2 Functional Description

2.1 Introduction

The block diagram in Figure 2.1 on page 2-3 shows the different data paths in the Spanner in its function as 68K target and PCI target. The figure also illustrates its role in translating during interrupts from 68K interrupters to the PCI bus. Detailed discussion of each of these functions is found in:

"Spanner as 68K Target (PCI Initiator)" on page 2-4

"Spanner as PCI Target (68K Initiator)" on page 2-8

"PCI Address Decoding" on page 2-12

"Interrupts" on page 2-14

"Register Access" on page 2-16

The various data paths in the Spanner are also described briefly below in the Functional Overview.

2.1.1 Functional Overview

The Spanner bridges a PCI bus to a 68040 compatible bus, with initiator and target capabilities on both interfaces. Most transactions between the 68K bus and the PCI bus are coupled, meaning that the source bus must wait for data acknowledgment from the destination bus. However, write transactions from the 68K bus to the PCI bus are decoupled using a posted write FIFO (PWFIFO). Data transfer through the Spanner is maximized for reads and writes originating from a 68K initiator. This allows for optimum throughput to and from high-bandwidth PCI-based slave devices.

Access from the 68040 Bus

Through the Spanner chip, 68K initiators can access 4 Gbytes of Memory space on the PCI bus (I/O space and Configuration space cannot be accessed by a 68K initiator). The Spanner does not provide programmable slave images on the 68K interface, so the user must supply external logic on the 68K side for address decoding.

Writes to the PCI bus are all performed through a PWFIFO, which is a longword (32 bits) wide and four entries deep. This optimizes the transfer of burst line transactions across the 68K interface. Reads are coupled, and burst-line read transactions from a 68K initiator are directly translated into a single PCI transaction with 4 longword data beats. Flow control on the PCI side is performed with IRDY#. Since the data acknowledgment on the 68K bus lags behind the PCI read by only one clock cycle, data transfer from the PCI side is virtually uninterrupted by the Spanner.

Interrupts from a 68K source are translated to the PCI bus as a single PCI interrupt signal, INTA#. The interrupt controller on the PCI bus generates an IACK cycle on the 68K bus by accessing a special 2 Kbyte address space in the Spanner's Fixed Offset PCI slave image.

Access from the PCI Bus

The Spanner chip provides 2 programmable images on the PCI interface. These images can be mapped anywhere, and to any size, within the available 4 Gbytes (with a granularity of 64 Kbytes). The Spanner programmable PCI images are in PCI memory space, so only PCI memory transactions are decoded by the Spanner. The Spanner also provides a 62 Kbyte Fixed Offset slave image which can be accessed in PCI Memory and/or I/O space. PCI transactions are mapped to a single transaction type on the 68K bus (corresponding to a 68040 normal supervisor data access). The only exception to this occurs during an access to a special space in the Fixed Offset image, which results in an IACK cycle on the 68040 bus.

Since there is no guarantee that data acknowledgment from the 68K side will meet PCI latency requirements, the Spanner chip performs a target disconnect after the first data beat of every PCI transaction. This means that all transactions from a PCI master are broken into single cycle transfers.

Endian Issues

The names for the two formats of data storage, big endian and little endian, come from a war described in Jonathan Swift's "Gulliver's Travel's", where thousands died in a dispute over whether to break eggs from the little end or the big end. Motorola®'s architecture is big endian, while PCI (from an Intel® origin) is little endian.

The Spanner maps between these two endian systems using Address Invariance. With Address Invariance, data structures and arrays maintain their order in memory independent of the endian architecture in which they reside. For example, when transferring a string of characters like "this will be legible", each character and word is in the same relative position after it is transferred to the other endian system. A CPU on either bus can still read back the text regardless of whether it was transferred byte by byte or word by word. The Spanner swaps byte lanes during transfers between the two buses to ensure Address Invariance is preserved.

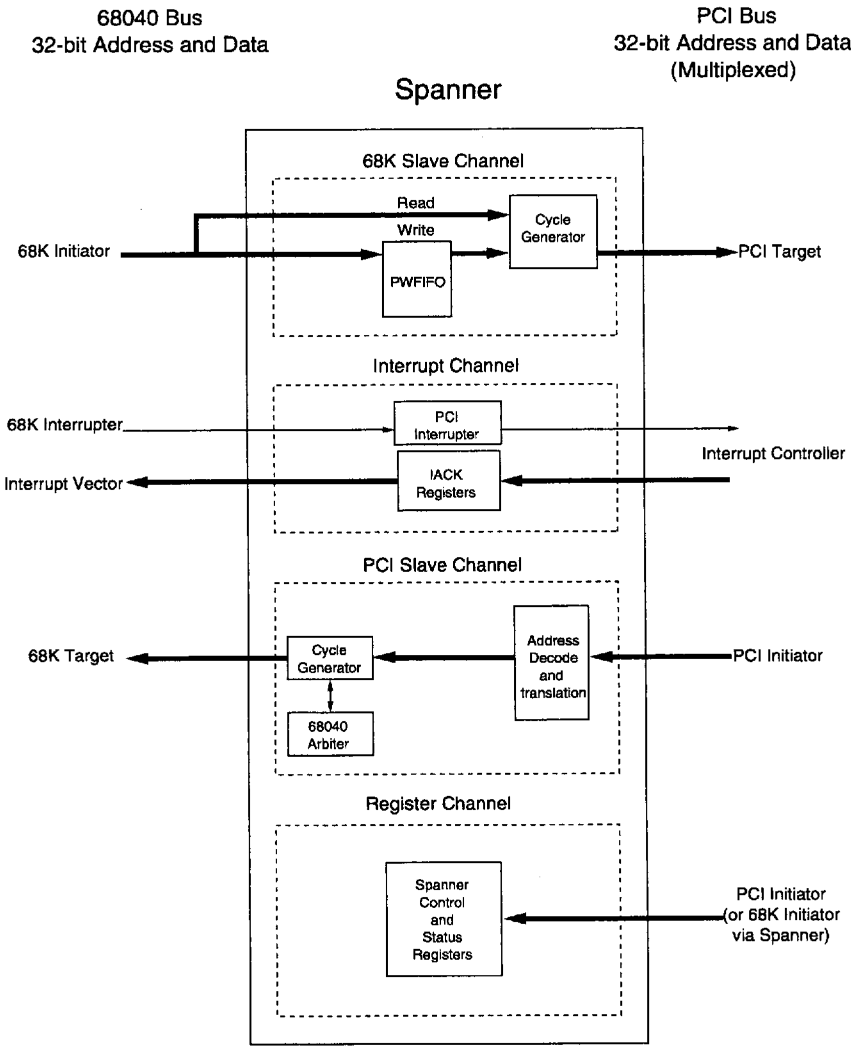


Figure 2.1 : Functional Block Diagram for the Spanner Chip

2.2 Spanner as 68K Target (PCI Initiator)

The Spanner becomes a 68K Target and PCI initiator when a 68K initiator attempts to access a PCI resource. In order for the Spanner to serve as PCI initiator, it must be enabled accordingly by setting the MASTR bit in the PCICOM register (Table A.4). Read transactions are coupled, while write transactions are always decoupled using the Posted Write FIFO. The Spanner's 68K port uses a synchronous 68040 protocol.

Note that if the Spanner requests the PCI bus and is granted it while another PCI master has ownership, then the Spanner will hold REQ# asserted until both FRAME# and IRDY# are sampled negated. On the rising edge of CLK when these two signals are sampled inactive, the Spanner negates REQ# and asserts FRAME#.

2.2.1 Address Phase

The Spanner becomes 68K target when $\overline{\text{LTS}}$ is asserted. Since the Spanner does not provide programmable images on the 68K interface (see "PCI Address Decoding" on page 2-12), the user needs to provide external address decoding for the Spanner on the 68K bus. Note that since there is no Spanner chip select pin, the user must use $\overline{\text{LTS}}$ to qualify 68K accesses to the Spanner.

The address bits on the 68K bus are mapped directly through to the PCI bus (no translation offsets are used by the Spanner with 68K-initiated transactions). All but the lower 2 bits of the address on the 68K bus are latched by the Spanner, on the rising edge of CLK where $\overline{\text{LTS}}$ is asserted,. However, all address and transfer attributes must be held valid until $\overline{\text{LTA}}$ or $\overline{\text{LTEA}}$ is asserted by the Spanner. The Spanner maps all 68K transactions to PCI Memory space.

2.2.2 Data Phase

All read transactions are handled by the Spanner as coupled (also called "pass-through", see Figure 2.2 below). Note that since TA lags behind TRDY# by only one clock cycle, a burst-line read from a 68K initiator is directly translated to a single PCI transaction (with 4 longword data beats).

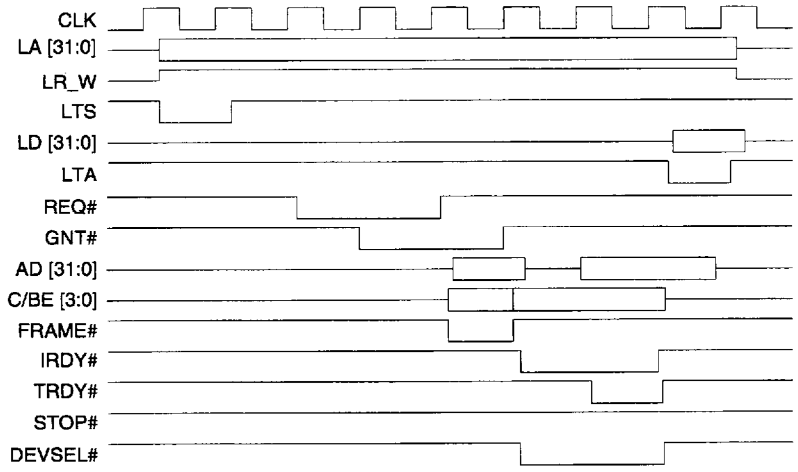


Figure 2.2 : 68040-Initiated Read Transaction

Write transactions from a 68K initiator are always sent to the PCI bus through a Posted Write FIFO (PWFIFO). The 68K initiator writing to the Spanner receives immediate data acknowledgment once the transaction is latched by the Spanner and does not need to wait for data acknowledgment from the PCI bus (see Figure 2.3 below).

If a line write from a 68040 initiator is attempted at a non-32 byte aligned address, then the burst write on the PCI bus may or may not be linear depending on the response of the PCI target. For example, in a linear burst the next longword address after 0x0C is 0x10. However, if the address wraps then the next longword address after 0x0C is 0x00. The address will wrap only if the PCI target breaks up the burst (through a target disconnect) on a wrappable boundary. For example, consider a burst transaction with longword addresses 0x04, 0x08. 0x0C, and 0x10. If the PCI target breaks the transaction into two double longword bursts, then the address will not wrap. However, if the PCI target breaks the transaction into single longword transactions, then the address will wrap (data appears at addresses 0x04, 0x08. 0x0C, and 0x00).

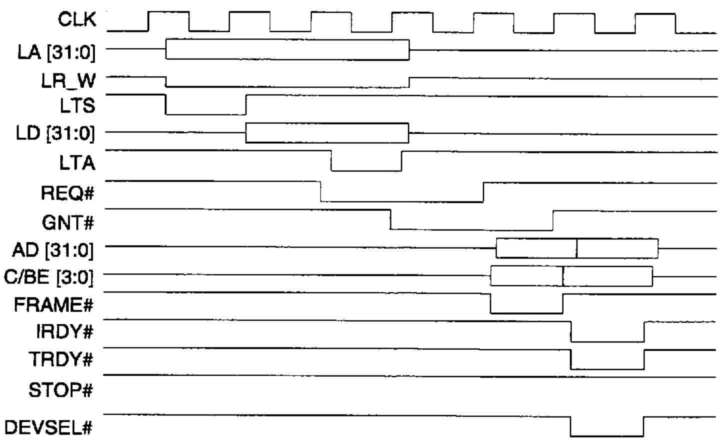


Figure 2.3 : 68040-Initiated Write Transactions

The PWFIFO is a longword (32 bits) wide and 4 entries deep. The Spanner writes to a PCI target from the PWFIFO until the FIFO is empty. Note that there is no packing/unpacking performed by the PWFIFO. Only one transaction is held in the PWFIFO at one time, and the Spanner inserts wait states on the 68K bus (bus keeping $\overline{\text{LTA}}$ negated) if a 68K initiator attempts to write to the PWFIFO before it is empty.

Byte lanes from the 68K bus are automatically byte swapped to the appropriate byte lanes on the PCI bus so that Address Invariance is maintained between the big-endian and little-endian memory structures. In addition, the Spanner directly maps byte lane activity on the 68040 bus to the PCI bus by enabling the appropriate PCI byte lanes.

The Spanner generates parity on the PCI bus during a write as PCI initiator, but does not monitor parity during reads.

2.2.3 Terminations

Since write transactions are decoupled, errors from the PCI target are not passed back to the 68K initiator. However, note that if PCI transactions terminate with Target Abort or Master Abort, then this will be recorded in the PCI Command and Status register (Table A.4). If the Spanner terminates with a master abort, it sets the RCVMTA bit in the PCICS register (Table A.4). Similarly, if it receives a target abort while it is PCI initiator, it sets the RCTVA bit in the PCICS register. These status bits are cleared by writing a “1” to them. During read transactions (which are coupled), PCI terminations are translated to the 68K bus according to Table 2.1 below.

Table 2.1 : Translation of PCI Terminations to the 68K Interface

PCI Termination	Signals on 68K Interface
Target Disconnect	TA
Target Abort	TEA
Target Retry	see text below

If there is a target retry on a 68040-initiated read or write, then the Spanner continues to retry the PCI target until it receives TRDY# or a target abort. If the retry occurs during a read, then the acknowledgment from the PCI bus (when it is eventually received) is relayed to the 68040 bus. The 68040 initiator must not be backed off or retried once an access is made to the Spanner until the Spanner responds with LTA or LTEA to terminate the transaction. For retries during write transactions, any subsequent cycles from the 68K bus to the PCI bus are blocked until the retried transaction is completed.

Caution: A 68K time-out cannot be used to halt PCI retries.

2.3 Spanner as PCI Target (68K Initiator)

The Spanner becomes PCI target when a PCI initiator accesses one of the Spanner's two programmable images, its Fixed Offset image, or the Spanner register space. Since register access is discussed elsewhere (see "Register Access" on page 2-16), this section only describes transactions where a PCI initiator is accessing a 68K resource through the Spanner's PCI images.

All accesses from the PCI bus to the 68K bus are coupled (single cycle) transactions. Note that the operation of the 68K interface is essentially identical to the 68040 protocol.

2.3.1 Address Phase

The Spanner decodes any access to its two programmable PCI slave images or its one Fixed Offset slave image (see "PCI Address Decoding" on page 2-12). The two programmable images can be located anywhere within the 4 Gbytes of PCI Memory space (dual address cycles are ignored). The 62 Kbyte Fixed Offset image can be located together with the Spanner's register space in Memory and/or I/O space with a base address programmed by the MEMBASE and IOBASE registers, respectively.

The Spanner's two programmable images decode Memory Reads and Memory Writes on the PCI bus. Memory Read Line transactions are accepted as Memory Reads. Memory Write Line and Memory Write Invalidate transactions are accepted as Memory Writes. The Fixed Offset image decodes Memory reads and writes and/or I/O reads and writes. All decoded PCI transactions (to the programmable images or the Fixed Offset image) are mapped to a single address space on the 68K bus (function code lines are set as LTT[1:0]=0 and LTM[2:0]=5, equivalent to normal supervisor data access on a 68040 bus). The only exception to this is during accesses to the upper 2 Kbytes of the Fixed Offset image. In this case, the LTT[1:0] and LTM[2:0] lines are set to generate an IACK cycle on the 68040 bus (see "Interrupt Acknowledgment" on page 2-14).

The address decoded by the two programmable images is modified for the 68040 bus according to the translation offset programmed for that PCI image (see "PCI Address Decoding" on page 2-12). The translation offset allows the user to map the PCI image anywhere within the 4 Gbytes of 68K address space.

The Spanner does not decode LOCK# on the PCI interface.

2.3.2 Arbitration on the 68K Bus

The Spanner has an internal arbiter which can arbitrate between three devices on the 68K bus (including itself). The three request inputs are $\overline{\mathrm{ABR}}$[2:0] and the three request outputs are $\overline{\mathrm{ABG}}$[2:0]. The arbiter follows 68040 priority arbitration protocol where abr2 has the highest priority and abr0 has the lowest. Note that the Spanner's internal arbiter does not support locking, so users may wish to employ an external arbiter for 68K designs using a locking protocol.

If an external arbiter is implemented, the Spanner asserts its bus request signal ($\overline{\mathrm{LBR}}$) and should receive the bus grant signal ($\overline{\mathrm{LBG}}$) in accordance to 68040 protocol. The Spanner will become master of the 68040 bus once $\overline{\mathrm{LBG}}$ has been sampled low, and will drive the bus when $\overline{\mathrm{LBB}}$ has been released by the previous master. Once it obtains the bus, it asserts the 68040 bus busy signal ($\overline{\mathrm{LBB}}$), and holds it asserted until completion of the single outstanding PCI access.

Parking the 68040 bus at the Spanner has no effect other than speeding up acquisition of the bus for the next transaction. If the bus is parked at the Spanner, then the Spanner starts driving the 68040 bus on the next rising clock edge after it drives the bus request.

2.3.3 Data Phase

All transactions from a PCI initiator are treated as coupled (also called "pass-through"), which means that the PCI transaction only terminates after the 68040 transaction has completed. After the address phase, the PCI bus lies idle while the Spanner establishes a coupled connection with the 68K resource. Once the coupled connection is established, the first data beat is sent to the 68K bus. For aligned data transfers, the Spanner performs beat-for-beat translation between the two buses. However, if the byte lane enabling on the PCI bus does not correspond to an aligned 68K transaction (i.e. unaligned transfers, or PCI transactions with noncontiguous byte enables), then the Spanner breaks the PCI transaction into multiple 68K transactions as required (see byte lane mapping in Table 2.2 below). If the transaction is broken into two 68040 cycles, then bus busy is held during the two 68K transactions (the 68K bus is held between the two cycles). Once the final 68K transfer has been acknowledged with $\overline{\mathrm{LTA}}$ (or $\overline{\mathrm{LTEA}}$), the Spanner asserts TRDY# (or target abort) to acknowledge the PCI data beat.

Note also that the Spanner performs byte swapping such that Address Invariance is preserved between the two endian systems (big endian on the 68K bus versus little endian on the PCI bus).

Table 2.2 : Byte Lane Mapping for PCI to 68K Transactions

BE[3:0]#	Transaction Number	LSIZ 1	LSIZ 0	LA 1	LA 0	Active 68040 Byte Lanes			
						LD [31:24]	LD [23:16]	LD [15:08]	LD [07:00]
1110	1	0	1	0	0	X	-	-	-
1101	1	0	1	0	1	-	X	-	-
1011	1	0	1	1	0	-	-	X	-
0111	1	0	1	1	1	-	-	-	X
1100	1	1	0	1	0	-	-	X	X
1001	1	0	1	0	1	-	X	-	-
	2	0	1	1	0	-	-	X	-
0011	1	1	0	1	0	-	-	X	X
0110	1	0	1	0	0	X	-	-	-
	2	0	1	1	1	-	-	-	X
1000	1	1	0	0	0	X	X	-	-
	2	0	1	1	0	-	-	X	-
0001	1	0	1	0	1	-	X		
	2	1	0	1	0	-	-	X	X
0101	1	0	1	0	1	-	X	-	-
	2	0	1	1	1	-	-	-	X
0010	1	0	1	0	0	X	-	-	-
	2	1	0	1	0	-	-	X	X
0100	1	1	0	0	0	X	X	-	-
	2	0	1	1	1	-	-	-	X
1010	1	0	1	0	0	X	-	-	-
	2	0	1	1	0	-	-	X	-
0000	1	0	0	0	0	X	X	X	X
1111	1	Note 1							

Notes:

1. No 68040 transaction is initiated and the Spanner performs a target disconnect on the PCI side.

Since there is no guarantee that PCI latency requirements can be met on the 68K bus, the Spanner performs a target disconnect on the PCI bus during the first data beat of every transaction by asserting STOP# together with TRDY#. This means that all transactions from a PCI initiator are broken into single cycle transactions.

The Spanner generates parity during reads as a PCI target, but does not monitor parity during writes.

2.3.4 Terminations

Terminations from the 68K interface are translated to the PCI initiator according to Table 2.3 below.

Table 2.3 : Translation of 68K Terminations

68 K Termination	PCI Termination
TA	Target Disconnect
TEA	Target Abort
TA + TEA (Retry)	Target Retry

Note that if the Spanner terminates the PCI transaction with target abort on the PCI bus, it sets the SIGTA bit in the PCICS register (Table A.4). Clear this status bit by writing a "1" to it. As PCI target, the Spanner will also terminate with target retry if it encounters deadlock due to simultaneous access by a 68K initiator to the PCI bus.

2.4 PCI Address Decoding

The Spanner recognizes two types of accesses on the PCI interface: accesses destined for a 68K resource, and accesses to its own register space. Address decoding for the Spanner's register space is described in "Register Access" on page 2-16. This section describes how to program the PCI images for 68K accesses.

2.4.1 Programmable PCI Slave Images

The Spanner provides two programmable images which can be mapped anywhere within the 4 Gbytes of PCI Memory space. Each of these images is enabled by setting its corresponding enable bit in the PCI Slave Image Control register (see Table A.11 or Table A.13).The two programmable images can be any size with 64 Kbyte granularity as long as their combined size does not exceed the 4 Gbyte limit. The most significant 16 bits for the PCI base address is programmed with the PCI Slave Image Base and Bound register (see Table A.10 or Table A.12). The upper 16 bits of the bound address of the PCI image is programmed with the same register. For example, if PCIBS is programmed as 0x2000 and PCIBD is programed with 0x2FFF, then the Spanner decodes any access between 0x2000 0000 and 0x2FFF FFFF.

When the PCI access is mapped to the 68K bus, an address offset can be used to translate the PCI address to any 68K address. This can be done using the PCI Slave Image Control register (see Table A.11 or Table A.13). The address on the 68K bus is the sum of the offset value and the original PCI address. In the event of overlap in the programming of the slave images, the register access image takes priority followed by PCI slave image 0.

2.4.2 The Fixed Offset Slave Image

The Fixed Offset slave image is programmed together with the Spanner's register space into Memory space and/or I/O space according to the base address set in the MEMBASE or IOBASE registers (see "Address Mapping for the Fixed Offset Image" on page 2-13). This fixed image is enabled in Memory and/or I/O space using the MEMSP and IOSP bits in the PCI Command register (Table A.4). Access to the lower 60 Kbytes of the Fixed Offset image results in a 68040 cycle similar to one produced by an access to the programmable PCI slave images (LTT[1:0]=0 and LTM[2:0]=5, equivalent to normal supervisor data access). However, access to the upper 2 Kbytes of the Fixed Offset image can be used to generate IACK cycles on the 68040 bus. See "Interrupt Acknowledgment" on page 2-14 for a description of how this 2 Kbyte range is used for 68040 IACK cycle generation. Accesses to the Fixed Offset image are mapped to 68K address such that the address is always aligned to FFF4 0000.

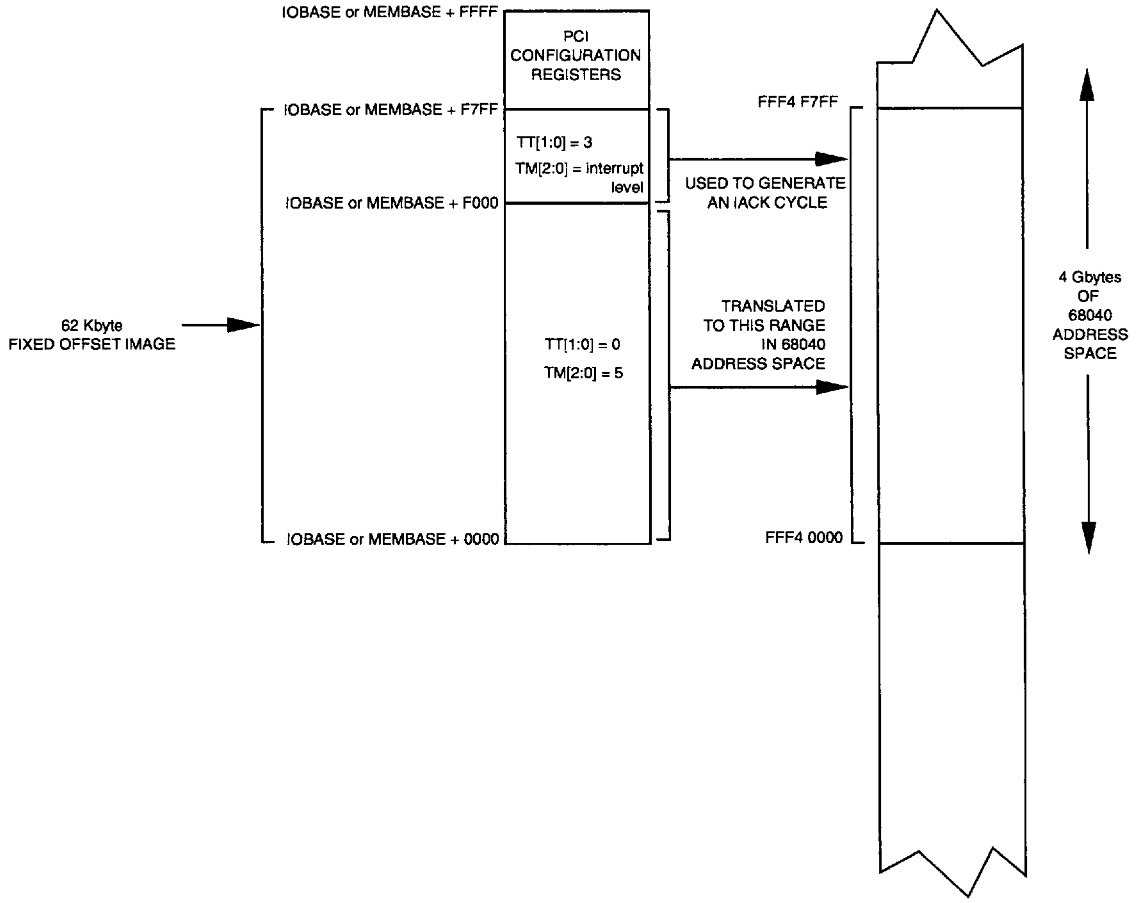

IOBASE or MEMBASE + FFFF
PCI CONFIGURATION REGISTERS
IOBASE or MEMBASE + F7FF
TT[1:0] = 3
TM[2:0] = interrupt level
IOBASE or MEMBASE + F000
USED TO GENERATE AN IACK CYCLE
FFF4 F7FF
62 Kbyte FIXED OFFSET IMAGE
TT[1:0] = 0
TM[2:0] = 5
TRANSLATED TO THIS RANGE IN 68040 ADDRESS SPACE
4 Gbytes OF 68040 ADDRESS SPACE
IOBASE or MEMBASE + 0000
FFF4 0000

Figure 2.4 : Address Mapping for the Fixed Offset Image

2.5 Deadlock Resolution

A deadlock occurs when the Spanner is being accessed as PCI target and 68K target simultaneously. When deadlocks occur (where a PCI and 68040 initiator attempt to reach the other bus at the same time), the deadlock is always resolved by retrying the PCI initiator. The 68K initiator will complete its cycle (read or write) on the PCI bus after the incoming PCI cycle is retried.

2.6 Interrupts

2.6.1 Interrupt Mapping

The Spanner bridges 68K interrupters to the PCI bus (interrupts are not relayed in the other direction). If interrupt mapping is enabled (with the IEN bit in the INTCS register, Table A.14), then an interrupt received on the 68K interrupt lines ($\overline{\text{LIPL}}$[2:0]) causes the Spanner to synchronously assert INTA# on the PCI bus. Any asserted combination of $\overline{\text{LIPL}}$[2:0] is accepted as a valid interrupt. The incoming interrupt level decoded from the $\overline{\text{LIPL}}$[2:0] lines is reflected by the ILVL bits in the INTCS register. The value of the ILVL bits reflects the most recent interrupt received from the 68K bus. The PCI bus interrupt is negated once the IPL lines are negated regardless of whether an IACK cycle has occurred on the 68040 bus or not.

Caution $\overline{\text{LIPL}}$*[2:0] are sampled synchronously on a rising clock edge and as such must not be treated as asynchronous signals.*

2.6.2 Interrupt Acknowledgment

When an interrupt from the 68040 bus is passed through to the PCI bus, the Spanner passes the incoming interrupt level to the ILVL bits in the Interrupt Control and Status (INTCS) register (Table A.14). The interrupt controller on the PCI bus must use this interrupt level to determine the address it needs to access in the upper 2 Kbytes of the Spanner's Fixed Offset image. The address which is accessed in this 2 Kbyte range controls the IACK cycle type that the Spanner generates on the 68040 bus (LTT[1:0]=3, and LTM[2:0]= interrupt level). The relationship between the Fixed Offset access and the resulting 68040 cycle is shown in Table 2.4 below. (Table 2.4 provides offsets for single byte offset cycles. For other IACK data widths, use the cycle mapping information in Table 2.2.) Essentially, the value presented on the PCI lines AD[4:2] are directly translated to the value generated on the 68040 lines LTM[2:0] to represent the interrupt level.

Table 2.4 : Using the Fixed Offset Image to Generate Single Byte 68040 IACK Cycles

Fixed Offset from IOBASE or MEMBASE	Corresponding IACK Level	68040 Cycle	
		LTT[1:0]	LTM[2:0]
xxxx F007	IACK Level 1	3	1
xxxx F00B	IACK Level 2	3	2
xxxx F00F	IACK Level 3	3	3
xxxx F013	IACK Level 4	3	4
xxxx F017	IACK Level 5	3	5
xxxx F01B	IACK Level 6	3	6
xxxx F01F	IACK Level 7	3	7
xxxx F020 - xxxx F7FF	IACK Levels Repeated	3	IACK Levels Repeated

Note that a 68K IACK cycle is only generated if the PCI initiator reads from the appropriate address in the upper 2 Kbytes of the Fixed Offset image. The interrupt vector is returned to the PCI interrupt controller according to the byte enable setting on the PCI bus. Writes are not screened and will appear on the 68040 bus with LTT[1:0] and LTM[2:0] lines set for an IACK cycle. In addition, the rules for cycle mapping listed in Table 2.2 on page 2-10 apply for any access to the Fixed Offset image. If an unaligned access is attempted to the upper 2 Kbytes of the Fixed Offset image, then it will be translated as two separate cycles on the 68040 bus with the LTT[1:0] and LTM[2:0] lines set according to Table 2.4 above.

2.7 Register Access

2.7.1 Control and Status Registers

The 2 Kbytes of the Spanner chip's Control and Status Registers (SCSR) are used to program PCI settings as well as all of the Spanner's operating parameters (see Figure 2.4 on page 2-13 for the location of the Spanner's control and status registers and Table 2.5 on page 2-18 for the Spanner register map). The SCSRs are accessible in PCI Configuration space at power up, which means that the configuration access is externally decoded and the Spanner is selected with IDSEL# (much like a standard chip select signal).

The SCSRs can be configured together with the Fixed Offset image (see "The Fixed Offset Slave Image" on page 2-12) so that they are also accessible in Memory and/or I/O space. The SCSRs are located in Memory and/or I/O space as address offsets from MEMBASE or IOBASE, respectively (see Figure 2.5 below). IOBASE and MEMBASE are programmed using the IOBASE and MEMBASE registers (see Table A.7 and Table A.8, respectively). The SCSRs are enabled as Memory and/or I/O images through the MEMSP and IOSP bits in the PCI Command register (Table A.4).

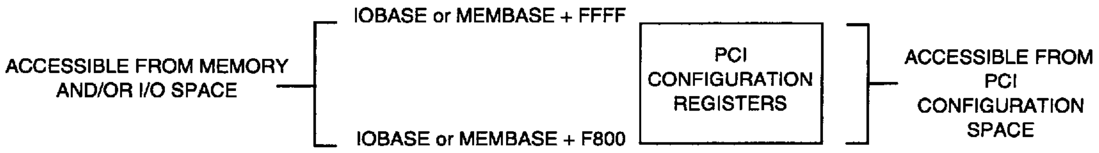

IOBASE or MEMBASE + FFFF
ACCESSIBLE FROM MEMORY AND/OR I/O SPACE
PCI CONFIGURATION REGISTERS
ACCESSIBLE FROM PCI CONFIGURATION SPACE
IOBASE or MEMBASE + F800

Figure 2.5 : Accessing Control and Status Register in Configuration, I/O and Memory Space

A 68040 initiator accesses the SCSR space through the PCI bus. This means that the 68040 initiator must first gain ownership of the PCI bus (Spanner as PCI initiator) before reaching the SCSRs (making the Spanner PCI target as well). Since all 68040 transactions are mapped to PCI memory space by the Spanner, the SCSR space is accessible to a 68040 initiator only in PCI memory space.

In order for a 68K initiator to gain access to the SCSRs, they must first be programmed by a PCI host so that the registers are accessible in memory space. In addition, the Spanner must be enabled at power up as a PCI master (see Appendix C for external logic that enables a 68K initiator to access the Spanner registers and PCI configuration space without a PCI host).

If a PCI initiator attempts burst reads or writes to the SCSRs, then the Spanner performs target disconnects after the first data beat of every transaction. This is also true if a 68040 initiator attempts a line read or line write to the Spanner's registers through the PCI bus (i.e. via the Spanner). The Spanner as PCI initiator attempts to perform a four longword burst write, but the Spanner (as PCI target) performs a target disconnect after the first data beat forcing the burst to be broken into 4 single data beat transactions. (Note also that the address will wrap in this case, see page 2-4.)

2.7.2 Register Maps

Register offsets in Table 2.5 and Table 2.4 below are given with the upper 21 address bits as variables (e.g. xxxx xx0A). The upper 21 address bits vary depending upon whether the PCI Configuration registers are accessed from Memory or I/O space, or from PCI Configuration space.

If the PCI Configuration registers are accessed from Memory space, then the upper 21 address bits will be MEMBASE + 0000 F8xx, with the lower 11bits supplied from the PCI bus address lines. Likewise, if the PCI Configuration registers are accessed from I/O space, then the upper 21 address bits will be IOBASE + 0000 F8xx, with the lower 11bits supplied from the PCI bus address lines.

If the PCI Configuration registers are accessed through PCI Configuration space, then the upper 21 address bits will depend upon the implementation of IDSEL# and the mapping of the SCSRs in the PCI Configuration space.

Table 2.5 : Spanner PCI Configuration Map

Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C
I/O Map Base Address and Control				10
Memory Base Address and Control				14
Not Supported				18
Not Supported				1C
Not Supported				20
Not Supported				24
Reserved				28
Reserved				2C
Not Supported				30
Reserved				34
Not Supported				38
Max Latency	Minimum Grant	Interrupt Pin	Interrupt Line	3C
PCI Slave Bound 0		PCI Slave Base Address 0		40
Reserved		Address Offset and Enable bit for PCI Slave Image 0		44
PCI Slave Bound 1		PCI Slave Base Address		48
Reserved		Address Offset and Enable bit for PCI Slave Image 1		4C
Reserved			Interrupt Enable and Level	50

Table 2.6 : Spanner PCI Configuration Registers

Register Address	Register	Name
xxxx xx00	ID Register	ID
xxxx xx04	PCI Command and Status	PCICSR
xxxx xx08	Configuration Class Register	CLASS
xxxx xx0C	Miscellaneous Configuration Register 0	MISCON0
xxxx xx10	PCI I/O Base Address	IOBASE
xxxx xx14	PCI Memory Base Address	MEMBASE
xxxx xx3C	Miscellaneous Configuration Register 1	MISCON1
xxxx xx40	PCI Slave Image Base/Bound 0	SI0_BSBD
xxxx xx44	PCI Slave Image 0 Control	SI0_CTL
xxxx xx48	PCI Slave Image Base/Bound 1	SI1_BSBD
xxxx xx4C	PCI Slave Image 1Control	SI1_CTL
xxxx xx50	Interrupt Status and Control	INTSC

2.8 Clocks and Resets

2.8.1 Clocking

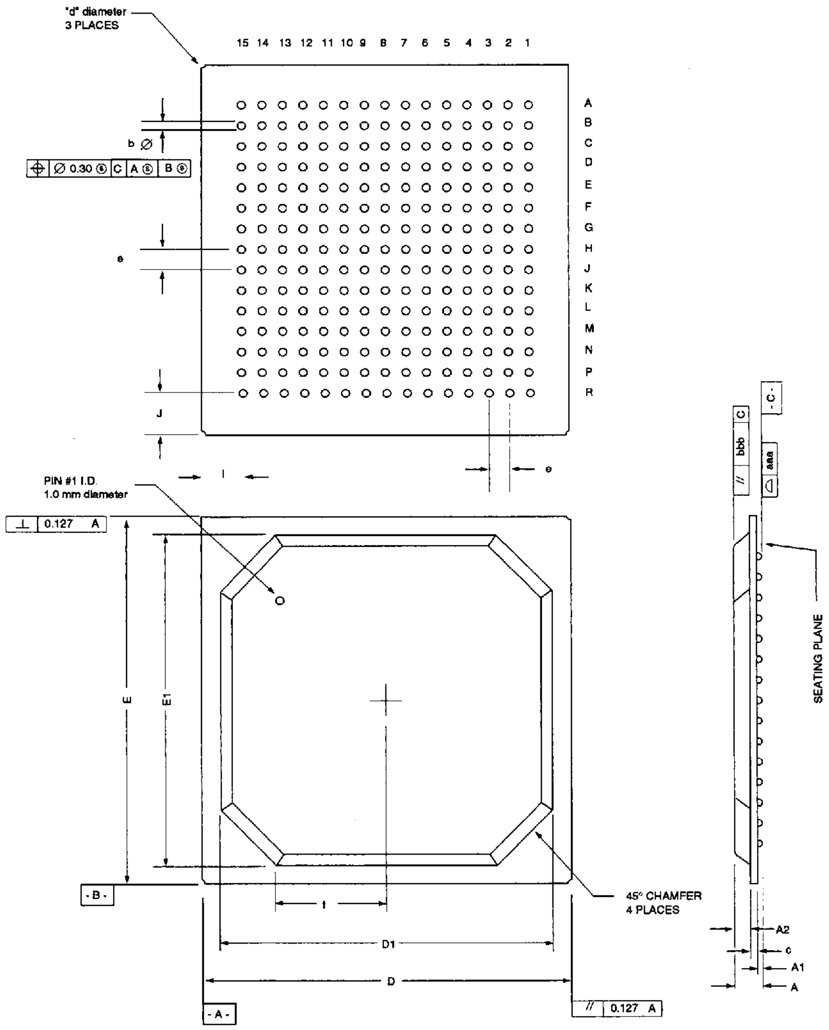
Timing for the Spanner is taken only from the PCI bus on the CLK input. Since the Spanner is a synchronous part, this means that it expects both buses to operate at the same clock frequency.

2.8.2 Reset

The Spanner has a single reset input from the PCI bus. When RST is asserted, all Spanner signals (on the PCI and 68K interfaces) go inactive and tristate. There is no specific logic for reset. All outputs are tri-stated upon assertion of RST# and all register values return to their default state (see App A).

Appendix F Mechanical and Ordering Information

F.1 Mechanical Information



Controlling Dimensions in mm

Dimension	Min	Nom	Max
A	1.90	2.09	2.30
A1	0.50	0.60	0.70
A2	1.12	1.17	1.22
D	26.80	27.00	27.20
D1		24.00	24.70
E	26.80	27.00	27.20
E1		24.00	24.70
I	3.00 REF		
J	3.00 REF		
M	15		
b	0.60	0.76	0.90
c	0.28	0.32	0.38
d	1.00 REF		
e	1.50 REF		
f	8.05 REF		
aaa			0.15
bbb			0.15

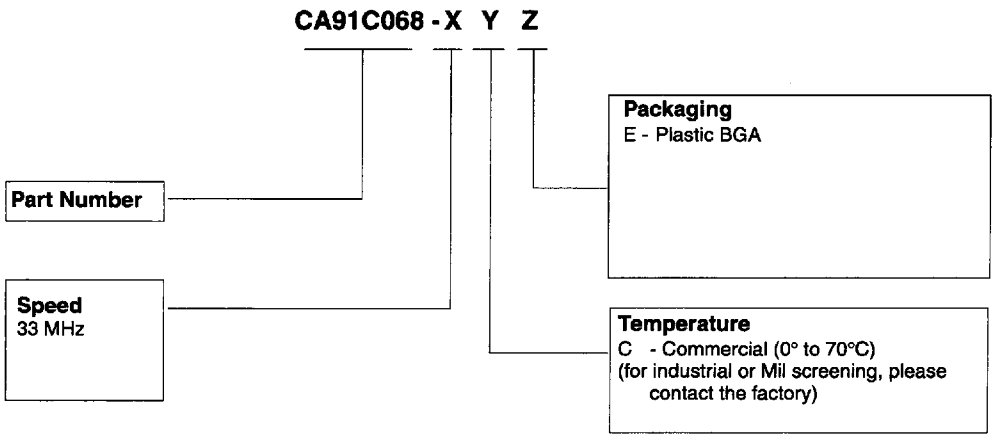
App F

Figure F.1 : Mechanical Dimensions for 225-Pin PBGA

6588101 0003458 685

F.2 Ordering Information

Newbridge Microsystems products are designated by a Product Code. When ordering, refer to products by their full code. For detailed mechanical drawings or alternative packaging requirements, please contact our factory directly


CA91C068 - X Y Z
Part Number
Speed
33 MHz
Packaging
E - Plastic BGA
Temperature
C - Commercial (0° to 70°C)
(for industrial or Mil screening, please contact the factory)