Sun SCSI Programmers' Manual

W. M. Bradley

ABSTRACT

This manual describes the architecture and operation of Sun Microsystems' interface to the Small Computer Systems Interface (SCSI) bus.

Overview

The SCSI bus is an ANSI standard interface bus which is capable of connecting processors to peripherals and to other processors. It is intended to provide an interface protocol which is largely independent of the manufacturer of the peripheral device attched. SCSI peripheral devices may include disk drives, tape drives, printers, network interfaces, and others. A *Host Adapter* is needed to interface between the SCSI bus and the processor which wants to use the SCSI peripherals. The Sun SCSI Host Adapter comes in two versions: as a Multibus card, and as part of of the Sun-2 single board. Except for minor (and inevitable) differences relating to the fact that the Multibus is not the same as the internal bus on the Sun-2 single board, the two versions are identical. For further information about the SCSI bus, consult the *SCSI Buyers' Guide,* published by ADES. The *Controller/System Interface Specification* by Adaptec, Inc is also useful.

Capabilities

This interface supports the SCSI specification with parity and without disconnect/reconnect. Arbitration is not supported, so there can only be one *INITIATOR* (the Sun host adapter). SCSI messages may be supported via host software, but there is insufficient hardware support to allow reconnection. Data transfers may be done via DMA, but commands and status must be transferred using programmed IO. The host may be interrupted when a status transfer is requested or when a message is sent or requested. Data transfers may be done either 8 bits or 16 bits at a time. Command and Status transfers may only be done 8 bits at a time.

Basic SCSI

The SCSI bus is a communications bus that is designed for medium speed transfer of data between peripherals and computers. In its most general form, it is similar to an IBM multiplexor channel, supporting overlapped operation of numerous peripheral devices. Subsets of the complete specification are allowed. The subset described here is the level of implementation represented by the Sun host adapter.

The SCSI bus is a bidirectional bus with 18 signals; 8 data lines with parity, and 9 control lines. Transfers are asynchronous, with an explicit *REQuest/ACKnowledge* handshake for every byte transferred. The complete SCSI spec allows for a total of up to 8 controllers and/or host adapters to be connected to the bus. Each has a unique *ID*. A host adapter is a bus interface that is associated with a smart device such a computer. A controller connects the SCSI bus to a peripheral device like a disk drive. The complete SCSI spec allows more than one host adapter to be on the same SCSI bus. This implementation supports only one host adapter per SCSI bus, and as many as 7 controllers. Each controller may have up to 8 devices attached to it, depending on the capabilities of the individual controller.

A data transfer occurs between 2 devices on the bus. The device which initiates the transfer is called the *INITIATOR*. For this implementation, only the host adapter may serve as an *INITIATOR*. The device which responds to the initiator's command is called the *TARGET*. An *INITIATOR* starts a transaction by SELecting a target. From then on, the *TARGET* is in control of the action. Instead of the *INITIATOR* force-feeding a command to the *TARGET*, the *TARGET REQuests* the command from the *INITIATOR*. After the *TARGET* has requested and received the command, the *TARGET* performs the action implied by the command. This usually involves a data transfer between the *INITIATOR* and the *TARGET*. This data transfer is controlled by the *TARGET*, by *REQuesting* that a data byte be sent from or accepted by the *INITIATOR*. The *INITIATOR* responds by handling the data byte and ACKnowledging.

After all data bytes have been transferred, the *TARGET* returns STATUS, again by a *REQuest/ACKnowledge* procedure. The transaction is finished when the *TARGET* sends a COMMAND COMPLETE message to the initiator.

The following control signals are used to make this happen.

BSY (Busy) means that the bus is in use and must not be disturbed except as needed to carry out the ongoing transaction.

RST (Reset) causes the bus and all devices on it to be initialized. *RST* may be asserted at any time.

SEL (Select) is asserted to begin a transaction. *SEL* is asserted concurrently with the *ID* of the *TARGET* device to which the transaction applies and the Host Adapter's own *ID*. There are only 8 possible *ID*s, each of which is assigned a single bit on the DATA bus. If device number 5 is the desired *TARGET*, and the Host Adapter is device number 7, bits 5 and 7 of the data bus should be asserted, with all other bits negated.

REQ (Request) and *ACK* (Acknowledge) are handshaking lines which are used to control the asynchronous transfer of each byte of command, status, or data on the SCSI bus. *REQ* is asserted by the *TARGET* when it is ready to accept or provide a new byte. The *INITIATOR* responds by asserting *ACK* when it is prepared to deal with that byte (either by sending or receiving it, as needed). The *TARGET* negates *REQ* when it sees *ACK* asserted, and then the *INITIATOR* negates *ACK* completing the handshake. Host adapters with disconnect/reconnect take care of all of this transparently, since all bytes to be transferred to and from the *TARGET* are communicated with the host system via DMA. For this implementation, the software must transfer commands, status, and messages with programmed IO, but it need not worry about the explicit handshake. The software may sense a request by reading a bit in a register (the control register). It then reads or writes another register, which causes the actual SCSI transfer to occur. The *ACK* line is automatically sequenced by the host adapter hardware as a result of the access to the second register. The software does not need to watch the *REQ* signal go away and return, since the Sun host adapter has a bit in the control register which is only true if a new unacknowledged request is pending. As soon as a REQuest is ACKnowledged, this bit goes false and does not go true again until the SCSI *REQ* line goes false and then true again.

MSG (message), *CMD* (command), and *I/O* (input/output) are lines used to specify which sort of byte transfer is to take place in order to satisfy a *REQ* for a byte. They are asserted by the *TARGET* and read by the *INITIATOR*. The encoding is:

MSG	*CMD*	*I/O*	Transfer Type
0	0	0	Data to *TARGET*
0	0	1	Data from *TARGET*
0	1	0	Command to *TARGET*
0	1	1	Status from *TARGET*
1	0	0	Not Used
1	0	1	Not Used
1	1	0	Message to *TARGET*
1	1	1	Message from *TARGET*

ATN (Attention) is asserted by the *INITIATOR* to inform the *TARGET* that the *INITIATOR* wishes to send a message. This is necessary because, once a *TARGET* has been selected, the *TARGET* is in control of the sequencing of byte transfers. The *ATN* mechanism provides a means of forcing the *TARGET* to ask the *INITIATOR* for a message. If *ATN* is asserted at the end of the Selection phase, the *TARGET* will assume that the *INITIATOR* supports Disconnect/Reconnect. The *ATN* line is not implemented on the Sun Host Adapter, because it is useless without disconnect/reconnect.

Architecture

Figure 1 is a block diagram of the SCSI interface.

The control section takes care of byte packing, handshake generation, interrupt generation, DMA cycles, and parity.

The byte packing hardware allows host transfers to proceed 16 bits at a time, while transfers over the SCSI data bus proceed 8 bits at a time.

The handshake generator takes care of the *REQuest/ACKnowledge* handshake that accompanies every byte transferred over the SCSI data bus. When a word is needed from the host, the *REQuest* from the device is not acknowledged until the host transfer is done. When the device *REQuest* may be satisfied by a byte that is already stored in the byte-packing hardware, the *REQuest* is automatically acknowledged.

Parity generation and checking may be optionally performed. If parity is to be used, all devices on the SCSI bus must have parity enabled.

The DMA controller allows Data transfers on the SCSI bus to occur using Direct Memory Access. Only data (as opposed to commands, status, or messages) may be transferred with DMA.

Software Interface

The SCSI host adapter has 5 registers, an SCSI Data register, an SCSI Command/Status register, an Interface Control register, a DMA Address register, and a DMA Count register. Consult the Register Addresses section for their addresses.

The DMA address register sets the starting address for DMA transfers. It is a write-only register. Attempts to read it will not cause a bus error, but the resulting data returned is undefined. It is 24 bits wide, writeable as two 16 bit halves. The upper 8 bits of the most-significant word are ignored. The most-significant-word is at the lower memory address, so this register may be written as a 68000 longword. For the Multibus version, only 20 bits of the address are actually used. After a data byte or word is transferred, the DMA Address register auto-increments by either 1 or 2, depending on the size of the data operand transferred.

The DMA Count register is a 16-bit read/write write register which may be used to determine how many bytes were actually transferred during an SCSI transaction. Also, it may be used to enforce a maximum DMA count. It auto-increments at the same time as the DMA address register. Both registers always count bytes. When the DMA Address register is loaded

with the base address for the transfer, the DMA Count register should be loaded with the negative of the maximum number of data bytes to be transferred, minus 1. At the end of the transfer, it may then be read to determine how many bytes were actually transferred with DMA. If the maximum number of bytes were actually transferred, the DMA Count register will read as -1 at the end of the transfer. If a Target device tries to transfer more that the maximum number of bytes, the Count register will end up at 1 if in Word Mode or 0 if not in Word Mode, instead of -1. Bus Error will be signalled, which will cause an interrupt if interrupts are enabled.

Here is how this works. When the maximum number of bytes have been transferred via DMA, the Count register will have counted to -1. When the count is -1, the DMA circuit will not issue a read strobe or a write strobe to the bus. If a data request comes in over the SCSI bus, the DMA circuit will try to do a DMA operation, but the read or write strobe won't actually be issued. Since there is no read or write strobe, no bus device will respond to the DMA cycle, and the DMA cycle will time out. This will result in a Bus Error condition on the SCSI board, which will prevent any further DMA requests from occurring. The Count register will be incremented at the end of the cycle causing the Bus Error, so it will read 0 (if not Word-Mode) or 1 (if Word Mode). This makes it possible to differentiate between a DMA overrun (Count = 0 or 1) and a Bus Error from some other cause (Count < 0). As described later, it is necessary to reset the board to clear a Bus Error condition.

The Data Register is a 16-bit wide read/write register that is used to transfer bytes of data to and from the SCSI bus. The SCSI data bus is only 8 bits wide. The data register serves to pack and unpack 2 bytes so that only one 16-bit memory access is required for every two bytes transferred over the SCSI data bus. The hardware takes care of sequencing the bytes within the word so that they are sent over the SCSI bus in the proper order. The high-order, or most-significant, byte is the first one which is actually sent over the SCSI bus.

It is possible to disable the packing/unpacking, so that each byte transferred over the SCSI data bus results in a separate one byte access to host memory. The Wordmode bit in the Control register (described later) controls this. If packing is turned off, only the high-order byte is sent. This means that in single-byte transfers, the intended data byte should be written to the high-order byte (for 68000-based systems, the one at the lower memory address) of the Data Register.

Since the *TARGET* device is in control of the transfer once it has been selected, it is possible for the host not to know in advance exactly how many bytes will be transferred. If an odd number of bytes is transferred, but packing is enabled so that memory accesses are 2 bytes at a time, then the last byte will be left over in the data register, and will not be DMA'ed into memory. This can only occur when the direction of transfer is from the *TARGET* to the host. This condition is detectable as described later, and the last byte may be recovered by simply reading the SCSI data register. The byte will be in the most-significant portion of the register.

If data is being transferred in the other direction, the *TARGET* may request that an odd number of bytes be sent to it. In this case, all the necessary bytes will be transferred to the *TARGET,* but the DMA count register will indicate that an even number of bytes were actually read from memory. The indicated count will be one more than the number of bytes actually transferred to the *TARGET.* This condition is similarly detectable.

The most-significant byte of the Data register is also used to send the *TARGET* and Host Adapter *IDs* during the selection phase. This is accomplished by writing the appropriate *ID* bits to the Data register before the selection line is asserted.

The SCSI Command/Status register is used to send commands and messages to the SCSI *TARGET* devices, and to receive status and incoming messages. It is 8 bits wide, and is only accessible with programmed IO.

After an SCSI *TARGET* device has been selected, it will request that a command byte be sent. The host should respond by writing the first byte of the command packet to the Command/Status register. This has the effect of making the byte available for the *TARGET*

device to read, and of acknowledging the request. The *TARGET* will continue to request bytes until it has received the entire packet.

After the data transfer portion of a transaction is completed, the *TARGET* device will send back a status byte. The host gets this byte by reading the Command/Status register, which also sends an acknowledgement to the *TARGET*. The *TARGET* may send back several more status bytes, each of which must be read individually from the Command/Status register. After all the status has been returned, the *TARGET* sends back a command complete message (1 byte), which also appears in the Command/Status register. The different types of requests (command, status, message, data) may be distinguished by bits in the Interface Control register.

The Interface Control register is a 16-bit wide read/write register that is used to control the SCSI control lines, to read the current state of the SCSI control lines, and to control some internal functions of the host adapter. Its bits are: (Bit 0 is the least-significant bit)

Interface Control Register

Bit 0 Interrupt Enable Read/Write

This bit enables and disables the generation of host interrupts 1=Enable, 0=Disable. Clearing this bit causes any pending interrupts to be immediately cleared, as well as preventing any further interrupts. When this bit is set, the SCSI interface will interrupt the host when the *TARGET* sends back status (usually upon completion of a transfer), and when the *TARGET* sends a message (usually right after status is sent). A Bus Error (DMA timeout or overrun) will cause an interrupt if this bit is set. The Bus Error bit may be used to distinguish a Bus Error interrupt from a status or message interrupt.

Bit 1 DMA Enable Read/Write

Enables or disables the generation of DMA requests. DMA will only occur during the Data Transfer phase of a transaction. Commands and Status are transferred using programmed IO. 1=Enable, 0=Disable

Bit 2 Word Mode Enable Read/Write

Enables or disables Word Mode (byte packing). When the interface is in Word Mode, both bytes from the Data Register will be sent over the SCSI bus in response to data *REQuests* from the target. When Word Mode is disabled, only the high byte will be sent. Word Mode is only effective during the data transfer phase of the SCSI transaction, since it only affects the data register. Commands, status, and messages are always sent one byte at a time (the state of the Word Mode bit does not matter during the Command phase of the transaction). Before a transaction is started, it is necesary to momentarily clear this bit in order to reset the byte packing hardware. This is necessary because the previous transfer may have involved an odd number of bytes. Do this before the selection phase starts, so that the proper *ID* will be asserted on the SCSI bus.

Bit 3 Parity Enable Read/Write

Enables or disables the generation and checking of SCSI data bus parity. The SCSI spec requires that either all of the devices on the SCSI bus have parity enabled, or none of them do. In practice, it may be possible and useful to connect two devices to the SCSI bus, one of which supports parity, and one of which does not. The host adapter is capable of generating and checking parity for the device which supports it, and ignoring parity for the device which does not.

Bit 4 *RST* Read/Write 1=asserted

Control or sense the state of the SCSI *RST* (Reset) line. Once this signal is asserted, the software should wait at least 25 microseconds before deasserting it. The hardware does not enforce this timing. Setting this bit also clears the bus error bit.

Bit 5 *SEL* Read/Write 1=asserted

Control or sense the state of the SCSI SELect line. This is used to start a transaction. Here's how: First clear the Wordmode bit to make sure the byte packing starts out on the right foot. Write the *ID* bit of the desired *TARGET* and the Host Adapter's own *ID* bit (bit 7 unless you can think of a reason to use some other bit) to the high byte of the SCSI Data register. When the SCSI bus isn't Busy (as indicated by the *BSY* line), assert *SEL* by writing a 1 to this bit. When the *TARGET* recognizes that it is being Selected, it will assert *BSY*, after which the host should negate *SEL*. *** Note: The SCSI specification calls for the *TARGET* device to wait until *SEL* is negated before proceeding to *REQuest* the command from the Host Adapter. In practice, the Xebec Disk controller does not do this, but goes ahead and *REQuest* s the command before *SEL* is negated. It only requires that *SEL* be negated before the transaction is finished. For future compatibility and for good form, negate *SEL* as soon as possible after the *TARGET* asserts *BSY*. *** *** Note: Some SCSI target devices do not require that the Host Adapter's *ID* bit be asserted concurrently with the target's *ID* bit. The ADES controller does require it, and the SCSI standard requires it, so both *IDs* should be asserted. I recommend bit 7 for the Host Adapter, since most controllers are shipped configured for bit 0 as the Target *ID*. ***

Bit 6 *BSY* Read Only 1=asserted

Sense that state of the *BSY* (Busy) line. A *TARGET* device asserts this line to indicate that it thinks it is servicing a transaction. The host should not try to Select a *TARGET* until this line is negated. It is okay to assert *RST* while *BSY* is asserted, but this is an asynchronous hard reset of everything on the SCSI bus, and will abort whatever is happening.

Bit 7 *PAR* Read Only

1=asserted Sense the state of the SCSI *PARity* line. This bit is provided for testing purposes, and may be ignored by system software in normal operation. This bit may be used to verify that the parity generation circuit on the board actually works. When the bus is otherwise idle, a byte may be written to the Data Register. If parity is enabled, the overall parity of the data byte plus the *PARity* line should be odd. In normal operation, the Parity Error bit is used to sense parity.

Bit 8 *I/O* Read Only 1=Input, 0=Output

Sense the state of the SCSI *I/O* line. Used with *MSG* and *C/D* to specify the direction and type of transfer that is currently *REQuest* ed. Input means that the direction of transfer is from the *TARGET* to the Host.

Bit 9 *C/D* Read Only 1=Command, 0=Data

Sense the state of the SCSI *C/D* line. Used with *MSG* and *I/O* to specify the direction and type of transfer that is currently *REQuest* ed.

Bit 10 *MSG* Read Only 1=asserted

Sense the state of the *MSG* (Message) line. In conjunction with *C/D* and *I/O*, *MSG* indicates what kind of transfer is being requested by the *TARGET*. When *MSG* is asserted (1), the transfer is a message, and the *I/O* line indicates whether the *TARGET* wants to send a message byte or receive a message byte. The only message that is absolutely required is Command Complete. This one is sent from the *TARGET* to the *INITIATOR* (that's us, remember?) and tells us that the transaction is over.

Bit 11 *REQ* Read Only 1=asserted

Sense the state of the *REQuest* line. A target uses this line to control the transfer of a byte over the SCSI bus. When doing programmed IO transfers on the SCSI bus, this bit may be used to indicate when it's time to transfer the next operand (word or byte) to or from the Data Register.

This bit does not exactly mirror the state of the SCSI *REQ* line. When the SCSI *REQ* line is first asserted, and the request type is Command, Status, or Message, then Bit 11 will read as 1. When the request is acknowledged by the CPU accessing the Command/Status register, Bit 11 will go to 0. It will remain at 0 until the SCSI *REQ* line is deasserted and then reasserted. This provides a positive handshake with the SCSI Target device. When Bit 11 is asserted, the CPU may be certain that it refers to a new request, rather than the one that was just handled. If the request type is Data, Bit 11 will only be asserted if DMA is not enabled. This makes it possible to do programmed I/O during the data phase by polling Bit 11 with DMA disabled. With DMA disabled, a Data request will only cause Bit 11 to be asserted if the CPU needs to do something in order to satisfy the request. In other words, if Wordmode is enabled and the byte-packing hardware either already has the next outgoing byte or has room for the next incoming byte, then Bit 11 won't be asserted. The upshot of this is that the CPU may poll the Control register looking for Bit 11 to go to 1. When it does, the CPU has to do something. If DMA is enabled, then Bit 11 may also be safely polled while waiting for a Status request, because the Data requests which will be satisfied by DMA will not cause Bit 11 to be asserted.

Bit 12 Interrupt Request Read Only

This bit will read as 1 if the SCSI host adapter is requesting a host interrupt. If interrupts are disabled, this bit may still read as 1, but no interrupt will actually occur. An interrupt request will occur as a result of a *REQuest* from the *TARGET* for Status or Message. *REQuest* s for Data result in a DMA request if DMA is enabled, and an interrupt request if DMA is not enabled. A Command request never causes an interrupt request. The reason is that after the ARGET *has been selected, it will very quickly issue the series of Command requests. There is no need for the CPU to go away and wait for an interrupt.*

Bit 13 Odd Length Read/Write 1=asserted

This bit reflects the state of the even/odd byte counter which is part of the byte packing/unpacking circuit. At the end of the data transfer when the status is returned, the host should sense this bit. If it is set, it means that there is a left-over byte in the data register which did not get DMA'ed into memory. This byte may be read from the data register with programmed IO. This applies to data that is coming in from the *TARGET* to the host, and then only when Wordmode is on. If data is going out and there is an odd length transfer, then one more byte than necessary will have been read from host memory (assuming Wordmode is on). This byte will be left in the data register, but it doesn't matter, since the *TARGET* will only request as many bytes as it actually wants. If Odd Length is set at the end of an outgoing transfer, it means that the value in the DMA Count register is one more than the number of bytes actually sent to the target. The DMA Count register accurately reflects the number of bytes that were read from the memory, unless there is a DMA overrun as described earlier.

Bit 14 Bus Error Read Only

If a DMA request resulted in the host bus being arbited for and won, but the data transfer does not successfully complete, then a bus error will be signalled. For the Multibus board, this means that transfer acknowledge was not returned within 13 microseconds of the start of the transfer. For the single board system, any 68000 bus error that occurs during an SCSI DMA transfer will cause this to happen. The bus error disables further DMA requests, so that the bus will not be loaded down

with repeated requests that can't be satisfied. An interrupt request is raised, which will cause an interrupt if interrupts are enabled. In order to clear this condition, the entire SCSI subsystem must be reset by writing to bit 4.

A DMA overrrun will also cause a Bus Error indication. The reason for this is that the Bus Error circuit is used to lock out further DMA requests after the first overrun attempt, again to prevent bus loading from repeated requests. The DMA count register may be used to distinguish a true Bus Error from a DMA overrun. Refer to the DMA Count Register description.

Bit 15 Parity Error Read Only

This bit will read as 1 if a parity error occurred during any byte of the previous transaction. This is latched when a parity error occurs, so that the host can read it at the end of the transaction. A parity error does not cause an interrupt request. At the end of a transaction, at the time that status is returned, the Parity Error bit may be checked. If a parity error occurred, the operation may be retried. To clear the Parity Error indication, momentarily clear the Parity Enable bit. Note that this parity check only applies to the SCSI bus itself. It does not check the parity of the memory that is involved in SCSI DMA operations.

Register Addresses

The base address of the register block is switch-selectable. For the single-board version, it may be set to any 16K boundary within the physical address space. For the Multibus version, the base address may be on any 16K boundary in Multibus memory space. The reason for the 16K boundary restriction is that the board contains some other functions which share decoding with the SCSI interface. Each of these other functions has its register set beginning on a 2K page boundary. These pages are contiguous in the address space, and the one for the SCSI registers is the first one in the group.

Address	Size	Register
base+0x00	16	Data (read/write)
base+0x02	8	Command (write)/Status (read)
base+0x04	16	Interface Control (read/write)
base+0x06		reserved
base+0x08	16*	DMA Address High Word (write)
base+0x0a	16	DMA Address Low Word (write)
base+0x0c	16	DMA Count (read/write)
base+0x0e		reserved

* Since the Multibus addressing is only 20 bits, only the lower 4 bits of this register are actually used for the Multibus version. Similarly, the single-board version, with 24-bit addressing, uses the lower 8 bits.

Registers may be accessed in sizes other than that specified by the table above. Thus, the Data Register may be read or written with a byte cycle. The byte order is the same as the 68000; the byte at the lower address is the most-significant byte. Similarly, the Command/Status register may be accessed with a word cycle. In that case, only the upper byte will be meaningful. Note that all the write