Bus commands indicate to the slave the type of transaction the master is requesting. Bus Commands are encoded on the C/BE[3:0] # lines during the address phase of a PCI cycle.

5.1.2 PCI BUS TRANSFER BASICS

Details of PCI Bus operations can be found in the *Peripheral Component Interconnect (PCI) Specification*. Only details of the PCI Bus unique to the SIO are included in this data sheet.

Table 7. PCI Commands

C/BE[3:0] #	Command Type As Slave	Supported As Slave	Supported As Master
0000	Interrupt Acknowledge	Yes	No
0001	Special Cycle(4)	No/Yes	No
0010	I/O Read	Yes	No
0011	I/O Write	Yes	No
0100	Reserved(3)	No	No
0101	Reserved(3)	No	No
0110	Memory Read	Yes	Yes
0111	Memory Write	Yes	Yes
1000	Reserved(3)	No	No
1001	Reserved(3)	No	No
1010	Configuration Read	Yes	No
1011	Configuration Write	Yes	No
1100	Memory Read Multiple	No(2)	No
1101	Reserved(3)	No	No
1110	Memory Read Line	No(2)	No
1111	Memory Write and Invalidate	No(1)	No

NOTES:
1. Treated as Memory Write.
2. Treated as Memory Read.
3. Reserved Cycles are considered invalid by the SIO and are to be competely ignored. All internal address decoding is ignored and DEVSEL # is never to be asserted.
4. The 82378 responds to a Stop Grant Special Cycle.

5.1.2.1 PCI Addressing

PCI address decoding uses the AD[31:0] signals. AD[31:2] are always used for address decoding while the information contained in the two low order bits AD[1:0] varies for memory, I/O, and configuration cycles.

For I/O cycles, AD[31:0] are decoded to provide a byte address. AD[1:0] are used for generation of DEVSEL# only and indicate the least significant valid byte involved in the transfer. For example, if only BE0# is asserted, AD[1:0] are 00. If only BE3# is asserted, then AD[1:0] are 11. If BE3# and BE2# are asserted, AD[1:0] are 10. If all BEx#'s are asserted, then AD[1:0] are 00. The byte enables determine which byte lanes contain valid data. The SIO requires that PCI accesses to byte-wide internal registers must assert only one byte enable.

When the SIO is the target of any PCI transaction in which BE[3:0]# = 1111, the SIO terminates the cycle normally by asserting TRDY#. No data is written into the SIO during write cycles and the data driven by the SIO during read cycles is indeterminate.

For memory cycles, accesses are decoded as Dword accesses. This means that AD[1:0] are ignored for decoding memory cycles. The byte enables determine which byte lanes contain valid data. When the SIO is a PCI master, it drives 00 on AD[1:0] for all memory cycles.

For configuration cycles, DEVSEL# is a function of IDSEL and AD[1:0]. DEVSEL# is selected during a configuration cycle only if IDSEL is active and both AD[1:0] = 00. The cycle is ignored by the SIO if either AD1 or AD0 is non-zero. Configuration registers are selected as Dwords using AD[7:2]. The byte enables determine which byte lanes contain valid data.

5.1.2.2 DEVSEL# Generation

As a PCI slave, the SIO asserts the DEVSEL# signal to indicate it is the slave of the PCI transaction. DEVSEL# is asserted when the SIO positively or subtractively decodes the PCI transaction. The SIO asserts DEVSEL# (claim the transaction) before it issues any other slave response, i.e., TRDY#, STOP#, etc. After the SIO asserts DEVSEL#, it does not negate DEVSEL# until the same edge that the master uses to negate the final IRDY#.

It is expected that most (perhaps all) PCI target devices will be able to complete a decode and assert DEVSEL# within 1 or 2 clocks of FRAME#. Since the SIO subtractively decodes all unclaimed PCI cycles (except configuration cycles), it provides a configuration option to pull in (by 1 or 2 clocks) the edge when the SIO samples DEVSEL#. This allows faster access to the expansion bus. Use of such an option is limited by the slowest positive decode agent on the bus. This is described in more detail in Section 5.5.1.4, Subtractively Decoded Cycles to ISA.

As a PCI master, the SIO waits for 5 PCICLKs after the assertion of FRAME# for a slave to assert DEVSEL#. If the SIO does not receive DEVSEL# in this time, it will master-abort the cycle. See Section 5.1.3.1, SIO as MasterMaster-Initiated Termination, for further details.

5.1.2.3 Basic PCI Read Cycles (I/O and Memory)

As a PCI master, the SIO only performs memory read transfers (i.e. I/O read transfers are not supported). When reading data from PCI memory, the SIO requests a maximum of 8 bytes via a two data phase burst read cycle to fill its internal 8 byte line buffer. If the line buffer is programmed for single transaction mode, fewer bytes are requested (refer to Section 5.6.1, DMA/ISA Master Line Buffer). Read cycles from PCI memory are generated on behalf of ISA masters and DMA devices.

As a PCI slave, the SIO responds to both I/O read and memory read transfers. For multiple read transactions, the SIO always target-terminates after the first data read transaction by asserting STOP# and TRDY# at the end of the first data phase. For single read transactions, the SIO finishes the cycle in a normal fashion, by asserting TRDY# without asserting STOP#.

5.1.2.4 Basic PCI Write Cycles (I/O and Memory)

As a PCI master, the SIO generates a PCI memory write cycle when it decodes an ISA-originated/PCI-bound memory write cycle. I/O write cycles are never initiated by the SIO. When writing data to PCI memory, the SIO writes a maximum of 4 bytes via a single data transaction write cycle. If the SIO's internal ISA master/DMA line buffer is programmed for single transaction mode, fewer bytes will be generated (refer to Section 5.6.1, DMA/ISA Master Line Buffer). In either case, only one data transaction will be performed. Cycles to PCI memory are generated on behalf of ISA masters, DMA devices, and the SIO when the SIO needs to flush the ISA master/DMA line buffer.

As a PCI master, the SIO drives the AD0 and AD1 signals low during the address phase of the cycle. This is done to indicate to the slave that the address will increment during the transfer. If there is no response on the PCI Bus, the SIO will master-abort due to the DEVSEL# time out.

As a PCI slave, the SIO will respond to both I/O write and memory write transfers. For multiple write transactions, the SIO will always target-terminate after the first data write transaction by asserting STOP# and TRDY# at the end of the first data phase. For single write transactions, the SIO will finish the cycle normally by asserting TRDY# without asserting STOP#.

5.1.2.5 Configuration Cycles

The configuration read or write command defined by the bus control signals C/BE[3:0]# is used to configure the SIO. During the address phase of the configuration read or write command, the SIO will sample its IDSEL (ID select). If IDSEL is active and AD[1:0] are both zero, the SIO generates DEVSEL#. Otherwise, the cycle is ignored by the SIO. During the configuration cycle address phase, bits AD[7:2] and C/BE[3:0]# are used to select particular bytes within a configuration register. Note that IDSEL is normally a "don't care" except during the address phase of a transaction.

NOTE:
An unclaimed configuration cycle is never forwarded to the ISA Bus.

5.1.2.6 Interrupt Acknowledge Cycle

The interrupt acknowledge command is a single byte read implicitly addressed to the SIO's interrupt controller. The address bits are logical "don't cares" during the address phase and the byte enables will indicate to the SIO an 8-bit interrupt vector is to be returned on AD[7:0]. The SIO converts this single cycle transfer into two cycles that the internal 8259 pair can respond to (see Section 5.8, Interrupt Controller). The SIO will hold the PCI Bus in wait states until the 8 bit interrupt vector is returned.

SIO responses to an interrupt acknowledge cycle can be disabled by setting bit 5 in the PCI Control Register to a 0. However, if disabled, the SIO will still respond to accesses to the interrupt register set and allow poll mode functions.

5.1.2.7 Exclusive Access

The SIO marks itself locked anytime it is the slave of the access and LOCK# is sampled negated during the address phase. As a locked slave, the SIO responds to a master only when it samples LOCK# negated and FRAME# asserted. The locking master may negate LOCK# at the end of the last data phase. The SIO unlocks itself when FRAME# and LOCK# are both negated. The SIO will respond by asserting STOP# with TRDY# negated (retry) to all transactions when LOCK# is asserted during the address phase.

Locked cycles are never generated by the SIO.

5.1.2.8 PCI Special Cycle

When the SCE bit (bit 3) in the COM PCI configuration register (configuration offset 04h) is set to a "0", the SIO will ignore all PCI Special Cycles. When the SCE bit is set to a "1", the SIO will recognize PCI Special Cycles.

The only PCI Special Cycle currently recognized is the Stop Grant Special Cycle which is broadcast onto the PCI bus when an S-series processor enters the Stop Grant State. The SCE bit must be set to a "1" when the Stop Clock feature is being used.

5.1.3 TRANSACTION TERMINATION

The SIO supports both Master-initiated Termination as well as Target-initiated Termination.

5.1.3.1 SIO As Master—Master-Initiated Termination

The SIO supports two forms of master-initiated termination:

1. Normal termination of a completed transaction.
2. Abnormal termination due to no slave responding to the transaction (Abort).

Figure 5 shows the SIO performing master-abort termination. This occurs when no slave responds to the SIO's master transaction by asserting DEVSEL# within 5 PCICLK's after FRAME# assertion. This master-abort condition is abnormal and it indicates an error condition. The SIO will not retry the cycle. The Received Master-abort Status bit in the PCI Status Register will be set indicating that the SIO experienced a master-abort condition.

If an ISA master or the DMA is waiting for the PCI cycle to terminate (CHRDY negated), the master-abort condition will cause the SIO to assert CHRDY to terminate the ISA cycle. Note that write data will be lost and the read data will be all 1's at the end of the cycle. This is identical to the way an unclaimed cycle is handled on the "normally ready" ISA Bus. If the line buffer is the requester of the PCI transaction, the master-abort mechanism will end the PCI cycle, but no data will be transferred into or out of the line buffer. The line buffer will not be allowed to retry the cycle.

5.1.3.2 SIO As A Master—Response To Target-Initiated Termination

SIO's response as a master to target-termination:

1. For a target-abort, the SIO will not retry the cycle. If an ISA master or the DMA is waiting for the PCI cycle to complete (CHRDY negated), the target-abort condition will cause the SIO to assert CHRDY and end the cycle on the ISA Bus. If the ISA master or DMA device was reading from PCI memory, the SIO will drive all 1's on the data lines of the ISA Bus. The Received Target-abort Status bit in the PCI Status Register will be set indicating that the SIO experienced a target-abort condition.
2. If the SIO is retried as a master on the PCI Bus, it will remove it's request for 2 PCI clocks before asserting it again to retry the cycle.
3. If the SIO is disconnected as a master on the PCI Bus, it will respond very much as if it had been retried. The difference between retry and disconnect is that the SIO did not see any data phase for the retry. Disconnect may be generated by a PCI slave when the SIO is running a burst memory read cycle to fill it's 8-byte Line Buffer. In this case, the SIO may need to finish a multi-data phase transfer, and thus, must recycle through arbitration as required for a retry. An example of this is when the on-board DMA requests an 8-byte Line Buffer transfer and the SIO is disconnected before the Line Buffer is completely filled.

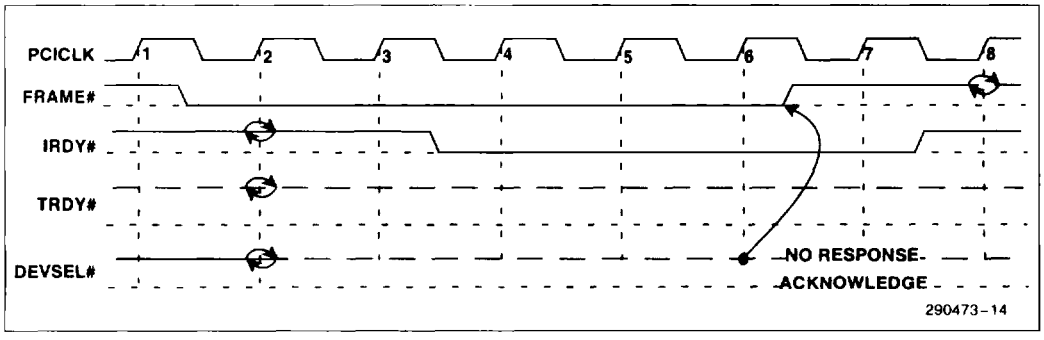

Figure 5. Master—Initiated Termination (Master-Abort)

5.1.3.3 SIO As A Target—Target-Initiated Termination

The SIO supports three forms of Target-initiated Termination:

Disconnect	Disconnect refers to termination requested because the SIO is unable to respond within the latency guidelines of the PCI specification. Note that this is not usually done on the first data phase.
Retry	Retry refers to termination requested because the target is currently in a state which makes it unable to process the transaction.
Abort	Abort refers to termination requested because the target will never be able to respond to the transaction.

The SIO will initiate Disconnect for PCI-originated/ISA-bound cycles after the first data phase due to incremental latency requirements. Since the SIO has only one Posted Write Buffer and every PCI to ISA incremental data phase will take longer than the specified 8 clocks, the SIO will always terminate burst cycles with a disconnect protocol. An example of this is when the SIO receives a burst memory write. Since the SIO only has one Posted Write Buffer, the transaction will automatically be disconnected after the first data phase.

The SIO will retry PCI masters:

1. For memory write cycles when the posted write buffer is full.
2. When the pending PCI cycle initiates some type of buffer management activity.
3. When the SIO is locked as a resource and a PCI master tries to access the SIO without negating the LOCK# signal in the address phase.
4. When the ISA Bus is occupied by an ISA master or DMA.

Target-abort is issued by the SIO when the internal SIO registers are the target of a PCI master I/O cycle and more than one byte enable is active. Accesses to the BIOS Timer Register and the Scatter/Gather Descriptor Table Pointer Registers are exceptions to this rule. Accesses to the Scatter/Gather Descriptor Table Pointer Register must be 32-bits wide and accesses to the BIOS Timer Register must be 16- or 32-bits wide. These accesses will not result in a SIO target-abort. The SIO responds with a target-abort since the registers must be accessed as 8-bit quantities. Target-abort resembles a retry, although the SIO also negates DEVSEL# along with the assertion of STOP#. Bit 11 in the Device Status Register is set to a 1 when the SIO target-aborts.

5.1.4 BUS LATENCY TIME-OUT

5.1.4.1 Master Latency Timer

Because the SIO only bursts a maximum of two Dwords, the PCI master latency timer is not implemented.

5.1.4.2 Target Incremental Latency Mechanism

As a slave, the SIO supports the Incremental Latency Mechanism for PCI to ISA cycles. The PCI specification states that for multi-data phase PCI cycles, if the incremental latency from current data phase (N) to the next data phase (N+1) is greater than 8 PCICLK's, then the slave must manipulate TRDY# and STOP# to stop the transaction upon completion of the current data phase (N). Since all PCI-originated (SIO is a slave)/ISA-bound cycles will require greater than the stated 8 PCICLK's, the SIO will automatically terminate these cycles after the first data phase. Note that latency to the first data phase is not restricted by this mechanism.

5.1.5 PARITY SUPPORT

As a master, the SIO generates address parity for read and write cycles, and data parity for write cycles. As a slave, the SIO generates data parity for read cycles. The SIO does not check parity and does not generate SERR#.

PAR is the calculated parity signal. PAR is "even" parity and is calculated on 36 bits; the 32 AD[31:0] signals plus the 4 C/BE[3:0]# signals. "Even" parity means that the number of 1's within the 36 bits plus PAR are counted and the sum is always even. PAR is always calculated on 36 bits, regardless of the valid byte enables. PAR is only guaranteed to be valid one PCI clock after the corresponding address or data phase.

5.1.6 RESET SUPPORT

The PCIRST# pin acts as the SIO hardware reset pin.

During Reset

AD[31:0], C/BE[3:0]#, and PAR are always driven low by the SIO from the leading edge of PCIRST#. FRAME#, IRDY#, TRDY#, STOP#, DEVSEL#, MEMREQ#, FLSHREQ#, CPUGNT#, GNT0#/ SIOREQ#, and GNT1#/ RESUME# are tri-stated from the leading edge of PCIRST#.

GNT2# and GNT3# are tri-stated from the leading edge of PCIRST#.

After Reset

AD[31:0], C/BE[3:0]#, and PAR are always tri-stated from the trailing edge of PCIRST#. If the internal arbiter is enabled (CPUREQ# sampled high on the trailing edge of PCIRST#), the SIO will drive these signals low again (synchronously 2-5 PCICLKs later) until the bus is given to another master. If the internal arbiter is disabled (CPUREQ# sampled low on the trailing edge of PCIRST#), these signals remain tri-stated until the SIO is required to drive them valid as a master or slave.

FRAME#, IRDY#, TRDY#, STOP#, and DEVSEL# remain tri-stated until driven by the SIO as either a master or a slave. MEMREQ#, FLSHREQ#, CPUGNT#, GNT0#/SIOREQ#, and GNT1#/RESUME# are tri-stated until driven by the SIO.

GNT2# and GNT3# are tri-stated until driven by the SIO.

After PCIRST, MEMREQ# and FLSHREQ# are driven inactive asynchronously from PCIRST# inactive. CPUGNT#, GNT0#/SIOREQ#, and GNT1#/ RESUME# are driven based on the arbitration scheme and the asserted REQx#'s.

GNT2# and GNT3# are also driven based on the arbitration scheme and the asserted REQx#'s.

5.1.7 DATA STEERING

Data steering logic internal to the SIO provides the assembly/disassembly, copy up/copy down mechanism for cycles between the 32-bit PCI data bus and the 16-bit ISA Bus. The steering logic ensures that the correct bytes are steered to the correct byte lane and that multiple cycles are run where applicable.

5.2 PCI Arbitration Controller

The 82378 contains a PCI Bus arbiter that supports six PCI masters; the Host Bridge, SIO, and four other masters. The SIO's REQ#/GNT# lines are internal. The integrated arbiter can be disabled by asserting CPUREQ# during PCIRST# (see Section 5.2.7, Power-up Configuration). When disabled, the SIO's REQ#, GNT#, and RESUME# signals become visible for an external arbiter. The internal arbiter is enabled upon power-up.

The internal arbiter contains several features that contribute to system efficiency:

- Use of a RESUME# signal to re-enable a backed-off initiator in order to minimize PCI Bus thrashing when the SIO generates a retry (Section 5.2.4.1).
- A programmable timer to re-enable retried initiators after a programmable number of PCICLK's (Section 5.2.4.2).
- The CPU (host bridge) can be optionally parked on the PCI Bus (Section 5.2.5).
- A programmable PCI Bus lock or PCI resource lock function (Section 5.2.6).

The PCI arbiter is also responsible for control of the Guaranteed Access Time (GAT) mode signals (Section 5.2.3.2).

5.2.1 ARBITRATION SIGNAL PROTOCOL

The internal arbiter follows the PCI arbitration method as outlined in the *Peripheral Component Interconnect (PCI) Specification*. The SIO's arbiter is discussed in this section.

5.2.1.1 Back-To-Back Transactions

The SIO as a master does not generate fast back-to-back accesses since it does not know if it is accessing the same target.

The SIO as a target supports fast back-to-back transactions. Note that for back-to-back cycles, the SIO treats positively decoded accesses and subtractively decoded accesses as different targets. Therefore, masters can only run fast back-to-back cycles to positively decoded addresses or to subtractively decoded addresses. Fast back-to-back cycles must not mix positive and subtractive decoded addresses. See the address decoding section to determine what addresses the SIO positively decodes and subtractively decodes.

5.2.2 PRIORITY SCHEME

The PCI arbitration priority scheme is programmable through the PCI Arbiter Priority Control and Arbiter Priority Control Extension Register. The arbiter consists of four banks that can be configured for the six masters to be arranged in a purely rotating priority scheme, one of twenty-four fixed priority schemes, or a hybrid combination (Figure 6).

Note that SIOREQ#/SIOGNT# are SIO internal signals.

The PCI Arbiter Priority Control (PAPC) and PCI Arbiter Priority Control Extension Register bits are shown below:

PCI Arbiter Priority Control Register Bits (PAPC)

Bit	Description
7	Bank 3 Rotate Control
6	Bank 2 Rotate Control
5	Bank 1 Rotate Control
4	Bank 0 Rotate Control
3	Bank 2 Fixed Priority Mode select B
2	Bank 2 Fixed Priority Mode select A
1	Bank 1 Fixed Priority Mode select
0	Bank 0 Fixed Priority Mode select

PCI Arbiter Priority Control Extension Register Bits (ARBPRIX)

Bit	Description
7:1	Reserved. Read as 0
0	Bank 3 Fixed Priority Mode select

PAPC defaults to 04h and ARBPRIX to 00h at reset selecting fixed mode # 10 (Table 8) with the CPU the highest priority device guaranteeing access to BIOS.

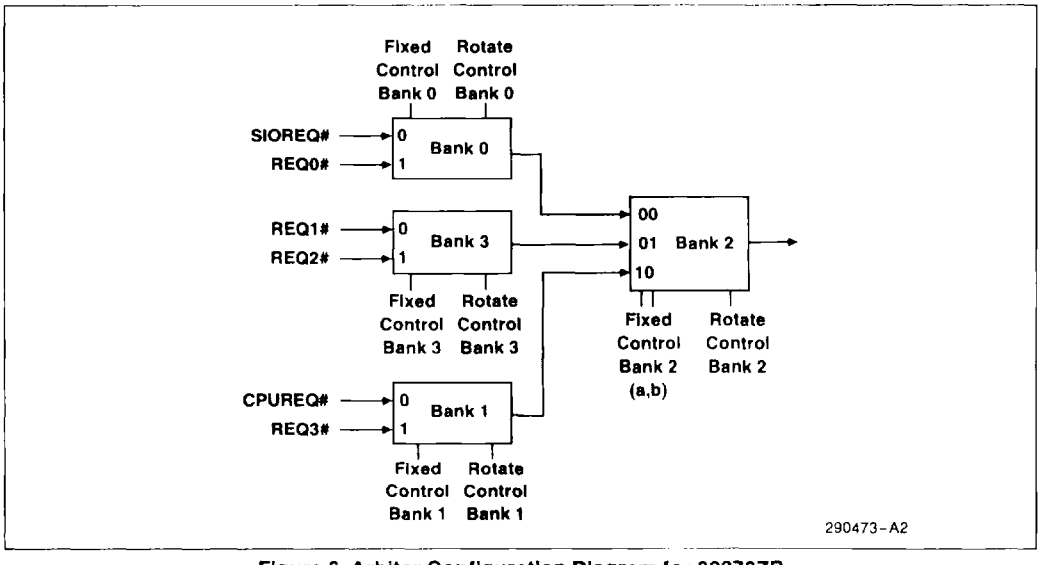

Figure 6. Arbiter Configuration Diagram for 82378ZB

5.2.2.1 Fixed Priority Mode

The 24 selectable fixed priority schemes are listed in Table 8.

Table 8. Fixed Priority Mode Bank Control Bits

Mode	Bank					Priority					
	3	2b	2a	1	0	Highest					Lowest
00	0	0	0	0	0	SIOREQ#	REQ0#	REQ2#	REQ3#	CPUREQ#	REQ1#
01	0	0	0	0	1	REQ0#	SIOREQ#	REQ2#	REQ3#	CPUREQ#	REQ#
02	0	0	0	1	0	SIOREQ#	REQ0#	REQ2#	REQ3#	REQ1#	CPUREQ#
03	0	0	0	1	1	REQ0#	SIOREQ#	REQ2#	REQ3#	REQ1#	CPUREQ#
04	0	0	1	0	0	CPUREQ#	REQ1#	SIOREQ#	REQ0#	REQ2#	REQ3#
05	0	0	1	0	1	CPUREQ#	REQ1#	REQ0#	SIOREQ#	REQ2#	REQ3#
06	0	0	1	1	0	REQ1#	CPUREQ#	SIOREQ#	REQ0#	REQ2#	REQ3#
07	0	0	1	1	1	REQ1#	CPUREQ#	REQ0#	SIOREQ#	REQ2#	REQ3#
08	0	1	0	0	0	REQ2#	REQ3#	CPUREQ#	REQ1#	SIOREQ#	REQ0#
09	0	1	0	0	1	REQ2#	REQ3#	CPUREQ#	REQ1#	REQ0#	SIOREQ#
0A	0	1	0	1	0	REQ2#	REQ3#	REQ1#	CPUREQ#	SIOREQ#	REQ0#
0B	0	1	0	1	1	REQ2#	REQ3#	REQ1#	CPUREQ#	REQ0#	SIOREQ#
0C-0F	0	1	1	x	x	Reserved					
10	1	0	0	0	0	SIOREQ#	REQ0#	REQ3#	REQ2#	CPUREQ#	REQ1#
11	1	0	0	0	1	REQ0#	SIOREQ#	REQ3#	REQ2#	CPUREQ#	REQ1#
12	1	0	0	1	0	SIOREQ#	REQ0#	REQ3#	REQ2#	REQ1#	CPUREQ#
13	1	0	0	1	1	REQ0#	SIOREQ#	REQ3#	REQ2#	REQ1#	CPUREQ#
14	1	0	1	0	0	CPUREQ#	REQ1#	SIOREQ#	REQ0#	REQ3#	REQ2#
15	1	0	1	0	1	CPUREQ#	REQ1#	REQ0#	SIOREQ#	REQ3#	REQ2#
16	1	0	1	1	0	REQ1#	SPUREQ#	SIOREQ#	REQ0#	REQ3#	REQ2#
17	1	0	1	1	1	REQ1#	CPUREQ#	REQ0#	SIOREQ#	REQ3#	REQ2#
18	1	1	0	0	0	REQ3#	REQ2#	CPUREQ#	REQ1#	SIOREQ#	REQ0#
19	1	1	0	0	1	REQ3#	REQ2#	CPUREQ#	REQ1#	REQ0#	SIOREQ#
1A	1	1	0	1	0	REQ3#	REQ2#	REQ1#	CPUREQ#	SIOREQ#	REQ0#
1B		1	0	1	1	REQ3#	REQ2#	REQ1#	CPUREQ#	REQ0#	SIOREQ#
1C-1F	1	1	1	x	x	Reserved					

2

The fixed bank control bit(s) selects which requester is the highest priority device within that particular bank. For fixed priority mode, bits[7:4] of the PAPC Register and bit zero of ARBPRIX must be 0's (rotate mode disabled).

The selectable fixed priority schemes provide 24 of the 64 possible fixed mode permutations possible for the six masters.

5.2.2.2 Rotating Priority Mode

When any bank rotate control bit is set to a one, that particular bank rotates between the requesting inputs. Any or all banks can be set in rotate mode. If all four banks are set in rotate mode, the six supported masters are all rotated and the arbiter is in a pure rotating priority mode. If, within a rotating bank, the highest priority device (a) does not have an active request, the lower priority device (b or c) will be granted the bus. However, this does not change the rotation scheme. When the bank toggles, device b is the highest priority. Because of this, the maximum latency a device can encounter is two complete rotations.

5.2.2.3 Mixed Priority Mode

Any combination of fixed priority and rotate priority modes can be used in different arbitration banks to achieve a specific arbitration scheme.

5.2.2.4 Locking Masters

When a master acquires the LOCK# signal, the arbiter gives that master highest priority until the LOCK# signal is negated and FRAME# is negated. This ensures that a master that locked a resource will eventually be able to unlock that same resource.

5.2.3 MEMREQ#, FLSHREQ#, AND MEMACK# PROTOCOL

Before an ISA master or the DMA can be granted the PCI Bus, it is necessary that all PCI system posted write buffers be flushed (including the SIO's Posted Write Buffer). Also, since the ISA originated cycle could access memory on the host bridge, it's possible that the ISA master or the DMA could be held in wait states (via IOCHRDY) waiting for the host bridge arbitration for longer than the 2.5 μs ISA specification. The SIO has an optional mode called the Guaranteed Access Time Mode (GAT) that ensures that this timing specification is not violated. This is accomplished by delaying the ISA REQ# signal to the requesting master or DMA until the ISA Bus, PCI Bus, and the System Memory Bus are arbitrated for and owned.

Three PCI sideband signals, MEMREQ#, FLSHREQ#, and MEMACK# are used to support the System Posted Write Buffer Flushing and Guaranteed Access Time mechanisms. The MEMACK# signal is the common acknowledge signal for both mechanisms. Note that when MEMREQ# is asserted, FLSHREQ# is also asserted. Table 9 shows the relationship between MEMREQ# and FLSHREQ#:

Table 9. FLSHREQ#, MEMREQ#

FLSHREQ#	MEMREQ#	Meaning
1	1	Idle
0	1	Flush buffers pointing towards PCI to avoid ISA deadlock
1	0	Reserved
0	0	GAT mode, Guarantee PCI Bus immediate access to main memory

5.2.3.1 Flushing the System Posted Write Buffers

Once an ISA master or the DMA begins a cycle on the ISA Bus, the cycle can not be backed-off. It can only be held in wait states via IOCHRDY. In order to know the destination of ISA master cycles, the cycle needs to begin. However, after the cycle has started, no other device can intervene and gain ownership of the ISA Bus until the cycle has completed and arbitration is performed. A potential deadlock condition exists when an ISA originated cycle to the PCI Bus finds the PCI target inaccessible due to an interacting event that also requires the ISA Bus. To avoid this potential deadlock, all PCI posted write buffers in the system must be disabled and flushed before DACK can be returned. The buffers must remain disabled while the ISA Bus is occupied by an ISA master or the DMA.

When an ISA master or the DMA requests the ISA Bus, the SIO asserts FLSHREQ#. FLSHREQ# is an indication to the system to flush all posted write buffers pointing towards the PCI Bus. The SIO also flushes it's own Posted Write Buffer. Once the posted write buffers have been flushed and disabled, the system asserts MEMACK#. Once the SIO receives the MEMACK# acknowledgment signal, it asserts the DACK signal giving the requesting master the bus. FLSHREQ# stays active as long as the ISA master or DMA owns the ISA Bus.

5.2.3.2 Guaranteed Access Time Mode

Guaranteed Access Time (GAT) Mode is enabled/disabled via the PCI Arbiter Control Register. When this mode is enabled, the MEMREQ# and MEMACK# signals are used to guarantee that the ISA 2.5 μs IOCHRDY specification is not violated.

When an ISA master or DMA slave requests the ISA Bus (DREQ# active), the ISA Bus, the PCI Bus, and the memory bus must be arbitrated for and all three must be owned before the ISA master or DMA slave is granted the ISA Bus. After receiving the DREQ# signal from the ISA master or DMA slave, MEMREQ# and FLSHREQ# are asserted (FLSHREQ# is driven active, regardless of GAT mode being enabled or disabled). MEMREQ# is a request for direct access to main memory. MEMREQ# and FLSHREQ# will be asserted as long as the ISA master or the DMA owns the ISA Bus. When MEMACK# is received by the SIO (all posted write buffers are flushed and the memory bus is dedicated to the PCI interface), it will request the PCI Bus. When it is granted the PCI Bus, it asserts the DACK signal releasing the ISA Bus to the requesting master or the DMA.

The use of MEMREQ#, FLSHREQ#, and MEMACK# does not guarantee functionality with ISA masters that don't acknowledge IOCHRDY. These signals just guarantee the IOCHRDY inactive specification.

NOTE:
Usage of an external arbiter in GAT mode will require special logic in the arbiter.

2

5.2.4 RETRY THRASHING RESOLVE

When a PCI initiator's access is retried, the initiator releases the PCI Bus for a minimum of two PCI clocks and will then normally request the PCI Bus again. To avoid thrashing the bus with retry after retry, the PCI arbiter provides REQ# masking. The REQ# masking mechanism differentiates between SIO target retries and all other retries.

For initiators which were retried by the SIO as a target, the masked REQ# is flagged to be cleared upon RESUME# active. All other retries trigger the Master Retry Timer, if enabled. Upon expiration of this timer, the mask is cleared.

5.2.4.1 Resume Function (RESUME#)

The conditions under which the SIO forces a retry to a PCI master and will mask the REQ# are:

1. Any required buffer management
2. ISA Bus occupied by ISA master or DMA
3. The PCI to ISA Posted Write Buffer is full
4. The SIO is locked as a resource and LOCK# is asserted during the address process.

The RESUME# signal is pulsed whenever the SIO has retried a PCI cycle for one of the above reasons and that condition has passed. When RESUME# is asserted, the SIO will unmask the REQ#'s that are masked and flagged to be cleared by RESUME#.

If the internal arbiter is enabled, RESUME# is an internal signal. The RESUME# signal becomes visible as an output when the internal arbiter is disabled. This allows an external arbiter to optionally avoid retry thrashing associated with the SIO as a target. The RESUME# signal is asserted for one PCI clock.

5.2.4.2 Master Retry Timer

To re-enable a PCI master's REQ# which resulted in a retry to a slave other than the SIO, a SIO programmable Master Retry Timer has been provided. This timer can be programmed for 0 (disabled), 16, 32, or 64 PCICLKs. Once the SIO has detected that a PCI slave has forced a retry, the timer will be triggered and the corresponding master's REQ# will be masked. All subsequent PCI retries by this REQ# signal will by masked by the SIO. Expiration of this timer will unmask all of the masked requests. This timer has no effect on the request lines that have been masked due to a SIO retry.

If no other PCI masters are requesting the PCI Bus, all of the REQ#'s masked for the timer will be cleared and the timer will be reset. This is necessary to assist the host bridge in determining when to re-enable any disabled posted write buffers.

5.2.5 BUS PARKING

The SIO arbitration logic supplies a mechanism for PCI Bus parking. Parking is only allowed for the device which is tied to CPUREQ# (typically the system CPU). When bus parking is enabled, CPUGNT# will be asserted when no other agent is currently using or requesting the bus. This achieves the minimum PCI arbitration latency possible. Enabling of bus parking is achieved by programming the Arbiter Control Register. REQ0#, REQ1#, and the internal SIOREQ# are not allowed to park on the PCI Bus.

Upon assertion of CPUGNT# due to bus parking enabled and the PCI Bus idle, the CPU (or the parked agent) must ensure that AD[31:0], C/BE[3:0], and (one PCICLK later) PAR are driven. If bus parking is disabled, the SIO takes responsibility for driving the bus when it is idle.

5.2.6 BUS LOCK MODE

As an option, the SIO arbiter can be configured to run in Bus Lock Mode or Resource Lock Mode. The Bus Lock Mode is used to lock the entire PCI Bus. This may improve performance in some systems that frequently run quick read-modify-write cycles. Bus Lock Mode emulates the LOCK environment found in today's PC by restricting bus ownership when the PCI Bus is locked. With Bus Lock enabled, the arbiter recognizes a LOCK# being driven by any initiator and does not allow any other PCI initiator to be granted the PCI Bus until LOCK# and FRAME# are both negated indicating the master released lock. When Bus Lock is disabled, the default resource lock mechanism is implemented (normal resource lock) and a higher priority PCI initiator could intervene between the read and write cycles and run non-exclusive accesses to any unlocked resource.

5.2.7 POWER-UP CONFIGURATION

The SIO's arbiter is enabled if CPUREQ# is sampled high on the trailing edge of PCIRST#. When enabled, the arbiter is set in fixed priority mode 4 with CPU bus parking turned off. Fixed mode 4 guarantees that the CPU will be able to run accesses to the BIOS in order to configure the system, regardless of the state of the other REQ#'s. Note that the Host Bridge should drive CPUREQ# high during the trailing edge of PCIRST#. When the arbiter is enabled, the SIO acts as the central resource responsible for driving the AD[31:0], C/BE[3:0]#, and PAR signals when no one is granted the PCI Bus and the bus is idle. The SIO is always responsible for driving AD[31:0], C/BE[3:0]#, and PAR when it is granted the bus and as appropriate when it is the master of a transaction. After reset, if the arbiter is enabled, CPUGNT#, GNT0#, GNT1#, and the internal SIOGNT# will be driven based on the arbitration scheme and the asserted REQ#'s.

If an external arbiter is present in the system, the CPUREQ# signal should be tied low. When CPUREQ# is sampled low on the trailing edge of PCIRST#, the internal arbiter is disabled. When the internal arbiter is disabled, the SIO does not drive AD[31:0], C/BE[3:0]#, and PAR as the central resource. In this case, the SIO is only responsible for driving AD[31:0], C/BE[3:0]#, and PAR when it is granted the bus. If the SIO's arbiter is disabled, GNT0# becomes SIOREQ#, GNT1# becomes RESUME#, and REQ0# becomes SIOGNT#. This exposes the normally embedded SIO arbitration signals.

NOTE:
Usage of an external arbiter in GAT mode will require special logic in the arbiter.

5.3 ISA Interface

5.3.1 ISA INTERFACE OVERVIEW

The SIO incorporates a fully ISA Bus compatible master and slave interface. The SIO directly drives six ISA slots without external data or address buffers. The ISA interface also provides byte swap logic, I/O recovery support, wait-state generation, and SYSCLK generation.

The ISA interface supports the following types of cycles:

- PCI-initiated I/O and memory cycles to the ISA Bus.
- DMA compatible cycles between PCI memory and ISA I/O and between ISA I/O and ISA memory, DMA type "A", type "B", and type "F" cycles between PCI memory and ISA I/O.
- ISA Refresh cycles initiated by either the SIO or an external ISA master.
- ISA master-initiated memory cycles to the PCI Bus and ISA master-initiated I/O cycles to the internal SIO registers.

The refresh and DMA cycles are shown and described in Section 5.4.

5.3.2 SIO AS AN ISA MASTER

The SIO executes ISA cycles as an ISA master whenever a PCI initiated cycle is forwarded to the ISA Bus. The SIO also acts as an ISA master on behalf of DMA and refresh.

ISYSCLK is an internal 8 MHz clock.

5.3.3 SIO AS AN ISA SLAVE

The SIO operates as an ISA slave when:

- An ISA master accesses SIO internal registers.
- An ISA master accesses PCI memory on the PCI Bus.

5.3.3.1 ISA Master Accesses To SIO Registers

An ISA Bus master has access to SIO internal registers as shown in Table 19. An ISA master to SIO register cycle will always run as an 8-bit extended cycle (IOCHRDY will be held inactive until the cycle is completed).

Table 10. Arbitration Latency

Bus Condition	Arbitration Latency
Parked	0 PCICLKs for Agent 0, 2 PCICLKs for All Other
Not Parked	1 PCICLK for All Agents

5.3.3.2 ISA Master Accesses to PCI Resource

An ISA master can access PCI memory, but not I/O devices residing on the PCI Bus. The ISA/DMA address decoder determines which memory cycles should be directed towards the PCI Bus. During ISA master read cycles to the PCI Bus, the SIO will return all 1's if the PCI cycle is target-aborted or does not respond.

If the SIO is programmed for GAT mode, the SIO arbiter will not grant the ISA Bus before gaining ownership of both the PCI Bus and system memory. However, if the SIO is not programmed in this mode, the SIO does not need to arbitrate for the PCI Bus before granting the ISA Bus to the ISA master. For more details on the arbitration, refer to Section 5.2.2.

All cycles forwarded to a PCI resource will run as 16-bit extended cycles (i.e. IOCHRDY will be held inactive until the cycle is completed).

Because the ISA bus size is different from the PCI bus size, the data steering logic inside the SIO is responsible for steering the data to the correct byte lanes on both buses, and assembling/disassembling the data as necessary.

5.3.4 ISA MASTER TO ISA SLAVE SUPPORT

During ISA master cycles to ISA slaves, the SIO drives several signals to support the transfer:

BALE:

This signal is driven high while the ISA master owns the ISA Bus.

AEN:

This signal is driven low while the ISA master owns the ISA Bus.

SMEMR# and **SMEMW#:**

These signals are driven active by the SIO whenever the ISA master drives a memory cycle to an address below 1 Mb.

Utility Bus Buffer Control Signals and Chip Select Signals:

These signals are driven active as appropriate whenever an ISA master accesses devices on the Utility Bus. For more details, see Section 5.9.

Data Swap Logic:

The data swap logic inside the SIO is activated as appropriate to swap data between the even and odd byte lanes. This is discussed in further detail in Section 5.3.5.

5.3.5 DATA BYTE SWAPPING

The data swap logic is integrated in the SIO. For slaves that reside on the ISA Bus, data swapping is performed if the slave (I/O or memory) and ISA bus master (or DMA) sizes differ and the upper (odd) byte of data is being accessed. Table 11 shows when data swapping is provided during DMA. Table 12 shows when data swapping is provided during ISA master cycles to 8-bit ISA slaves.

Table 11. DMA Data Swap

DMA I/O Device Size	ISA Memory Slave Size	Swap	Comments I/O	⟷	Memory
8-Bit	8-Bit	No	SD[7:0]	⟷	SD[7:0]
8-Bit	16-Bit	Yes	SD[7:0]	⟷	SD[7:0]
			SD[7:0]	⟷	SD[15:8]
16-Bit	8-Bit	No		Not Supported	
16-Bit	16-Bit	No	SD[15:0]	⟷	SD[15:0]

The SIO monitors the SBHE# and SA0 signals to determine when to swap the data. The SIO ensures that the data is placed on the appropriate byte lane.

Table 12. 16-Bit Master to 8-Bit Slave Data Swap

SBHE#	SA0	SD[15:8]	SD[7:0]	Comments
0	0	Odd	Even	Word Transfer (data swapping not required)
0	1	Odd	Even	Byte Swap(1,2)
1	0	—	Even	Byte Transfer (data swapping not required)
1	1	—	—	Not Allowed

NOTES:
1. For ISA master read cycles, the SIO swaps the data from the lower byte to the upper byte.
2. For ISA master write cycles, the SIO swaps the data from the upper byte to the lower byte.

5.3.6 ISA CLOCK GENERATION

The SIO generates the ISA system clock (SYSCLK). SYSCLK is a divided down version of the PCICLK (see Table 13). The clock divisor value is programmed through the ISA Clock Divisor Register.

Table 13. SYSCLK Generation from PCICLK

PCICLK (MHz)	Divisor (Programmable)	SYSCLK (MHz)
25	3	8.33
33	4 (default)	8.33

NOTE:
For PCI frequencies less than 33 MHz (not including 25 MHz), a clock divisor value must be selected that ensures that the ISA Bus frequency does not violate the 6 MHz to 8.33 MHz SYSCLK specification.

5.3.7 WAIT STATE GENERATION

The SIO will add wait states to the following cycles, if IOCHRDY is sampled active low. Wait states will be added as long as IOCHRDY is low.

- During Refresh and SIO master cycles (not including DMA) to the ISA Bus.
- During DMA compatible transfers between ISA I/O and ISA memory only.

For ISA master cycles targeted for the SIO's internal registers or PCI memory, the SIO will always extend the cycle by driving IOCHRDY low until the transaction is complete.

The SIO will shorten the following cycles, if ZEROWS# is sampled active.

- During SIO master cycles (not including DMA) to 8-bit and 16-bit ISA memory.
- During SIO master cycles (not including DMA) to 8-bit ISA I/O only.

For ISA master cycles targeted for the SIO's internal registers or PCI memory, the SIO will not assert ZEROWS#.

NOTE:
If IOCHRDY and ZEROWS# are sampled active at the same time, IOCHRDY will take precedence and wait-states will be added.

5.3.8 I/O RECOVERY

The I/O recovery mechanism in the SIO is used to add additional recovery delay between PCI originated 8-bit and 16-bit I/O cycles to the ISA Bus. The SIO automatically forces a minimum delay of four SYSCLKs between back-to-back 8- and 16-bit I/O cycles to the ISA Bus. This delay is measured from the rising edge of the I/O command (IOR# or IOW#) to the falling edge of the next BALE. If a delay of greater than four SYSCLKs is required, the ISA I/O Recovery Time Register can be programmed to increase the delay in increments of SYSCLKs. No additional delay is inserted for back-to-back I/O "sub cycles" generated as a result of byte assembly or disassembly .

5.4 DMA Controller

5.4.1 DMA CONTROLLER OVERVIEW

The DMA circuitry incorporates the functionality of two 82C37 DMA controllers with seven independently programmable channels (Channels 0–3 and Channels 5–7). DMA Channel 4 is used to cascade the two controllers and will default to cascade mode in the DMA Channel Mode (DCM) Register. In addition to accepting requests from DMA slaves, the DMA controller also responds to requests that are initiated by software. Software may initiate a DMA service request by setting any bit in the DMA Channel Request Register to a 1. The DMA controller for Channels 0–3 is referred to as "DMA-1" and the controller for Channels 4–7 is referred to as "DMA-2".

Each DMA channel may be programmed for 8- or 16-bit DMA device size and ISA-compatible, Type "A", Type "B", or Type "F" transfer timing. Each DMA channel defaults to the compatible settings for DMA device size: channels [3:0] default to 8-bit, count-by-bytes transfers, and channels [7:5] default to 16-bit, count-by-words (address shifted) transfers. The SIO provides the timing control and data size translation necessary for the DMA transfer between the PCI and the ISA Bus. ISA-compatible is the default transfer timing.

Full 32-bit addressing is supported as an extension of the ISA-compatible specification. Each channel includes a 16-bit ISA compatible Current Register which holds the 16 least-significant bits of the 32-bit address, and an ISA compatible Low Page Register which contains the eight second most significant bits. An additional High Page Register contains the eight most significant bits of the 32-bit address.

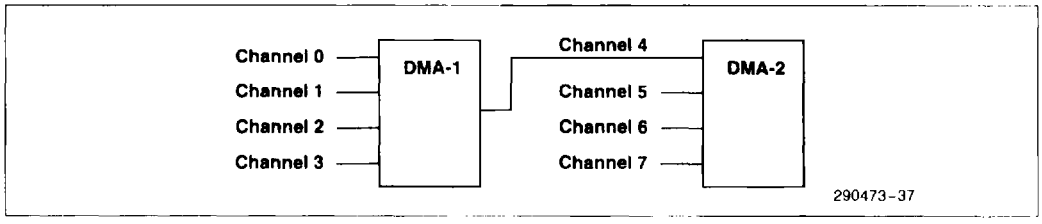

Figure 7. Internal DMA Controller

The DMA controller also features refresh address generation, and auto-initialization following a DMA termination.

The DMA controller receives commands from the ISA Bus arbiter to perform either DMA cycles or refresh cycles. The arbiter determines which requester from among the requesting DMA slaves, the PCI Bus, and refresh should have the bus.

The DMA controller is at any time either in master mode or slave mode. In master mode, the DMA controller is either servicing a DMA slave's request for DMA cycles, or allowing a 16-bit ISA master to use the bus via a cascaded DREQ signal. In slave mode, the SIO monitors both the ISA Bus and the PCI, decoding and responding to I/O read and write commands that address its registers.

Note that a DMA device (I/O device) is always on the ISA Bus, but the memory device is either on the ISA or PCI Bus. If the memory is decoded to be on the ISA Bus, then the DMA cycle will run as a compatible cycle. If the memory is decoded to be on the PCI Bus, the cycle can run as compatible, "A", "B", or "F" type. The ISA controller will not drive a valid address for type "A", "B", and "F" DMA transfers on the ISA Bus.

When the SIO is running a DMA cycle in compatible timing mode, the SIO will drive the MEMR# or MEMW# strobes if the address is less than 16 MBytes (00000000h-00FFFFFFh). These memory strobes will be generated regardless of whether the cycle is decoded for PCI or ISA memory. The SMEMR# and SMEMW# will be generated if the address is less than 1 MBytes (00000000h-000FFFFFh). If the address is greater than 16 MBytes (01000000h-FFFFFFFFh) the MEMR# or MEMW# strobe will not be generated in order to avoid aliasing problems. For type "A", "B", and "F" timing mode DMA cycles, the SIO will only generate the MEMR# or MEMW# strobe when the address is decoded for ISA memory. When this occurs, the cycle converts to compatible mode timing.

During DMA cycles, the ISA controller drives AEN high to prevent the I/O devices from misinterpreting the DMA cycle as a valid I/O cycle. The BALE signal is also driven high during DMA cycles. Also, during DMA memory read cycles to the PCI Bus, the SIO will return all 1's to the ISA Bus if the PCI cycle is either target-aborted or does not respond.

Further details can be found in the 82C37 data sheet.

5.4.2 DMA TIMINGS

ISA Compatible timing is provided for DMA slave devices. Three additional timings are provided for I/O slaves capable of running at faster speeds. These timings are referred to as Type "A", Type "B", and Type "F".

5.4.2.1 Compatible Timing

Compatible timing runs at 8 SYSCLKs during the repeated portion of a Block or Demand mode transfer.

5.4.2.2 Type "A" Timing

Type "A" timing is provided to allow shorter cycles to PCI memory. Type "A" timing runs at 6 SYSCLKs (720 ns/cycle) during the repeated portion of a block or demand mode transfer. This timing assumes an 8.33 MHz SYSCLK. Type "A" timing varies from compatible timing primarily in shortening the memory operation to the minimum allowed by system memory. The I/O portion of the cycle (data setup on write, I/O read access time) is the same as with compatible cycles. The actual active command time is shorter, but it is expected that the DMA devices which provide the data access time or write data setup time should not require excess IOR# or IOW# command active time. Because of this, most ISA DMA devices should be able to use type "A" timing.

5.4.2.3 Type "B" Timing

Type "B" timing is provided for 8-/16-bit ISA DMA devices which can accept faster I/O timing. Type "B" only works with PCI memory. Type "B" timing runs at 5 SYSCLKs (600 ns/cycle) during the repeated portion of a Block or Demand mode transfer. This timing assumes an 8.33 MHz SYSCLK. Type "B" timing requires faster DMA slave devices than compatible timing in that the cycles are shortened so that the data setup time on I/O write cycles is shortened and the I/O read access time is required to be faster. Some of the current ISA devices should be able to support type "B" timing, but these will probably be more recent designs using relatively fast technology.

5.4.2.4 Type "F" Timing

Type "F" timing provides high performance DMA transfer capability. These transfers are mainly for fast I/O devices (i.e. IDE devices). Type "F" timing runs at 3 SYSCLKs (360 ns/cycle) during the repeated portion of a Block or Demand mode transfer.

5.4.2.5 DREQ and DACK# Latency Control

The SIO DMA arbiter maintains a minimum DREQ to DACK# latency on DMA channels programmed to operate in compatible timing mode. This is to support older devices such as the 8272A. The DREQs are delayed by eight SYSCLKs prior to being seen by the arbiter logic. Software requests will not have this minimum request to DACK# latency.

5.4.3 ISA BUS/DMA ARBITRATION

The ISA Bus arbiter evaluates requests for the ISA Bus coming from several different sources. The DMA unit, the refresh counter, and the PCI Bus (primarily the Host CPU) may all request access to the ISA Bus. Additionally, 16-bit ISA masters may request the bus through a cascaded DMA channel.

The SIO ISA arbiter uses a three-way rotating priority arbitration method. At each level, the devices which are considered equal are given a rotating priority. On a fully loaded bus, the order in which the devices are granted bus access is independent of the order in which they assert a bus request. This is because devices are serviced based on their position in the rotation. The arbitration scheme assures that DMA channels access the bus with minimal latency.

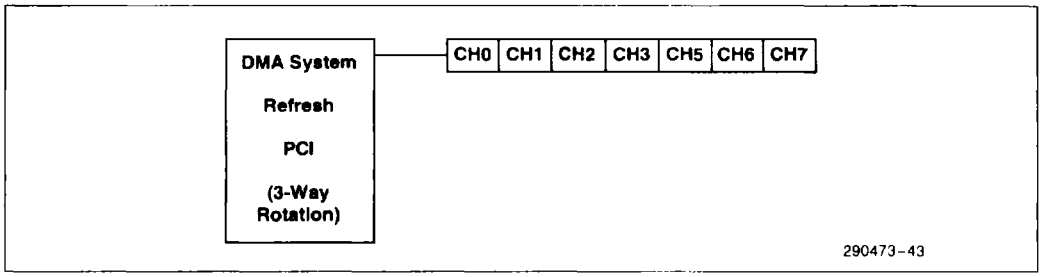

Figure 8. ISA Arbiter with DMA in Fixed Priority

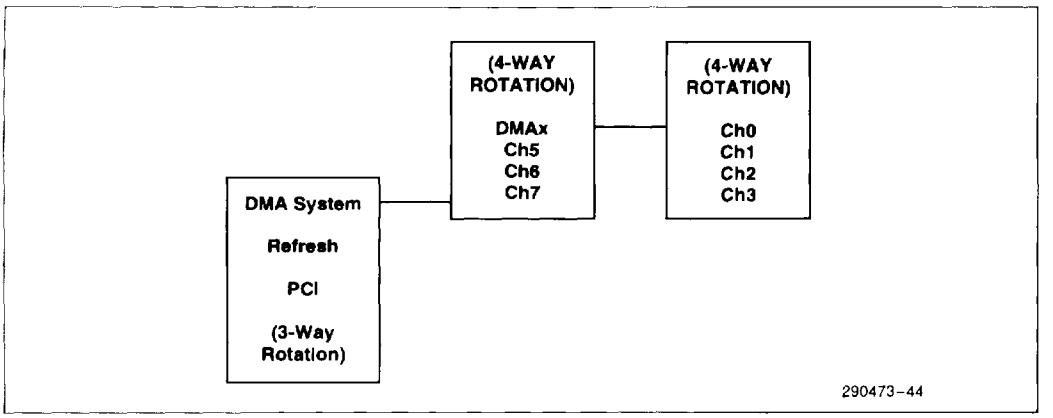

Figure 9. ISA Arbiter with DMA in Rotating Priority

2

5.4.3.1 Channel Priority

For priority resolution the DMA consists of two logical channel groups: channels 0–3 and channels 4–7 (see Figure 7). Each group may be in either fixed or rotate mode, as determined by the DMA Command Register.

For prioritization purposes, the source of the DMA request is transparent. DMA I/O slaves normally assert their DREQ line to arbitrate for DMA service. However, a software request for DMA service can be presented through each channel's DMA Request Register. A software request is subject to the same prioritization as any hardware request. Please see the detailed register description in Section 4.2.4 for Request Register programming information.

Fixed Priority

The initial fixed priority structure is as follows:

Table 14. Initial Fixed Priority Structure

High Priority . . . . . Low Priority
(0, 1, 2, 3), 5, 6, 7

The fixed priority ordering is 0, 1, 2, 3, 5, 6, and 7. In this scheme, Channel 0 has the highest priority, and channel 7 has the lowest priority. Channels [3:0] of DMA-1 assume the priority position of Channel 4 in DMA-2, thus taking priority over channels 5, 6, and 7.

Rotating Priority

Rotation allows for "fairness" in priority resolution. The priority chain rotates so that the last channel serviced is assigned the lowest priority in the channel group (0–3, 5–7).

Channels 0–3 rotate as a group of 4. They are always placed between Channel 5 and Channel 7 in the priority list.

Channel 5–7 rotate as part of a group of 4. That is, channels (5–7) form the first three positions in the rotation, while channel group (0–3) comprises the fourth position in the arbitration.

Table 15 demonstrates rotation priority.

Table 15. Rotating Priority Example

Programmed Mode	Action	Priority High .....Low
Group (0–3) is in Rotation Mode Group (4–7) is in Fixed Mode	1) Initial Setting	(0, 1, 2, 3), 5, 6, 7
	2) After Servicing Channel 2	(3, 0, 1, 2), 5, 6, 7
	3) After Servicing Channel 3	(0, 1, 2, 3), 5, 6, 7
Group (0–3) is in Rotation Mode Group (4–7) is in Rotation Mode	1) Initial Setting	(0, 1, 2, 3), 5, 6, 7
	2) After Servicing Channel 0	5, 6, 7, (1, 2, 3, 0)
	3) After Servicing Channel 5	6, 7, (1, 2, 3, 0), 5
	4) After servicing Channel 6	7, (1, 2, 3, 0), 5, 6
	5) After servicing Channel 7	(1, 2, 3, 0), 5, 6, 7

NOTE:
The first servicing of channel 0 caused double rotation.

5.4.3.2 DMA Preemption In Performance Timing Modes

A DMA slave device that is not programmed for compatible timing will be preempted from the bus by another device that requests use of the bus. This will occur, regardless of the priority of the pending request. For DMA devices not using compatible timing mode, the DMA controller stops the DMA transfer and releases the bus within 32 BCLK (4 μs) of a preemption. Upon the expiration of the 4 μs timer, the DACK will be inactivated after the current DMA cycle has completed. The bus will then be arbitrated for and granted to the highest priority requester. This feature allows flexibility in programming the DMA for long transfer sequences in a performance timing mode while guaranteeing that vital system services such as refresh are allowed access to the ISA Bus.

The 4 μs timer is not used in compatible timing mode. It is only used for DMA channels programmed for Type "A", Type "B", or Type "F" timing. It is also not used for 16-bit ISA masters cascaded through the DMA DREQ lines.

If the DMA channel that was preempted by the 4 μs timer was operating in Block Mode, an internal bit will be set so that the channel will be arbitrated for again, independent of the state of DREQ.

5.4.3.3. Arbitration during Non-Maskable Interrupts

If a non-maskable interrupt (NMI) is pending, and the CPU is requesting the bus, then the DMA controller will be bypassed each time it comes up for rotation. This will give the CPU the bus bandwidth it requires to process the interrupt as fast as possible.

5.4.3.4 Programmable Guaranteed Access Time Mode (GAT Mode)

The PCI Arbiter Register contains a bit for configuring the SIO in "Guaranteed Access Time Mode" (GAT Mode). This mode guarantees that the 2.5 μs CHRDY time-out specification for ISA masters running cycles to PCI will not be exceeded. When an ISA master or DMA slave arbitrates for the ISA Bus, and the SIO is configured in Guaranteed Access Time Mode, the MEMREQ# pin will be asserted by the PCI arbiter in order to gain ownership of main memory. The arbitration for the PCI and then the main memory bus must be completed prior to granting the DACK# to the ISA master or DMA slave. A MEMACK# signal to the SIO indicates that the SIO now owns main memory and can grant the DACK# to the ISA master or DMA slave. A detailed description is contained in Section 5.2.3.2.

5.4.4 REGISTER FUNCTIONALITY

Please see Section 4.2 for detailed information on register programming, bit definitions, and default values/functions of the DMA registers after a PCIRST#.

2

DMA Channel 4 is used to cascade the two DMA controllers together and should not be programmed for any mode other than cascade. The DMA Channel Mode Register for channel 4 will default to cascade mode. Special attention should also be take when programming the Command and Mask Registers as related to channel 4.

5.4.4.1 Address Compatibility Mode

Whenever the DMA is operating in address compatibility mode, the addresses do not increment or decrement through the High and Low Page Registers, and the High Page Register is set to 00h. This is compatible with the 82C37 and Low Page Register implementation used in the PC/AT. This mode is set when any of the lower three address bytes of a channel are programmed. If the upper byte of a channel's address is programmed last, the channel will go into extended address mode. In this mode, the high byte may be any value and the address will increment or decrement through the entire 32-bit address.

After PCIRST# is negated, all channels will be set to address compatibility mode. The DMA Master Clear command will also reset the proper channels to address compatibility mode.

5.4.4.2 Summary of DMA Transfer Sizes

Table 16 lists each of the DMA device transfer sizes. The column labeled "Current Byte/Word Count Register" indicates that the register contents represents either the number of bytes to transfer or the number of 16-bit words to transfer. The column labeled "Current Address Increment/Decrement" indicates the number added to or taken from the Current Address register after each DMA transfer cycle. The DMA Channel Mode Register determines if the Current Address Register will be incremented or decremented.

5.4.4.3 Address Shifting when Programmed for 16-Bit I/O Count by Words

To maintain compatibility with the implementation of the DMA in the PC/AT which used the 82C37, the DMA will shift the addresses when the DMA Channel Extended Mode Register is programmed for, or defaulted to, transfers to/from a 16-bit device count-by-words. Note that the least significant bit of the Low Page Register is dropped in 16-bit shifted mode. When programming the Current Address Register when the DMA channel is in this mode, the Current Address must be programmed to an even address with the address value shifted right by one bit. The address shifting is shown in Table 17.

5.4.4.4 Autoinitialize

By programming a bit in the DMA Channel Mode Register, a channel may be set up as an autoinitialize channel. During Autoinitialize initialization, the original values of the Current Page, Current Address and Current Byte/Word Count Registers are automatically restored from the Base Page, Address, and Byte/Word Count Registers of that channel following TC. The Base Registers are loaded simultaneously with the Current Registers by the microprocessor and remain unchanged throughout the DMA service. The mask bit is not set when the channel is in autoinitialize. Following Autoinitialize, the channel is ready to perform another DMA service, without CPU intervention, as soon as a valid DREQ is detected.

Table 16. DMA Transfer Size

DMA Device Date Size and Word Count	Current Byte/Word Count Register	Current Address Increment/Decrement
8-Bit I/O, Count by Bytes	Bytes	1
16-Bit I/O, Count by Words (Address Shifted)	Words	1
16-Bit I/O, Count by Bytes	Bytes	2

Table 17. Address Shifting in 16-Bit I/O DMA Transfers

Output Address	8-Bit I/O Programmed Address	16-Bit I/O Programmed Address (Shifted)	16-Bit I/O Programmed Address (No Shift)
A0	A0	0	A0
A[16:1]	A[16:1]	A[15:0]	A[16:1]
A[31:17]	A[31:17]	A[31:17]	A[31:17]

NOTE:
The least significant bit of the Low Page Register is dropped in 16-bit shifted mode.

5.4.5 SOFTWARE COMMANDS

There are three additional special software commands which can be executed by the DMA controller. The three software commands are:

1. Clear Byte Pointer Flip-Flop
2. Master Clear
3. Clear Mask Register

They do not depend on any specific bit pattern on the data bus.

5.4.5.1 Clear Byte Pointer Flip-Flop

This command is executed prior to writing or reading new address or word count information to/from the DMA controller. This initializes the flip-flop to a known state so that subsequent accesses to register contents by the microprocessor will address upper and lower bytes in the correct sequence.

When the Host CPU is reading or writing DMA registers, two Byte Pointer Flip-Flops are used; one for channels 0–3 and one for channels 4–7. Both of these act independently. There are separate software commands for clearing each of them (0Ch for channels 0–3, 0D8h for channels 4–7).

5.4.5.2 DMA Master Clear

This software instruction has the same effect as the hardware reset. The Command, Status, Request, and Internal First/Last Flip-Flop Registers are cleared and the Mask Register is set. The DMA controller will enter the idle cycle.

There are two independent master clear commands, 0Dh which acts on channels 0–3, and 0DAh which acts on channels 4–7.

5.4.5.3 Clear Mask Register

This command clears the mask bits of all four channels, enabling them to accept DMA requests. I/O port 00Eh is used for channels 0–3 and I/O port 0DCh is used for channels 4–7.

5.4.6 TERMINAL COUNT/EOP SUMMARY

This is a summary of the events that will happen as a result of a terminal count or external EOP when running DMA in various modes. (See Table 18.)

2

5.4.7 ISA REFRESH CYCLES

Refresh cycles are generated by two sources: the refresh controller inside the SIO component or by ISA bus masters other than the SIO. The ISA bus controller will enable the address lines SA[15:0] so that when MEMR# goes active, the entire ISA system memory is refreshed at one time. Memory slaves on the ISA Bus must not drive any data onto the data bus during the refresh cycle.

Counter 1 in the timer register set should be programmed to provide a request for refresh about every 15 μs.

Table 18. Terminal Count/EOP Summary Table

Conditions AUTOINIT	No		Yes	
Event				
Word Counter Expired	Yes	X	Yes	X
EOP Input	X	Asserted	X	Asserted
Result				
Status TC	set	set	set	set
Mask	set	set	—	—
SW Request	clr	clr	clr	clr
Current Register	—	—	load	load

NOTES:
load load current from base
— no change
X don't care
clr clear

5.4.8 SCATTER/GATHER DESCRIPTION

Scatter/Gather (S/G) provides the capability of transferring multiple buffers between memory and I/O without CPU intervention. In Scatter/Gather, the DMA can read the memory address and word count from an array of buffer descriptors, located in system memory (ISA or PCI), called the Scatter/Gather Descriptor (SGD) Table. This allows the DMA controller to sustain DMA transfers until all of the buffers in the SGD Table are transferred.

The S/G Command and Status Registers are used to control the operational aspect of S/G transfers. The SGD Table Pointer Register holds the address of the next buffer descriptor in the SGD Table.

The next buffer descriptor is fetched from the SGD Table by a DMA read transfer. DACK# will not be asserted for this transfer because the IO device is the SIO itself. The SIO will fetch the next buffer descriptor from either PCI memory or ISA memory, depending on where the SGD Table is located. If the SGD table is located in PCI memory, the memory read will use the line buffer to temporarily store the PCI read before loading it into the DMA S/G registers. The line buffer mode (8 byte or single transaction) for the S/G fetch operation will be the same as what is set for all DMA operations. If set in 8 byte mode, the SGD Table fetches will be PCI burst memory reads. The SGD Table PCI cycle fetches are subject to all types of PCI cycle termination (retry, disconnect, target-abort, master-abort). The fetched SGD Table data is subject to normal line buffer coherency management and invalidation. EOP will be asserted at the end of the complete link transfer.

To initiate a typical DMA Scatter/Gather transfer between memory and an I/O device, the following steps are required:

1. Software prepares a SGD Table in system memory. Each SGD is 8 bytes long and consists of an address pointer to the starting address and the transfer count of the memory buffer to be transferred. In any given SGD Table, two consecutive SGDs are offset by 8 bytes and are aligned on a 4-byte boundary.

   Each Scatter/Gather Descriptor for the linked list contains the following information:

   a. Memory Address (buffer start) 4 bytes

   b. Transfer Size (buffer size) 2 bytes

   c. End of Link List 1 bit (MSB)

2. Initialize the DMA Channel Mode and DMA Channel Extended Mode Registers with transfer specific information like 8-/16-bit I/O device, Transfer Mode, Transfer Type, etc.
3. Software provides the starting address of the SGD Table by loading the SGD Table Pointer Register.
4. Engage the Scatter/Gather function by writing a Start command to the S/G Command Register.
5. The Mask register should be cleared as the last step of programming the DMA register set. This is to prevent the DMA from starting a transfer with a partially loaded command description.
6. Once the register set is loaded and the channel is unmasked, the DMA will generate an internal request to fetch the first buffer from the SGD Table.

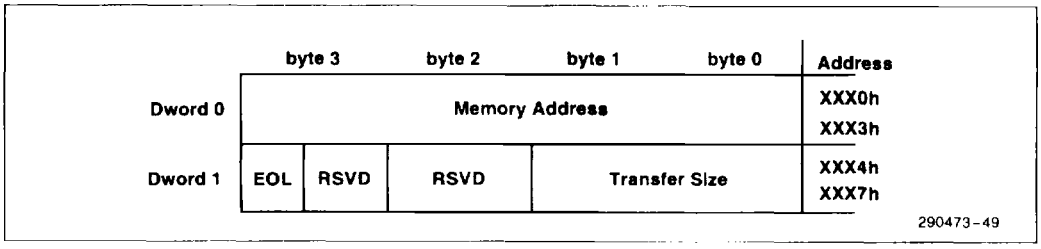

Figure 10. SGD Format

After the above steps are finished, the DMA will then respond to DREQ or software requests. The first transfer from the first buffer moves the memory address and word count from the Base register set to the Current register set. As long as S/G is active and the Base register set is not loaded and the last buffer has not been fetched, the channel will generate a request to fetch a reserve buffer into the Base register set. The reserve buffer is loaded to minimize latency problems going from one buffer to another. Fetching a reserve buffer has a lower priority than completing DMA transfers for the channel.

The DMA controller will terminate a Scatter/Gather cycle by detecting an End of List (EOL) bit in the SGD Table. After the EOL bit is detected, the channel transfers the buffers in the Base and Current register sets, if they are loaded. At terminal count the channel asserts EOP or IRQ13, depending on its programming and set the terminate bit in the S/G Status Register. If the channel asserted IRQ13, then the appropriate bit is set in the S/G Interrupt Status Register. The active bit in the S/G Status Register will be reset and the channel's Mask bit will be set.

5.5 Address Decoding

The SIO contains two address decoders; one to decode PCI master cycles and one to decode DMA/ISA master cycles. Two decoders are required to support the PCI and ISA Buses running concurrently. The PCI address decoder decodes the address from the multiplexed PCI address/data bus. The DMA/ISA master address decoder decodes the address from the ISA address bus for DMA and ISA master cycles. The address decoders determine how the cycle is handled.

5.5.1 PCI ADDRESS DECODER

PCI address decoding is always a function of AD[31:2]. The information contained in the two low order bits (AD[1:0]) varies for memory, I/O, and configuration cycles.

2

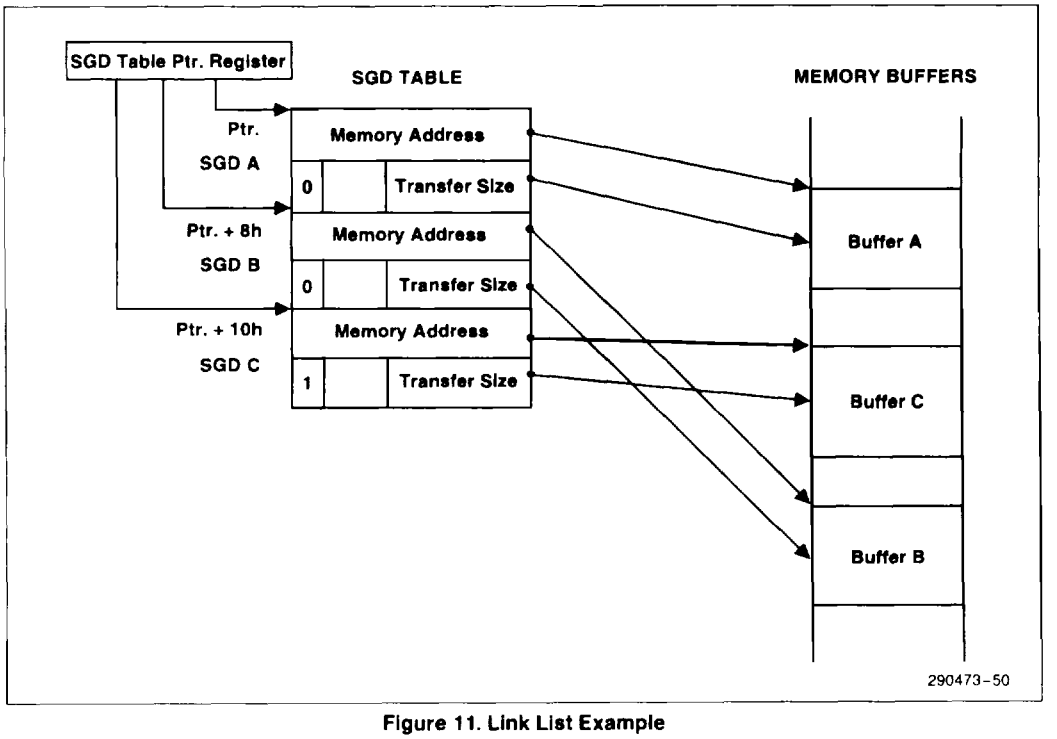

Figure 11. Link List Example

For I/O cycles, AD[31:0] are all decoded to provide a byte address. The byte enables determine which byte lanes contain valid data. The SIO requires that PCI accesses to byte-wide internal registers must assert only one byte enable.

For memory cycles, accesses are decoded as Dword accesses. This means that AD[1:0] are ignored for decoding memory cycles. The byte enables are used only to determine which byte lanes contain valid data.

For configuration cycles, DEVSEL# is a function of IDSEL# and AD[1:0]. DEVSEL# is generated only when AD[1:0] are both zero. If either AD[1:0] are non-zero, the cycle is ignored by the SIO. Individual bytes of a configuration register can be accessed with the byte enables. A particular configuration register is selected using AD[7:2]. Again, the byte enables determine which byte lanes contain valid data.

All PCI cycles decoded in one of the following ways result in the SIO generating DEVSEL#. The PCI master cycle decoder decodes the following addresses based on the settings of the relevant configuration registers:

SIO I/O Addresses: Positively decodes I/O addresses for registers contained within the SIO (exceptions: 60h, 92h, 3F2h, 372h, and F0h).

BIOS Memory Space: Positively decodes BIOS memory space.

MEMCS# Address Decoding: Decodes memory addresses that reside on the other side of the Host bridge and generates the MEMCS# signal. (SIO does not generate DEVSEL# in this case). The address range(s) used for this decoding is selected via the MEMCSCON, MEMCSBOH, MEMCSTOH, MEMCSTOM, MAR1, MAR2, and MAR3 Registers (see Section 4.1).

Subtractively Decoding Cycles to ISA: Subtractively decodes cycles to the ISA Bus. Accesses to registers 60h, 92h, 3F2h, 372h, and F0h are also subtractively decoded to the ISA Bus.

One of the PCI requirements is that, upon power-up, PCI agents do not respond to any address. Typically, the only access to a PCI agent is through the IDSEL configuration mechanism until it is enabled through configuration. The SIO is an exception to this, since it controls access to the BIOS boot code. All addresses decoded by the PCI address decoder, that are enabled after chip reset, are accessible immediately after power-up.

5.5.1.1 SIO I/O Addresses

These addresses are the internal, non-configuration SIO register locations and are shown in the SIO Address Decoding Table, Table 19. These addresses are fixed. Note that the Configuration Registers, listed in Table 3, are accessed with PCI configuration cycles as described in Section 5.1.2.5

In general, PCI accesses to the internal SIO registers will not be broadcast to the ISA Bus. However, PCI accesses to addresses 70h, 60h, 92h, 3F2h, 372h, and F0h are exceptions. Read and write accesses to these SIO locations are broadcast onto the ISA Bus. PCI master accesses to SIO registers will be retried if the ISA Bus is owned by an ISA master or the DMA controller. All of the registers are 8 bit registers. Accesses to these registers must be 8 bit accesses. Target-abort is issued by the SIO when the internal SIO non-configuration registers are the target of a PCI master I/O cycle and more than one byte enable is active. Refer to Table 19 for the SIO Address Decoding Map.

Accesses to the BIOS Timer Register (78h–7Bh) are broadcast to the ISA bus only if this register is disabled. If this register is enabled, the cycle is not broadcast to the ISA bus.

The address decoding logic includes the read/write cycle type. For example, read cycles to write only registers are not positively decoded and get forwarded to the ISA bus via subtractive decoding.

Table 19. SIO Address Decoding

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
0000h	0000	0000	000x	0000	r/w	DMA1 CH0 Base and Current Address	DMA
0001h	0000	0000	000x	0001	r/w	DMA1 CH0 Base and Current Count	DMA
0002h	0000	0000	000x	0010	r/w	DMA1 CH1 Base and Current Address	DMA
0003h	0000	0000	000x	0011	r/w	DMA1 CH1 Base and Current Count	DMA
0004h	0000	0000	000x	0100	r/w	DMA1 CH2 Base and Current Address	DMA
0005h	0000	0000	000x	0101	r/w	DMA1 CH2 Base and Current Count	DMA
0006h	0000	0000	000x	0110	r/w	DMA1 CH3 Base and Current Address	DMA
0007h	0000	0000	000x	0111	r/w	DMA1 CH3 Base and Current Count	DMA
0008h	0000	0000	000x	1000	r/w	DMA1 Status(r) Command(w) Register	DMA
0009h	0000	0000	000x	1001	wo	DMA1 Write Request Register	DMA
000Ah	0000	0000	000x	1010	wo	DMA1 Write Single Mask Bit	DMA
000Bh	0000	0000	000x	1011	wo	DMA1 Write Mode Register	DMA
000Ch	0000	0000	000x	1100	wo	DMA1 Clear Byte Pointer	DMA
000Dh	0000	0000	000x	1101	wo	DMA1 Master Clear	DMA
000Eh	0000	0000	000x	1110	wo	DMA1 Clear Mask Register	DMA
000Fh	0000	0000	000x	1111	r/w	DMA1 Read/Write All Mask Register Bits	DMA
0020h	0000	0000	001x	xx00	r/w	INT 1 Control Register	PIC
0021h	0000	0000	001x	xx01	r/w	INT 1 Mask Register	PIC
0040h	0000	0000	010x	0000	r/w	Timer Counter 1—Counter 0 Count	TC
0041h	0000	0000	010x	0001	r/w	Timer Counter 1—Counter 1 Count	TC
0042h	0000	0000	010x	0010	r/w	Timer Counter 1—Counter 2 Count	TC
0043h	0000	0000	010x	0011	wo	Timer Counter 1 Command Mode Register	TC
0060h	0000	0000	0110	0000	ro	Reset UBus IRQ12	Control
0061h	0000	0000	0110	0xx1	r/w	NMI Status and Control	Control
0070h	0000	0000	0111	0xx0	wo	CMOS RAM Address and NMI Mask Register	Control
0078h(1)	0000	0000	0111	10xx	r/w	BIOS Timer	TC

2

Table 19. SIO Address Decoding (Continued)

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
0080h	0000	0000	100x	0000	r/w	DMA Page Register (Reserved)	DMA
0081h	0000	0000	100x	0001	r/w	DMA Channel 2 Page Register	DMA
0082h	0000	0000	1000	0010	r/w	DMA Channel 3 Page Register	DMA
0083h	0000	0000	100x	0011	r/w	DMA Channel 1 Page Register	DMA
0084h	0000	0000	100x	0100	r/w	DMA Page Register (Reserved)	DMA
0085h	0000	0000	100x	0101	r/w	DMA Page Register (Reserved)	DMA
0086h	0000	0000	100x	0110	r/w	DMA Page Register (Reserved)	DMA
0087h	0000	0000	100x	0111	r/w	DMA Channel 0 Page Register	DMA
0088h	0000	0000	100x	0100	r/w	DMA Page Register (Reserved)	DMA
0089h	0000	0000	100x	1001	r/w	DMA Channel 6 Page Register	DMA
008Ah	0000	0000	100x	1010	r/w	DMA Channel 7 Page Register	DMA
008Bh	0000	0000	100x	1011	r/w	DMA Channel 5 Page Register	DMA
008Ch	0000	0000	100x	1100	r/w	DMA Page Register (Reserved)	DMA
008Dh	0000	0000	100x	1101	r/w	DMA Page Register (Reserved)	DMA
008Eh	0000	0000	100x	1110	r/w	DMA Page Register (Reserved)	DMA
008Fh	0000	0000	100x	1111	r/w	DMA Low Page Register Refresh	DMA
0090h	0000	0000	100x	0000	r/w	DMA Page Register (Reserved)	DMA
0092h	0000	0000	1001	0010	r/w	System Control Port	Control
0094h	0000	0000	100x	0100	r/w	DMA Page Register (Reserved)	DMA
0095h	0000	0000	100x	0101	r/w	DMA Page Register (Reserved)	DMA
0096h	0000	0000	100x	0110	r/w	DMA Page Register (Reserved)	DMA
0098h	0000	0000	100x	1000	r/w	DMA Page Register (Reserved)	DMA
009Ch	0000	0000	100x	1100	r/w	DMA Page Register (Reserved)	DMA
009Dh	0000	0000	100x	1101	r/w	DMA Page Register (Reserved)	DMA
009Eh	0000	0000	100x	1110	r/w	DMA Page Register (Reserved)	DMA
009Fh	0000	0000	100x	1111	r/w	DMA low page Register Refresh	DMA
00A0h	0000	0000	101x	xx00	r/w	INT 2 Control Register	PIC
00A1h	0000	0000	101x	xx01	r/w	INT 2 Mask Register	PIC
00B2h	0000	0000	1011	0010	r/w	Advanced Power Management Control Port	PM
00B3h	0000	0000	1011	0011	r/w	Advanced Power Management Status Port	PM
00C0h	0000	0000	1100	000x	r/w	DMA2 CH0 Base and Current Address	DMA

Table 19. SIO Address Decoding (Continued)

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
00C2h	0000	0000	1100	001x	r/w	DMA2 CH0 Base and Current Count	DMA
00C4h	0000	0000	1100	010x	r/w	DMA2 CH1 Base and Current Address	DMA
00C6h	0000	0000	1100	011x	r/w	DMA2 CH1 Base and Current Count	DMA
00C8h	0000	0000	1100	100x	r/w	DMA2 CH2 Base and Current Address	DMA
00CAh	0000	0000	1100	101x	r/w	DMA2 CH2 Base and Current Count	DMA
00CCh	0000	0000	1100	110x	r/w	DMA2 CH3 Base and Current Address	DMA
00CEh	0000	0000	1100	111x	r/w	DMA2 CH3 Base and Current Count	DMA
00D0h	0000	0000	1101	000x	r/w	DMA2 Status(r) Command(w) Register	DMA
00D2h	0000	0000	1101	001x	wo	DMA2 Write Request Register	DMA
00D4h	0000	0000	1101	010x	wo	DMA2 Write Single Mask Bit	DMA
00D6h	0000	0000	1101	011x	wo	DMA2 Write Mode Register	DMA
00D8h	0000	0000	1101	100x	wo	DMA2 Clear Byte Pointer	DMA
00DAh	0000	0000	1101	101x	wo	DMA2 Master Clear	DMA
00DCh	0000	0000	1101	110x	wo	DMA2 Clear Mask Register	DMA
00DEh	0000	0000	1101	111x	r/w	DMA2 Read/Write All Mask Register Bits	DMA
00F0h	0000	0000	1111	0000	wo	Coprocessor Error	Control
0372h	0000	0011	0111	0010	wo	Secondary Floppy Disk Digital Output Reg.	Control
03F2h	0000	0011	1111	0001	wo	Primary Floppy Disk Digital Output Reg.	Control
040Ah	0000	0100	0000	1010	ro	Scatter/Gather Interrupt Status Register	DMA
040Bh	0000	0100	0000	1011	wo	DMA1 Extended Mode register	DMA
0410h(1)	0000	0100	0001	0000	wo	CH0 Scatter/Gather Command	DMA
0411h(1)	0000	0100	0001	0001	wo	CH1 Scatter/Gather Command	DMA
0412h(1)	0000	0100	0001	0010	wo	CH2 Scatter/Gather Command	DMA
0413h(1)	0000	0100	0001	0011	wo	CH3 Scatter/Gather Command	DMA
0415h(1)	0000	0100	0001	0101	wo	CH5 Scatter/Gather Command	DMA
0416h(1)	0000	0100	0001	0110	wo	CH6 Scatter/Gather Command	DMA
0417h(1)	0000	0100	0001	0111	wo	CH7 Scatter/Gather Command	DMA
0418h(1)	0000	0100	0001	1000	ro	CH0 Scatter/Gather Status	DMA
0419h(1)	0000	0100	0001	1001	ro	CH1 Scatter/Gather Status	DMA
041Ah(1)	0000	0100	0001	1010	ro	CH2 Scatter/Gather Status	DMA
041Bh(1)	0000	0100	0001	1011	ro	CH3 Scatter/Gather Status	DMA

Table 19. SIO Address Decoding (Continued)

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
041Dh(1)	0000	0100	0001	1101	ro	CH5 Scatter/Gather Status	DMA
041Eh(1)	0000	0100	0001	1110	ro	CH6 Scatter/Gather Status	DMA
041Fh(1)	0000	0100	0001	1111	ro	CH7 Scatter/Gather Status	DMA
0420h(1)	0000	0100	0010	00xx	r/w	CH0 Scatter/Gather Descriptor Table Pointer	DMA
0424h(1)	0000	0100	0010	01xx	r/w	CH1 Scatter/Gather Descriptor Table Pointer	DMA
0428h(1)	0000	0100	0010	10xx	r/w	CH2 Scatter/Gather Descriptor Table Pointer	DMA
042Ch(1)	0000	0100	0010	11xx	r/w	CH3 Scatter/Gather Descriptor Table Pointer	DMA
0434h(1)	0000	0100	0011	01xx	r/w	CH5 Scatter/Gather Descriptor Table Pointer	DMA
0438h(1)	0000	0100	0011	10xx	r/w	CH6 Scatter/Gather Descriptor Table Pointer	DMA
043Ch(1)	0000	0100	0011	11xx	r/w	CH7 Scatter/Gather Descriptor Table Pointer	DMA
0481h	0000	0100	1000	0001	r/w	DMA CH2 High Page Register	DMA
0482h	0000	0100	1000	0010	r/w	DMA CH3 High Page Register	DMA
0483h	0000	0100	1000	0011	r/w	DMA CH1 High Page Register	DMA
0487h	0000	0100	1000	0111	r/w	DMA CH0 High Page Register	DMA
0489h	0000	0100	1000	1001	r/w	DMA CH6 High Page Register	DMA
048Ah	0000	0100	1000	1010	r/w	DMA CH7 High Page Register	DMA
048Bh	0000	0100	1000	1011	r/w	DMA CH5 High Page Register	DMA
04D0	0000	0100	1101	0000	r/w	INT CNTRL-1 Edge Level Control Register	Control
04D1	0000	0100	1101	0001	r/w	INT CNTRL-2 Edge Level Control Register	Control
04D6h	0000	0100	1101	0010	wo	DMA2 Extended Mode Register	DMA

NOTE:
1. The I/O address of this register is relocatable. The value shown in this table is the default address location.

5.5.1.2 BIOS Memory Space

The 128 Kb BIOS memory space is located at 000E0000h to 000FFFFFh (top of 1 Mb), and is aliased at FFFE0000h to FFFFFFFFh (top of 4 Gb) and FFEE0000h to FFEFFFFFh (top of 4 Gb-1 Mb). The aliased regions account for the CPU reset vector and the uncertainty of the state of A20GATE when a software reset occurs. This 128 Kb block is split into two 64 Kb blocks. The top 64 Kb is always enabled while the bottom 64 Kb can be enabled or disabled (the aliases automatically match). Enabling the lower 64 Kb BIOS space (000E0000h to 000EFFFFh, 896 Kb-960 Kb) results in positively decoding this region and enables the BIOSCS# signal generation. The upper 64 Kb is positively decoded only if bit 6 = 1 in the ISA Clock Divisor Register. Otherwise this region is subtractively decoded. Positively decoding these cycles expedites BIOS cycles to the ISA Bus. Note that both of these regions are subtractively decoded if bit 4 in the MEMCS# Control Register is set to a 1.

When PCI master accesses to the 128 Kb BIOS space at 4 Gb-1 Mb are forwarded to the ISA Bus, the LA20 line is driven to a 1 to avoid aliasing at the 15 Mb area. The 4 Gb-1 Mb BIOS decode area accounts for the condition when A20M# is asserted and an ALT-CTRL-DEL reset is generated. The CPU's reset vector will access 4 Gb-1 Mb. When this gets forwarded to ISA, AD[32:24] are truncated and the access is aliased to 16 Mb-1 Mb = 15 Mb space. If ISA memory is present at 15 Mb, there will be contention. Forcing LA20 high aliases this region to 16 Mb. The alias here is permissible since this is the 80286 reset vector location.

In addition to the normal 128 Kb BIOS space, the SIO supports an additional 384 Kb BIOS space. The SIO can support a total of 512 Kb BIOS space. The additional 384 Kb region can only be accessed by PCI masters and resides at FFF80000h to FFFDFFFFh. When enabled via the UBCSA Register, memory accesses within this region will be positively decoded, forwarded to the ISA Bus, and encoded BIOSCS# will be generated. When forwarded to the ISA Bus, the PCI AD[23:20] signals will be propagated to the ISA LA[23:20] lines as all 1's which will result in aliasing this 512 Kb region at the top of the 16 Mb space. To avoid contention, ISA add-in memory must not be present in this space.

All PCI cycles positively decoded in the enabled BIOS space will be broadcast to the ISA Bus. Since the BIOS device is 8 or 16 bits wide and typically has very long access times, PCI burst reads from the BIOS space will invoke "disconnect target termination" semantics after the first data transaction in order to meet the PCI incremental latency guidelines.

The following tables and diagrams describe the operation of the SIO in response to PCI BIOS space accesses.

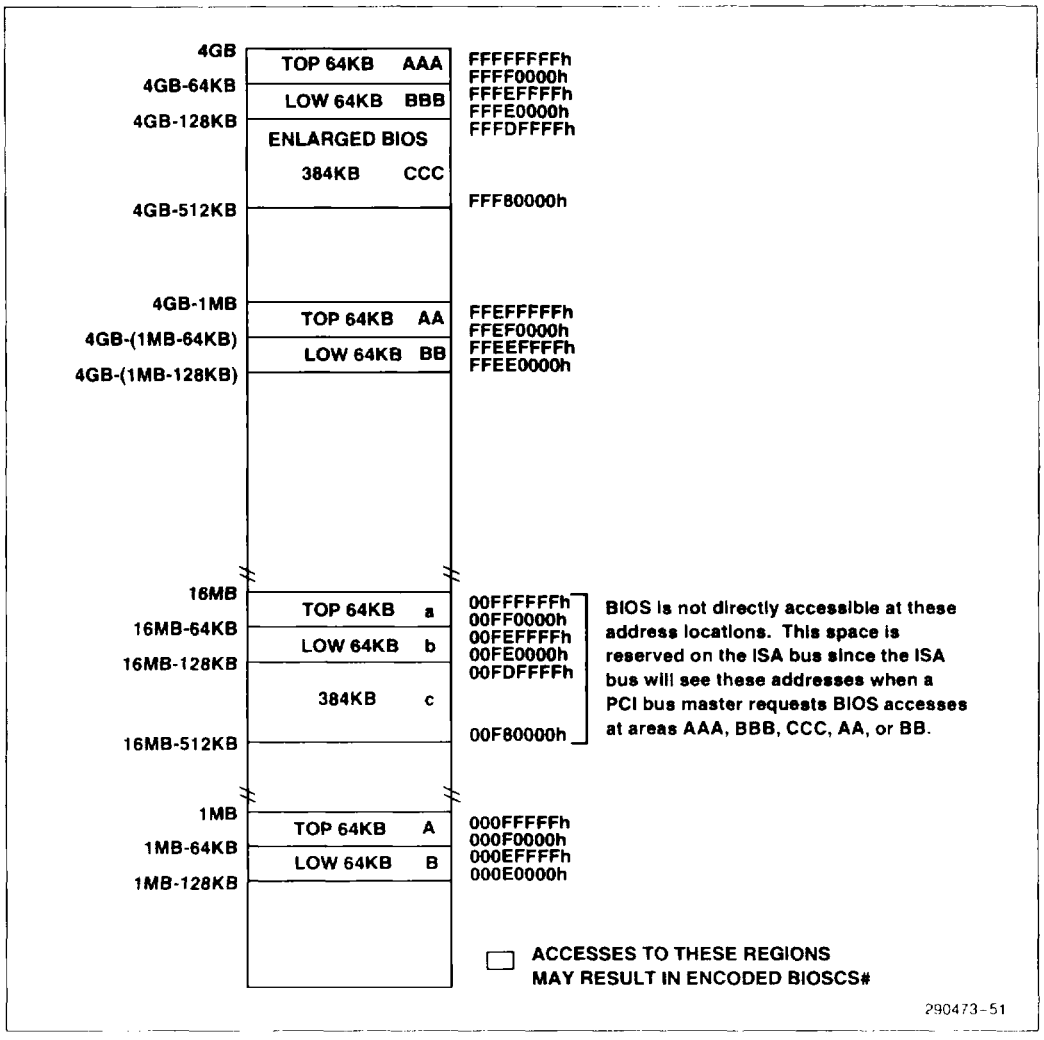

Figure 12. BIOS Space Decode Map

The BIOS space decode map, Figure 12, shows the possible BIOS spaces and the aliases throughout the memory space. The various regions are designated with code letters; "a's" for the top 64 Kb, "b's" for the low 64 Kb, and "c's" for the enlarged space.

Table 20 indicates the SIO's response to PCI BIOS space accesses based on its configuration state.

Table 20. PCI Master BIOS Space Decoding

Master	Region	Top 64 Kb BIOS Positive Decode Enabled(1)	Low 64 Kb BIOS Enabled(2)	Enlarged BIOS Enabled(3)	Encoded BIOSCS# Generated	LA20	Positive PCI Decode	Subtractive PCI Decode
PCI	A	0	x	x	Yes	Pass (0)	No	Yes
PCI	A	1	x	x	Yes	Pass (0)	Yes(5)	No(5)
PCI	B	x	0	x	No	Pass (0)	No	Yes
PCI	B	x	1	x	Yes	Pass (0)	Yes(5)	No(5)
PCI	a	1	x	x	No	Pass (1)	No	Yes
PCI	b	x	0	x	No	Pass (1)	No	Yes
PCI	c	x	x	0	No	Pass (1)	No	Yes
PCI	AA	0	x	x	Yes	1	No	Yes(4)
PCI	AA	1	x	x	Yes	1	Yes(4,5)	No(5)
PCI	BB	x	0	x	No	x	No	No
PCI	BB	x	1	x	Yes	1	Yes(4,5)	No(5)
PCI	AAA	0	x	x	Yes	Pass (1)	No	Yes(4)
PCI	AAA	1	x	x	Yes	Pass (1)	Yes(4,5)	No(5)
PCI	BBB	x	0	x	No	x	No	No
PCI	BBB	x	1	x	Yes	Pass (1)	Yes(4,5)	No(5)
PCI	CCC	x	x	0	No	x	No	No
PCI	CCC	x	x	1	Yes	Pass (1)	Yes(4)	No

NOTES:
1. The column labeled "Top 64 Kb BIOS Positive Decode Enable" shows the value of the ISA Clock Register bit 6. This bit determines the decoding for memory regions A, AA, and AAA (1 = positive, 0 = negative decoding).
   Note that if bit 4 in the MEMCS# Control Register is set to a 1 (Global MEMCS# decode enabled), the positive decoding function enabled by having ISA Clock Register bit 6 = 1 is ignored. Subtractive decoding is provided, instead.
2. The column labeled "Low 64 Kb BIOS Enable" shows the value of the Utility Bus Chip Select Enable A Bit 6. This bit determines whether memory regions B, BB, and BBB are enabled (bit = 1) or disabled (bit = 0).
3. The column labeled "Enlarged BIOS Enabled" shows the value of the Utility Bus Chip Select Enable A Bit 7. This bit determines whether memory region CCC is enabled (bit = 1) or disabled (bit = 0).
4. ISA memory is not allowed to be enabled at the corresponding aliased areas or contention will result.
5. When bit 4 in the MEMCS# Control Register is set to a 1 (Global MEMCS# decode enabled), positive decoding for these areas will be disabled. The SIO will only provide subtractive decoding in this case.

5.5.1.3 MEMCS# Decoding

For MEMCS# decoding, the SIO decodes sixteen ranges. Fourteen of these ranges can be enabled or disabled independently for both read and write cycles. The fifteenth range (0 KB–512 KB) and sixteenth range (programmable from 1 MB up to 512 MB in 2 MB increments) can be enabled or disabled only. Addresses within these enabled regions generate a MEMCS# signal that can be used by the host bridge to know when to forward PCI cycles to main memory. A seventeenth range is available that can be used to identify a "memory hole". Addresses within this hole will not generate a MEMCS#. The address regions are summarized:

- 0 KB to 512 KB Memory (can only be disabled if MEMCS# is completely disabled)
- 512 KB to 640 KB Memory
- (1 MB–64 KB) to 1 MB Memory (BIOS Area)
- 768 KB to 918 KB in 16 KB sections (total of 8 sections)
- 918 KB to 983 KB in 16 KB sections (total of 4 sections)
- 1M-to-programmable boundary on 2 MB increments from 2 MB up to 512 MB
- programmable "memory hole" in 64 KB increments between 1 MB and 16 MB

Table 21 and Figure 13 show the registers and decode areas for MEMCS#.

Table 21. MEMCS# Decoding Register Summary

MAR Registers	Attribute	Memory Segments	Comments
MCSCON[4] = 0	Disable	Disable MEMCS# Function	Enable/Disable MEMCS# Function
MCSCON[4] = 1	Enable	Enable MEMCS# Function	When Enabled, 0 KB to 512 KB Range is also Automatically Enabled (RE/WE)
MCSTOH/ MCSBOH	MEMCS# Hole	100000h–0FFFFFFh	1 MB to 16 MB Hole in MEMCS# Region
MCSTOM	MEMCS# Top	200000h–1FFFFFFFh	2 MB to 512 MB Top of MEMCS# Region
MCSCON[1:0]	[0] = RE[1] = WE	080000h–09FFFFh	512 KB to 640 KB R/W Enable
MCSCON[3:2]	[2] = RE[3] = WE	0F0000h–0FFFFFh	BIOS Area R/W Enable
MAR1[1:0]	[0] = RE[1] = WE	0C0000h–0C3FFFh	ISA Add-On BIOS R/W Enable
MAR1[3:2]	[2] = RE[3] = WE	0C4000h–0C7FFFh	ISA Add-On BIOS R/W Enable
MAR1[5:4]	[4] = RE[5] = WE	0C8000h–0CBFFFh	ISA Add-On BIOS R/W Enable
MAR1[7:6]	[6] = RE[7] = WE	0CC000h–0CFFFFh	ISA Add-On BIOS R/W Enable
MAR2[1:0]	[0] = RE[1] = WE	0D0000h–0D3FFFh	ISA Add-On BIOS R/W Enable
MAR2[3:2]	[2] = RE[3] = WE	0D4000h–0D7FFFh	ISA Add-On BIOS R/W Enable
MAR2[5:4]	[4] = RE[5] = WE	0D8000h–0DBFFFh	ISA Add-On BIOS R/W Enable
MAR2[7:6]	[6] = RE[7] = WE	0DC000h–0DFFFFh	ISA Add-On BIOS R/W Enable
MAR3[1:0]	[0] = RE[1] = WE	0E0000h–0E3FFFh	BIOS Extension R/W Enable
MAR3[3:2]	[2] = RE[3] = WE	0E4000h–0E7FFFh	BIOS Extension R/W Enable
MAR3[5:4]	[4] = RE[5] = WE	0E8000h–0EBFFFh	BIOS Extension R/W Enable
MAR3[7:6]	[6] = RE[7] = WE	0EC000h–0EFFFFh	BIOS Extension R/W Enable

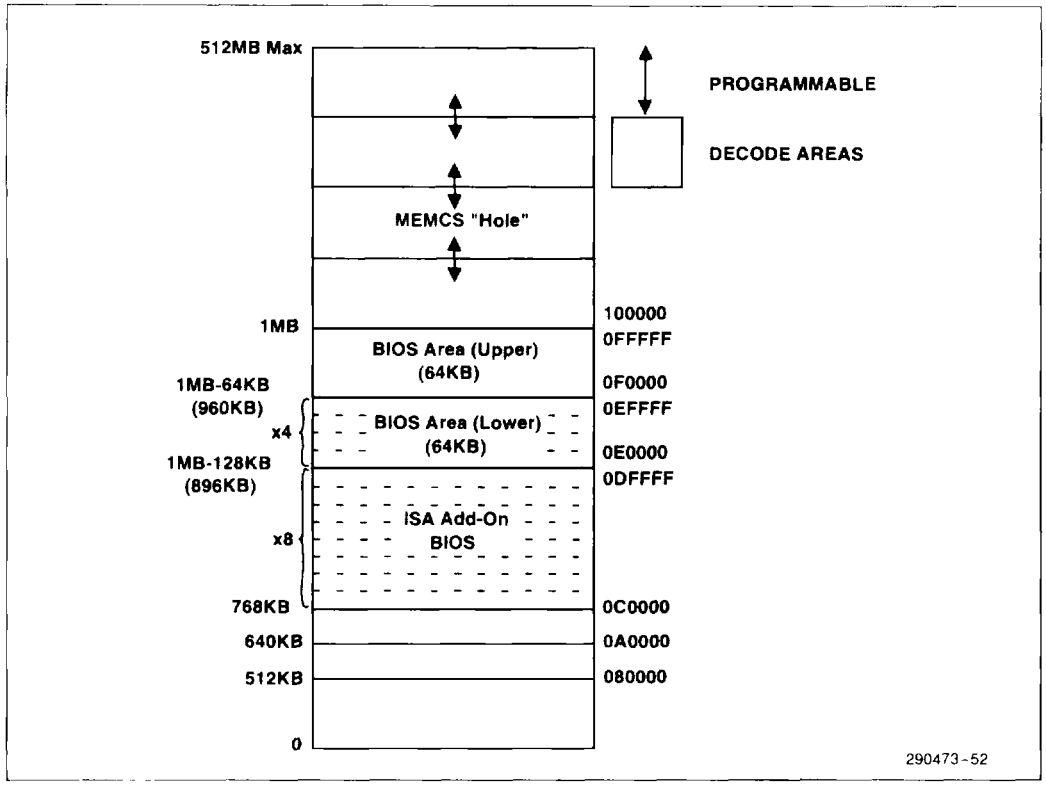

Figure 13. MEMCS# Decode Areas

The SIO generates MEMCS# from the PCI address. MEMCS# is generated from the clock edge after FRAME# is sampled active. MEMCS# will only go active for one PCI clock period. The SIO does not take any other action as a result of this decode other than generating MEMCS#. It is the responsibility of the device using the MEMCS# signal to generate DEVSEL#, TRDY# and any other cycle response. The device using MEMCS# will always generate DEVSEL# on the next clock. This fact can be used to avoid an extra clock delay in the subtractive decoder described in the next section.

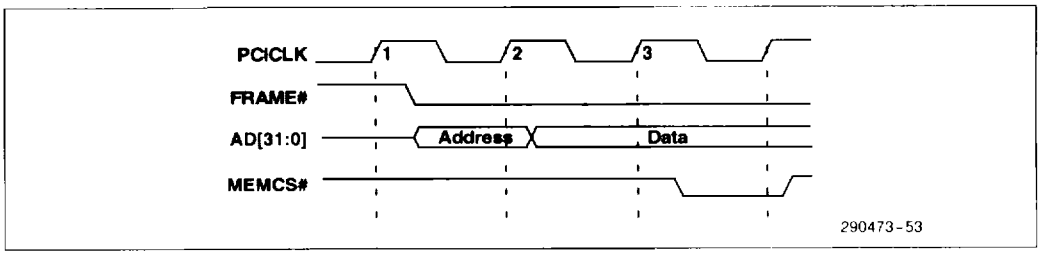

Figure 14. MEMCS# Generation

5.5.1.4 Subtractively Decoded Cycles to ISA

The addresses that reside on the ISA Bus could be highly fragmented. For this reason, subtractive decoding is used to forward PCI cycles to the ISA Bus. An inactive DEVSEL# will cause the SIO to forward the PCI cycle to the ISA Bus. The DEVSEL# sample point can be configured for three different settings. If the "fast" point is selected, the cycle will be forwarded to ISA when DEVSEL# is inactive at the F sample point as shown in Figure 15. If the "typical" point is selected, DEVSEL# will be sampled on both F and T, and if inactive, will be forwarded to the ISA Bus. Likewise, if the "slow" point is selected, DEVSEL# will be sampled at F, T, and S. The sample point should be configured to match the slowest PCI device in the system. This capability reduces the latency to ISA slaves when all PCI devices are "fast" and also allows for devices with slow decoding. Note that when these unclaimed cycles are forwarded to the ISA Bus, the SIO will drive the DEVSEL# active.

Since an active MEMCS# will always result in an active DEVSEL# at the "Slow" sample point, MEMCS# is used as an early indication of DEVSEL#. In this case, if the device using MEMCS# is the only "slow" agent in the system, the sample point can be moved in to the "typical" edge.

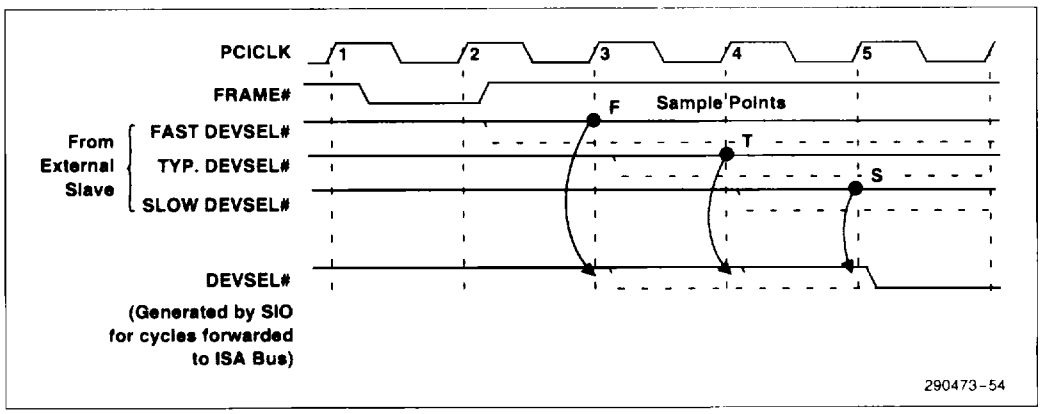

Figure 15. DEVSEL# Generation

Unclaimed PCI cycles with memory addresses above 16M and I/O addresses above 64K will not be forwarded to the ISA Bus. The SIO will not respond with DEVSEL# (BIOS accesses are an exception to this). This is required to avoid the possibility of aliasing. Under this condition, these unclaimed cycles will be recognized as such by the originating master and the master will use "master-abort" semantics to terminate the PCI cycle.

5.5.2 DMA/ISA MASTER CYCLE ADDRESS DECODER

The SIO also contains a decoder which is used to determine the destination of ISA master and DMA master cycles. This decoder provides:

Positive Decode to PCI: Positively decodes addresses to be forwarded to the PCI Bus. This includes addresses residing directly on PCI as well as addresses that reside on the back side of PCI bridges (Host Bridges).

Access to SIO Internal Registers: Positively decodes addresses to registers within the SIO.

BIOS Accesses: Positively decodes BIOS memory accesses and generates encoded BIOSCS#.

Utility Bus Chip Selects: Positively decodes utility bus chip selects.

Subtractive Decode: Subtractively decodes cycles to be contained to the ISA Bus.

5.5.2.1 Positive Decode to PCI

ISA master or DMA addresses that are positively decoded by this decoder will be propagated to the PCI Bus. This is the only way to forward a cycle from an ISA master or the DMA to the PCI Bus. If the cycle is not decoded by this decoder it will *not* be forwarded to the PCI Bus.

This decoder has several memory address regions to positively decode cycles that should be forwarded to the PCI Bus. These regions are listed below. Regions "a" through "e" are fixed and can be enabled or disabled independently. Region "f" defines a space starting at 1M with a programmable upper boundary up to 16 MB. Within this region a hole can be opened. Its size and location are programmable to allow a hole to be opened in the memory space. A memory address above 16 MB will be forwarded to the PCI Bus automatically. This is possible only during DMA cycles in which the DMA has been programmed for 32-bit addressing above 16 MB.

a. Memory: 0 KB–512 KB

b. Memory: 512 KB–640 KB

c. Memory: 640 KB–768 KB (Video buffer)

d. Memory: 768 KB–896 KB in eight 16K sections (Expansion ROM)

e. Memory: 896 KB–960 KB (lower BIOS area)

f. Memory: 1 MB-to-X MB (up to 16 MB) within which a hole can be opened. Accesses to the hole are not forwarded to PCI. The top of the region can be programmed on 64 KByte boundaries up to 16 MB. The hole can be between 64 KB and 8 MB in size in 64 KB increments located on any 64 KB boundary. (Refer to the ISA Address Decoder Register in the register description section, Section 5.5.2)

g. Memory: > 16 MB automatically forwarded to PCI

Figure 16 shows a map of the ISA master/DMA decode regions and Table 22 summarizes the registers used to configure the decoder.

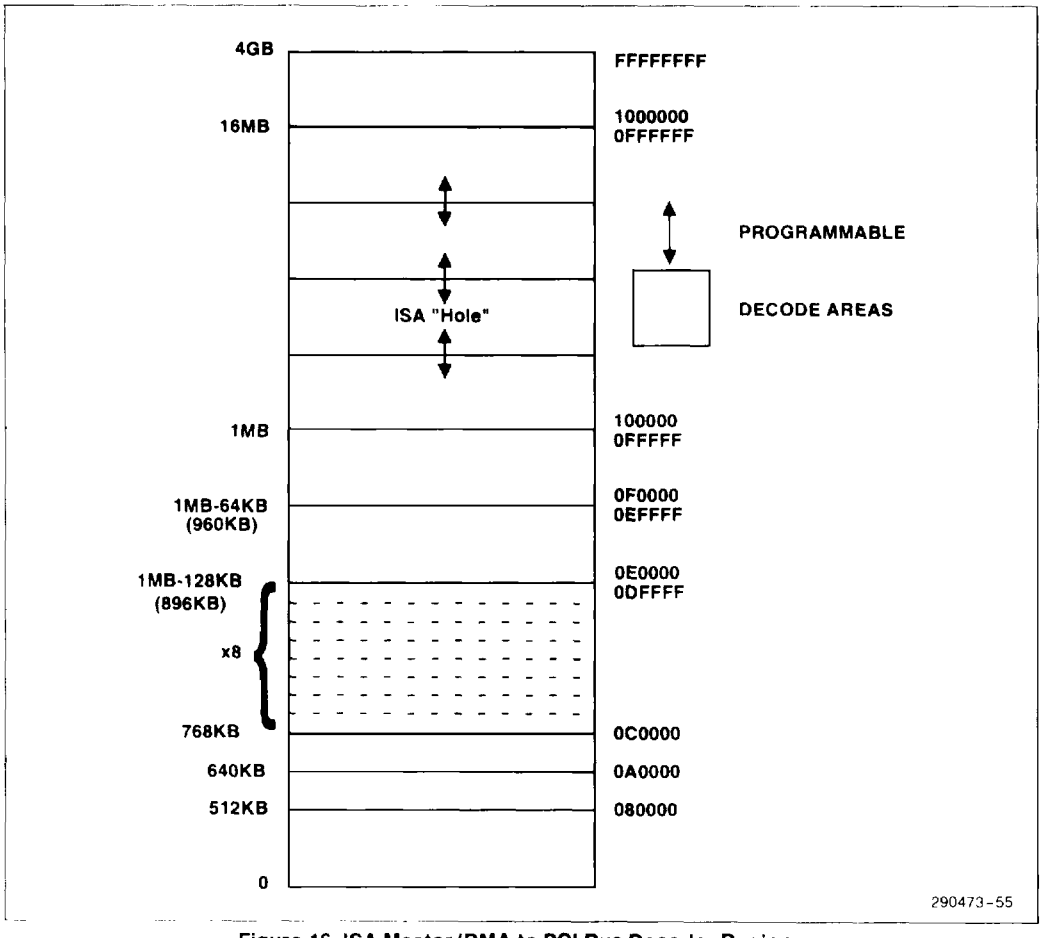

Figure 16. ISA Master/DMA to PCI Bus Decoder Regions

Table 22. ISA Master/DMA to PCI Bus Decoding Register Summary

MAR Registers	Attribute	Memory Segments	Comments
IADCON[7:4]	ISA Memory Top	100000h–0FFFFFFh	1 MB to 16 MB Top of ISA Region
IADTOH/IADBOH	ISA Hole	100000h–0FFFFFFh	1 MB to 16 MB Hole in ISA Region
IADCON[0]	Enable/Disable	000000h–07FFFFh	0 to 512 KB Enable/Disable
IADCON[1]	Enable/Disable	080000h–09FFFFh	512 KB to 640 KB Enable/Disable
IADCON[2]	Enable/Disable	0A0000h–0BFFFFh	640 KB to 768 KB Enable/Disable
IADCON[3]*	Enable/Disable	0E0000h–0EFFFFh	896 KB to 960 KB Lower BIOS Enable/Disable
IADRBE[0]	Enable/Disable	0C0000h–0C3FFFh	ISA Add-On BIOS (Expansion ROM) Enable
IADRBE[1]	Enable/Disable	0C4000h–0C7FFFh	ISA Add-On BIOS (Expansion ROM) Enable
IADRBE[2]	Enable/Disable	0C8000h–0CBFFFh	ISA Add-On BIOS (Expansion ROM) Enable
IADRBE[3]	Enable/Disable	0CC000h–0CFFFFh	ISA Add-On BIOS (Expansion ROM) Enable
IADRBE[4]	Enable/Disable	0D0000h–0D3FFFh	ISA Add-On BIOS (Expansion ROM) Enable
IADRBE[5]	Enable/Disable	0D4000h–0D7FFFh	ISA Add-On BIOS (Expansion ROM) Enable
IADRBE[6]	Enable/Disable	0D8000h–0DBFFFh	ISA Add-On BIOS (Expansion ROM) Enable
IADRBE[7]	Enable/Disable	0DC000h–0DFFFFh	ISA Add-On BIOS (Expansion ROM) Enable

NOTE:
* This can be overridden by bit 6 of the UBCSA Register being set to a 1.

5.5.2.2 SIO Internal Registers

Most of the internal SIO registers are accessible by ISA masters. Table 19 lists the registers that are not accessible by ISA masters. Registers accessed by ISA masters are run as 8-bit extended I/O cycles.

5.5.2.3 BIOS Accesses

The 128K BIOS memory space is located at 000E0000h to 000FFFFFh, and is aliased at FFFE0000h to FFFFFFFFh (top of 4 GB) and FFEE0000h to FFEFFFFFh (top of 4 GB–1 MB). The aliased regions account for the CPU reset vector and the uncertainty of the state of A20GATE when a software reset occurs. This 128K block is split into two 64K blocks. The top 64K is always enabled while the bottom 64K can be enabled or disabled (the aliases automatically match). ISA masters can only access BIOS in the 000E0000 to 000FFFFFh region.

ISA originated accesses to the enabled 64K sections of the BIOS space (000E0000h–000FFFFFh) will activate the encoded BIOSCS# signal. ISA originated cycles will not be forwarded to the PCI Bus. Encoded BIOSCS# is combinatorially generated from the ISA, SA, and LA address bus. Encoded BIOSCS# is disabled during refresh and DMA cycles. The ISA Master/DMA BIOS Decoding Table indicates the SIO's response to BIOS accesses based on the configuration state.

Table 23. ISA Master/DMA BIOS Decoding

Cycle		SIO Configuration			SIO Response		
Master	Region[1]	Top 64 KB PCI Positive Decode Enabled[2]	Low 64 KB BIOS Enabled[3]	Forward Low 64 KB to PCI Enabled[4]	Encoded BIOSCS# Generated	Forward to PCI	Contain to ISA
ISA/DMA	A	x	x	x	Yes	No	Yes
ISA/DMA	B	x	0[5]	0	No	No	Yes
ISA/DMA	B	x	0[5]	1	No	Yes	No
ISA/DMA	B	x	1	x	Yes	No	Yes
ISA/DMA	a	These cycles will be forwarded to PCI dependent on the state of the ISA Address Decoder Configuration Registers. Encoded BIOSCS# will not be generated for any of these cycles.					
ISA/DMA	b						
ISA/DMA	c						

NOTES:
1. The memory sections referenced can be found in Figure 12.
2. The column labeled "Top 64 KB BIOS Positive Decode Enabled" shows the value of the ISA Clock Divisor Configuration Register bit 6. This bit determines how the memory region is decoded (0 – subtractively decoded, 1 positively decoded).
3. The column labeled "Low 64 KB BIOS Enable" shows the value of the Utility Bus Chip Select Enable A Configuration Register bit 6. This bit determines if the memory region is enabled (bit 1) or disabled (bit – 0).
4. The column labeled "Forward Low 64 KB to PCI Enables" shows the value of the ISA Address Decoder Control Configuration Register Bit 3. This bit determines whether PCI Bus forwarding is enabled (bit 1) or disabled (bit 0).
5. Forward to PCI if IADCON Bit 6 1.

5.5.2.4 Utility Bus Encoded Chip Selects

The SIO generates encoded chip selects for certain functions that are located on the utility bus (formerly X-Bus). The encoded chip selects are generated combinatorially from the ISA SA[15:0] address bus. The encoded chip selects are decoded externally (see Figure 19).

The encoded chip select table (Table 24) shows the addresses that result in encoded chip select generation. Chip selects can be enabled or disabled via configuration registers. In general, the addresses shown in Table 24 do not reside in the SIO itself. Write only addresses 70h, 372h, 3F2h are exceptions since particular bits from these registers reside in the SIO. For ISA master cycles, the SIO will respond to writes to address 70h, 372h, and 3F2h by generating IOCHRDY and writing to the appropriate bits.

Note that the SIO monitors read accesses to address 60h to support the mouse function. In this case, IOCHRDY is not generated.

Table 24. Encoded Chip Select Table

Address	Address				Type	Name	Encoded Chip Select
	FEDC	BA98	7654	3210			
0060h	0000	0000	0110	00x0	r/w	Keyboard Controller	KEYBRDCS#
0064h	0000	0000	0110	01x0	r/w	Keyboard Controller	KEYBRDCS#
0070h	0000	0000	0111	0xx0	w	Real Time Clock Address	RTCALE#

Table 24. Encoded Chip Select Table (Continued)

Address	Address				Type	Name	Encoded Chip Select
	FEDC	BA98	7654	3210			
0071h	0000	0000	0111	0xx1	r/w	Real Time Clock Data	RTCCS#
0170h	0000	0001	0111	0000	r/w	Secondary Data Register	IDECS0#
0171h	0000	0001	0111	0001	r/w	Secondary Error Register	IDECS0#
0172h	0000	0001	0111	0010	r/w	Secondary Sector Count Register	IDECS0#
0173h	0000	0001	0111	0011	r/w	Secondary Sector Number Register	IDECS0#
0174h	0000	0001	0111	0100	r/w	Secondary Cylinder Low Register	IDECS0#
0175h	0000	0001	0111	0101	r/w	Secondary Cylinder High Register	IDECS0#
0176h	0000	0001	0111	0110	r/w	Secondary Drive/Head Register	IDECS0#
0177h	0000	0001	0111	0111	r/w	Secondary Status Register	IDECS0#
01F0h	0000	0001	1111	0000	r/w	Primary Data Register	IDECS0#
01F1h	0000	0001	1111	0001	r/w	Primary Error Register	IDECS0#
01F2h	0000	0001	1111	0010	r/w	Primary Sector Count Register	IDECS0#
01F3h	0000	0001	1111	0011	r/w	Primary Sector Number Register	IDECS0#
01F4h	0000	0001	1111	0100	r/w	Primary Cylinder Low Register	IDECS0#
01F5h	0000	0001	1111	0101	r/w	Primary Cylinder High Register	IDECS0#
01F6h	0000	0001	1111	0110	r/w	Primary Drive/Head Register	IDECS0#
01F7h	0000	0001	1111	0111	r/w	Primary Status Register	IDECS0#
0278h	0000	0010	0111	1x00	r/w	LPT3 PP Data Latch	LPTCS#
0279h	0000	0010	0111	1x01	r	LPT3 PP Status	LPTCS#
027Ah	0000	0010	0111	1x10	r/w	LPT3 PP Control	LPTCS#
027Bh	0000	0010	0111	1x11	r/w		LPTCS#
02F8h	0000	0010	1111	1000	r/w	COM2 SP Transmit/Receive Register	COM2CS#
02F9h	0000	0010	1111	1001	r/w	COM2 SP Interrupt Enable Register	COM2CS#
02FAh	0000	0010	1111	1010	r	COM2 SP Interrupt Identification Register	COM2CS#
02FBh	0000	0010	1111	1011	r/w	COM2 SP Line Control Register	COM2CS#
02FCh	0000	0010	1111	1100	r/w	COM2 SP Modem Control Register	COM2CS#
02FDh	0000	0010	1111	1101	r	COM2 SP Line Status Register	COM2CS#
02FEh	0000	0010	1111	1110	r	COM2 SP Modem Status Register	COM2CS#
02FFh	0000	0010	1111	1111	r/w	COM2 SP Scratch Register	COM2CS#

Table 24. Encoded Chip Select Table (Continued)

Address	Address				Type	Name	Encoded Chip Select
	FEDC	BA98	7654	3210			
0370h	0000	0011	0111	0000	r/w	Secondary Floppy Disk Extended Mode Register	FLOPPYCS#
0371h	0000	0011	0111	0001	r/w	Secondary Floppy Disk Extended Mode Register	FLOPPYCS#
0372	0000	0011	0111	0010	w	Secondary Floppy Disk Digital Output Register	FLOPPYCS#
0373h	0000	0011	0111	0011	r/w	Reserved	FLOPPYCS#
0374h	0000	0011	0111	0100	r/w	Secondary Floppy Disk Status Register	FLOPPYCS#
0375h	0000	0011	0111	0101	r/w	Secondary Floppy Disk Data Register	FLOPPYCS#
0376h	0000	0011	0111	0110	r/w	Secondary Alternate Status Register	IDECS1#
0377h	0000	0011	0111	0111	r	Secondary Drive Address Register	IDECS1#
0377h*	0000	0011	0111	011x	r/w	Secondary Floppy Disk Digital Input Register	FLOPPYCS#
0378h	0000	0011	0111	1x00	r/w	LPT2 PP Data Latch	LPTCS#
0379h	0000	0011	0111	1x01	r	LPT2 PP Status	LPTCS#
037Ah	0000	0011	0111	1x10	r/w	LPT2 PP Control	LPTCS#
037Bh	0000	0011	0111	1x11	r/w		LPTCS#
03BCh	0000	0011	1011	1100	r/w	LPT1 PP Data Latch	LPTCS#
03BDh	0000	0011	1011	1101	r	LPT1 PP Status	LPTCS#
03BEh	0000	0011	1011	1110	r/w	LPT1 PP Control	LPTCS#
03BFh	0000	0011	1011	1111	r/w		LPTCS#
03F0h	0000	0011	1111	0000	r/w	Primary Floppy Disk Extended Mode Register	FLOPPYCS#
03F1h	0000	0011	1111	0001	r/w	Primary Floppy Disk Extended Mode Register	FLOPPYCS#
03F2h	0000	0011	1111	0010	w	Primary Floppy Disk Digital Output Register	FLOPPYCS#
03F3h	0000	0011	1111	0011	r/w	Reserved	FLOPPYCS#
03F4h	0000	0011	1111	0100	r/w	Primary Floppy Disk Status Register	FLOPPYCS#
03F5h	0000	0011	1111	0101	r/w	Primary Floppy Disk Data Register	FLOPPYCS#
03F6h	0000	0011	1111	0110	r/w	Primary Drive Alternate Status Register	IDECS1#
03F7h	0000	0011	1111	0111	r	Primary Drive Address Register	IDECS1#
03F7h*	0000	0011	1111	011x	r/w	Primary Floppy Disk Digital Input Register	FLOPPYCS#

Table 24. Encoded Chip Select Table (Continued)

Address	Address				Type	Name	Encoded Chip Select
	FEDC	BA98	7654	3210			
03F8h	0000	0011	1111	1000	r/w	COM1 SP Transmit/Receive Register	COM1CS#
03F9h	0000	0011	1111	1001	r/w	COM1 SP Interrupt Enable Register	COM1CS#
03FAh	0000	0011	1111	1010	r	COM1 SP Interrupt Identification Register	COM1CS#
03FBh	0000	0011	1111	1011	r/w	COM1 SP Line Control Register	COM1CS#
03FCh	0000	0011	1111	1100	r/w	COM1 SP Modem Control Register	COM1CS#
03FDh	0000	0011	1111	1101	r	COM1 SP Line Status Register	COM1CS#
03FEh	0000	0011	1111	1110	r	COM1 SP Modem Status Register	COM1CS#
03FFh	0000	0011	1111	1111	r/w	COM1 SP Scratch Register	COM1CS#
0800h–08FFh	0000	1000	xxxx	xxxx	r/w		CFIGMEMCS#
0C00h	0000	1100	0000	0000	r/w		CPAGECS#

NOTE:
*If both the IDE and Floppy Drive are located on the UBUS, FLOPPYCS# will not be generated, IDECS1# will be generated.

5.5.2.5 Subtractive Decode to ISA

ISA master and DMA cycles not positively decoded by the ISA decoder are contained to the ISA Bus.

5.6 Data Buffering

The SIO contains data buffers to isolate the PCI Bus from the ISA Bus. The buffering is described from two perspectives: PCI master accesses to the ISA Bus (Posted Write Buffer) and DMA/ISA master accesses to the PCI Bus (Line Buffer). Temporarily buffering the data requires buffer management logic to ensure that the data buffers remain coherent.

5.6.1 DMA/ISA MASTER LINE BUFFER

An 8-byte Line Buffer is used to isolate the ISA Bus's slower I/O devices from the PCI Bus. The Line Buffer is bi-directional and is used by ISA masters and the DMA controller to assemble and disassemble data. Only memory data written to or read from the PCI Bus by an ISA master or DMA is assembled/disassembled using this 8 byte line buffer. I/O cycles do not use the buffer.

Bits 0 and 1 of the PCI Control Register set the buffer to operate in either single transaction mode (bit = 0) or 8-byte mode (bit = 1). Note that ISA masters and DMA controllers can have their buffer modes configured separately.

In single transaction mode, the buffer will store only one transaction. For DMA/ISA master writes, this single transaction buffer looks like a posted write buffer. As soon as the ISA cycle is complete, a PCI cycle is scheduled. Subsequent DMA/ISA master writes are held off in wait-states until the buffer is empty. For DMA/ISA master reads, only the data requested is read over the PCI Bus. For instance, if the DMA channel is programmed in 16-bit mode, 16 bits of data will be read from PCI. As soon as the requested data is valid on the PCI bus, it is latched into the Line Buffer and the ISA cycle is then completed, as timing allows. Single transaction mode will guarantee strong read and write ordering through the buffers.

In 8 byte mode, for write data assembly, the Line Buffer acts as two individual 4 byte buffers working in ping pong fashion. For read data disassembly, the Line Buffer acts as one 8 byte buffer.

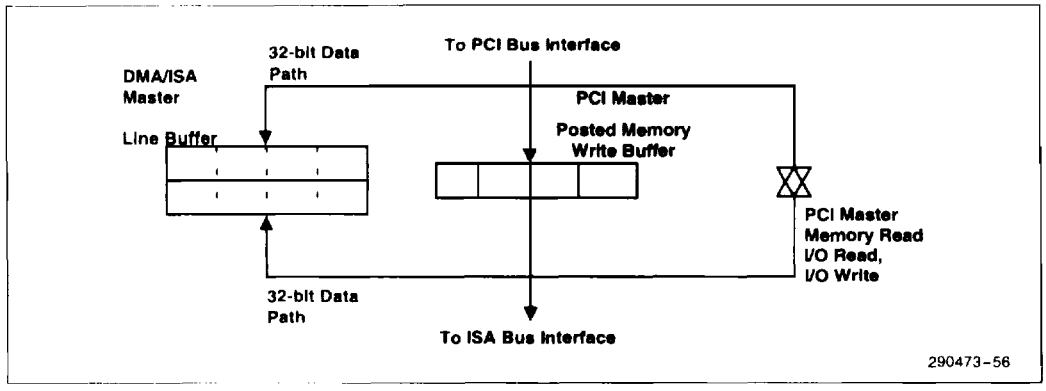

Figure 17. SIO Buffer Diagram

5.6.2 PCI MASTER POSTED WRITE BUFFER

PCI master memory write cycles destined to ISA memory are buffered in a 32-bit Posted Write Buffer. The PCI Memory Write and Memory Write and Invalidate commands are all treated as a memory write and can be posted, subject to the Posted Write Buffer status. The Posted Write Buffer has an address associated with it. A PCI master memory write can be posted any time the posted write buffer is empty and write posting is enabled (bit 2 of the PCI Control Configuration Register is set to a 1). Also, the ISA Bus must not be occupied. If the posted write buffer contains data, the PCI master write cycle is retried. If the posted write buffer is disabled, the SIO's response to a PCI master memory write is dependent on the state of the ISA Bus. If the ISA Bus is available and the posted write buffer is disabled, the cycle will immediately be forwarded to the ISA Bus (TRDY# will not be asserted until the ISA cycle has completed). If the ISA Bus is busy and the posted write buffer is disabled, the cycle is retried.

Memory read and I/O read and I/O write cycles do not use the 32-bit Posted Write Buffer.

5.6.3 BUFFER MANAGEMENT

Any time data is temporarily stored in the buffers between the ISA Bus and the PCI Bus, there are potential data coherency problems.

The SIO contains buffer management circuitry which guarantees data coherency by intercepting synchronization protocol between the buses and managing the buffers before synchronization communication between the buses is complete. The buffers are flushed or invalidated as appropriate before a bus cycle is allowed to occur in cases where data coherency could be lost.

5.6.3.1 DMA/ISA Master Line Buffer—Write State

When the DMA/ISA Master Line Buffer contains data that is to be written to the PCI Bus, it is in the Write State. The 8-byte line buffer is flushed when the line becomes full, when a subsequent write is a line miss, when a subsequent write would overwrite an already valid byte, or when a subsequent cycle is a read. The ISA master or DMA cycle that triggers the buffer flush will be held in wait-states until the flush is complete. The buffer is also flushed whenever there is a change in ISA Bus ownership as indicated by any DACK# signal going inactive.

Once the buffer is scheduled to be flushed to PCI, any PCI cycle to the SIO or ISA Bus will get retried by the SIO.

5.6.3.2 DMA/ISA Master Line Buffer—Read State

When the DMA/ISA Master Line Buffer contains data that has been read from the PCI Bus, it is in the Read State. The data in the buffer will be invalidated when the SIO accepts a PCI memory or I/O write cycle. The line buffer in the read state is also invalidated when a subsequent read is a line miss, or when a subsequent cycle is a write. The line buffer in the read state is not invalidated on a change of ISA ownership. Note that as bytes are disassembled from the line buffer, they are invalidated so that subsequent reads to the same byte will cause a line buffer miss.

5.6.3.3 PCI Master Posted Write Buffer

As soon as a PCI master has posted a memory write into the posted write buffer, the buffer is scheduled to be written to the ISA Bus. Any subsequent PCI master cycles to the SIO (including ISA Bus) will be retried until the posted write buffer is empty.

Prior to granting the ISA Bus to an ISA master or the DMA, the PCI master posted memory write buffer is flushed. Also, as long as the ISA master or DMA owns the ISA Bus, the posted write buffer is disabled. A PCI master write can not be posted while an ISA master or the DMA owns the ISA Bus.

5.7 SIO Timers

5.7.1 INTERVAL TIMERS

The SIO contains three counters that are equivalent to those found in the 82C54 programmable interval timer. The three counters are contained in one SIO timer unit, referred to as Timer-1. Each counter output provides a key system function. Counter 0 is connected to interrupt controller IRQ0 and provides a system timer interrupt for a time-of-day, diskette time-out, or other system timing functions. Counter 1 generates a refresh request signal and Counter 2 generates the tone for the speaker. Note that the 14.31818 MHz counters use OSC for a clock source.

Full details of this counter can be found in the 82C54 data sheet.

2

Table 25. Interval Timer Functions Table

Interval Timer Functions	
Function	**Counter 0—System Timer**
Gate	Always On
Clock In	1.193 MHz (OSC/12)
Out	INT-1 IRQ0
Function	**Counter 1—Refresh Request**
Gate	Always On
Clock In	1.193 MHz (OSC/12)
Out	Refresh Request
Function	**Counter 2—Speaker Tone**
Gate	Programmable-Port 61h
Clock In	1.193 MHz (OSC/12)
Out	Speaker

5.7.1.1 Interval Timer Address Map

Table 26 shows the I/O address map of the interval timer counters.

Table 26. Interval Timer Counters I/O Address Map

I/O Address	Register Description
040h	System Timer (Counter 0)
041h	Refresh Request (Counter 1)
042h	Speaker Tone (Counter 2)
043h	Control Word Register

Counter 0, System Timer

This counter functions as the system timer by controlling the state of IRQ0 and is typically programmed for Mode 3 operation. The counter produces a square wave with a period equal to the product of the counter period (838 ns) and the initial count value. The counter loads the initial count value one counter period after software writes the count value to the counter I/O address. The counter initially asserts IRQ0 and decrements the count value by two each counter period. The counter negates IRQ0 when the count value reaches 0. It then reloads the initial count value and again decrements the initial count value by two each counter period. The counter then asserts IRQ0 when the count value reaches 0, reloads the initial count value, and repeats the cycle, alternately asserting and negating IRQ0.

Counter 1, Refresh Request Signal

This counter provides the refresh request signal and is typically programmed for Mode 2 operation. The counter negates refresh request for one counter period (833 ns) during each count cycle. The initial count value is loaded one counter period after being written to the counter I/O address. The counter initially asserts refresh request, and negates it for 1 counter period when the count value reaches 1. The counter then asserts refresh request and continues counting from the initial count value.

Counter 2, Speaker Tone

This counter provides the speaker tone and is typically programmed for Mode 3 operation. The counter provides a speaker frequency equal to the counter clock frequency (1.193 MHz) divided by the initial count value. The speaker must be enabled by a write to port 061h (see Section 4.5.1 on the NMI Status and Control Register).

5.7.2 BIOS TIMER

5.7.2.1 Overview

The SIO provides a system BIOS Timer that decrements at each edge of its 1.04 MHz clock (derived by dividing the 8.33 MHz SYSCLK by 8). Since the state of the counter is undefined at power-up, it must be programmed before it can be used. Accesses to the BIOS Timer are enabled and disabled through the BIOS Timer Base Address Register. The timer continues to count even if accesses are disabled.

A BIOS Timer Register is provided to start the timer counter by writing an initial clock value. The BIOS Timer Register can be accessed as a single 16-bit I/O port or as a 32-bit port with the upper 16-bits being "don't care" (reserved). It is up to the software to access the I/O register in the most convenient way. The I/O address of the BIOS Timer Register is software relocatable. The I/O address is determined by the value programmed into the BIOS Timer Base Address Register.

The BIOS Timer clock has a value of 1.04 MHz using an 8.33 MHz SYSCLK input (an 8 to 1 ratio will always exist between SYSCLK and the timer clock). This allows the counting of time intervals from 0 ms to approximately 65 ms. Because of the PCI clock rate, it is possible to start the counter and read the value back in less than 1 μs. The expected value of the expired interval is 0, but depending on the state of the internal clock divisor, the BIOS Timer might indicate that 1 ms has expired. Therefore, accuracy of the counter is + 1 μs.

5.7.2.2 BIOS Timer Operations

A write operation to the BIOS Timer Register will initiate the counting sequence. The timer can be initiated by writing either the 16-bit data portion or the whole 32-bit register (upper 16 bits are "don't care"). After initialization, the BIOS timer will start decrementing until it reaches zero. Then it will stop decrementing (and hold a zero value) until initialized again.

After the timer is initialized, the current value can be read at any time and the timer can be reprogrammed (new initial value written), even before it reaches zero.

All write and read operations to the BIOS timer Register should include all 16 counter bits. Separate accesses to the individual bytes of the counter must be avoided since this can cause unexpected results (wrong count intervals).

5.8 Interrupt Controller

The SIO provides an ISA compatible interrupt controller which incorporates the functionality of two 82C59 interrupt controllers. The two controllers are cascaded so that 14 external and two internal interrupts are possible. The master interrupt controller provides IRQ[7:0] and the slave interrupt controller provides IRQ [15:8] (see Figure 18). The two internal interrupts are used for internal functions only and are not available to the user. IRQ2 is used to cascade the two controllers together and IRQ0 is used as a system timer interrupt and is tied to Interval Timer 1, Counter 0. The remaining 14 interrupt lines (IRQ1, IRQ3-IRQ15) are available for external system interrupts. Edge or level sense selection is programmable on a by-controller basis.

The Interrupt Controller consists of two separate 82C59 cores. Interrupt Controller 1 (CNTRL-1) and Interrupt Controller 2 (CNTRL-2) are initialized separately and can be programmed to operate in different modes. The default settings are: 80x86 Mode, Edge Sensitive (IRQ0-15) Detection, Normal EOI, Non-Buffered Mode, Special Fully Nested Mode disabled, and Cascade Mode. CNTRL-1 is connected as the Master Interrupt Controller and CNTRL-2 is connected as the Slave Interrupt Controller.

Note that IRQ13 is generated internally (as part of the coprocessor error support) by the SIO when bit 5 in the ISA Clock Divisor Register is set to a 1. When this bit is set to a 0, then the FERR#/IRQ13 signal is used as an external IRQ13 signal and has the same functionality as the normal IRQ13 signal. IRQ12/M is generated internally (as part of the mouse support) by the SIO when bit 4 in the ISA Clock Divisor Register is set to a 1. When set to a 0, the standard IRQ12 function is provided.

2

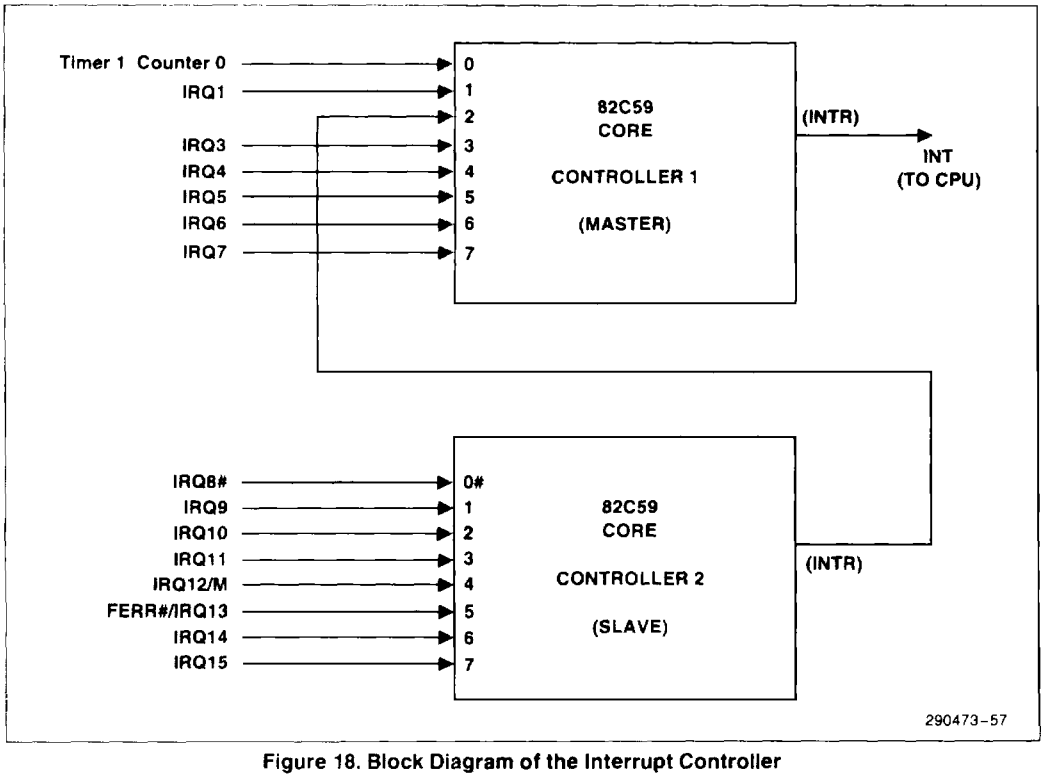

Figure 18. Block Diagram of the Interrupt Controller

Table 27 lists the I/O port address map for the interrupt registers:

Table 27. Interrupt Registers I/O Port Address Map

Interrupts	I/O Address	# of Bits	Register
IRQ[7:0]	0020h	8	CNTRL-1 Control Register
IRQ[7:0]	0021h	8	CNTRL-1 Mask Register
IRQ[15:8]	00A0h	8	CNTRL-2 Control Register
IRQ[15:8]	00A1h	8	CNTRL-2 Mask Register

IRQ0, IRQ2, (and possibly IRQ13 and IRQ12 if the "mouse" or floating point error logic is disabled in the ISA Clock Divisor Register), are connected to the interrupt controllers internally. The other interrupts are always generated internally and their typical functions are shown in Table 28:

Table 28. Typical Interrupt Functions

Priority	Label	Controller	Typical Interrupt Source
1	IRQ0	1	Interval timer 1, Counter 0 OUT
2	IRQ1	1	Keyboard
3-10	IRQ2	1	Interrupt from Controller 2
3	IRQ8#	2	Real Time Clock
4	IRQ9	2	Expansion Bus Pin B04
5	IRQ10	2	Expansion Bus Pin D03
6	IRQ11	2	Expansion Bus Pin D04
7	IRQ12/M	2	Mouse Interrupt
8	FERR#/IRQ13	2	Coprocessor Error
9	IRQ14	2	Fixed Disk Drive Controller Expansion Bus Pin D07
10	IRQ15	2	Expansion Bus Pin D06
11	IRQ3	1	Serial Port 2, Expansion Bus B25
12	IRQ4	1	Serial Port 1, Expansion Bus B24
13	IRQ5	1	Parallel Port 2, Expansion Bus B23
14	IRQ6	1	Diskette Controller, Expansion Bus B22
15	IRQ7	1	Parallel Port 1, Expansion Bus B21

5.8.1 EDGE AND LEVEL TRIGGERED MODES

There are two ELCR registers, one for each 82C59 bank. They are located at I/O ports 04D0h (for the Master Bank, IRQ[0:1,3:7]#) and 04D1h (for the Slave Bank, IRQ[8:15])#. They allow the edge and level sense selection to be made on an interrupt by interrupt basis instead of on a complete bank. Interrupts reserved for ISA use MUST be programmed for edge sensitivity (to ensure ISA compatibility). That is, IRQ (0,1,2,8#,13) must be programmed for edge sensitive operation. The LTIM bit (Edge/Level Bank select, offsets 20h, A0h) is disabled in the SIO. The default programming is equivalent to programming the LTIM bit (ICW1 bit 3) to a 0.

If an ELCR bit is equal to "0", an interrupt request will be recognized by a low to high transition on the corresponding IRQ input. The IRQ input can remain high without generating another interrupt.

If an ELCR bit is equal to "1", an interrupt request will be recognized by a "low" level on the corresponding IRQ input, and there is no need for an edge detection. For level triggered interrupt mode, the interrupt request signal must be removed before the EOI command is issued or the CPU interrupt must be disabled. This is necessary to prevent a second interrupt from occurring.

In both the edge and level triggered modes the IRQ inputs must remain active until after the falling edge of the first INTA#. If the IRQ input goes inactive before this time a DEFAULT IRQ7 will occur when the CPU acknowledges the interrupt. This can be a useful safeguard for detecting interrupts caused by spurious noise glitches on the IRQ inputs. To implement this feature the IRQ7 routine is used for "clean up" simply executing a return instruction, thus ignoring the interrupt. If IRQ7 is needed for other purposes a default IRQ7 can still be detected by reading the ISR. A normal IRQ7 interrupt will set the corresponding ISR bit, a default IRQ7 won't. If a default IRQ7 routine occurs during a normal IRQ7 routine, however, the ISR will remain set. In this case it is necessary to keep track of whether or not the IRQ7 routine was previously entered. If another IRQ7 occurs it is a default.

5.8.2 REGISTER FUNCTIONALITY

For a detailed description of the Interrupt Controller register set, please see Section 4.4, Interrupt Controller Register Description.

5.8.3 NON-MASKABLE INTERRUPT (NMI)

An NMI is an interrupt requiring immediate attention and has priority over the normal interrupt lines (IRQx). The SIO indicates error conditions by generating a non-maskable interrupt.

NMI interrupts are caused by the following conditions:

1. System Errors on the PCI Bus. SERR# is driven low by a PCI resource when this error occurs.
2. Parity errors on the add-in memory boards on the ISA expansion bus. IOCHK# is driven low when this error occurs.

The NMI logic incorporates two different 8-bit registers. These registers are addressed at locations 061h and 070h. The status of Port (061h) is read by the CPU to determine which source caused the NMI. Bits set to 1 in these ports show which device requested an NMI interrupt. After the NMI interrupt routine processes the interrupt, the NMI status bits are cleared by the software. This is done by setting the corresponding enable/disable bit high. Port (070H) is the mask register for the NMI interrupts. This register can mask the NMI signal and also disable or enable all NMI sources.

The individual enable/disable bits clear the NMI detect flip-flops when disabled.

All NMI sources can be enabled or disabled by setting Port 070h bit 7 to a 0 or 1. This disable function does not clear the NMI detect flip-flops. This means, if NMI is disabled then enabled via Port 070h, then an NMI will occur when Port 070h is re-enabled if one of the NMI detect flip-flops had been previously set.

To ensure that all NMI requests are serviced, the NMI service routine software needs to incorporate a few very specific requirements. These requirements are due to the edge detect circuitry of the host microprocessor, 80386 or 80486. The software flow would need to be the following:

1. NMI is detected by the processor on the rising edge of the NMI input.
2. The processor will read the status stored in port 061h to determine what sources caused the NMI. The processor may then set to 0 the register bits controlling the sources that it has determined to be active. Between the time the processor reads

the NMI sources and sets them to a 0, an NMI may have been generated by another source. The level of NMI will then remain active. This new NMI source will not be recognized by the processor because there was no edge on NMI.

3. The processor must then disable all NMI's by setting bit 7 of port 070H to a 1 and then enable all NMI's by setting bit 7 of port 070H to a 0. This will cause the NMI output to transition low then high if there are any pending NMI sources. The CPU's NMI input logic will then register a new NMI.

Section 4.5 Control Registers, contains a detailed description of the NMI Status and Control Register (port 061h) and the NMI Enable and Real-Time Clock Address Register at port 070h.

5.9 Utility Bus Peripheral Support

The Utility Bus is a secondary bus buffered from the ISA Bus used to interface with peripheral devices that do not require a high speed interface. The buffer control for the lower 8 data signals is provided by the SIO via two control signals; UBUSOE# and UBUSTR. Figure 19 shows a block diagram of the external logic required as part of the decode and Utility Bus buffer control.

The SIO provides the address decode and three encoded chip selects to support:

1. Floppy Controller
2. Keyboard Controller
3. Real Time Clock
4. IDE Drive
5. 2 Serial Ports (COM1 and COM2)
6. 1 Parallel Port (LPT1, 2, or 3)
7. BIOS Memory
8. Configuration Memory (8 Kbyte I/O Mapped)

The SIO also supports the following functions:

1. Floppy DSKCHG Function
2. Port 92 Function (Alternate A20 and Alternate Reset)
3. Coprocessor Logic (FERR# and IGNNE# Function)

The binary code formed by the three Encoded Chip Selects determines which Utility Bus device is selected. The SIO also provides an Encoded Chip Select Enable signal (ECSEN#) that is used to select between the two external decoders. A zero selects decoder 1 and a one selects decoder 2. The table below shows the address decode for each of the Utility Bus devices.

Table 29. NMI Source Enable/Disable and Status Port Bits

NMI Source	I/O Port Bit for Status Reads	I/O Port Bit for Enable/Disable
IOCHK#	Port 061h, Bit 6	Port 061h, Bit 3
SERR#	Port 061h, Bit 7	Port 061h, Bit 2

Table 30. Encoded Chip Select Summary Table

ECSADDR2	ECSADDR1	ECSADDR0	ECSEN#	Address Decoded	External Chip Select	Note	Cycle Type
Decoder 1							
0	0	0	0	70h, 72h, 74, 76h	RTCALE#		I/O W
0	0	1	0	71h, 73h, 75h, 77h	RTCCS#		I/O R/W
0	1	0	0	60h, 62h, 64h, 66h	KEYBRDCS#		I/O R/W
0	1	1	0	000E0000h–000FFFFFh FFFE0000h–FFFFFFFFh FFF80000h–FFFDFFFFh	BIOSCS#	1	MEM R/W
1	0	0	0	3F0h–3F7h (primary) 370h–377h (secondary)	FLOPPYCS#	2	I/O R/W
1	0	1	0	1F0h–1F7h (primary) 170h–177h (secondary)	IDECS0#	2	I/O R/W
1	1	0	0	3F6h–3F7h (primary) 376h–377h (secondary)	IDECS1#	2	I/O R/W
1	1	1	0	Reserved			
Decoder 2							
0	0	0	1	Reserved			
0	0	1	1	0C00h	CPAGECS#	3	I/O R/W
0	1	0	1	0800h–08FFh	CFIGMEMCS#	3	I/O R/W
0	1	1	1	3F8h–3FFh (COM1) -or- 2F8h–37Fh (COM2)	COMACS#	4	I/O R/W
1	0	0	1	3F8h–3FFh (COM1) -or- 2F8h–37Fh (COM2)	COMBCS#	4	I/O R/W
1	0	1	1	3BCh–3BFh (LPT1) 378h–37Fh (LPT2) 278h–27Fh (LPT3)	LPTCS#	5	I/O R/W
1	1	0	1	Reserved			
1	1	1	1	Idle State			

NOTES:

1. The encoded chip select signals for BIOSCS# will always be generated for accesses to the upper 64 KB at the top of 1 MByte (F0000h–FFFFFh) and its aliases at the top of the 4 GB and 4 GB-1 MByte. Access to the lower 64 KByte (E0000h–EFFFFh) and its aliases at the top of 4 GB and 4GB–1MB can be enabled or disabled through the SIO. An additional 384 KB of BIOS memory at the top of 4 GB (FFFD0000h–FFFDFFFFh) can be enabled for BIOS use.
2. The primary and secondary locations are programmable through the SIO. Only one location range can be enabled at any one time. The floppy and IDE share the same enable and disable bit (i.e. if the floppy is set for primary, the IDE is also set for primary).
3. These signals can be used to select additional configuration RAM.
4. COM1 and COM2 address ranges can be programmed for either port A (COMACS#) or port B (COMBCS#).
5. Only one address range (LPT1, LPT2, or LPT3) can be programmed at any one time.

Port 92h Function

The SIO integrates the Port 92h Register. This register provides the alternate reset (ALTRST) and alternate A20 (ALT_A20) functions. Figure 19 shows how these functions are tied into the system.

DSKCHG Function

DSKCHG is tied directly to the DSKCHG signal of the floppy controller. This signal is inverted and driven onto system data line 7 (SD7) during I/O read cycles to floppy address locations 3F7h (primary) or 377 (secondary) as indicated by Table 31.

Table 31. DSKCHG Summary Table

FLOPPYCS# Decode	IDECSx# Decode	State of SD7 (Output)	State of UBUSOE#
Enabled	Enabled	Tri-stated	Enabled
Enabled	Disabled	Driven via DSKCHG	Disabled
Disabled	Enabled	Tri-stated	Enabled(1)
Disabled	Disabled	Tri-stated	Disabled

NOTE:
1. For this mode to be supported, extra logic is required to disable the U-bus transceiver for accesses to 3F7/377. This is necessary because of potential contention between the Utility bus buffer and a floppy on the ISA Bus driving the system bus at the same time during shared I/O accesses.

Coprocessor Error Support

If bit 5 in the ISA Clock Divisor Register is set to a one, the SIO will support coprocessor error reporting through the FERR#/IRQ13 signal.

FERR# is tied directly to the Coprocessor error signal of the CPU. If FERR# is driven active in this mode (coprocessor error detected by the CPU), an internal IRQ13 is generated and the INT output from the SIO is driven active. When a write to I/O location F0h is detected, the SIO negates IRQ13 and drives IGNNE# active. IGNNE# remains active until FERR# is driven inactive. Note that IGNNE# is not generated unless FERR# is active.

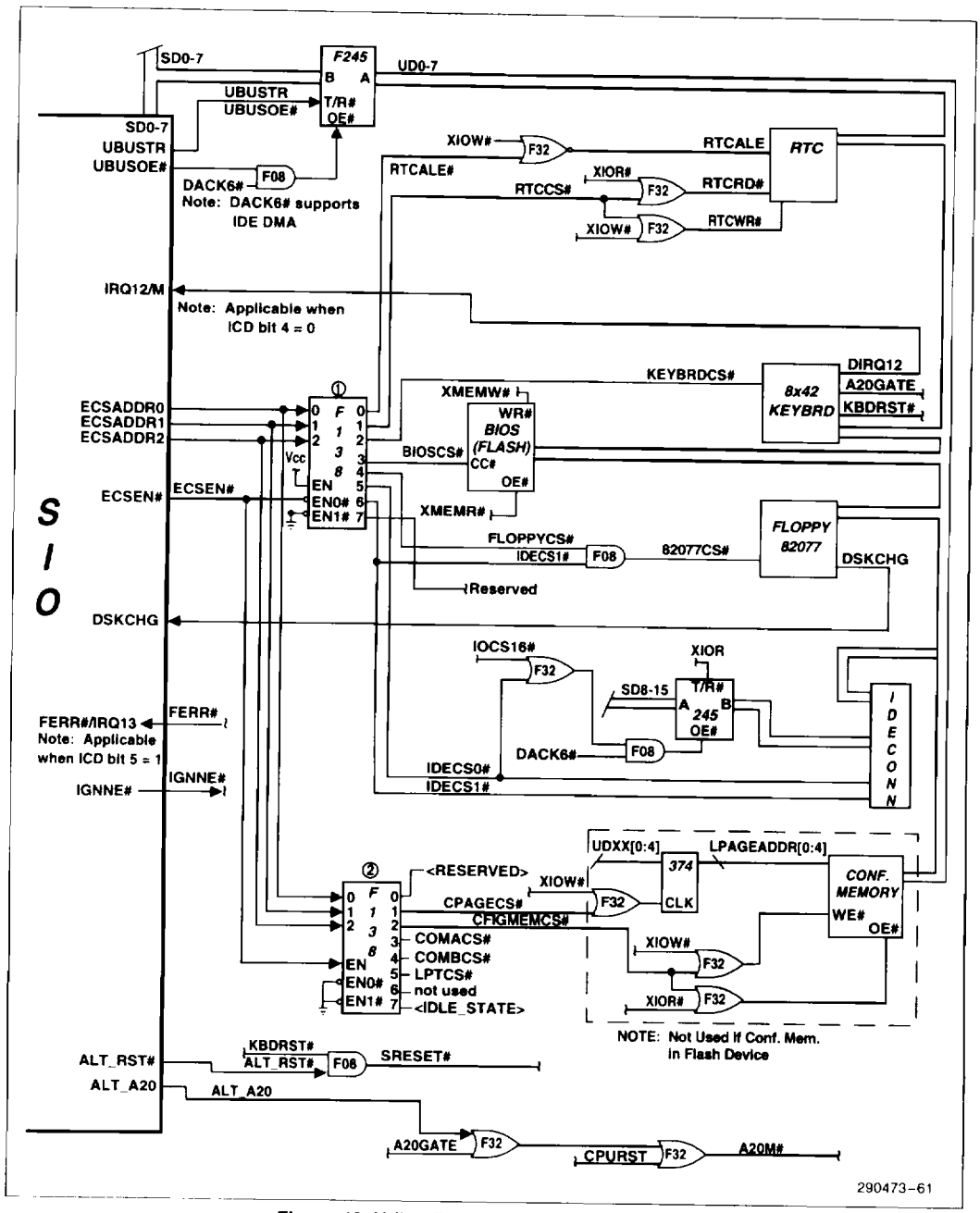

Figure 19. Utility Bus External Support Logic

Utility Bus accesses by the SIO, by an ISA master, and by the DMA is shown in Figure 20 and Figure 21. UBUSOE# and UBUSTR are driven differently for DMA cycles as shown in Figure 21.

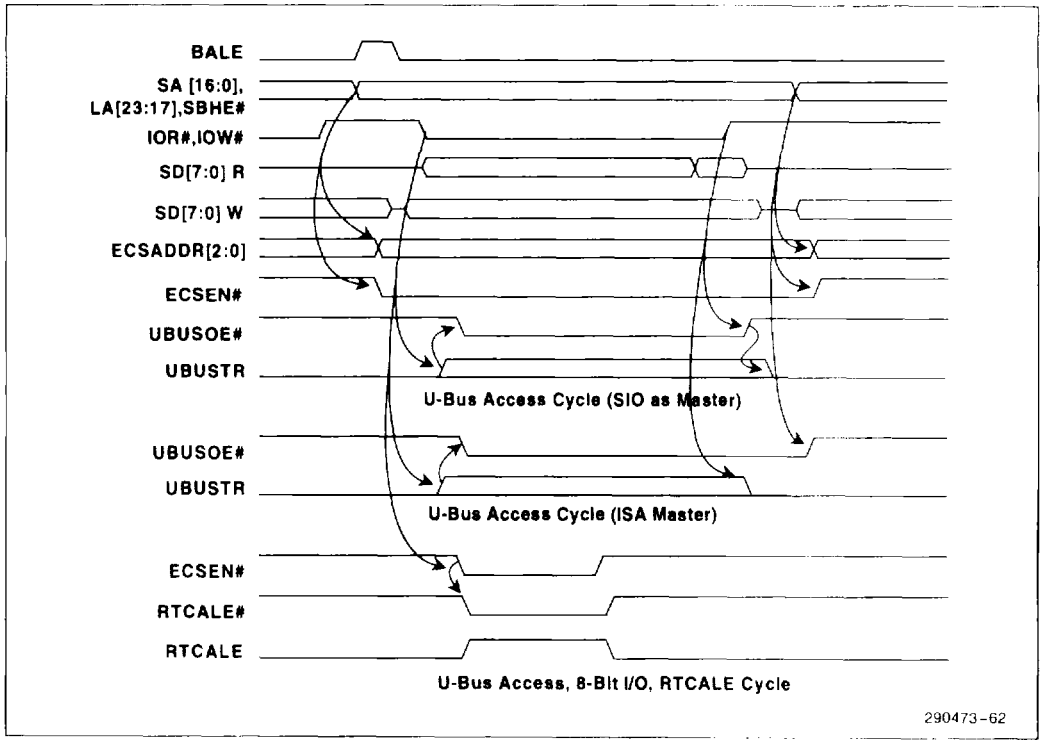

Figure 20. Utility Bus Access (SIO and ISA Master)

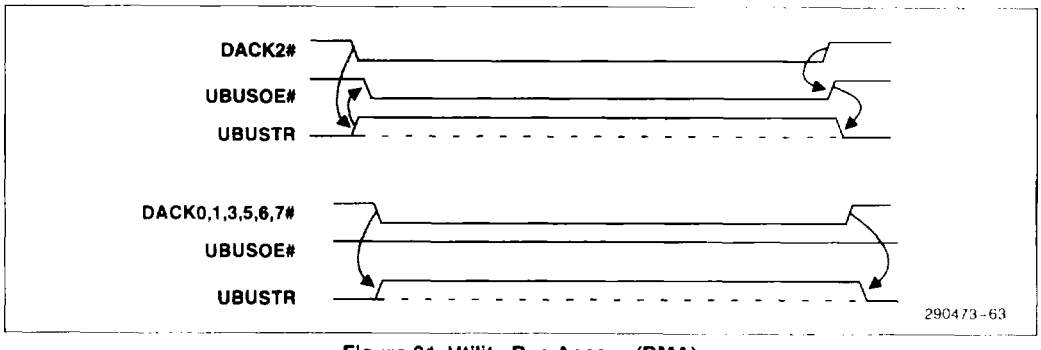

Figure 21. Utility Bus Access (DMA)

5.10 Power Management

The SIO's power management architecture is based around three core functions:

1. SMM (System Management Mode)
2. Clock Throttling
3. APM (Advanced Power Management Interface)

SMM is a mode during which an S Series Processor is executing SMM code from a secure memory space (SMRAM). SMM is invoked through the assertion of an SMI (System Management Interrupt). Physically, this is signaled over the SMI# pin. SMI's are triggered by various hardware and software events. SMRAM is used to store the SMM code which is really the SMI interrupt handler routine.

Clock Throttling will be used to reduce the power consumption of the CPU. STPCLK# is the physical signal used to control the CPU's clock.

APM creates an interface to allow the Operating System to communicate with the SMM code.

Figure 22 shows how the power management signals are connected in a Saturn based system with an S-series CPU.

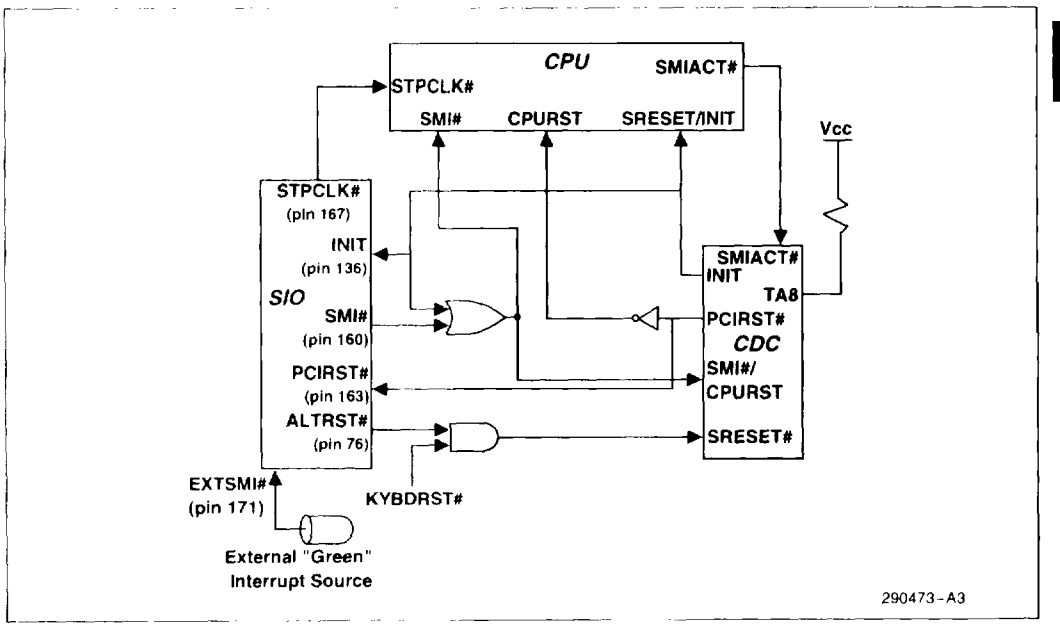

Figure 22. Power Management

6.0 ELECTRICAL CHARACTERISTICS

6.1 Absolute Maximum Ratings*

Case Temperature under Bias	65°C to + 110°C
Storage Temperature	65°C to + 150°C
Supply Voltages with Respect to Ground	0.5V to V_{CC} + 0.5V
Voltage On Any Pin	0.5V to V_{CC} + 0.5V

** WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*