T A B L E   O F   C O N T E N T S

Appendices

LIST OF FIGURES

L I S T   O F   T A B L E S

1. PURPOSE

This document describes the design and performance requirements of an intelligent host bus adapter that interfaces a SCSI bus (multiple target or initiator) to a multi-user IBM AT system. This SCSI to AT Interface Channel (WD7000-ASC) contains an onboard Z80 CPU, along with the necessay drivers, to handle all of the data transfer between these 2 buses.

It is assumed that the reader is familiar with the ANSI SCSI specification, the SBIC data sheet, the AT bus specification, and has a working knowledge of a multi-user environment.

Throughout this document, signals which are active low true will be denoted by an appended negation symbol "-". Signals which are active high true, will have no appended notation.

Throughout this specification the host bus adapter will be referred to as the WD7000-ASC or as the ASC.

2. REFERENCE DOCUMENTS

All of the reference documents specified below, or otherwise implied, are an integral part of this specification.

The following list of documents will be made available from Western Digital's document control.

Additionally all source files and documents generated by board products Engineering are also available on IBM compatible 5.25" floppy diskettes. These are for internal use only and not available for general distribution.

The ASC source files were created using a text editor like wordstar in non document mode and all other documents or specifications were processed using WORDSTAR 3.3 in document mode. The list files can be re-created using the procedures outlined in the respective software packages.

	Description/Title	Document Num./Source
1.	ASC Test Specification	96-000495
2.	ASC Parts list	61-600049
3.	ASC Schematics	68-600049
4.	ASC PCB Fab (Drill) Dwg.	60-600049
5.	ASC Assembly Drawing	61-600049
6.	ASC Artwork	65-600049

7.	ASC Source List	75-000294
8.	ASC Characterization Report	95-000064
9.	ASC CPU Procurement Spec.	96-000XXX
10.	ASC MT Exerciser Diagnostic Diskette	79-000115
11.	ASC PAL Specifications	

It is highly recommended that the serious reader be very familiar with all of the references indicated below. The SCSI Engineering dept. maintains several copies of these references as well.

13.	ANSI SCSI Specification X3T9.2	ANSI Committee
14.	WD33C93 SBIC Data Sheet	W.D. Publication
15.	IBM AT Technical Reference Manual	IBM
16.	Z80 Technical & Programming Manuals	Zilog
17.	WD37C65 Floppy Disk Controller	W.D. Publication

3. GENERAL DESCRIPTION

As the name implies, ASC is an interface channel between two multi-user worlds: the AT bus and the SCSI bus. All necessary driver/receivers are included, permitting direct cable connections to the SCSI bus through a 50 pin connector and the AT interface is through 2 card edges as defined by the AT Technical Manual. Included are jumper configurable options so that the address space, DMA channels, and Interrupt requests can be selected to suit the user's application.

To facilitate data transfer between these 2 buses ASC contains an onboard CPU (Z80) and a 16 byte FIFO on the AT bus to maintain up to 4.0 Mbyte/sec synchronous SCSI data transfer rate. First party DMA is employed to handle most of the command, data, and status information via an outgoing and incoming mail box system, so that several logical units can be connected simultaneously. All other handshaking required to transfer the data is done by the Local CPU (LCPU).

The SCSI interface consists of the WD33C93 SCSI Bus Interface Controller (SBIC) device that handles all of the arbitration, selection, disconnection, and reselection, leaving the LCPU to manage data transfers and interrupts more efficiently. The SBIC is usually configured to operate as the initiator but it can operate as a target as well in a multi-initiator system. Selection as a target generates an unsolicited interrupt to the host AT computer, if so enabled by the host. Further communication is controlled totally by the host AT system, with the LCPU managing all of the required handshake between the SBIC and the host CPU.

ASC supports the common command set including all of the extended SCSI commands. SCSI commands are queued by the ASC and can also be linked by any initiator in the system. The SCSI bus parity option is fully supported as well. The on-board diagnostic capability, down to the component level, makes field replacements a breeze.

Additionally an onboard floppy disk controller and a BIOS ROM are provided for the convenience of the host. The WD37C65 floppy disk controller is controlled entirely by the host CPU and the BIOS ROM enables the host to boot off a SCSI drive, if desired.

The ASC is based on a proprietary chip set specifically designed to add a SCSI port to the AT bus.

3.1. ASC Features

- 0 to 55 Deg. C. operation
- Intelligent Z80 CPU Channel Control
- WD37C65 Floppy Disk Controller (optional)
- BIOS ROM (optional)
- WD33C93 SBIC for SCSI interface
- User Selectable Device Address, DMA & Interrupt Channels
- 4.0 Mbytes/sec SCSI synchronous data transfer rate
- 2 Mbytes/sec SCSI asynchronous data transfer rate
- First party DMA data transfers
- 16 byte FIFO on AT bus
- Any Block Size Transfers up to 16 MBytes
- Even or Odd DMA Start Address
- Programmable DMA Bus On/Off Times
- Multiple logical thread connection up to 16
- Operates as Initiator or Target
- Bus Arbitration Including Disconnect & Reconnect
- Supports The Full SCSI Command Sets
- SCSI Bus Parity Option
- Queued SCSI Commands
- Linked SCSI Commands
- On Board Controller Diagnostics
- IBM XT Form Factor
- Single +5V Supply

4. ASC ARCHITECTURE

A block diagram of the internal architecture of the WD7000-ASC controller is shown in figure 4.1. The diagram shows all of the major elements and it should be referred to throughout the following description. The functional blocks are:

1. Local Microprocessor (LCPU)

2. Program Memory (ROM)

3. Scratchpad RAM (SRAM)

4. First Party DMA logic (FPDMA)

5. Host Interface Logic (HIL)

6. SCSI Bus Interface Controller (SBIC)

7. Floppy Disk Controller (FDC)

8. Bios ROM (BIOS)

9. Control Support Logic (CS)

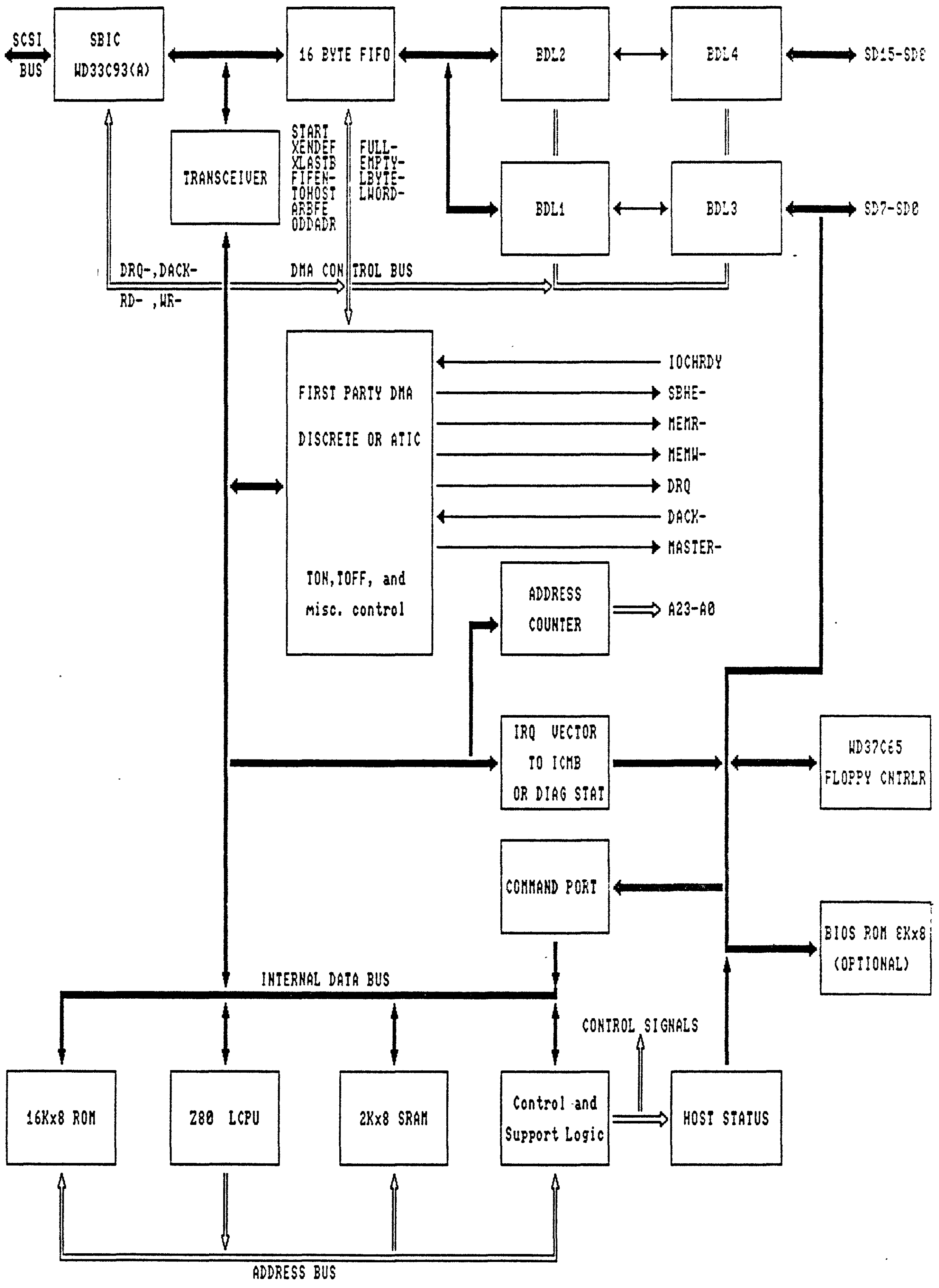
All pertinent portions of the ASC design will be discussed in this and the following sections. Complete details of the major components can be found in the references. Address space configuration jumpers and connector pinouts are described in section 7.

4.1. Local Central Processing Unit

The LCPU (Z80 uP) is the heart of the control logic for the ASC. By interpreting commands sent from the host, the LCPU configures the other logic blocks to perform the desired operation. This includes setting the data transfer direction, DMA starting address, time on and time off AT bus counters, as well as the control of the SBIC.

Host access to the ASC via the command port causes the LCPU to read the command byte immediately. The command is decoded and either executed if it is an ASC local command or put in the SBIC queue if it is meant for a SCSI device. In all cases, the host is informed within 70 uS whether the command byte has been accepted or not by the ASC.

The actual execution time for a particular command, that has been put in a queue, is governed by the programmed DMA bus on/off times, the number of mailboxes available, the length of the data transfers, and any exception handling involved. These and other features of the ASC will be discussed in the appropriate sections described later on.



WD7000-ASC Engineering Specification

Fig. 4-1: WD7000-ASC Block Diagram

4.1.1. LCPU Clock

The LCPU uses a 4 MHZ clock obtained by dividing the 8 MHZ clock used for the SBIC device. This gives a LCPU a basic clock period of 250 nS referred to as the T cycle. The machine cycles vary from 4 to 23 T states. A complete description of the instuction set and execution times can be found in the references.

4.2. Program Memory

The 8K X 8 PROM contains the control firmware for the LCPU with a provision for up to 32K with small changes in the hardware design for any future enhancements. The other 32K of the address space, differentiated by A15, is reserved for data memory.

4.3. Scratchpad RAM

The 2K X 8 SRAM is used by the LCPU to store variables and queued commands issued by the host. Since SCSI command data is directly passed between the host and the SBIC, no provision has been made to buffer any data, thus eliminating any latencies required to store and then transfer the data.

The present RAM size is sufficient to handle 16 logical threads which implies that up to 16 commands for different SCSI LUNs can be active at any given instant of time. In addition, the SCAN mailbox command (described later) can be used to issue more than 16 commands without requiring any further host intervention than simply issuing the command. However, more logical threads can be supported by increasing the size of the scratchpad RAM depending upon future applications at the cost of minor hardware modifications.

For all memory map information, refer to the source listing.

4.4. First Party DMA Logic

First Party DMA provides high speed data transfers by allowing WD7000-ASC to be master of the AT bus. The hardware consists of a 16 byte FIFO that can transfer a word across the AT bus every 375 nS, DMA control logic to transfer data in word mode or byte mode, on even or odd addresses, and a 24 bit counter to address up to 16M of host memory. The bus on/off times are fully programmable by the host. The DMA state machine control is implemented using PALS as detailed in the references.

The LCPU sets data transfer direction, starting DMA address, time on and time off the AT bus and then issues a START. For WD7000-ASC to AT transfers, the FPDMA will wait until there is data in the FIFO, then arbitrate for the AT bus by asserting DRQ. Upon receiving DACK- the FPDMA will assert MASTER- and then start writing to the AT's memory. The FPDMA will continue as master until the time on the bus is exausted. It will stay off the bus until the time off the bus is exausted then re-arbitrate if data is still in the FIFO. AT to WD7000-ASC transfers are similar except reads are done from the AT's memory and data is stored in the FIFO. The WD7000-ASC will continue to re-arbitrate for the bus until the FIFO is full.

The FDMA also provides the necessay signals to transfer data between the SBIC and FIFO and between the LCPU and FIFO.

4.4.1. DMA Transfer Detail

The FPDMA logic consists of a state machine (SM) that controls the data transfers between the SBIC & FIFO and the FIFO & AT, 24 bit address counters, bi-directional data latches, a 16 byte FIFO, bus on/off time latches, and combinatorial logic. Most parts of this logic are directly controlled by the LCPU, except for the real time control of the data transfer, once the process has been initiated. Refer to appendix A for the I/O address map and description of the FIFO control byte.

A data transfer from the SCSI port to the AT memory generates the following sequence of events.

- The LCPU loads the 24 bit address counter with the desired AT starting address.

- The LCPU writes to the FIFO Control Register (FCR) to set the data transfer direction (ASC-AT), takes the FIFO out of reset, and enables the FIFO transfer. If the AT address is odd, FCR bit 0 is set. If arbitration for the AT bus is desired only when the FIFO is full, bit 1 is set. If terminating bus master of the AT bus is desired when the FIFO is empty, bit 6 is set.

- The SBIC generates a REQ when it has a byte of data to transfer. If the FIFO is not full, the SM responds with an ACK and stores the byte in the FIFO. This sequence continues until either the FIFO is full or the SBIC stops generating REQs.

- With a byte in the FIFO, the SM starts arbitrating for the AT bus if FCR bit 1 is false and the Time Off Bus Counter (TOFBC) is at terminal count. If FCR bit 1 is true, the FIFO must be full before arbitration can begin.

- As the SM fills the FIFO, it also attempts to empty it by writing to Bi-Directional Latches (BDL) 1, then BDL2. If these latches are full, the SM must wait for them to empty.

- After successfully arbitrating for the AT bus, the SM asserts MASTER to gain host bus control. IF BDL 1 and 2 are full, the SM transfers their contents to BDL 3 and 4 and starts a memory write operation. BDL 1 and 2 empty, signals the SM to reload them if the FIFO is not empty. The SM can reload these latches before the current write operation is complete thus producing a transfer rate of 2 bytes every 375 ns.

- The SM continues to master the bus until the Time On Bus Counter (TONBC) reaches terminal count or if the FIFO is empty (FCR bit 6 must be set).

- The SM does not rearbitrate until the TOFBC reaches terminal count. If FCR bit 1 is set the FIFO must also be full.

- The arbitrate, master and release bus sequence continues until all of the data is transferred. One byte stays in the FIFO if; an even number of bytes is transferred and the starting address is odd, or if an odd number of bytes is transferred and the starting address is even. The LCPU detects this condition via the FSR, then sets FCR bit 4 to command the SM to transfer this last byte.

A data transfer from AT memory to the SCSI port is similar to the SCSI to AT transfer. It generates the following sequence of events.

- The LCPU loads the 24 bit address counter with the desired AT starting address.

- The LCPU writes to the FIFO Control Register (FCR) to set the data transfer direction (AT-ASC), takes the FIFO out of reset, and starts the FIFO transfer. If the AT address is odd, FCR bit 0 is set. If arbitration for the AT bus is desired only when the FIFO is empty, bit 1 is set. If terminating bus master of the AT bus is desired when the FIFO is full, bit 6 is set.

- With the FIFO empty, the SM immediately starts arbitrating for the AT bus. The SBIC can at any time generate an REQ but the SM does not respond until the FIFO is not empty.

- After successfully arbitrating for the AT bus, the SM asserts MASTER and now controls the bus. The SM performs a memory read operation and stores the 2 bytes into BDL 3 and 4. BDL 3 and 4 full, signals the SM to transfer their contents to BDL 1 and 2, then into the FIFO if it is not full. BDL 3 and 4 empty signals the SM to start another memory read. The SM can empty BDL 1 and 2 before a read cycle can complete thus producing a data transfer rate of 2 bytes every 375 ns.

- The SM continues to master the bus until the TONBC reaches terminal count or if the FIFO is full (FCR bit 6 must be set).

- The SM does not rearbitrate until TOFBC reaches terminal count. If FCR bit 1 is set, the FIFO must also be empty.

- The arbitrate, master, release bus sequence continues until all of the data is transferred.

4.5. Host Interface Logic

The HIL contains the requisite logic to buffer all data and control between the AT bus and the WD7000-ASC. The host interface consists of 16 data lines, a 24 bit address bus, and control lines. For a description of the signals and pinouts see the Interface and Connectors section 7. The HIL also contains a Command Register, WD7000-ASC Status Resgister, Interrupt Status Register, WD7000-ASC Interrupt Strobe, Static Reset, and SCSI Bus Reset. These are described in detail in section 5.

4.6. SCSI Bus Interface Controller

The WD33C93 SBIC is a single chip SCSI controller that can operate either as a Target or an Initiator. When the SBIC status register indicates reselection, it is configured as the Initiator and if the selection status is set, then it is configured as a Target. The SBIC, under the control of the LCPU, generates all of the necessary signals and timing to command the SCSI bus and transfer information across it.

For complete details on it's operation, refer to the data sheets mentioned in the references.

4.7. Floppy Disk Contoller

The WD37C65 is a single chip floppy controller. With minimal hardware, it interfaces directly to the AT bus and enables the host to control up to 4 drives.

For a description of the connector pinouts, refer to the Interface Connectors section. Details on the operation of the WD33C65 can be found in the data sheets mentioned in the references.

4.8. BIOS ROM

The BIOS, consisting of an 8KX8 ROM, enables the host to boot directly from a hard drive on the SCSI bus and to determine number of logical units connected to the bus at power on. It's other main function in life is to convert the host SCSI protocol to the mailbox protocol, including the CDB block construction, as described in sections 5 and 6 of this specification.

The address space of the BIOS, shown in section 7, is configurable in the range C0XXX-DEXXX (hex), with the LSB 13 bits being used to address the 8K ROM.

4.9. Control Support Logic

The CS consists mainly of decoders in the I/O address space of the LCPU, the reset circuitry, miscellaneous combinatorial logic and memory elements needed to control the various functional blocks. Due to it's very nature, control support logic tends to be logically distributed as required instead of being mostly centralized as other blocks described above. Nevertheless it can functionally be thought of as a block that is used by the LCPU to implement all of the real time control functions required to direct the signal flow within the ASC channel.

4.9.1. Reset Logic

The reset pulse for the ASC can come from 3 sources: power-up reset, host, or the SCSI port.

The reset from the SCSI port is used to reset the SBIC device but not the LCPU. This was done to maintain all of the host command queue so that these commands can be re-started without having the host re-issue the commands again.

After the assertion of RESETDRV signal at the host interface, generally on power-up, the host must write a logic "1" to the onboard reset D-FF so that the ASC can come out of the reset state and execute it's onboard diagnostics. This same register is set/reset by the host at other times whenever the ASC needs to be reset.

A similar set/reset D-FF is provided to reset all of the devices on the SCSI interface. Host access to these 2 reset registers is shown in the next section along with all of the other host addressable ASC ports.

4.9.2. I/O Address Space

In general, the upper 4 address lines were decoded to select a major functional block and the lower 4 address lines were used to access a specific register within that device. Since not all of the I/O space was being fully utilized some of the device chip selects were combined as indicated by the I/O address assignments shown in the tables in appendix A.

5. HOST INTERFACE PROTOCOL

The interface between the host AT system and the ASC controller occurs at 3 levels: host ASC control, SCSI DMA access, and Interrupt generation. For the most part the hardware aspects of the interface do not enter the I/O drivers with the exception of the maximum bus on/off times and the reset pulsewidth requirements. Bus on/off time, in this context, is defined as the amount of time the ASC is on the AT bus during DMA transfers and the amount of time that elapses after a packet of data has been transferred before the next packet of data transferred. The host bus on/off times are both programmable parameters of the host adapter and can be specified via the ASC initialization command discussed in the next section. On the SCSI side, disconnection can occur as required by the target provided all of the required SCSI protocol has been properly followed.

Before specifying the actual handshake between the AT-ASC-SCSI in either direction, a general signal flow will be discussed to get an overall feel of the interaction between all of the major logic components in the system. For all timing parameters, see the AC timing specification section. For detailed command execution, refer to the source listing or to the flowcharts in appendix B, designed to illustrate the general command execution flow within the ASC from the user's point of view. Appendix A contains useful tables, error codes, and appendix C has an example source listing, written & compiled using Microsoft "C", illustrating the procedure used to enable the ASC DMA and IRQ channels with the host system.

5.1. General Signal Flow

All communication between the host and ASC takes place via the HIL, using a command control byte and a mail box system. The host issues commands to the ASC by first selecting the ASC through it's address bus and writing a control byte to the command port. The command strobe clears the COMMAND PORT READY status bit alerting the host and the onboard LCPU that command execution is now in progress. The host must provide additional parameter bytes, if required, by writing to the command port when the COMMAND PORT READY status bit is set. The service time between 2 command port bytes is around 70 uS. This 8-bit programmed I/O data transfer handshake is repeated until the required number of bytes to complete the command have been transferred. The start I/O command byte contains a mail box number, which is used to compute the absolute address of the mailbox containing the command block pointer of the SCSI/ASC command to be executed. All further communication to the host is done using the onboard first party DMA controller in the master mode.

The 16/32 bytes of the command block accessed from the system memory are stored in the SRAM before being decoded for execution. The command is then executed by the LCPU or is re-issued to the SBIC for a SCSI access. The SBIC device handles all of the SCSI protocol including arbitration, selection or reselection, etc.

For any write operation to a SCSI LUN additional bytes of data are accessed from the host system memory using first party DMA transfers controlled by the bus hold time programmable parameters. This sector of data is written directly to the SBIC using the SBIC's DRQ-/DACK- handshake until all of the bytes have been transferred. For all other commands, that do not require data, this phase of the operation is skipped. At the end of the data transfer phase, the SBIC obtains the command status and generates an SINTRQ to the LCPU which in turn sets appropriate status and error register bits in the host command descriptor block (CDB). This status is then read by the host upon command completion, signalled by an interrupt to the host.

On a read operation from the selected LUN, arbitration & selection takes place as before except that the flow of data is reversed. All message transfers and protocol requirements to complete a given SCSI command is handled by the ASC totally transparent to the host. As an added protection, a direction bit can be set by the host to prevent a runaway target from writing into the host memory. This bit is set on all read operations or when writing to the host memory.

For the same logical unit, commands can be linked via the host CDB. The next host CDB will be accessed and executed when the prior command has been completed. This new host CDB can in turn point to another command to be executed. In this fashion several commands may be chained together. Interrupts to the host can be programmed to be generated at the end of every command or on completion of the entire chain of commands.

When the ASC is selected as the target, whether anticipated or not, an interrupt is generated to the LCPU, which in turn sets an appropriate incoming mail box and interrupts the host, if so enabled. The command, data, and status bytes are then transferred in 3 or more packets. Each information packet is only initiated after the ASC receives an appropriate command from the host to accept that packet from the SCSI Initiator. In other words, each SCSI command, is executed by the AT in several small steps. At the completion of a single phase command an interrupt is generated to the host. This process is repeated, under host control, until all of the neccessary information packets have been transferred to the host to execute that command as desired by the SCSI Initiator.

As an alternative to the above target mode of operation, the host can reserve a block of system memory in anticipation of a command request from another initiator in the system. Using the SCSI SEND command for initiator to initiator communication the entire command, data, and status phase information can be passed provided the 2 initiators have previously agreed to the data block format. For each new initiator within the system, a new block of host memory space must be allocated using the open receive buffer command.

Since the ASC is a single threaded channel resource, commands can only be queued from either side, SCSI port or the host AT port, when no data transfers are taking place across the channel. To limit LCPU overhead and SRAM space, no more than 16 commands, can be queued at any given time. The queued commands are then executed in the order received by the LCPU as described above.

5.1.1. Power-Up Sequence

On RESET-, the ASC executes a power-up sequence to setup internal parameters and intialize the onboard circuitry properly. The LCPU resets and initializes the SBIC device, followed by initialization of any required tables and default values. Further details can be obtained from the ASC source listing. The minimum required RESET pulsewidth is 25.0 uS. Both the SCSI and the ASC reset strobes may be asserted simultaneouly if desired. If improper response, then strobe the SCSI reset first, followed by the ASC reset strobe.

Onboard ASC diagnostics will be executed for about 2 seconds. If an error is detected, the appropraite error code is set in the interrupt status register that can be read by the host after the LCPU has set the command port READY line. The status port will contain a 4FH value since the initialization sequence should not have been started yet (the lower nibble should be masked off by all application drivers). Hence, in order to detect any power-up diagnostic failures, the host should first read this interrupt status before requesting the ASC to execute a command even though an IRQ is not set. A value of 01 indicates no diagnostic errors have occured. Additionally the LCPU does not turn off an LED to visually signal a diagnostic error condition. Details of the diagnostics are discussed under the command section.

The LCPU is interrupt driven by the COMMAND PORT READY line from the host controller for command execution. Having read the diagnostic status, the host must then issue an initialization command and pass 9 additional bytes using programmed I/O. If any of these bytes are rejected (status = 60H), the host can simply resend the byte until it does not receive a rejected status (40H). The ASC channel has been properly initialized when the host can read a status of 5FH from the status port. The host must then enable the ASC DMA logic if desired, followed by the enabling of the system master mode. The LCPU will then direct the DMA control logic to read off the command block as defined in the following paragraphs and prepare the ASC for command execution.

5.2. Host Hardware Interface

The host utilizes four I/O ports for ASC control as defined in the table below. Using only 4 ports keeps the requirements on host I/O space to a minimum so that only 2 of the lower addresses are decoded to access the R/W ports. Address lines A9-A3 are jumper configurable so that the ASC can reside in any address space from 300-3FF. Additionally the DMA & IRQ channels are user selectable. See section 7 for all of the jumper options available.

Addr	Read Port	Type	Write Port	Type
0	ASC Status	D7-D0	Command Register	Byte
1	Host Interrupt Stat.	Byte	ASC Interrupt Ack.	strobe
2	Reserved		Host Control reg.	D3-D0
3	Reserved		Reserved	

Table 5-1: ASC I/O Port Definition

5.2.1. Status Port

The host must read the status on power-up or before issuing any command through the command port. On Reset the upper nibble of the byte is cleared and the lower nibble is set (status byte = 0FH) since no hardware is driving the lower nibble of the status byte. For future compatibility, this lower nibble should be masked off by all application driver routines. Hence the power-on status byte should appear as 00H and not 0FH as mentioned before. From now on, this lower nibble will be assumed to be zero. After the power-on diagnostics have been completed, indicated by status of 40H, the ASC is ready for the initialization bytes. The completion of the initialization is indicated by a status of 50H. If any command port byte is to be rejected, the status is set to 70H until the next byte is written to the command port. The format of the status byte is shown on the next page:

Bit	Description	Meaning
7	Interrupt Image Flag	0 = Interrupt Inactive 1 = Interrupt Active
6	Command Port Ready	0 = Port Busy 1 = Port Ready
5	Command Port Byte Rejected	0 = Byte Accepted 1 = Illegal Cmd/Param.
4	ASC Initialized Flag	0 = Init. Required 1 = ASC Initialized
3	Reserved.	Set to "1" Mask off this bit
2	Reserved.	Set to "1" Mask off this bit
1	Reserved.	Set to "1" Mask off this bit
0	Reserved.	Set to "1" Mask off this bit

Table 5-2: ASC Status Byte Format

5.2.1.1. Interrupt Image Flag (D7)

This reflects the image of the hardwired interrupt to the host. This enables the host to poll the interrupt line when the ASC interrupt is disabled on the system board specially when 2 or more I/O cards share the same interrupt channel. This bit will stay asserted until an ASC reset occurs or the host acknowledges the interrupt by writing to the logical port at address 1.

5.2.1.2. Command Port Ready (D6)

This bit is useful to transfer command bytes in the programmed I/O mode. The host can send the command byte when bit 6 of the status port is a 1. Writing a byte to the command port will reset this bit. The ASC sets the bit when it is ready for the next byte. Using this handshake technique, the pre-arranged number of command bytes can be transferred. The host should not write the next byte until this bit has been set. The service time between 2 programmed I/O bytes is around 70 uS but with a faster LCPU clock, the service time will be reduced to about 35 uS.

5.2.1.3. Commmand Byte Rejected (D5)

When the LCPU sets the READY line, the host should also examine bit 5 of the status byte to check if the byte written was accepted. This is required since there is no other means other than the ICMB to warn the host about an illegal command or parameter. The ICMB technique involves DMA transfer after the fact making it rather difficult for the host to take corrective action.

This bit also informs the host that an SCSI command has been entered in the SBIC queue or that a local ASC command via the command port has completed execution. If the SBIC queue is full, then the host command byte will be always rejected until space is available in the queue.

On power-up or during the Initialization command, this bit is set if any command other than the Initialization command is issued. For the 9 subsequent bytes, it indicates a parameter out of range and the host is required to send that byte again before transmitting the other command parameters. Any subsequent initialization attempts by the host will always be rejected unless preceded by an ASC reset.

5.2.1.4. ASC Initialized Flag (D4)

When this bit is reset, the host must issue an INITIALIZATION command to establish the mailbox parameters so that commands can be executed properly using first party DMA.

This bit is reset by host master reset (HMR or RESET DRV) at the interface or the ASC reset port. After the INITIALIZATION command has been executed, this flag is set by the LCPU. Following a successful initialization, if the host attempts to re-initialize the ASC, the command will always be rejected. To re-initialize the host must reset the ASC.

5.2.2. Interrupt Status Byte

The host read only registers also includes an interrupt information byte that contains the mailbox number pertaining to the current interrupt. Normally the interrupt information byte is valid only when the interrupt status bit is set implying that the host CPU must read this register before acknowledging the interrupt. However, the diagnostic error code is also latched into this register since this event and a host interrupt request are mutually exclusive events, with minimal hardware required. The diagnostic error code is valid after power-up, reset, or a diagnostic command, after the LCPU has set the READY signal to the host. The byte is formatted as follows:

76543210	Definition
00	Power-on condition, no diagnostics executed
01	No Diagnostic Error Occured
02	RAM failed
03	FIFO R/W failed
04	SBIC reg. R/W failed
05	Initialization D-FF failed
06	Host IRQ D-FF failed
07	ROM checksum error
10NNNNNN	Outgoing mail box number 'N' is available. Used only when "Interrupt on Free OGMB" command has been issued.
11NNNNNN	Incoming mail box 'N' needs service.

NOTE: Codes 00-07 valid after READY is asserted in the status port
Codes 05-06 valid only after when ASC is first powered-up

Table 5-3: Interrupt Status Byte / Diagnostic Error Code

5.2.3. Command Port

The host issues ASC control commands via the first write port to get it's attention. Most control commands are 1 byte instructions except the initialization command which must be sent to the ASC after power on or ASC reset and the SCSI soft reset command. The format of these commands is shown in section 6 under supported commands.

Refer to table 6.1 for a list of all commands the host can issue using the command port.

The command port allows the host to queue commands at any time desired as opposed to just scanning the mail boxes once when the START MULTIPLE I/O command is issued. For the present no more than 16 host commands can be queued at any given time. When the queue is full, the command port byte will be rejected by the LCPU.

Also if a previous command to the same target and LUN has not completed execution, then the second command to the same target & LUN will be terminated without being issued to the desired target. If the host wishes to issue more than one command to the same target LUN, then SCSI command linking should be invoked.

5.2.4. Interrupt Acknowledge

This input strobe clears the hardware Interrupt Request to the host. In the case of queued commands, the ASC will not set the Interrupt unless the previously set interrupt has been acknowledged by the host.

The IRQ signal status to the host is queued but only after the command status has been written to the host via the incoming mailbox described in the next section. The interrupt acknowledge signal clears the previous IRQ signal and also frees up an ICMB so that it can be re-used. Hence if the number of these mailboxes allocated is small, then the host should service the interrupts as quickly as possible, else the execution time for the next command in the queue will be proportionately larger.

5.2.5. Host Control Register

The host can write to the control port to selectively enable or disable the functions indicated in the table below. On power-up Reset or RESET DRV, all registers are cleared to all zeros. If the upper nibble is not cleared, the application drivers should mask off this upper nibble.

Bit	Description	Meaning
7	Reserved.	Not Used
6	Reserved.	Not Used
5	Reserved.	Not Used
4	Reserved.	Not Used
3	Interrupt Enable	0 = Interrupt Off 1 = Interrupt On
2	DMA Request Enable	0 = DRQ Disabled 1 = DRQ Enabled
1	SCSI Port Reset	0 = SCSI Reset Off 1 = SCSI Reset On
0	ASC Reset	0 = ASC Reset Off 1 = ASC Reset On

Table 5-4: Host Control Byte Format

5.2.5.1. ASC Reset (D0)

This 1 bit register allows the host to reset only the ASC controller instead of the entire system using the RESET DRV input line at the host interface. The host writes twice to this port, a "1" followed by a "0", so that the reset pulsewidth can be tailored and onboard logic minimized without excessive use of the bus bandwidth on the AT backplane. The recommended reset pulsewidth is 25 uS. Following this reset, the ASC must be re-initialized using the 10 byte sequence.

5.2.5.2. SCSI Reset (D1)

Similar to the ASC Reset except that the SCSI port is reset instead of the ASC or the entire AT system. For software resets, refer to the commands described in the next section. If desired both the ASC reset and the SCSI hardware resets can be asserted simultaneously by writing a 03H to this port followed by a 00H keeping the minimum pulsewidth requirements in mind.

Usually, this reset will not generate an interrupt to the host unless the unsolicited interrupt mask is enabled. If there are commands being executed by the ASC, then an interrupt will be generated regardless of the state of the unsolicited interrupt mask. To differentiate between these 2 types of interrupts, 2 different ICMB status codes are generated as shown by the vendor unique status/error codes listed in appendix A.

5.2.5.3. DMA Request Enable (D2)

The host can program the DRQ signal from being enabled/disabled on the host bus by simply writing a "1/0" to this D-FF. This is done so multiple devices can share the same DMA channel. The host must write a "1" to this register before first party DMA transfers by the ASC can begin. Since writing to this register enables the selected tri-state DRQX driver on the host bus, this register must first be enabled before the system DMA is enabled or else the ASC may behave unpredictably. A typical DMA sequence is as follows:

```
O    322  4              ;enable ASC DMA tri-state driver
O    D6   C2             ;set DMA on channel 6 into cascade mode
O    D4   2              ;enable DMA master mode
```

5.2.5.4. Interrupt Request Enable (D3)

The host can program the IRQ signal from being enabled/disabled on the host bus by simply writing a "1/0" to this D-FF. This is done so multiple devices can share the same IRQ channel. The host must write a "1" to this register in order to enable the IRQX line to generate a hardware interrupt to the host or poll the status of the IRQ signal via the host status register.

As done with the DMA enable, the IRQ channel must be enabled first. For the convenience of users, both the DMA and the IRQ enable drivers are included in appendix C. The drivers are written using microsoft C. If different channels (other than the factory configuration) are used, then these source programs must be changed slightly. These source files may be obtained from your marketing representive or by writing to the marketing dept. of Western Digital Corporation.

As done with the 2 resets above, both the DMA and the Interrupt request enables can be set/reset simultaneouly if required by just setting the individual bits to their desired states: a "1" to enable the desired function & a "0" to turn it off.

5.3. Mail Boxes

In order to fetch and execute a command block setup by the host in it's system memory, the host must provide the I/O channel with the starting address of the block. Likewise, when the ASC completes a command, it must indicate to the host the address of the CDB it has just completed. These handshakes are accomplished via Outgoing and Incoming mail box systems. The terms Outgoing/Incoming are with respect to the host system. At system startup the host issues the initialization command which sets the number of outgoing/incoming mail boxes and the starting address of mail box block. It is assumed that all of the outgoing mailboxes are followed by the incoming mailboxes and that these do not occur in pairs. In fact, one may specify more incoming mailboxes than the outgoing mailboxes and vice versa. The relative mail box number is given in the command port control opcodes. The format of a typical mailbox in memory is shown below:

Mail Type	Contents
O.G.M.B. #n	Mail Status (0=empty, else full) Command Block Pointer (MSB) Command Block Pointer Command Block Pointer (LSB)
I.C.M.B. #m	Action Status Command Block Pointer (MSB) Command Block Pointer Command Block Pointer (LSB)

OGMB Address = Starting address of mail block + mail box number n x 4

ICMB Address = Starting address of mail block + mail box number m x 4 + 4 (total number of OGMB's)

where 0 <= n <= p-1 & 0 <= m <= q-1

p,q = number of OGMBs/ICMBs resp. (max. = 64)
m,n = mail box offset numbers from 0 to 63 (max.)

Table 5-5: Mail Box Format

5.3.1. Outgoing Mail

An outgoing mail box is marked as full or empty by the mail box status byte. A value of zero means that the ASC has picked up any mail that may have been in the box and the host may issue a new command via that particular box. A non zero value indicates a full box which the ASC has not had time to service.

After the host has set up the OGMB command pointer, it sets the status byte to full (non zero). The host then notifies the ASC to either check all the boxes, starting at the base address of the OGMB with a particular mail box signature, or just a single mail box via the direct ASC START commands. The ASC reads the necessary amount of mail boxes and marks them empty. Generally the commands are executed on a first come basis. However, if disconnect is supported within the SCSI system, several commands could be in progress and the order of execution may not be the same as the start.

Since the ASC has a limited number of SRAM & I/O sub-channels, it cannot concurrently process an infinite number of commands. Once the ASC has occupied all of it's sub-channels it will respond to 'START' commands with rejected status. It will not empty a mail box until it has completed a pending I/O command. The host will know that the ASC limit has been reached by the fact that all the OGMB's are full and any additional host commands are being rejected. At this point the host has 2 options: it may continue to poll for an empty mail box or it may issue an "Interrupt on Free OGMB" command. As soon as the ASC has freed an OGMB an interrupt is generated and the proper status is reported to the host as shown in table 5.3. This interrupt is generally followed by the command completion interrupt triggered by that OGMB.

5.3.2. Incoming Mail

The ICMB number can be read from the interrupt status port after the interrupt has occured. The ICMB is used to notify the host which command has completed execution or which other SCSI Initiator has requested service. The ASC code, indicating completion or host attention, is the first byte in an ICMB. The next three bytes point to the command block in memory that the ASC just finished executing. These bytes are the link back from the ASC to the operating system I/O driver that generated the command block. In the case of another SCSI initiator these 3 bytes are defined as indicated in the ICMB return status byte table, otherwise they are meaningless. A value of 0FFH is used whenever these bytes are undefined or meaningless.

Refer to the interrupt status byte table 5.3 for the 2 valid codes returned for commands associated with a mailbox.

5.3.2.1. ICMB Return Status

The ICMB return status bytes are broken down to two classes. The first class (00-7F) is known as solicted interrupt status. These are codes accompanying interrupts which were in response to a host command to the ASC. These are further broken up into several major categories to help decode error codes quickly and aid field replacements. These divisions are somewhat arbitrary but self explanatory. A new ICMB status code should be created whenever another category not implied by the existing codes is uncovered.

The second class, unsolicited interrupts, accompany an interrupt as a result of a ASC detected condition. These interrupts do not have a corressponding host request (CDB) for an interrupt. Besides host service attention status, these codes are typically used for reporting an error condition that can be treated as solicited interrupts if an error logger exists in the host system to record their occurance. The host can disable or enable unsolicited interrupts via the commands described in the next section. The codes which can be returned in the return status bytes are defined in appendix A along with the associated vendor unique error codes.

The vendor unique error codes are once again divided into major groups as far as possible. Although no hard and fast rules were used to generate these codes but a pattern such as hardware errors, illegal command parameters, SCSI protocol errors, ASC detected command status, etc. should be obvious by looking at the vendor unique error codes defined in the appendix.

5.4. Command Blocks (CB)

In order to perform a ASC Interface Channel operation, the host must first build a command block in memory. Two types of command blocks are defined. The first type, a SCSI Command Block (SCB), is intended to be processed by an SCSI device utilizing the host adapter as the DMA channel into host memory. The SCB is always used when the ASC is operating as the Initiator. The Interface Command Block (ICB) contains a functional command to be executed by the ASC which may sometimes involve the SCSI bus, specially when selected as a target by another SCSI initiator, else it is intended for local execution.

Whenever a parameter field is greater than a single byte, the MSB byte is always specified first. This is done to conform to the SCSI standard where the MSB byte is always specified first within the CDB. Since the SCSI CDB is really part of the SCB, it helps avoid some confusion if another alternate standard were to be adopted.

5.4.1. SCSI Command Execution Block (SCB)

	Bit Position / Description							
Byte	7	6	5	4	3	2	1	0
00	S/I	Command Opcode						
01	Target ID			0	0	L.U.N.		
02	SCSI Command Block (Byte # 0)							
thru	SCSI Command Block Bytes 1 - 10							
13	SCSI Command Block (Byte # 11)							
14	SCSI Return Status Byte							
15	Vendor Unique Error Code							
16	Maximum Data Transfer Length (MSB)							
17	Maximum Data Transfer Length in Bytes							
18	Maximum Data Transfer Length (LSB)							
19	SCSI Data Block Pointer (MSB)							
20	SCSI Data Block Pointer							
21	SCSI Data Block Pointer (LSB)							
22	Next Command Link Pointer (MSB)							
23	Next Command Link Pointer							
24	Next Command Link Pointer (LSB)							
25	Direc	Reserved for Future Expansion						
thru	Reserved for Future Expansion							
30	Reserved for Future Expansion							
31	Reserved for Future Expansion							

Table 5-6: SCSI Command Execution Block

The most significant bit of the first byte (00) indicates whether the CB is intended for an SCSI port execution or meant for execution by the ASC. A value of 00 specifies a SCSI command where the ASC acts as the SCSI Initiator. Values of 01-7F (hex) are reserved for future expansion. Values of 80-FF are used as ASC command codes.

Byte 1 contains sufficient information for the LCPU to direct the SBIC to start arbitration, the selection process, and ensure that the data transfer,if any, is initiated.

Bytes 2 through 13 are written to the SBIC device and the other bytes are used to access additional parameters required to complete the command. The command completion status, as generated by the selected SCSI LUN, is logged into byte 14. Any errors detected by the ASC are logged into byte 15. Note that not all of these error codes imply a command failure. Some of these only imply a warning or a status to the host driver e.g. vue = 40H implies less data was transferred by the target than the space allocated by the maximum transfer count field (bytes 16-18). For a listing of all possible error codes, see appendix A.

Bytes 16-18, data transfer length, are provided as an input to the ASC to keep a "run away" target from overwriting host memory. This field specifies the maximum number of bytes which the ASC will allow the target to write or read. If an attempt is made to exceed this value, a ASC error will be generated and the command terminated. The next 3 bytes, form a 3 byte data pointer that allows I/O channel addressing of 16M bytes.

Bytes 22-24 provide the next SCB address which will be used by the ASC with chained commands i.e. next command to be executed after the current command has been completed, depending upon the LINK bit in the SCSI CDB. If the target does not support link commands, an error will be generated and the link chain will be broken. If the link flag in the SCSI CDB control byte is set, then a host interrupt will be generated between each link command. If the link flag is reset, then only one host interrupt will be generated at the end of the link or on completion of a command in the link for which a SCSI status other than good completion is returned by the target.

Bit 7 of byte 25 is the direction bit and is set whenever data is to be written to the host. Hence all SCSI read operations that involve data transfer should have this bit set. Conversely all write operations to the SCSI bus should have this bit reset. This bit guarantees the integrity of the data transfer in the case of a runaway target that does not conform to the direction implied by the CDB within the 32 byte SCB.

All other bits and bytes are reserved for future expansion and should be set to zeros for compatibilty reasons.

The figures in appendix A illustrate the start of command execution and completion using the mailbox scheme described earlier.

5.4.2. Interface Command block (ICB)

The ICB is comprised of the command byte, 14 parameter bytes, and a status byte as shown in the table below. The parameter bytes are defined by the various ASC commands discussed in the next section.

Byte	Bit Position / Description 7	6	5	4	3	2	1	0
00	S/I	Command Opcode						
01	ICB Parameter Byte # 1							
02	ICB Parameter Byte # 2							
03	ICB Parameter Byte # 3							
04	ICB Parameter Byte # 4							
05	ICB Parameter Byte # 5							
06	ICB Parameter Byte # 6							
07	ICB Parameter Byte # 7							
08	ICB Parameter Byte # 8							
09	ICB Parameter Byte # 9							
10	ICB Parameter Byte # 10							
11	ICB Parameter Byte # 11							
12	ICB Parameter Byte # 12							
13	ICB Parameter Byte # 13							
14	ICB Parameter Byte # 14							
15	ASC Vendor Unique Error Status							

Table 5-7: ICB Command Execution Block

5.5. Command Queuing

To keep the LCPU overhead and the RAM requirements to a minimum, 16 logical threads will be supported. In other words no more than 16 host and/or 16 SCSI commands can be pending execution on the ASC. Commands will be processed on a FIFO (first in first out) basis. No more than 1 command per LUN can be queued i.e. any given LUN can only have 1 command pending or in execution but never both. To manage command execution properly, 4 queues are used. The host command queue, the SBIC queue, and the SCSI command completion queue all support 16 commands each and the host interrupt queue can handle up to 32 command completions subject to the number of ICMBs specified by the host. These 4 queues are serviced in a round robin fashion as shown in the figure below:

```
New SCSI cmd issued if        Host write to         IRQ if previous
disconnect & cmds left        ASC cmd port          ACKed, frees ICMB
   <----+                        +<----                   +----->
        |                        |   INC CPQCNT           | DEC IRQCNT
  +-----+-----+            +-----+-----+            +-----+-----+
  |           |            |           |no data xfer|           |
  |    SBQ    | if SBQ is  |    CPQ    | ICMB avail.|    IRQ    |
  |           | not full   |           |IRQ not full|           |
  |  16 X 2   |<-----------+  16 X 1   +----------->|  32 X 1   |
  |           | DEC CPQCNT |           |            |           |
  | 2 byte ptr| INC SBQCNT |1 byte cmd | DEC CPQCNT |           |
  |           | INC SBQLFT |           |            |           |
  +---+---+---+            +-----------+            | 1 byte of |
      |   |                                         | ICMB num. |
      |   +-------->+  SCSI cmd completion buffer   |           |
      |             |  order of compl. is random    |           |
  +---+-------+     |      +-----------+            +-----+-----+
  |           |     |      |           |if no data xfer   |
  |SCB storage|     |      |    CCQ    | ICMB avail.      |
  |           |     |      |           |IRQ not full      |
  |  16 X 32  |     +----->|  16 X 2   +----------------->+
  |32 byte cmd|            |2 byte ptr | DEC SBQCNT
  |           |            |           |
  +-----------+            +-----------+
  accessed via current
  SCB pointer (CURSCB)
```

Fig. 5-1: Command Queue Flow

To begin, the host issues a command by writing a byte into the command port using programmed I/O. If the command specifies a mailbox access, it is queued, else it is executed immediately. In either case, the READY signal is set in about 70 us, and the host should read the status register to check if the command byte has been accepted.

After the command has been entered in the host queue, the specified mailbox is accessed to begin command execution. If the command is an SCSI command, then it is entered in a separate queue, or else it is a local ASC command that is to be executed. As long as the command to be executed does not require any additional data for completion, it is executed without further delay. However, if data transfer is required, then the host memory is accessed again provided no previous data transfer is in progress. If the bus is being used by the DMA channel, then command execution is suspended until freed by the DMA channel. By the same token, if the data bus is being used by the LCPU to complete command execution, then DMA transfers cannot start until the LCPU has finished accessing the host memory.

When the command has completed execution, the host is notified via the ICMB. Once again the same data transfer in progress restrictions (as described above) apply to write the ICMB data to the host. After successful ICMB completion, the command is removed from the host queue, and the interrupt status byte specifying the ICMB used is entered in another queue designed solely to interrupt the host.

If the previous interrupt has been acknowledged by the host, then the host will be notified of the next command completion entered in this queue. The interrupt acknowledge is also used to free a previously used ICMB so that it can be re-used. If the number of ICMBs specified by the host is small, then the interrupt request should be serviced by the host quickly in order to avoid delaying the execution of the next command entered in the host queue. This is specially true in the case of an ICB type command type command since a new host command will not be de-queued for execution unless the previous command under execution has been added to the interrupt queue after the appropriate ICMB and vendor unique error codes have been updated.

New commands are issued to the desired target providing the previous target has disconnected from the SBIC device. If no other exception handling is required, then command completion is reported to the host in a similar manner as described above. The completed command is then removed from the SBIC queue.

The entire internal command flow is summarized by the flowcharts contained in appendix B.

5.6. SCSI Protocol

The SCSI protocol followed is defined by the ANSI SCSI specification mentioned in the references. The WD33C93 SBIC data sheet specifies the protocol requirements used for the ASC design for all communication with the SCSI port.

5.6.1. ASC Supported Messages

The SCSI message system allows communication between an initiator and the target for the purpose of physical path management.

Most SCSI messages, shown in the ANSI SCSI specification, are handled by the LCPU totally transparent to the host CPU. In addition, hooks for some messages like command ABORT and SCSI RESET messages have been provided via the I/O command port, since these are useful at the host level as well. These are described in the next section describing all of the ASC commands supported, under "SCSI soft reset". Among the extended messages, only the synchronous data transfer request message is supported for the present.

For a detailed description of all the messages supported, refer to the ANSI SCSI specification or the ASC source listing. Only the EXTENDED IDENTITY and the MODIFY DATA POINTER messages are not supported. The link message set is supported, providing the selected SCSI target supports these 2 messages.

5.7. Exception Handling

An exception is defined as a condition from which a command cannot be executed to it's normal expected completion. Accordingly, whenever an unrecoverable error condition is detected by the ASC, the command is terminated with appropriate SCSI status and vendor unique error codes set in the ICMB byte 0, SCB bytes 14, 15 or simply the ICB byte 15. Refer to appendix A for all possible status and error codes that can be returned. New codes will be added to this list as the need for new error conditions is uncovered. For the most part these codes are self-explanatory but a brief "error philosophy" shall be outlined here.

In most cases, an interrupt will be returned to the host if the command is unsuccesful. The normal rule is that all commands associated with an OGMB will receive a host interrupt. The only exception to this rule is if the target disconnects normally and never reconnects to finish the operation. For these cases, the host must use a timeout that selects the nature of the target attached used in the application. No standard timeout value was used by the ASC since these values differ greatly depending upon the specific application requirements.

5.7.1. Error Reporting

For all commands associated with an OGMB, the host is informed about the command completion status via a host interrupt using the ICMB0 as the vendor unique status code, the SCSI status byte itself, and a vendor unique error code to further break down the type of error encountered.

For commands with no associated mailbox, an unsolicited interrupt is generated, provided the mask bits are properly enabled. For all other commands issued by the host, the command byte will be rejected as discussed earlier in this section.

Basically, the error handling procedure consists of the following general guidelines:

If an unrecoverable error is detected due to parity, unexpected information phase, etc., the target will be issued an abort command message and the command terminated with the error reported back to the host. In other words, whenever possible, the target will be forced to disconnect if an unknown state occurs during the SCSI session.

If a catastrophic error causes the SBIC to report an invalid command interrupt status, the command is suspended with the target connected since the SBIC is basically lost. This condition will require a host reset of the ASC. This condition should rarely occur, if ever.

Parity error retries are limited to 2 attempts. However, this number can be changed mainly for in-house testing purposes as defined by the set execution parameters command.

5.7.1.1. ICMB status byte 0

These are mainly divided into 2 classes. The first class (01-7FH) is known as solicited interrupts which are returned in response to a host command to the WD7000-ASC. This byte will always contain "01" if the command finished execution successfully with a SCSI status phase completion. If the command did not get a status phase completion or was not executed for some reason (e.g. illegal command), then the command is deemed to have failed. Accordingly, a status code other than "01" is assigned, depending upon the type of error encountered. If the command completed but some anormality was observed, then an ICMB status code of 02 is returned. For example, the maximum transfer count field was set at 00 00 20 (hex) but only 10 (hex) bytes were received by the ASC, then a vendor unique error (vue) code (byte 15 of SCB/ICB) will be returned to the host even though no actual errors occured in the data transfer. This type of vue is really a warning or a status to the host rather than a real error. In other cases, a real error may have occured, like a hardware error or a parity error.

The second class (80-FFH), known as the unsolicited interrupts, accompany an interrupt as a result of the ASC detected condition. These typically do not have a corresponding host request for an interrupt. These can occur when another initiator treats the ASC as a target unknown to the host. Depending upon the application, the host may or may not wish to deal with these interrupts. Accordingly, these can be turned off/on by the ENABLE UNSOLICITED INTERRUPT & DISABLE UNSOLICITED INTERRUPT and the unsolicited interrupt mask byte defined in the SET/READ PARAMETRS data block. Bits 4-0 will result in interrupts 84-80 as defined in the next section. These are generally maskable only if no current commands are being executed by the ASC. If a command is currently under execution, and a maskable interrupt occurs (e.g. SCSI hard reset), then a host interrupt will always be generated regardless of the mask bit.

5.7.1.2. SCSI Status Byte

The SCSI status byte reported by the target is always returned to the host in byte 14 of the SCB. This byte is not modified by the ASC in any way. If this status byte is anything other than "00" then an ICMB status of 02 will be returned. If no status byte is reported by the target, then an ICMB status higher than 02 will be returned depending upon the type of error encountered.

5.7.1.3. Vendor Unique Errors

The vue is always returned in byte 15 of the SCB or the ICB. These codes are roughly divided into several groups merely for convenience and ease of field replacements. The major groups are hardware errors, illegal command parameters, and SCSI protocol related errors or command status. Some of these error codes indicate the phase when the unrecoverable parity errors occured, but these are more for debugging purposes. The operating system can regard all of these as unrecoverable SCSI parity errors. The host may retry the command if desired simply by re-issuing it.

5.7.2. SCSI Parity Errors

The number of retries used for SCSI bus parity errors can be programmed by the user using the set parameter command described later. The default value, as recommended by the ANSI standard, is 2. For the most part, the target is allowed to control the number of retries it wishes. If this count is exhausted, the target will disconnect. For all other cases (i.e. messages in initiator mode), if a parity error was detected, then that phase will be retried until the ASC retry count is exhausted. The ASC will send a SCSI abort message to the target to terminate the operation with ICMB0 = 04 and vue = 10, 50, 52, 35, etc. reflecting the error condition found, returned to the host in the normal fashion.

5.7.3. Hardware Errors

If the SBIC were to return an unexpected interrupt status, the command is terminated with the target still connected to the bus. No attempt is made to abort the SCSI command under execution since the SBIC device used to communicate to the SCSI port has failed. The host must reset the SCSI bus using the physical hardware line through the ASC control port and re-issue all unfinished commands or attempt to issue the SCSI soft reset command described in the next section.

If a SCSI reset (soft or hard) were to occur for a command under execution, then the host will be notified the usual way via the ICMB with a valid CDB address field. At this point, the host may choose to abort one or more commands using the SCSI soft reset command described in the next section.

If no command was under execution and if the unsolicited interrupt mask bits were enabled, then the address field will be all FF's. This latter case is only useful statistical analysis for a system where an error logger exists.

6. COMMANDS

All of the commands supported by the ASC are discussed in this section. These fall into two broad categories. The SCSI commands are merely passed through the ASC with the LCPU handling all of required handshakes to accomplish such a transfer. For a description of the SCSI commands supported, refer to the ANSI SCSI specification or the list of commands supported by the AT host and the SCSI devices used in the system.

The second category of commands is executed by the ASC. These include setup and execution of commands by the host as the INITIATOR or as the TARGET. In the INITIATOR mode the SCB contains all of the necessary information to execute the entire command. But in the TARGET mode, the host sets up the ASC, using the ICB format, to execute the command in several small steps. In case of data transfers, the scsi status phase may be combined with the data phase, to eliminate 1 host interrupt. For inititiator to initiator mode of communication 2 commands are provided combining all of the SCSI information phases so that the host has to deal only with a single interrupt. In either mode, the command port is always available to the host to control the ASC. Please note that the ICB format can also be used to control the ASC, if desired. The distinction is that command port commands are usually 1 byte action commands using programmed I/O, whereas for the SCB or the ICB, first party DMA access is required to complete the command.

Before any commands can be executed by the ASC following power-up or ASC hardware reset, the host must issue the initialization command when the ASC READY status is asserted, even if a diagnostic error may have occured. The initialization command is discussed in the next section. Following this command, the host should enable the DMA and the IRQ signals on the ASC in order to drive these signals on the host bus, before enabling the corresponding functions in the host system. This is required to turn on the tri-state drivers associated with these signals, else spurious signals may be received by the host due to the tri-state nature of these 2 signals.

For complete command execution details, refer to the source listing mentioned in the references or the appendices.

6.1. Command Port Control Opcodes

These control commands, in the table below, are issued by the host by writing a single byte to the command port. Some commands like the initialization and the SCSI soft reset require more than 1 byte. The disable/enable unsolicited interrupt commands may also be issued using the set parameters command if desired.

Opcode	Definition
00	No Operation.
01	Initialization (10 Byte sequence)
02	Disable Unsolicited Interrupts
03	Enable Unsolicited Interrupts
04	Interrupt on Free OGMB
05	SCSI Soft Reset (2 byte sequence)
06	SCSI Hard Reset Acknowledge
07-7F	Reserved
10NNNNNN	Start Command in OGMB #N
11NNNNNN	Start Multiple I/O (Scan Mailboxes) N = Scan signature 6 bits max.

Table 6-1: ASC Command Port Opcodes

6.1.1. No Operation (00)

This command does nothing other than toggle bit 6 of the read status port 0. It is useful for diagnostic reasons to make sure that the LCPU is at least functioning partially.

6.1.2. Initialization (01)

The initialization command must be sent to the ASC after power on or ASC reset. The host can send the command byte when 6 of the ASC Status port is 1. Writing a byte to the command port will reset this bit. The ASC sets the bit when it is ready for the next byte. Using this programmed I/O technique, the remaining 9 bytes of the initialization command can be sent to the ASC. These 9 bytes establish the number of outgoing and incoming mail boxes, the starting address of the entire mail box block, and the bus on/off times. All other programmable parameters can be selected by the host via an ICB.

It is assumed that the OGMB's are followed by the ICMB's so that the relative mail box number, specified by the control port command byte, can be added to the 3 byte base address specified in the table below to compute the absolute mail box address. Furthermore, it has been assumed that the mailbox or any of the data blocks do not wrap around memory from the highest 24 bit memory address to the lowest memory address of zero.

In order to change any of the parameters defined by this command, the host must reset the ASC. At all other times this command will be rejected by the ASC.

In the table below, a value of zero in bytes 8 and 9 will be taken to imply that at least 1 OGMB and 1 ICMB are allocated by the host. Hence the maximum value that the host can specify in these 2 bytes is 64 instead of 63.

Similarly if byte 2 is set to zero and a DMA transfer is required at least 1 word will be transferred by the first party DMA controller before giving up the host bus. A value of zero in byte 3 implies that the DMA controller will relinquish the bus for only 125 nS before re-arbitrating for it again. The maximum on time used should be less than 15 uS less ALL overhead time required to allow the host to service memory refresh cycles, including DMA bus arbitration time. The ASC DMA controller transfers a word every 375 nS once it has control of the bus. For optimum performance the host should use a bus on time of 3.0 uS to transfer 8 words, the length of the FIFO. Bus arbitration overhead is about 1 to 2 us.

Byte Num.	Bit Pos. 7654 3210	Definition
00	0000 0001	Command Opcode
01	0000 0SSS	3 Bit SCSI ASC ID Field
02	TTTT TTTT	Bus On Time (125 nS per count value)
03	TTTT TTTT	Bus Off Time (125 nS per count value)
04	0000 0000	Reserved
05	AAAA AAAA	Starting Address of Mail Block (MSB)
06	AAAA AAAA	Starting Address of Mail Block
07	AAAA AAAA	Starting Address of Mail Block (LSB)
08	0NNN NNNN	Number of OGMB (max. 64) (0,1 = 1)
09	0NNN NNNN	Number of ICMB (max. 64) (0,1 = 1)

Table 6-2: Initialization Command Format

6.1.3. Disable Unsolicited Interrupts (02)

This command allows the host to turn off interrupts generated by another initiator on the SCSI bus. In this mode, the ASC will not respond as a target.

By resetting bit 0, byte 22, via the set parameter command, the user can accomplish the same effect.

6.1.4. Enable Unsolicited Interrupts (03)

This command allows the host to turn on interrupts generated by another initiator on the SCSI bus. In this mode, the ASC is enabled to respond as a target. The interrupts due to a SCB/ICB command execution are always set.

By setting bit 0, byte 22, via the set parameter command, the user can accomplish the same effect.

6.1.5. Interrupt On Free OGMB (04)

This command is issued by the host when all of the OGMB's have been used and it does not wish to scan for an available OGMB. The ASC will generate an interrupt to the host as soon as one is available. The relative OGMB number can be read by the host from the interrupt status byte (ASC read port 01).

By setting bit 1, byte 22, via the set parameter command, the user can accomplish the same effect.

6.1.6. SCSI Soft Reset (05)

Since SCSI soft reset functions are handled at the Message system level, this command must be implemented as a ASC function as opposed to a SCSI command level function. Hence the LCPU sets up the SBIC to transmit a reset or abort message to the desired target as opposed to passing the SCB to the SCSI device as done by the SCB format.

	Bit Position / Description							
Byte	7	6	5	4	3	2	1	0
00	0	0	0	0	0	1	0	1
01	SCSI Target ID			0	R.T.	L.U.N.		

LEGEND: R.T. = Reset Type : 0 = Abort SCSI command message
1 = SCSI Bus Device Reset message

Table 6-3: SCSI Soft Reset Format

The ICB format to implement this command cannot be used since all these types of commands are normally queued. This might be too late to kill a SCSI command already in progress. Hence this command is implemented as a 2 byte sequence via the command port to get immediate attention of the ASC. When aborting a SCSI command issued by the host, only the target ID and LUN need be specified (byte 1) since the host is not expected to have more than 1 command active per unique value of byte 1.

If a SCSI soft reset is already in progress, a second soft reset cannot be issued by the host until the present soft reset has completed execution. The host is informed of the command completion in the normal fashion using an ICMB containing the absolute address of the command terminated, along with a vue code in SCB15. The abort message terminates a specific SCSI command whereas the reset message terminates all commands issued to a given target regardless of the LUN field. Hence for a reset message, the host can expect more than one interrupt depending upon the number of commands outstanding.

It is legal to issue this command even if there have been no SCSI commands issued to the desired target. Refer to the ANSI specification for the effect of these 2 types of soft resets on the target.

6.1.7. SCSI Hard Reset Acknowledge (06)

In order to facilitate SCB execution recovery, in the event of a hardware reset on the SCSI interface, the host will be notified only if:

a. There are commands in the SCSI Q to be executed.
b. The command completed Q is empty.
c. All previous host interrupts have been acknowledged.

If there are no commands in the SCSI Q, the host will not be informed about the SCSI hard reset unless the proper unsolicited interrupt mask bits are set. When a SCSI command is outstanding the host will be informed via an ICMB, regardless of any mask bits set by the host. At this point, the ASC will enter a pseudo idle loop. The only commands that the host should enter at this point are the reset/abort type commands described above. The host may abort as many previous commnds issued as it wishes or none at all. As each abort sequence is completed, the host will be notified via an ICMB, at which point another abort command may be issued. This process will continue until all desired commands have been aborted.

The end of the pseudo idle state is indicated by another "SCSI RESET ACKNOWLEDGE" command (opcode = 06). At this point normal command processing is resumed by the ASC.

This special sequence is required during a SCSI harware reset, since a given command can be anywhere in it's processing. Besides, during a non-reset state, several other targets may have to be serviced ahead of the abort/reset message depending upon who wins the arbitration phase. Hence the host must wait for the ICMB status before issuing another reset/abort message command.

6.1.8. Start Command (80 - BF)

These opcodes are used to start the execution of a single I/O command pointed by a specific OGMB. The lower 6 bits of the opcode are used to specify the OGMB number.

6.1.9. Start Multiple I/O (C0-FF)

This command is provided so that the host can initiate multiple commands at the same time. The ASC LCPU will scan all the OGMB's and execute the commands pointed to, if the mail box full status has been set by the host. This non zero byte can be used by the host to identify the 6 bit signature passed with the scan command so that only mailboxes bearing the correct signature will be executed. The order of execution is on a FIFO basis, starting with the base address of the OGMB, up to the maximum number of OGMBs specified with the initialization command. At the completion of each command a separate interrupt will be generated. This simplifies the ICMB information reported back to the host since holes can exist in the OGMB's.

It should be pointed out that the scan mailbox command may be somewhat inefficient because the mailbox memory is scanned in blocks of 16 bytes per command execution poll sequence starting from the base address. Hence depending upon which OGMB contains the command to be executed, the overhead incurred in command execution could be significant. Also note that compared to executing a command contained in a specific OGMB, the computation time of the OGMB address and all associated handling, even though small, adds to the execution time of a given command.

On the completion of the scan command, an additional host interrupt is generated informing the host about the completion of the scan mailbox command.

6.2. ASC Commands (ICB)

The ASC commands include all of the host adapter functions which are outside the scope of the SCSI INITIATOR specification. These functions include operation as a target in a host/host communication activity, local ASC commands, specifying programmable parameters, etc.

For all of the commands supported, the ASC will signal completion of the command by loading an ICMB and generating an IRQ to the host. The interrupt status byte contains the relative mail box number, which in turn contains the command status and the 3 byte address of the CDB that initiated the command. The host must acknowledge the interrupt by sending an interrupt acknowledge to complete the handshake, only after all of the error handling has been completed.

For commands initiated by another SCSI bus initiator (other than the local host), it is possible for the ASC to complete all phases of the target mode operation provided, all of the requisite data has been preset ahead of the anticipated SCSI commands such as SEND, RECEIVE, REQUEST SENSE, & INQUIRY. In order to enable this target mode of operation, the host must enable unsolicited interrupts as expained under Set Execution Parameters command and have opened the appropriate buffers containing the required data to complete these commands. Since there are no CDB pointers associated with these 3 commands, the host interrupt generated will have the following altered ICMB format to identify the correct command completion. Bit 7 of byte 0 of the ICMB is set to indicate a command completion IRQ that has not been initiated by the local host. Valid values are 80-87, depending upon the interrupt mask bit enabled and the SCSI request received.

ICMB Byte 0 = command completion status byte (normally 81)
ICMB Byte 1 = any applicable vendor unique error code
ICMB Byte 2 = packed SCSI ID + LUN (like byte 1 of the SCB/ICB format)

ICMB Byte 3 = 0FFH or reserved for future use

In case of any errors are exception conditions, all combination commands are immediately halted with an interrupt generated to the host. The host must then complete the operation manually using the command, data and/or status ICB commands if possible. For parity and unrecoverable error conditions, the host should prepare the appropriate request sense data packet, per the applicable ANSI standard or the SCSI device OEM manual, before acknowledging the ASC interrupt. If the proper request sense data buffer is not opened, the host will be interrupted when the request sense command is received, else the operation will be completed with a single interrupt after the anticipated request sense is fully executed.

Some vendor unique error codes, like less data transferred, should not be viewed as a true error condition but merely as a warning to the host so that the scratchpad CDB buffer contents can be used to determine the number of bytes actually transferred, if required.

The ICB commands supported are summarized in the table below:

Opcode	Definition
80	Open Inbound Data Buffer (Init-to-Init)
81	Receive Command from another Initiator
82	Receive Data from another Initiator
83	Receive Data with Status from another Initiator
84	Send Data to another Initiator
85	Send Data with Status to another Initiator
86	Send Command Status to another Initiator
87	Open Scratchpad CDB (for exception handling)
88	Read Initialization Bytes
89	Read ASC SCSI ID
8A	Set Execution Parameters
8B	Read Execution Parameters
8C	Read Firmware Revision Level
8D	Execute Onboard Diagnostics
8E	Prepare Sense Data
8F	Prepare Inquiry Data
90	Open Outbound Data Buffer (Init-to-Init)
91	Set Host Bus On/Off Times

Table 6-4: ICB Command Opcodes

6.2.1. Open Inbound Data Buffer (80)

	Bit Position / Description							
Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	0	0	0	0
01	LUN Acceptance Vector							
02	Data Buffer Length in Bytes (MSB)							
03	Data Buffer Length in Bytes							
04	Data Buffer Length in Bytes (LSB)							
05	Data Buffer Starting Address (MSB)							
06	Data Buffer Starting Address							
07	Data Buffer Starting Address (LSB)							
08	Open	Reserved				Initiator ID		
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-5: Open Inbound Data Buffer ICB Format

To operate as a TARGET the ASC must be given a place to put an incoming network message. In other words, the host must define a buffer starting address and buffer length for an anticipated message. This operation is performed by the "Open Inbound Data Buffer" command for receiving the data associated with the command initiated by another SCSI bus initiator.

After the host has determined that another SCSI initiator wishes to send a message of some sort, it issues this command to the ASC. When the ASC executes this command, it saves the buffer address and length for the initiator specified. This then enables the ASC to act as a target and accept an SCSI SEND command (6 byte CDB as defined by the ANSI standard for initiator to initiator mode of communication) to any of the host LUNs specified in the LUN acceptance vector. The LUN acceptance vector defines the target LUNs (i.e. the logical units associated with the AT bus) which will accept ASC write data from an initiator. Each bit position corresponds to a LUN. To enable responses on LUNs 7 and 0, the acceptance vector would be set to 81H. To enable all LUNs set this vector to 0FFH. When the specified initiator selects the ASC, the ASC will automatically fetch the SCSI CDB. If that command is a SCSI SEND command, the ASC will transfer the data from the initiator into the host buffer specified. Then, the ASC will send status, complete the command, and interrupt the host system. When the CDB received is not SEND or an unexpected CDB, or when an error occurs, the command bytes are written to the CDB buffer for inspection by the host (see Open SCSI CDB Buffer command, opcode = 87H).

Up to 8 such buffers can be opened, one for each SCSI bus ID, with the local host ID slot serving only as a place holder since it cannot communicate with itself. In order to open a data buffer, the host must set bit 7 of byte 8, with the default being closed (or zero). The host may close this buffer any time prior to the automatic execution of the SCSI SEND command. Once the SCSI SEND command has been received and executed, the appropriate data buffer will be closed in order to prevent rewriting to the same data space in case of linked commands or another SEND command by the same initiator. The data format of this message block must be pre-arranged between the 2 initiators if a meaningful dialogue to execute SCSI commands or some mutual specific action is the desired end goal. To ensure smooth handling by the ASC, the host must re-open a new data buffer (or the same as the previous, if desired) by re-issuing the open inbound data buffer command prior to acknowledging the IRQ indicating the completion of the SCSI SEND command.

This command gives the host system an efficient way to prepare for receiving information from other processors on the SCSI bus. The ASC handles all details when the SCSI SEND command occurs, rather than using the phase specific target Host CCBs. The host may send responses to the initiators by selecting them as processor targets, and issuing SCSI SEND command CDBs by way of a SCB command format. For Initiators that cannot be selected targets by other initiators, the complementary SCSI RECEIVE command can be used by issuing an "Open Outbound Data Buffer" shown later (opcode 90H).

6.2.2. Receive Command Block (81)

Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	0	0	0	1
01	SCSI Device ID			0	0	L.U.N.		
02	Command Buffer Length (MSB)							
03	Command Buffer Length							
04	Command Buffer Length (LSB)							
05	Command Buffer Starting Address (MSB)							
06	Command Buffer Starting Address							
07	Command Buffer Starting Address (LSB)							
08	Reserved							
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-6: Receive Command ICB Format

The receive command specifies a command buffer to the ASC so that it can request and receive command information from another initiator. This information is stored in the system memory deignated by the host. A receive command can only be issued in response to the ASC being selected by another initiator, usually after an exception condition has been detected or an unanticipated command has been received. Following reconnection, the ASC will send the disconnect message and disconnect from the SCSI bus after receiving the specified number of command bytes. The number of the command bytes to be received are specified by command buffer length field with a range from 6 to 12 bytes. Values from 1 to 5 are also acceptable, except that these are not supported by the ANSI standard. As usual the ASC will signal completion of the command by loading an ICMB and interrupting. Depending upon the command received, the host can then determine the next course of action.

6.2.3. Receive Data (82)

	Bit Position / Description							
Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	0	0	1	0
01	SCSI Device ID			0	0	L.U.N.		
02	Data Buffer Length (MSB)							
03	Data Buffer Length							
04	Data Buffer Length (LSB)							
05	Data Buffer Starting Address (MSB)							
06	Data Buffer Starting Address							
07	Data Buffer Starting Address (LSB)							
08	Reserved							
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-7: Receive Data ICB Format

Reveive data commands the ASC to reconnect to an initiator, request and receive data, and disconnect from the bus.

Specifically, the ASC arbitrates for the bus, reselects the initiator, sends the Identify message, requests and receives the specified number of data bytes, and stores them into the designated data buffer in the host memory. It then sends the Save Data Pointer message (so that additional data packets may be transferred) followed by the Disconnect message and then disconnects from the SCSI bus.

6.2.4. Receive Data With Status (83)

Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	0	0	1	1
01	SCSI Device ID			0	0	L.U.N.		
02	Data Buffer Length (MSB)							
03	Data Buffer Length							
04	Data Buffer Length (LSB)							
05	Data Buffer Starting Address (MSB)							
06	Data Buffer Starting Address							
07	Data Buffer Starting Address (LSB)							
08	Reserved							
09	Reserved							
thru	Reserved							
14	SCSI Status Byte to be sent							
15	Vendor Unique Error Code							

Table 6-8: Receive Data With Status ICB Format

Reveive data with status commands the ASC to reconnect to an initiator, request and receive data, send status and then disconnect from the bus to terminate the I/O operation.

Specifically, the ASC arbitrates for the bus, reselects the initiator, sends the Identify message, requests and receives the specified number of data bytes, and stores them into the designated data buffer in the host memory. If a recoverable error occurs, the ASC sends the Save Data Pointer message followed by a single completion status byte with data as specified by the host (usually 00H). It then sends the Command Completion message and disconnects from the SCSI bus.

If a nonrecoverable error occurs, the ASC does not send the Save Data Pointer message and the completion status byte. Instead, it disconnects from the bus without sending a Disconnect or Command Complete message. In either case, the ASC will load an ICMB and interrupt the host after disconnecting from the SCSI bus.

6.2.5. Send Data (84)

Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	0	1	0	0
01	SCSI Device ID			0	0	L.U.N.		
02	Data Buffer Length (MSB)							
03	Data Buffer Length							
04	Data Buffer Length (LSB)							
05	Data Buffer Starting Address (MSB)							
06	Data Buffer Starting Address							
07	Data Buffer Starting Address (LSB)							
08	Reserved							
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-9: Send Data ICB Format

Send data commands the ASC to reconnect to an initiator, request and send data, and disconnect from the bus.

Specifically, the ASC arbitrates for the bus, reselects the initiator, sends the Identify message, requests and sends the specified number of data bytes, from the designated data buffer in the host memory. It then sends the Save Data Pointer message followed by the Disconnect message and then disconnects from the SCSI bus.

6.2.6. Send Data With Status (85)

Byte	Bit Position / Description							
	7	6	5	4	3	2	1	0
00	1	0	0	0	0	1	0	1
01	SCSI Target ID			0	0	L.U.N.		
02	Data Buffer Length (MSB)							
03	Data Buffer Length							
04	Data Buffer Length (LSB)							
05	Data Buffer Starting Address (MSB)							
06	Data Buffer Starting Address							
07	Data Buffer Starting Address (LSB)							
08	Reserved							
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-10: Send Data With Status ICB Format

Send data with status commands the ASC to reconnect to an initiator, request and send data, send status and then disconnect from the bus to terminate the I/O operation.

Specifically, the ASC arbitrates for the bus, reselects the initiator, sends the Identify message, requests and sends the specified number of data bytes from the designated data buffer in the host memory. If a recoverable error occurs, the ASC sends the Save Data Pointer message followed by a single completion status byte with data equal to 00H. It then sends the Command Completion message and disconnects from the SCSI bus.

If a nonrecoverable error occurs, the ASC does not send the Save Data Pointer message and the completion status byte. Instead, it disconnects from the bus without sending a Disconnect or Command Complete message. In either case, the ASC will load an ICMB and interrupt the host after disconnecting from the SCSI bus.

6.2.7. Send Status (86)

Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	0	1	1	0
01	SCSI Device ID			0	0	L.U.N.		
02	Reserved							
03	Reserved							
04	Reserved							
05	Reserved							
06	Reserved							
07	Reserved							
08	Reserved							
09	Reserved							
thru	Reserved							
14	Completion Status Byte							
15	Vendor Unique Error Code							

Table 6-11: Send Status ICB Format

Send status commands the ASC to reconnect to an initiator, request and send status, and disconnect from the bus.

Specifically, the ASC arbitrates for the bus, reselects the initiator, sends the Identify message, requests and sends the specified completion status byte(s), from the ICB in the host memory. It then sends the Command Complete message and disconnects from the SCSI bus.

6.2.8. Open Scratchpad CDB (87)

Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	0	1	0	1
01	SCSI Non-standard CDB length (Group 3)							
02	SCSI Non-standard CDB length (Group 4)							
03	SCSI Non-standard CDB length (Group 5)							
04	SCSI Non-standard CDB length (Group 7)							
05	CDB Buffer Starting Address (MSB)							
06	CDB Buffer Starting Address							
07	CDB Buffer Starting Address (LSB)							
08	Reserved							
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-12: Open Scrathpad CDB Format

The Open Scratchpad CDB Buffer command indicates to the ASC where to put SCSI CDBs received when acting as a target. The buffer size is fixed at 32 bytes, with the same identical format as the SCB defined earlier. Only one buffer can be opened. This buffer is used whenever the ASC has no instructions on how to handle a particular CDB; specifically:

- The CDB is not a SEND, REQUEST SENSE, or INQUIRY command.
- The CDB is a SEND command, but no receive buffer is open.
- The CDB is a REQUEST SENSE command, but no sense data has been prepare
- The CDB is an INQUIRY command, but no inquiry data has been prepared.
- The CDB is one of the above three commands, but an exception condition was detected.

The CDB is transferred to host system memory at the buffer location specified. The four SCSI Non-Standard CDB Length fields specify the command lengths for the SCSI CDB group codes not defined by the SCSI standard. The acceptable values for these fields are from six to twelve. These fields are necessary for the ASC to receive the correct number of command phase bytes.

The CDB Buffer is considered available to the ASC when the interrupt informing the host of the selection and CDB transfer is acknowledged by the host. If another selection and CDB transfer occurs before the acknowledge, BUSY status will be reported to the initiator.

6.2.9. Read Initialization Bytes (88)

	Bit Position / Description							
Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	1	0	0	0
01	Reserved							
02	Data Buffer Length (MSB) (set to zero)							
03	Data Buffer Length (set to zero)							
04	Data Buffer Length (LSB) (max. 9)							
05	Data Buffer Starting Address (MSB)							
06	Data Buffer Starting Address							
07	Data Buffer Starting Address (LSB)							
08	Reserved							
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-13: Read Initialization Bytes ICB Format

This command reads the initialization bytes written by the host into the system memory area designated by the host. Bytes 2 through 4 are written depending upon the data field length choosen. A value of zero will result in no bytes being written by the ASC and a value greater than 9 will be reported as an error. The format and order of the bytes is the same as the Initialization Command. Since the initialization command opcode is not stored internally, it is not reported back to the host.

6.2.10. Read ASC Device Address (89)

This commands the ASC to write it's own SCSI ID into byte 01, bits 7-5, of the ICB command block.

	Bit Position / Description							
Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	1	0	0	1
01	SCSI ASC ID			0	0	0	0	0
02	Reserved							
03	Reserved							
04	Reserved							
05	Reserved							
06	Reserved							
07	Reserved							
08	Reserved							
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-14: Read ASC Device Address ICB Format

6.2.11. Set Execution Parameters (8A)

Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	1	0	1	0
01	Reserved							
02	Data Transfer Length (MSB)							
03	Data Transfer Length							
04	Data Transfer Length (LSB)							
05	Data Buffer Starting Address (MSB)							
06	Data Buffer Starting Address							
07	Data Buffer Starting Address (LSB)							
08	Index relative to parameter within DB (MSB)							
09	Index (LSB)							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-15: Set/Read Execution Parameters ICB Format

The set execution parameters is generally issued following an ASC initialization command to change some or all of the default parameters assumed. These include synchronous mode, SCSI timeout, bus parity, retries, etc. To issue this command, the host must setup the CDB as defined above, along with a data block with parameters to be changed. The format of this data block is defined below and summarized in appendix A as well.

Since the index value is added to the base of the internal tables to access the parameter desired, it enables the host to change 1 or more contiguous parameters with minimum requirements on the use of the host memory. No large chunks of memory need be allocated for this purpose - only up to the number of bytes required to change or read these parameter values, as specified by the data transfer length (1 to a max = 25 or 19 hex) and the index value.

6.2.11.1. Synchronous Rates (0-F)

The host can specify the desired SBIC synchronous rates for 8 possible targets including the 7000-ASC, using the even bytes. Byte 0 is the rate for target 0, byte 2 for target 1, and so on. The upper nibble of this byte defines the transfer period and the lower nibble defines the buffer offset. The range & format of this byte is described in the SBIC data sheets (reg address 11 hex).

The odd bytes 1-F, are reserved to report the actual negotiated or agreed transfer rate between the initiator and the target. These values are NOT to be modified by the host. They are merely reported back to the host for information purposes. The SCSI device IDs are implied by position as defined above.

To specify the synchronous rate for SCSI device #2 only, the host must setup the CDB bytes as shown below:

```
Data buffer length = 1
Index value        = 4
Byte 0 of data block = sync rate desired = 45H (recommended)
```

To include the rates for SCSI devices 3, 4 as well set these values:

```
Data buffer length = 5 or 6
Index value        = 4
Byte 0,2,4 of data block = sync rates desired = 63H
```

To specify all of the rates:

```
Data buffer length = 15 or 16 decimal.
Index value        = 0
Byte 0,2,4,6,8,10,12,14 of data block = sync rates desired = 45H
```

If these sync rates are not specified by the host, default async values of 40 (hex) will be used for all SCSI devices. The synchronous transfer rates are only established during the selection phase of a particular target. This is indicated by the MSB bit of an even byte being set for a given target. If no communication occurs between the ASC and the target, this bit will not be set and the corresponding odd byte rate is meaningless. The host may reset bit 7 of any even byte (between 0-F) to force the ASC to re-negotiate the synchronous transfer rate, preferably when there are no other SCSI commands being executed by the ASC.

6.2.11.2. SBIC ID Register (10H)

This byte is reserved for future enhancements of the SBIC device.

6.2.11.3. SBIC Control Register (11H)

This byte determines the mode of operation of the SBIC device. From the host's point of view, only bit 0 may be set to enable SCSI bus parity. All other bits must not be altered, else the ASC may malfunction.

6.2.11.4. SCSI Timeout (12H)

The contents of this byte control the timeout period required by the selection and reselection phases on the SCSI bus. This value may be calculated as a function of the input clock frequency and the desired timeout:

$$\text{byte value} = \frac{\text{timeout period (ms)} * \text{SBIC clock frequency (MHZ)}}{\text{80 (scale factor)}}$$

The user should round the resulting "register byte value" up to the next integral value to ensure that the user's minimum timeout requirement is met. The default value is 25 (19H) for a timeout period of 250 ms.

6.2.11.5. Target LUN Register (13H)

This byte is reserved for future enhancements of the SBIC device.

6.2.11.6. Destination ID Register (14H)

This byte is reserved for future enhancements of the SBIC device.

6.2.11.7. Source ID Register (15H)

When bit 7 of this register is set, disconnects of targets is permitted by the ASC. When reset, no target disconnect is permitted. Similarly, when bit 6 is set, selection by another SCSI initiator is allowed.

The default value for this byte is 80H. All other bits are reserved for future enhancements.

6.2.11.8. User Flags (16H)

The flags specified by this byte allow the user to set various operating modes as defined below: Default value = 00

Bit 2 = 1 : Return number of bytes not transferred or the last transfer count value of the SBIC data register. This value is returned in bytes 16-18 of the SCB. The host must maintain the previous count value and subtract the value returned to determine exactly the number of bytes transferred. This is mainly useful for diagnostic purposes.

Bit 1 = 1 : Interrupt on free OGMB; also available as a command via the command port. When all of the OGMBs have been used, the host may issue this command or the alternate 1 byte command, to determine the availabilty of an OGMB. This relieves the host from having to poll byte 0 of the OGMB all the time. When the ASC has picked up the mail from an OGMB, the host will be interrupted. The low 6 bits of the host interrupt status byte will contain the number of the OGMB available. To repeat, the host must re-issue the command.

Bit 0 = 1 : Enable Unsolicited Interrupts as determined by the interrupt mask specified by the next byte. This master enable/disable of the unsolicited interrupts can also be achieved with a single byte using the host command port. This bit must be set to enable the ASC to operate as a target.

6.2.11.9. Set Unsolicited Interrupt Mask (17H)

This command allows the host to choose the error conditions which will generate an unsolicited interrupt. If any bit, of byte 1, is set, then the error condition corresponding to that bit will generate an interrupt. Byte 0 of the ICMB will contain 80-87, corresponding to the interrupt conditions defined below. This helps to differentiate an unsolicited interrupt from a SCSI command issued by the host.

On power-up, this byte is reset so that any error condition listed will NOT cause an unsolicited interrupt until the interrupt has been set by the host, along with the master enable flag defined above.

Bit	Unsolicited Interrupt	Bit	Unsolicited Interrupt
0	Unexpected Reselection	4	Bus Reset with no cmds
1	Unexpected Selection	5	Unexpected Request Sense
2	Abort message received	6	Reserved
3	Bus Device Reset msg received	7	Reserved

<u>Unexpected Reselection</u>

When an unexpected reselection occurs, i.e. no SCSI commands are open for that target and logical unit, an immediate abort message will be sent, following the reselection handshake and the Identify message transfer. If this bit, then the host system will be interrupted with the modified ICMB format since a valid host SCB does not exist. If this bit is reset, no host interrupt is generated. Note that regardless of the state of this bit, the an unexpected reselection is always aborted.

Unexpected Selection

When enabled, a host interrupt occurs whenever the ASC is selected by an initiator and no host commands are open for that initiator and logical unit. The ASC will respond to the selection and transfer the identify message. If the scratchpad CDB buffer has been opened, the ASC will transfer the SCSI CDB to the host so that the host can react appropriately. If allowed to by the initiator, the ASC will disconnect. When disabled, any unexpected selection will generate a SCSI BUSY status, or just not respond to the selection until enabled by the host.

Interrupt on an Abort Message Received

When a selection is in progress, an abort message will always be reported to the host and the current operation terminated immediately. If no current target mode operation is in progress, then a host interrupt will be generated only if this bit is set. If reset, it is treated as a nuisance and no host interrupt is generated.

Interrupt on a Reset Message Received

Same action as the above except that the host is reponsible for terminating any pending operations for the remaining logical units for that initiator as described before for the ASC initiator mode of operation. This is done so that the host can customize the effect of a reset message for it's particular environment.

Interrupt on a SCSI Hardware Reset

If any ASC operations are outstanding, a host interrupt will be generated regardless of the state of this bit. The host is then expected to terminate all or some of the commands outstanding for that initiator and logical unit, followed by a reset acknowledge. The vendor unique error code returned is 05H. If no SCSI commands are outstanding, then the vendor unique error code returned is 84H if this bit is set. If reset, it is treated as a nuisance and no host interrupt is generated.

Interrupt on an Unexpected Request Sense Command

When enabled, a host interrupt will occur whenever the ASC receives a Request Sense command from an initiator for which no Prepare Sense Data command has been issued. The SCSI CDB will be written to the buffer defined by the Open Scratchpad CDB command. When disabled, the ASC will respond with a 'no sense' data block to the initiator, and as usual, no host interrupt will be generated.

6.2.11.10. Parity Retries (18H)

The default number of retries is set to 2. This field may be changed by the user, but it was mainly implemented for in house testing purposes. By changing this value, all retryable operations will be affected.

6.2.12. Read Execution Parameters (8B)

This command allows the host to read the mask established by the previous Set Unsolicited Interrupt Mask command or the default parameter bytes assumed after the power-up condition. It is recommended that the host read the parameter data block using the same format as the set execution parameter command, except for the command opcode, and then change the desired values to ensure other default or previous settings are not unintentionly affected.

6.2.13. Read Firmware Revision Level (8C)

This command allows the host to read the ASC firmware revision level. Byte 1 provides the primary revision level and byte 2 represents the secondary revision level.

	Bit Position / Description							
Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	1	1	0	0
01	Primary Revision Level (written by ASC)							
02	Secondary Revision Level (written by ASC)							
03	Reserved							
04	Reserved							
05	Reserved							
06	Reserved							
07	Reserved							
08	Reserved							
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-16: Read ASC Firmware Revision ICB Format

6.2.14. Execute Diagnostics (8D)

Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	1	1	0	1
01	Diagnostic Type Code							
02	Data Buffer Length (MSB)							
03	Data Buffer Length							
04	Data Buffer Length (LSB)							
05	Data Buffer Starting Address (MSB)							
06	Data Buffer Starting Address							
07	Data Buffer Starting Address (LSB)							
08	ASC RAM Starting Address (MSB)							
09	ASC RAM Starting Address (LSB)							
10	Initial Test Pattern							
11	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

LEGEND : Diagnostic Type Codes (for user)

00	= Execute Normal Short Diagnostics only
01	= Execute Walking 1's Pattern Diagnostics
02	= Execute DMA System Memory Diags with fixed pattern
06	= Execute DMA System Memory Diags with incr pattern

Diagnostic Type Codes (for in house testing)

90	= Execute Code from RAM
A0	= Write ASC RAM Buffer
C0	= Read ASC RAB Buffer

Table 6-17: Execute ASC Diagnostics ICB Format

This command allows the host to execute ASC diagnostics whenever desired. The power up ASC disgnostics does not exercise the first party DMA controller since the system memory will be destroyed. However, this command allows the host to designate a specific memory area for such a purpose, if desired.

Diagnostic codes designated for in house testing should not be used by the user as this could possibly trash internal RAM without the source listing in hand. In addition a certain sequence of keywords is required to enable this feature of the diagnostic.

To invoke the diagnostics, the host may assert the ASC reset line or issue an ASC command using the ICB format described above.

A power-up reset, determined by reading a 2 byte key, will always invoke the long walking 1's pattern diagnostic whereas a warm reset will invoke the short version of the diagnostics. No interrupt is generated since a command was not issued by the host. Instead the host must poll the status register until the ready status (40) is asserted. At this time the host may read the interrupt status register to detect any ASC diagnostic failures, if desired. This interrupt status register will remain unchanged until the host has issued it's first command to the ASC. The ASC then expects the host to furnish the 10 initialization bytes. If any byte is rejected, a status of 60 will be posted. The host may resubmit the corrected byte or begin again by asserting reset. After the successful completion of this 10 byte sequence, using the ready status, as a programmed I/O byte handshake, the host must wait for a status of 50 before any host commands may be issued. It is assumed that the host has already enabled the ASC DMA and IRQ channels appropriately.

If an ICB command is issued, usually when there are no other commands being processed by the ASC, a host interrupt will be generated with the diagnostic error code, if any, posted in byte 15 of the ICB. The host does not have to re-initialize the ASC as mentioned above. If there are commands pending, then these will be lost.

6.2.14.1. Normal Short Diagnostics (00)

The main criterion for this type of diagnostics is that the completion time should be less than 250 ms. According only fixed test patterns are used to verify the functionality of the onboard logic.

The test executed are ROM checksum, RAM stack test, SBIC register tests, FIFO and RAM tests, followed by the initialization of all internal variables. If a particular test fails, the rest of the diagnostics are skipped and initialization is attempted even though a particular test may have failed.

The ROM checksum simply consists of a 2 byte sum of the entire ROM locations excluding the checksum bytes which are stored in the last 2 locations of the ROM.

The RAM stack test does a quick check of the stack pointer before using any sub-routine calls.

The SBIC register test pattern consists of a 2 pass checkerboard byte which is written and then read back for verification. The reset condition of the device is also tested.

The FIFO test patterns used are identical to the ones used for the SBIC tests except that the FIFO register length is 16 instead of 22.

The portion of the RAM that is actually used by the ASC is tested by writing all FF's followed by all 00's so that the RAM is in a known state at the conclusion of the diagnostics.

If any test failure is detected, the status is posted in the interrupt status register, and the green LED stays lit. If no failures were detected, the LED is turned off with the interrupts status register set to 01. For all other error codes, see the appendix.

6.2.14.2. Walking 1's Diagnostics (01)

These tests are somewhat more extensive since a 1 bit is written and read in every position of a given register. The devices tested include the SBIC, FIFO and the entire onboard RAM. The execution format is the same as the one used for the short diagnostics described above. The entire exection time for these tests is around 2 seconds.

Whenever a cold reset or power-up condition is detected, 2 additional tests are conducted just prior to the SBIC, FIFO and RAM tests. These include the IRQ channel and the ASC initialized D-FFs. It is for this reason that the host is required not to interpret the ASC status until ready has been asserted. These 2 tests are conducted during a power-up sequence only and cannot be invoked by the user using a command.

6.2.14.3. DMA Diagnostics (02/06)

When the host wishes to exercise the DMA address counters, the host must allocate an EXPENDABLE portion of the system memory using bytes 2-7 and byte 10, which specifies the test pattern to be used. The data buffer length specified must be no larger than 64K.

If the diagnostic type code specified is an 02, then a fixed pattern of data is first written and then read back using bytes 2-4 as the transfer length and bytes 5-7 as the staring address of the host system memory. A vendor unique error code is posted in byte 15 in the usual fashion.

If in addition bit 2 of the diagnostic type code is set, then an incremental test pattern is used with byte 10 as the first data byte to be written and read back for verification. In all other respects, this test is identical to the fixed pattern test described above.

6.2.14.4. Vendor Unique Diagnostics (90, A0, C0)

In order to execute these vendor unique diagnostics, the user must execute a sequence of keywords in the exact order specified in the source listing. Additionally when writing to the RAM, the user must ensure that only unused portions of the RAM are used or that only specific areas of RAM are being altered by design. Once this diagnostic mode has been unlocked, the user may read, write, or execute code from RAM any number of times as indicated by the diagnostic type code used. Since no parameter checks are made when using the internal RAM, the user is cautioned to be very careful, or else no meaningful communications with the ASC will result. To recover, simply reset the ASC and start over again.

6.2.15. Prepare SCSI Sense Data (8E)

Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	1	1	1	0
01	Reserved					Target LUN		
02	Data Buffer Length in Bytes (MSB)							
03	Data Buffer Length in Bytes							
04	Data Buffer Length in Bytes (LSB)							
05	Data Buffer Starting Address (MSB)							
06	Data Buffer Starting Address							
07	Data Buffer Starting Address (LSB)							
08	Open	Reserved				Initiator ID		
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-18: Prepare Sense Data ICB Format

The Prepare Sense Data command is similar to the Open Data Buffer command, in that it anticipates an action by an initiator. The buffer is set up by the host system to contain SCSI sense data for the initiator specified. If the next command from the initiator (for the target LUN) is a SCSI REQUEST SENSE command, the data in the specified buffer area will be transferred to the initiator. If the next command was not a REQUEST SENSE command, this command will be canceled (i.e. the appropriate buffer will be closed) and the host system is interrupted.

If the ASC receives a REQUEST SENSE command from an initiator, and this command is not pending for that initiator, the ASC will respond as defined for bit 5 in the Unsolicited Interrupt Mask. Briefly, if this bit is set, the host system will be interrupted with the initiator CDB in the scratchpad area. The next course of action is up to the host. If reset, no host interrupt will be generated and the ASC will send a 'no sense' data block to the initiator without any host intervention.

This command may be issued while a command is pending for the initiator ID and target LUN specified; e.g., prior to issuing a Transmit Status for a CHECK CONDITION status to the initiator. The host may issue up to 8 such commands, one for each of the initiator ID's specified.

6.2.16. Prepare Inquiry Data (8F)

	Bit Position / Description							
Byte	7	6	5	4	3	2	1	0
00	1	0	0	0	1	1	1	1
01	LUN Acceptance Vector							
02	Data Buffer Length in Bytes (MSB)							
03	Data Buffer Length in Bytes							
04	Data Buffer Length in Bytes (LSB)							
05	Data Buffer Starting Address (MSB)							
06	Data Buffer Starting Address							
07	Data Buffer Starting Address (LSB)							
08	Initiator ID Acceptance Vector							
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-19: Prepare Inquiry Data ICB Format

The Prepare Inquiry Data command is similar to the Prepare Sense Data command, except that the buffer specifies the data that is to be returned when the ASC receives a SCSI Inquiry command. The LUN acceptance vector defines which local LUNs the data applies to, and the initiator acceptance vector defines which initiators the data applies to. Hence all combinations of the LUNs and IDs, as specified by the initiator, will receive the same inquiry data from the ASC. This command stays active unless aborted or superseded by another

Prepare Inquiry Data command.

If the ASC receives an Inquiry command from an initiator, and this command is not pending for that initiator and/or for the target LUN address by the initiator, the ASC responds as defined in the Unsolicited Interrupt Mask.

The host system may issue up to eight of these commands to specify data for all possible LUNs and IDs. A typical case would be to issue one Prepare Inquiry Data command for all valid LUN responses for all initiators, and issue a second Prepare Inquiry Data command for all invalid or not present LUN responses for all initiators. Six additional such combinations can be programmed by the host. To cancel all of the Prepare Inquiry Buffer commands, both the acceptance vectors for the LUNs and IDs (LAV/IAV), may be set to zero. If only one of the 2 vectors is set to zero, then the buffer specified by the previous eigth such command will be preserved. If at least 1 combination of the LAV/IAV is non zero, then the present pointers will supercede any previous buffer pointers allocated.

This command may be issued prior to enabling target mode operation via the Enable/Disable Unsolicited Interrupt mask commands (or the master enable/disable target mode ASC operation), and/or the Set Execution Parameters command.

6.2.17. Open Outbound Data Buffer (90)

	Bit Position / Description							
Byte	7	6	5	4	3	2	1	0
00	1	0	0	1	0	0	0	0
01	LUN Acceptance Vector							
02	Data Buffer Length in Bytes (MSB)							
03	Data Buffer Length in Bytes							
04	Data Buffer Length in Bytes (LSB)							
05	Data Buffer Starting Address (MSB)							
06	Data Buffer Starting Address							
07	Data Buffer Starting Address (LSB)							
08	Open	Reserved				Initiator ID		
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-20: Open Outbound Data Buffer ICB Format

For initiator-to-initiator mode of communication, with all scsi phases handled by the ASC with a single host interrupt, the ASC must be given the source of data for an outgoing network message. In other words, the host must define a buffer starting address and buffer length for the anticipated data transfer. This operation is performed by the "Open Outbound Data Buffer" command for sending data to an initiator, that cannot be selected as a target later.

After the host has determined that another SCSI initiator wishes to receive a data packet of some sort, it issues this command to the ASC. When the ASC executes this command, it saves the buffer address and length for the initiator specified. This then enables the ASC to act as a target and accept an SCSI RECEIVE command (6 byte CDB as defined by the ANSI standard for initiator to initiator mode of communication) to any of the host LUNs specified in the LUN acceptance vector. The LUN acceptance vector definition is identical to the open inbound data buffer defined earlier. When the specified initiator selects the ASC, the ASC will automatically fetch the SCSI CDB. If that command is a SCSI RECEIVE command, the ASC will transfer the host data buffer specified to the initiator. Then, the ASC will send status, complete the command, and interrupt the host system. When the CDB received is not RECEIVE or an unexpected CDB, or when an error occurs, the command bytes are written to the CDB buffer for inspection by the host (see Open SCSI CDB Buffer command, opcode = 87H).

Up to 8 such buffers can be opened, as explained in the complementary open inbound data buffer command. Once again, the data format of this message block must be pre-arranged to include all required information to execute some mutual specific operation between the 2 initiators. To ensure smooth handling by the ASC, the host must re-open a new data buffer (or the same as the previous, if desired) by re-issuing the open outbound data buffer command prior to acknowledging the IRQ indicating the completion of the SCSI RECEIVE command.

6.2.18. Set DMA Bus On/Off Times (91)

Byte	7	6	5	4	3	2	1	0
00	1	0	0	1	0	0	0	1
01	DMA Bus On Time							
02	DMA Bus Off Time							
03	Reserved							
04	Reserved							
05	Reserved							
06	Reserved							
07	Reserved							
08	Reserved							
09	Reserved							
thru	Reserved							
14	Reserved							
15	Vendor Unique Error Code							

Table 6-21: Set DMA Bus On/Off Times ICB Format

In order to customize the host bus bandwidth for a given application, the host may override the DMA bus on/off times without having to re-initialize the ASC by issuing this command. When this command is received and executed, the programmed bus on/off times provided with the initialization command will be over written. The new values specified will be used starting with the write of the ICMB for this command.

This command may be used as often as desired, but in order to avoid confusion, it is recommended that it only be used when there are no outstanding ASC commands. The confusion, if any, will arise in that a partial data packet will be transferred with the old set of values (due to SCSI target disconnects) and the rest with the new set of values specified when this command is executed in the order issued. Data integrity is maintained with no other side effects.

7. INTERFACE CONNECTORS

The ASC controller has four onboard connectors. These connectors consist of 2 host interface edge connectors, a 50 pin SCSI interface connector, and a 34 pin connector for the floppy interface. Power for the ASC is supplied by the host interface.

The host interface connectors provide interface signals that are compatible with the IBM PC/AT on the mother board. For a complete description of the interface, refer to the AT Technical Reference Manual.

The SCSI interface is completely defined by the ANSI standard X3T9.2 for SCSI.

The configuration jumper options are defined in the final sub-section.

7.1. Host Interface Connectors (P1, P2)

The host interface is through the 62 pin board edge connector (P1) and a 36 pin board edge connector (P2), where the computer I/O bus and power is available to the board. Connector pins A1 through A31, and C1 through C18, are located on the component side of the board. Whereas pins B1 through B31, and D1 through D18 are on the conductor side. The ASC does not load the computer I/O bus with more than 2 LSTTL DC loads per line. The edge connectors pin-outs are shown on the next 2 pages.

7.1.1. Data Bus (D7 thru D0)

Positive true 8-bit, tri-state, bi-directional, data bus for data, commands, and status communication between the ASC and host. D7-D0 form the least significant byte of a 16-bit data transfer.

7.1.2. Address Enable (AEN)

Address ENable is used to degate the host CPU and other devices from the I/O channel to allow DMA transfers to take place. When this line is active, the ASC DMA controller has control of the host address bus, memory control, and data busses. When AEN is active, I/O port addresses are no longer generated for I/O port access. The BIOS ROM can still be addressed via A19-A0.

Conversely, if AEN is inactive (or low), and a valid port address (XX0-XX3) is present on the address bus, then the ASC is in programmed I/O mode.

7.1.3. Address Bus (A19-A0)

Positive true 20-bit address. The least significant 10 bits are decoded to obtain the address, ports XX0 thru XX3, for processor I/O instruction execution. They are inhibited during DMA by AEN. The full 20 address bits are decoded to address the on-board ROM, regardless of the state of AEN.

I/O Pin	I/O	Signal Name
A1	NC	IOCHCK-
A2	I/O	D7
A3	I/O	D6
A4	I/O	D5
A5	I/O	D4
A6	I/O	D3
A7	I/O	D2
A8	I/O	D1
A9	I/O	D0
A10	NC	IOCHRDY-
A11	I	AEN
A12	I/O	A19
A13	I/O	A18
A14	I/O	A17
A15	I/O	A16
A16	I/O	A15
A17	I/O	A14
A18	I/O	A13
A19	I/O	A12
A20	I/O	A11
A21	I/O	A10
A22	I/O	A9
A23	I/O	A8
A24	I/O	A7
A25	I/O	A6
A26	I/O	A5
A27	I/O	A4
A28	I/O	A3
A29	I/O	A2
A30	I/O	A1
A31	I/O	A0

I/O Pin	I/O	Signal Name
B1	GND	GND
B2	I	HMR
B3	+5	+5VDC
B4	O	IRQ9
B5	-5	-5VDC
B6	O	DRQ2
B7	-12	-12VDC
B8	O	OWS
B9	+12	+12VDC
B10	GND	GND
B11	NC	SMEMW-
B12	NC	SMEMR-
B13	I/O	IOW-
B14	I/O	IOR-
B15	I	DACK3-
B16	O	DRQ3
B17	I	DACK1-
B18	O	DRQ1
B19	I/O	REFRESH-
B20	I	CLK
B21	O	IRQ7
B22	O	IRQ6
B23	O	IRQ5
B24	O	IRQ4
B25	O	IRQ3
B26	I	DACK2-
B27	NC	TC
B28	NC	BALE
B29	+5	+5VDC
B30	NC	OSC
B31	GND	GND

Table 7-1: Host Interface Connector P1

I/O Pin	I/O	Signal Name
C1	I/O	SBHE
C2	I/O	LA23
C3	I/O	LA22
C4	I/O	LA21
C5	I/O	LA20
C6	I/O	LA19
C7	I/O	LA18
C8	I/O	LA17
C9	I/O	MEMR-
C10	I/O	MEMW-
C11	I/O	D8
C12	I/O	D9
C13	I/O	D10
C14	I/O	D11
C15	I/O	D12
C16	I/O	D13
C17	I/O	D14
C18	I/O	D15

I/O Pin	I/O	Signal Name
D1	O	MEMCS16-
D2	O	IOCS16-
D3	O	IRQ10
D4	O	IRQ11
D5	O	IRQ12
D6	O	IRQ15
D7	O	IRQ14
D8	I	DACK0-
D9	O	DRQ0
D10	I	DACK5-
D11	O	DRQ5
D12	I	DACK6-
D13	O	DRQ6
D14	I	DACK7-
D15	O	DRQ7
D16	+5	+5VDC
D17	O	MASTER-
D18	GND	GND

Table 7-2: Host Interface Connector P2

7.1.4. Buffered Address Latch Enable (BALE)

A19-A0 are latched on the falling edge of BALE to indicate a valid address by the host. During DMA cycles the host forces BALE- high. BALE is not used by the ASC controller.

7.1.5. Memory Read (MEMR-)

Memory read signal, active on all memory read cycles instructs the memory devices to drive data onto the data bus. MEMR- may be driven by any microprocessor or DMA controller in the system. When the ASC drives the MEMR- signal, the address lines are valid at least one system clock period before driving MEMR- active.

7.1.6. Memory Write (MEMW-)

Memory write signal, active on all memory write cycles instructs the memory devices to store data from the data bus. MEMR- may be driven by any microprocessor or DMA controller in the system. When the ASC drives the MEMW- signal, the address lines are valid at least one system clock period before driving MEMW- active.

7.1.7. I/O Read (IOR-)

I/O Read signal, active when the host (uP or DMA controller) or the ASC (uP or DMA controller) resident on the I/O channel, instructs an I/O device to drive data onto the data bus.

7.1.8. I/O Write (IOW-)

I/O Write signal, active when the host (uP or DMA controller) or the ASC (uP or DMA controller) resident on the I/O channel, instructs an I/O device to read data from the data bus.

7.1.9. System Memory access (SMEMR-, SMEMW-)

These system memory read and write strobes are used by the host to address any memory that is mapped within the low 1Mb of the memory space. These strobes are not used by the ASC controller.

7.1.10. Reset (RST)

When asserted, RST places the ASC in its initial power-up condition.

7.1.11. I/O Channel Ready (IOCHRDY-)

This input signal to the AT bus is pulled low by a memory or an I/O device to lengthen I/O or memory cycles by an integral number of clock cycles (167 nS). Not used by the ASC controller.

7.1.12. I/O Channel Check (IOCHCHK-)

This signal to the host AT bus provides the system board with parity (error) information about memory or devices on the I/O channel. When this signal is active, it indicates an uncorrectable system error. This output line to the host is not used by the ASC controller.

7.1.13. Zero Wait State (0WS)

The zero wait signal tells the host processor that it can complete the present bus cycle without inserting any additional wait cycles. The address space of any onboard memory accessible to the host along with the read and write strobes can be used to generate this signal, if desired. Not presently used by the ASC controller.

7.1.14. Refresh Dynamic Memory (REFRESH-)

The REFRESH- strobe, generated about every 15 uS by the host, is used by the ASC to surrender the AT bus, so that the host can refresh the system dynamic memory.

7.1.15. System Clock (CLK)

The 6/8 MHz synchronous system clock is not used by the ASC controller.

7.1.16. Oscillator Clock (OSC)

The oscillator clock, 70 nS (14.31818 MHz) period, is not used in this design.

7.1.17. Terminal Count (TC)

The terminal count pulse is provided by the host when the terminal count for any DMA channel is reached. Not used by the ASC controller.

7.1.18. +5VDC (B3, B29, D16)

+5 volt power supply.

7.1.19. -12VDC (B7)

-12 volt power supply.

7.1.20. +12VDC (B7)

+12 volt power supply.

7.1.21. Ground (B1, B10, B31, D18)

Ground return for all signals.

7.1.22. Bus Master (MASTER-)

This signal is used with a DRQ line to gain control of the host system bus. The ASC DMA controller issues a DRQ to a DMA channel in cascade mode and receives a DACK-. Upon receiving the DACK-, the ASC controller asserts MASTER-, which allows it to control the system address, data, and control lines for no more than 15 uS so that the host can refresh the system memory at regular intervals.

The ASC DMA controller waits at least one system clock period before driving the address and data lines, and 2 clock periods before issuing any memory read or write strobes.

7.1.23. DMA Data Request (DRQ0 to DRQ3, DRQ5 to DRQ7)

DMA Requests 0-3 and 5-7, are asynchronous channel requests used by peripheral devices to gain DMA sevice or control of the host bus. They are prioritized, with DRQ0 having the highest priority and DRQ7 having the lowest priority. DRQ0-DRQ3 are used for 8-bit DMA transfers and hence not used by the ASC controller. DRQ5-DRQ7 will perform 16-bit DMA transfers and are available as jumperable options on the ASC controller.

7.1.24. Data Acknowledge (DACK0- to DACK3-, DACK5- to DACK7-)

DMA ACKnowledge, asserted by the host in responce to a DMA request from the ASC. These are active low signals and are jumper selectable on the ASC controller.

7.1.25. Interrupt Request (IRQX)

Interrupt Requests 3-7, 9-12, and 14-15 are used to signal the host that an I/O device needs attention. A jumper configurable interrupt request is asserted by the ASC to interrupt the host upon completion of an operation.

The interrupt requests are prioritized, with IRQ9-IRQ15 having a higher priority than IRQ3-IRQ7. Between the 2 groups IRQ9 has the highest priority and IRQ7 the lowest priority. The line is held high until the host acknowledges the interrupt request.

7.1.26. Memory Chip Select 16-bit (MEMCS16-)

This signals the host system board that the present data transfer is a 1 wait-state, 16 bit, memory cycle, derived from the decode of LA17-LA23.

7.1.27. I/O Chip Select 16-bit (IOCS16-)

Similar to MEMCS16-, this signal indicates to the host system board that the present data transfer is a 1 wait-state, 16 bit, I/O cycle. It is derived from an address decode.

7.1.28. System Bus High Enable (SBHE)

The SBHE indicates that a data transfer on the upper byte (D15-D8) of the data bus. The present bus master uses this signal to condition the data bus buffers driving D15-D8 lines.

7.1.29. Data Bus (D15-D0)

Lines D15-D0 form the upper byte of the 16 bit data bus used by the static controller.

7.1.30. Unlatched Address Bus (LA23-LA17)

These unlatched signals can be used to address memory and I/O devices within the sytem, extending the address space to 16Mb. These lines are valid when BALE is high and are latched by the falling edge of BALE when input to the ASC controller. Their purpose is to generate memory decodes for 1 wait-state memory cycles.

7.2. SCSI Connector

A 50 pin vertical header consisting of 2 rows of 25 male pins (2.54 mm) apart forms the SCSI interface. The single ended cable length shall not exceed 6.0 meters as specified by the ANSI specification. The pinouts are shown below. All odd pins except pin 25 are connected to ground.

The ASC is configured as an initiator or a target with signal descriptions as defined by the WD33C93-SBIC data sheets.

I/O Pin	I/O	Signal Name
2	I/O	DB0-
4	I/O	DB1-
6	I/O	DB2-
8	I/O	DB3-
10	I/O	DB4-
12	I/O	DB5-
14	I/O	DB6-
16	I/O	DB7-
18	I/O	DBP-
20	GND	GND
22	GND	GND
24	GND	GND
26	NC	TERMPWR

I/O Pin	I/O	Signal Name
28	GND	GND
30	GND	GND
32	O	ATN-
34	GND	GND
36	I/O	BSY-
38	O	ACK-
40	I/O	RST-
42	I	MSG-
44	I/O	SEL-
46	I	C-/D
48	I	REQ-
50	I	I-/O

NOTE: All odd pins except pin 25 are connected to ground.

Table 7-3: SCSI Interface Connector

7.3. Floppy Connector

The Floppy port is controlled entirely by the host. The ASC hosts the floppy interface hardware but does not control it in any fashion. For the description of the floppy interface signals refer to the WD37C65 data sheets or the OEM manual of the floppy drive manufacturer. The pinout of the 34 pin connector is shown below:

Signal Ground	Signal Pin #	Dir I/O	Signal Mnemonic	Signal Description
1	2	O	RWC-	Reduce Write Current
3	4	NC	-	No Connection
5	6	NC	-	No Connection
7	8	I	IDX-	Index Pulse
9	10	O	MOA-	Motor Enable A
11	12	O	DSB-	Drive Select B
13	14	O	DSA-	Drive Select A
15	16	O	MOB-	Motor Enable B
17	18	O	DIRC-	Direction In
19	20	O	STEP-	Step Pulse
21	22	O	WD-	Write Data
23	24	O	WG-	Write Gate
25	26	I	TK00-	Track 00
27	28	I	WP-	Write Protect
29	30	I	RD-	Read Data
31	32	O	HS-	Head Select 0/1
33	34	I	DCHNG-	Diskette Change

Table 7-4: Floppy Interface Connector

7.4. Configuration Jumper Options

The various configuration jumpers are shown in the following subsections. Installing the jumper in the positions shown will force that option to a logic low level. Conversely, removing the jumper from the position indicated will force that option to a logic high level. However, for the purposes of the jumper tables, a "1" implies that the jumper is installed and a "0" implies that NO jumper is installed, with an "*" indicating the default factory settings.

7.4.1. ASC I/O Address Space (W3)

These jumpers can be used to change the ASC I/O address space when desired. Address lines SA7-3 are selectable, SA9-8 are fixed at a high level, and all other lines are just driven by the host bus as required. The ASC is currently selected to decode 320-323 (SA9-SA0), the LSB 3 bits are used to select onboard ASC registers. SA9-SA8 are fixed at +5V or a logic "1" level.

```
* Default Config.        W3
                     +-------+
   SA3          1    | o---o |  2
   SA4          3    | o---o |  4
   SA5          5    | o   o |  6
   SA6          7    | o---o |  8
   SA7          9    | o---o |  10
                     +-------+
```

SA7	SA6	SA5	SA4	SA3	I/O ADDR
0	0	0	0	0	3F8H
0	0	0	0	1	3F0H
0	0	0	1	0	3E8H
0	0	0	1	1	3E0H
0	0	1	0	0	3D8H
0	0	1	0	1	3D0H
0	0	1	1	0	3C8H
0	0	1	1	1	3C0H
0	1	0	0	0	3B8H
0	1	0	0	1	3B0H
0	1	0	1	0	3A8H
0	1	0	1	1	3A0H
0	1	1	0	0	398H
0	1	1	0	1	390H
0	1	1	1	0	388H
0	1	1	1	1	380H
1	0	0	0	0	378H
1	0	0	0	1	370H
1	0	0	1	0	368H
1	0	0	1	1	360H
1	0	1	0	0	358H
1	0	1	0	1	350H
1	0	1	1	0	348H
1	0	1	1	1	340H
1	1	0	0	0	338H
1	1	0	0	1	330H
1	1	0	1	0	328H
1	1	0	1	1	320H *
1	1	1	0	0	318H
1	1	1	0	1	310H
1	1	1	1	0	308H
1	1	1	1	1	300H

Table 7-5: ASC I/O Address Space Jumpers

7.4.2. BIOS ROM Address Space (W4)

These jumpers can be used to change the BIOS ROM address space when desired. Address lines SA16-13 are selectable, SA19-18 are fixed at a high level, SA17 is fixed at a low level and all other lines are just driven by the host bus as required to access an 8KX8 ROM. The ASC is currently selected to decode C0 (SA19-SA13).

```
+-------------------------+
|    Address Select (W4)  |
+-------------------------+
|    SA19 fixed at +5 V   |
|    SA18 fixed at +5 V   |
|    SA17 fixed at  0 V   |
|    SA16  7 o---o  8     |
|    SA15  5 o---o  6     |
|    SA14  3 o---o  4     |
|    SA13  1 o---o  2     |
|    GND     |   |  4.7K  |
+-------------------------+
```

SA16	SA15	SA14	SA13	BIOS ADDR
0	0	0	0	DE00H
0	0	0	1	DC00H
0	0	1	0	DA00H
0	0	1	1	D800H
0	1	0	0	D600H
0	1	0	1	D400H
0	1	1	0	D200H
0	1	1	1	D000H
1	0	0	0	CE00H
1	0	0	1	CC00H
1	0	1	0	CA00H
1	0	1	1	C800H
1	1	0	0	C600H
1	1	0	1	C400H
1	1	1	0	C200H
1	1	1	1	C000H *

Table 7-6: BIOS ROM Address Space Jumpers

7.4.3. DRQ/DACK Jumpers (W2)

ASC uses DMA channels 5, 6, and 7 with channel 6 being the standard configuration. No other configurations are permitted since a 16 bit host bus is assumed. Also note that the host must turn on the ASC DMA enable first and then enable the system DMA channel. This is required since the DMA line is a tri-state control signal. If this is not done, spurious DMA signals might occur if the host system DMA is enabled first since the ASC will tri-state this signal on power up. Any other I/O card that shares the same DMA channel can also be used to drive this signal to avoid spurious signals.

```
                     (W2)
                  +--------+
    DRQ7      1   | o    o |  2
  * DRQ6      3   | o----o |  4
    DRQ5      5   | o    o |  6
    DACK7-    7   | o    o |  8
  * DACK6-    9   | o----o |  10
    DACK5-   11   | o    o |  12
                  +--------+
```

Table 7-7: DMA Channel Configuration Jumpers

7.4.4. IRQ Jumpers (W1, W2)

The ASC supports IRQ channels 3-5, 7, 9-12, and 14-15 with channel 3 being the standard configuration. Since this signal is tri-state driven, the same precaution as the DMA channel enable applies here as well.

```
                (W1)          &                  (W2)
             +--------+                       +--------+
* IRQ3    1  | o----o |  2      IRQ15    13   | o    o |  14
  IRQ4    3  | o    o |  4      IRQ14    15   | o    o |  16
  IRQ5    5  | o    o |  6      IRQ12    17   | o    o |  18
  IRQ7    7  | o    o |  8      IRQ11    19   | o    o |  20
  IRQ9    9  | o    o |  10     IRQ10    21   | o    o |  22
             +--------+                       +--------+
```

Table 7-8: IRQ Channel Configuration Jumpers

7.4.5. Terminator Power (W5)

The SCSI bus devices are connected together using a daisy chain cable. This cable should have terminating resistors at each end of the cable. These resistors are designated Z1 and Z2 on the ASC. All SCSI devices bewtween the two ends of the cable should not have the terminating resistor packs but one of them must supply power to these packs via a jumper similar to W5 as shown below:

```
                    (W5)
                 +--------+
      TRMPWR   1 | o---o  | 2
                 +--------+
```

7.4.6. Dual Speed Spindle (W7)

When the jumper is installed, it enables a dual speed, 360 RPM, floppy spindle. Without the jumper, it disables the dual speed spindle for a single speed, 300 RPM, floppy.

```
                    W7 = enable dual speed
                 +--------+
      DRV      1 | o    o | 2
                 +--------+
```

7.4.7. Floppy Write Precomp Select (W8)

With the jumper installed, the write precomp value set is 187 NS (for a 1.2 MB floppy) and with the jumper removed, the write precomp value selected is 125 NS (usually for a 360 KB floppy).

```
                 W8 = 187 NS precomp (1.2 MB), else
                    = 125 NS precomp (360 KB)
                 +--------+
     PRECOMP   1 | o    o | 2
                 +--------+
```

7.4.8. Floppy Control (W6) & (W9)

```
           +--------+
  W6     1 | o---o  | 2
           +--------+
           +--------+
  W9     1 | o---o  | 2
           +--------+
```

W6	W9	DESCRIPTION
0	0	Onboard Floppy Enabled with NO Hard or Combo Card
0	1	Onboard Floppy Enabled with Hard Card only
1	X	Onboard Floppy Disabled *

Table 7-9: Floppy Control Jumpers

8. SPECIFICATIONS

8.1. Performance

Host Transfer Rate	5.3 MByte/sec Burst on 8 MHZ AT
SCSI Transfer Rate	4.0 MByte/sec Synchronous
	2.0 MByte/sec Asynchronous
LCPU Clock	4 MHZ
SBIC Clock	8 MHZ
Selection Timeout	250 mS
Logical Threads	16
AT Bus On/Off Times	Programmable
MTBF:	10,000 POH
MTTR:	30 minutes

8.2. Power requirements

Voltage:	+5 V +-5%
Current:	X.0 amps max, X.0 amps typical
Noise Ripple:	0.1 volts max, typical 25 mV
12 Volt Supply:	+12V +-10%, 100 mA Max

8.3. Physical

Formfactor:	IBM PC XT	IBM AT
Length:	13.1 in.	13.3 in.
Width:	4.2 in.	4.9 in.
Height:	0.5 in.	0.5 in.

Host Interface:	16 bit bi-directional AT bus
	62 Pin Card Edge Connector
	38 Pin Card Edge Connector
SCSI Interface:	8 bit bi-directional SCSI bus
Drive cable lengths:	20 ft. max. (6 Meters)
SCSI Connector:	50 Pin

The locations of connectors and other critical dimensions are indicated on the assembly/drill drawings mentioned in the references.

8.4. Enviromental

Ambient temperature:	0-55 Deg. C.
Relative Humidity (non-condensing):	10% - 95%
Air flow at 1/2" from component surface:	100 linear ft/min
Altitude (operating):	10,000 ft. max.
Altitude (storage):	15,000 ft. max.

8.5. Electrical Specifications

8.5.1. DC Characteristics

The Host Interface requirements are shown below. Input drive requirements are one standard "LS" load. Detailed specifications can be found in the references.

Vcc = 5V +/- 5% ; GND = 0V ; TA = 0 to 55 Deg. C.

SYMBOL	PARAMETER	MIN	MAX	UNITS	CONDITIONS
VIH	Input high voltage	2.0		V	
VIL	Input low voltage		0.8	V	
VOL1	SCSI Output Low Voltage		0.5	V	IOL = 48.0 mA
VOL2	Host Output Low Voltage		0.5	V	IOL = XX.0 mA
VOH1	SCSI Output High Voltage	2.4		V	IOH = -400 uA
VOH2	Host Output High Voltage			V	IOH = XXX uA

Table 8-1: DC Characteristics

8.5.2. AC Characteristics

The AC characteristics for the SCSI and the host interface timings are completely defined in references of section 2. Portions pertinent to the ASC design are discussed in this section.

The host executes I/O reads from the ASC ports or the BIOS, and then writes to the ports to execute a command or generates a control strobe. The FPDMA is used for data transfers between the ASC and the host system RAM memory.

Data transfers across the SCSI bus use the REQ-/ACK- handshake as defined by the SBIC data sheets.

8.5.2.1. Programmed I/O or BIOS ROM Read

For addressing the 4 ports decoded as XX0-XX3 (hex), the LSB 10 bits of the 20 bit address bus are used. AEN should be low or de-asserted. The full 20 bit address bus is used to read the ROM regardless of AEN. The ROM address may optionally be gated with SMEMR-, if so desired. The data bus, D7-D0, is qualified by the read strobes, IOR- or SMEMR-, but not both.

SYMBOL	CHARACTERISTIC	MIN.	MAX.	UNITS	NOTES
t1	Read Address Setup Time	30		nS	
t2	Read Active Time (ROM)	260		nS	
	(I/O)	50			
t3	Address Hold Time	0		nS	
t4	Data Valid Time (ROM)		250	nS	
	(I/O)		40		
t5	Data Hold Time	0	x	nS	

Table 8-2: Host Prog. I/O or ROM Read Parameters

8.5.2.2. Programmed I/O Write

Similar to programmed I/O read, except that the ROM cannot be obviously written to.

SYMBOL	CHARACTERISTIC	MIN.	MAX.	UNITS	NOTES
t6	Write Address Setup Time	30		nS	
t7	Write Active Time (I/O)	50		nS	
t8	Address Hold Time	20		nS	
t9	Data Setup Time	0		nS	
t10	Data Valid Time	10		nS	
t11	Data Hold Time	30		nS	

Table 8-3: Host Prog. I/O Write Parameters

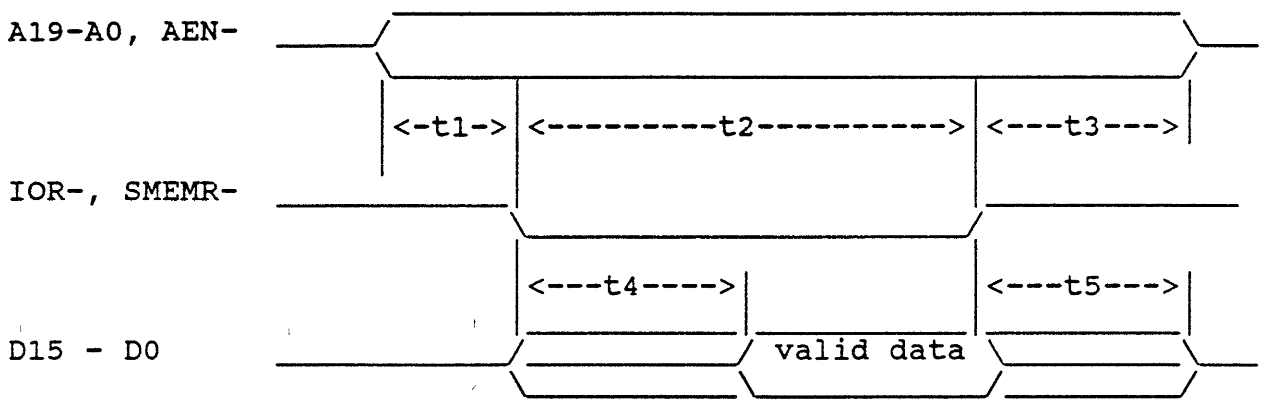


Fig. 8-1: Host Prog. I/O or ROM Read Timing

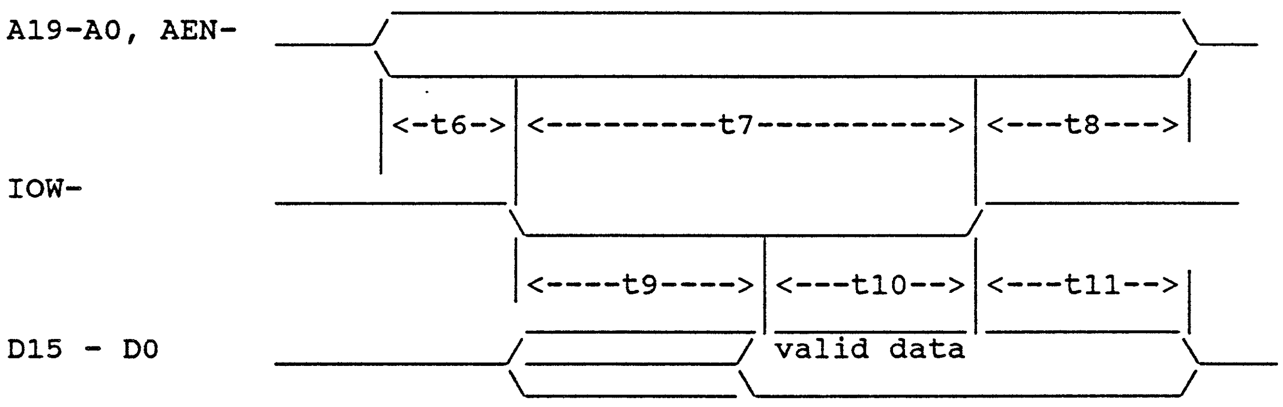


Fig. 8-2: Host I/O Write Timing

8.5.2.3. DMA Read Cycle

The ASC DMA memory cycle is indicated by the absence of any valid address lines. These are disabled by AEN being asserted high. Instead of the address lines, the DMA processor asserts DACKX- to begin a byte transfer. DRQX is gated with DACKX-, hence it is de-asserted when DACKX- is asserted. These two lines complete the handshake between the DMA processor and the host, until all bytes of data have been transferred to the SCSI device buffer.

SYMBOL	CHARACTERISTIC	MIN.	MAX.	UNITS	NOTES
t12	DACKX- Setup Time	125	200	nS	
t13	Read Active Time	250		nS	
t14	Data Valid Setup Time	10		nS	
t15	Data Hold Time	0		nS	

Table 8-4: Host DMA Read Parameters

8.5.2.4. DMA I/O Write Cycle

Similar to DMA I/O read, except that the SBIC buffer data is transferred to the host system memory.

SYMBOL	CHARACTERISTIC	MIN.	MAX.	UNITS	NOTES
t16	DACKX- Setup Time	125	200	nS	
t17	Read Active Time	250		nS	
t18	Data Valid Setup Time		20	nS	
t19	Data Hold Time	65		nS	

Table 8-5: Host DMA Write Parameters

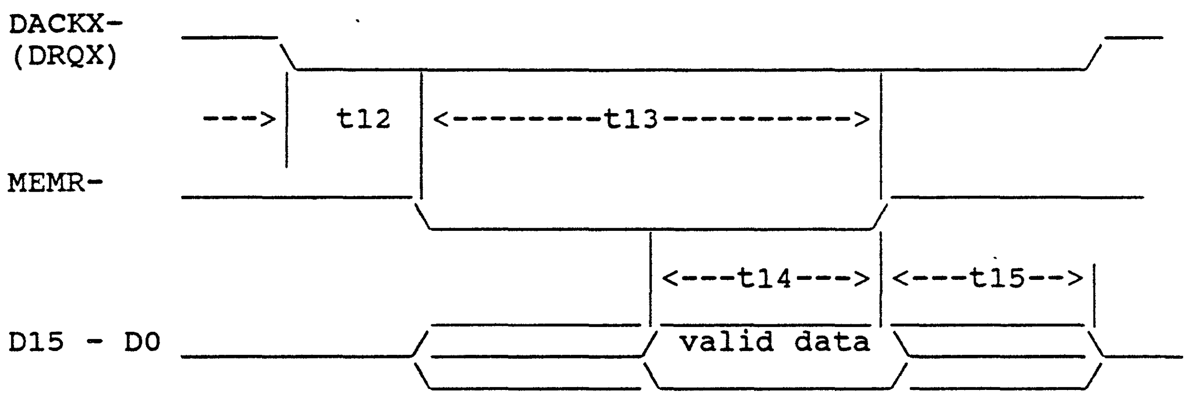


Fig. 8-3: DMA Read Timing

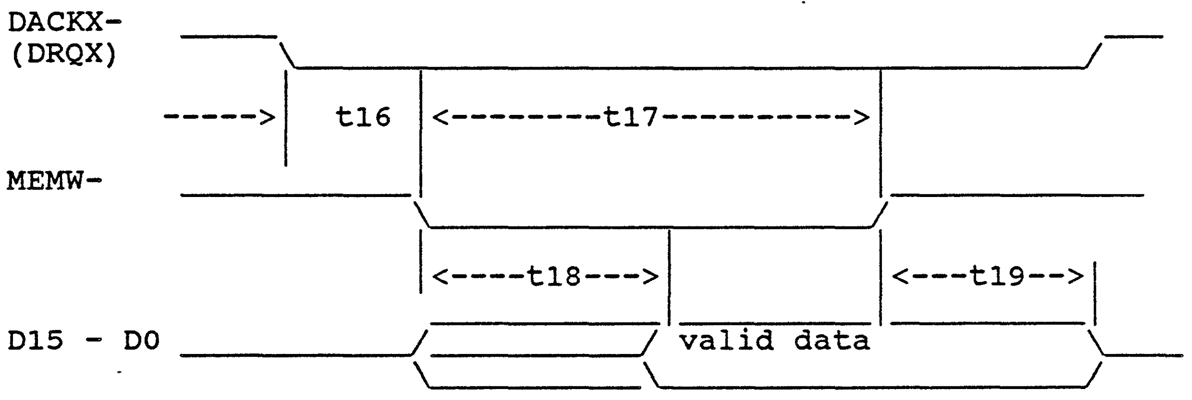


Fig. 8-4: DMA I/O Write Timing

APPENDIX A

A. WD7000-ASC Tables and Error Codes

The most frequently used tables are summarized here for convenience of an experienced ASC user. Details about individual fields can be obtained from sections 5 and 6.

Included in this appendix are ICMB status codes and all of the vue codes. For the most part, these codes are self-explanatory. If an ambiguity occurs, the user must reference the ASC source listing.

A.1. Host I/O Port Access Registers

Shown below are the LSB nibbles of the host I/O address space. The upper nibbles/bytes of the I/O address may be configured by the user using the jumper tables shown in section 7.

Addr	Read Port	Type	Write Port	Type
0	ASC Status	D7-D0	Command Register	Byte
1	Host Interrupt Stat.	Byte	ASC Interrupt Ack.	strobe
2	Reserved		Host Control reg.	D3-D0
3	Reserved		Reserved	

Table A-1: ASC I/O Port Definition

A.2. Host Control Register

The host may reset both the SCSI and the ASC by writing a 03 followed by 00 to this register, with a minimum time of 25 uS between the two writes in order to ensure the minimum reset pulsewidth requirements. Following proper ASC initialization, the ASC DMA port must be initialized first in order to drive the DRQ channel to the host. Similarly the selected IRQ channel may be enabled at the same time or later if desired. The most important point to remember is that since these lines are tri-statable, some I/O device must drive these lines in order to prevent spurious signals from occuring. Alternatively, the host IRQ/DRQ channels should be disabled, while the I/O device lines are tri-stated.

Bit	Description	Meaning
7	Reserved.	Not Used
6	Reserved.	Not Used
5	Reserved.	Not Used
4	Reserved	Not Used
3	Interrupt Enable	0 = Interrupt Off 1 = Interrupt On
2	DMA Request Enable	0 = DRQ Disabled 1 = DRQ Enabled
1	SCSI Port Reset	0 = SCSI Reset Off 1 = SCSI Reset On
0	ASC Reset	0 = ASC Reset Off 1 = ASC Reset On

Table A-2: Host Control Byte Format

A.3. Host Command Protocol

The next command byte or parameter cab be written to the ASC as soon as the READY status is read by the host. If a byte is rejected, it is either illegal or the host command queue capacity has been exceeded. The host can resend the corrected command/parameter or issue an interrupt on free OGMB command in case the host queue has been filled. Once a SCSI soft reset command has been issued, a new soft reset command must NOT be issued until the aborted command status has been returned to the host via the ICMB. For a SCSI hard reset during a host command execution, a reset acknowledge command is required.

Opcode	Definition
00	No Operation
01	Initialization (10 Byte sequence)
02	Disable Unsolicited Interrupts
03	Enable Unsolicited Interrupts
04	Interrupt on Free OGMB
05	SCSI Soft Reset (2 byte sequence)
06	SCSI Hard Reset Acknowledge
07-7F	Reserved
10NNNNNN	Start Command in OGMB #N
11NNNNNN	Start Multiple I/O (Scan Mailboxes) N = Scan signature 6 bits max.

Table A-3: ASC Command Port Opcodes

The initialization sequence can only be initiated following an ASC reset, after the READY status has been set by the ASC. At all other times, the command will be rejected. Following the execution of this command the host is required to wait for a status of 50H, with the lower nibble of the status byte being masked off by the application driver routine.

The maximum on time used should be less than 15 us less ALL overhead time required to allow the host to service memory refresh cycles, including DMA bus arbitration time. The ASC DMA controller transfers a word every 375 ns once it has control of the bus. For optimum performance the host should use a bus on time of 3.0 us to transfer 8 words, the length of the FIFO. Bus arbitration overhead is about 1 to 2 us. The minimum off time can be as low as .125 uS.

Typical on time used = 40H = 8.0 uS (practical maximum = 12.0 uS)
Typical off time used = 0FH = 1.875 uS

Byte Num.	Bit Pos. 7654 3210	Definition
00	0000 0001	Command Opcode
01	0000 0SSS	3 Bit SCSI ASC ID Field
02	TTTT TTTT	Bus On Time (125 ns per count value)
03	TTTT TTTT	Bus Off Time (125 ns per count value)
04	0000 0000	Reserved
05	AAAA AAAA	Starting Address of Mail Block (LSB)
06	AAAA AAAA	Starting Address of Mail Block
07	AAAA AAAA	Starting Address of Mail Block (MSB)
08	0NNN NNNN	Number of OGMB (max. 64) (0,1 = 1)
09	0NNN NNNN	Number of ICMB (max. 64) (0,1 = 1)

Table A-4: Initialization Command Format

Byte	Bit Position / Description						
	7	6	5	4	3	2	1	0
00	S/I	Command Opcode						
01	Target ID			0	0	L.U.N.		
02	SCSI Command Block (Byte # 0)							
thru	SCSI Command Block Bytes 1 - 10							
13	SCSI Command Block (Byte # 11)							
14	SCSI Return Status Byte							
15	Vendor Unique Error Code							
16	Maximum Data Transfer Length (MSB)							
17	Maximum Data Transfer Length in Bytes							
18	Maximum Data Transfer Length (LSB)							
19	SCSI Data Block Pointer (MSB)							
20	SCSI Data Block Pointer							
21	SCSI Data Block Pointer (LSB)							
22	Next Command Link Pointer (MSB)							
23	Next Command Link Pointer							
24	Next Command Link Pointer (LSB)							
25	Direc	Reserved for Future Expansion						
thru	Reserved for Future Expansion							
30	Reserved for Future Expansion							
31	Reserved for Future Expansion							

Table A-5: SCSI Command Execution Block

Typically bytes 2-4 of the ICB are used for the host buffer transfer length and bytes 5-7 contain the absolute address of the buffer, with the MSB specified first.

Byte	7	6	5	4	3	2	1	0
00	S/I	Command Opcode						
01	ICB Parameter Byte # 1							
02	ICB Parameter Byte # 2							
03	ICB Parameter Byte # 3							
04	ICB Parameter Byte # 4							
05	ICB Parameter Byte # 5							
06	ICB Parameter Byte # 6							
07	ICB Parameter Byte # 7							
08	ICB Parameter Byte # 8							
09	ICB Parameter Byte # 9							
10	ICB Parameter Byte # 10							
11	ICB Parameter Byte # 11							
12	ICB Parameter Byte # 12							
13	ICB Parameter Byte # 13							
14	ICB Parameter Byte # 14							
15	ASC Vendor Unique Error Status							

Table A-6: ICB Command Execution Block

The ICB commands supported are summarized in the table below:

Opcode	Definition
80	Open Inbound Data Buffer (init-to-init)
81	Receive Command from another Initiator
82	Receive Data from another Initiator
83	Receive Data with Status from another Initiator
84	Send Data to another Initiator
85	Send Data with Status to another Initiator
86	Send Command Status to another Initiator
87	Open Scratchpad CDB (for exception handling)
88	Read Initialization Bytes
89	Read ASC SCSI ID
8A	Set Execution Parameters
8B	Read Execution Parameters
8C	Read Firmware Revision Level
8D	Execute Onboard Diagnostics
8E	Prepare Sense Data
8F	Prepare Inquiry Data
90	Open Outbound Data Buffer (init-to-init)
91	Set Host Bus On/Off Times

Table A-7: ICB Command Opcodes

In the host memory, all of the OGMBs occupy contiguous memory locations follwed by all of the ICMBs. The number of ICMBs are independent of the number of OGMBs. The only advantage of having more number of ICMBs is that the host can be informed of the command completion sooner and/or reduce the overall execution time between 2 host commands. Up to 32 IRQs can be queued internally to the ASC. To see the remaining interrupts the host must acknowledge the previous interrupt after having completely processed it.

Mail Type	Contents
O.G.M.B. #n	Mail Status (0=empty, else full) Command Block Pointer (MSB) Command Block Pointer Command Block Pointer (LSB)
I.C.M.B. #m	Action Status Command Block Pointer (MSB) Command Block Pointer Command Block Pointer (LSB)

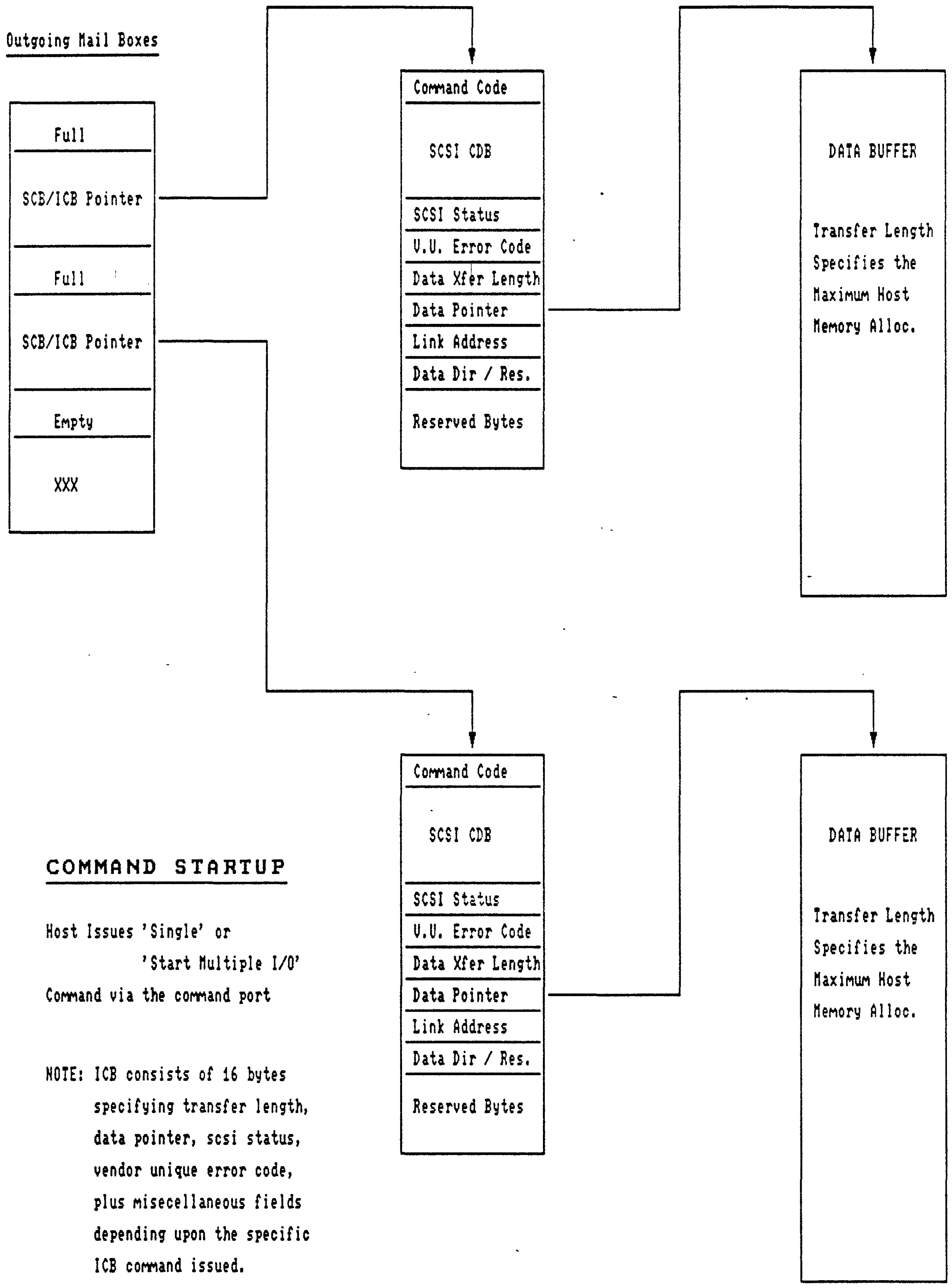
```
OGMB Address = Starting address of mail block + mail box number n x 4

ICMB Address = Starting address of mail block + mail box number m x 4
                + 4 (total number of OGMB's)

     where    0 <= n <= p-1   &   0 <= m <= q-1

              p,q = number of OGMBs/ICMBs resp.  (max. = 64)
              m,n = mail box offset numbers from 0 to 63 (max.)
```

Table A-8: Mail Box Format

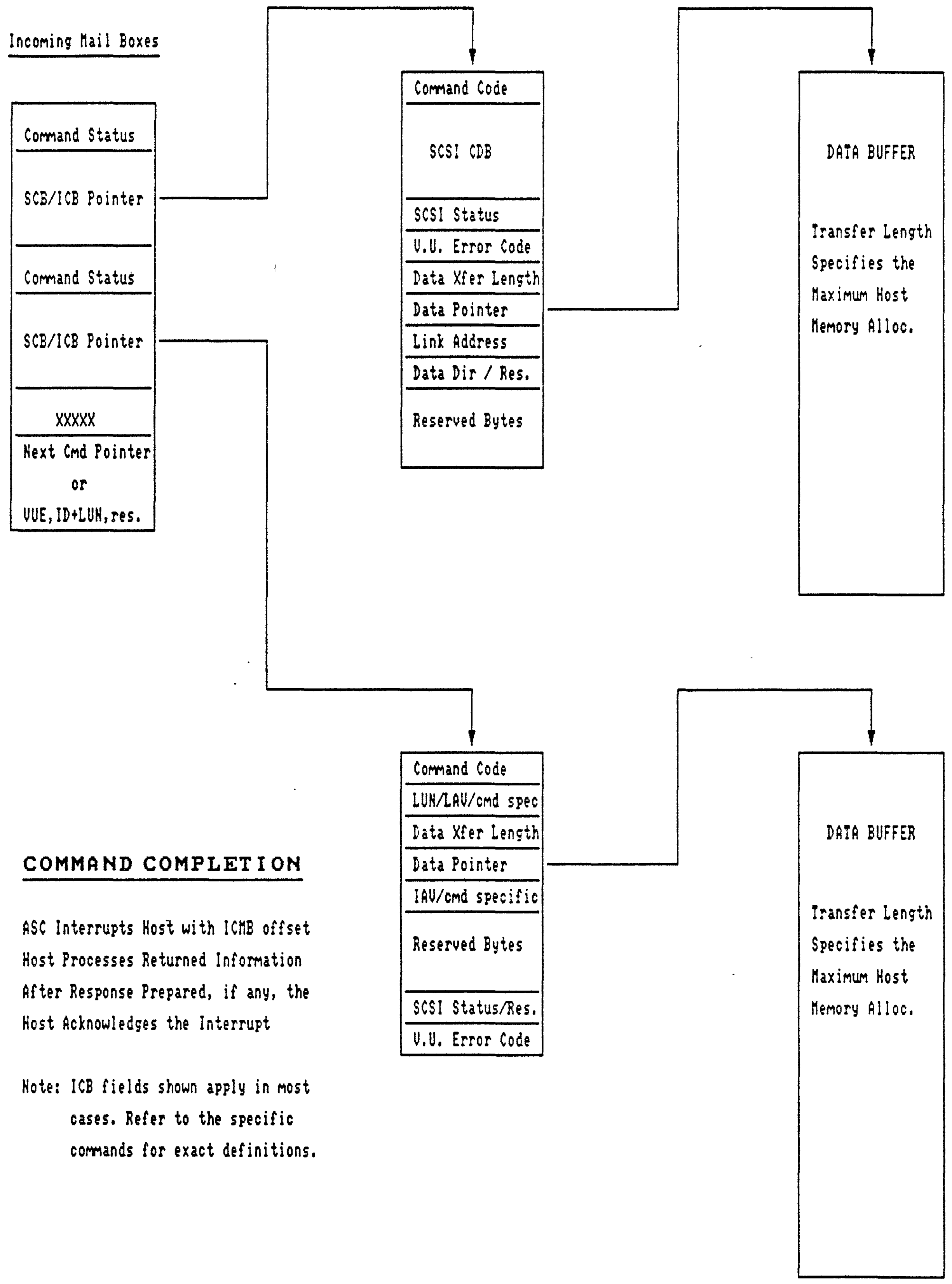


COMMAND STARTUP

Host Issues 'Single' or 'Start Multiple I/O' Command via the command port

NOTE: ICB consists of 16 bytes specifying transfer length, data pointer, scsi status, vendor unique error code, plus misecellaneous fields depending upon the specific ICB command issued.

Fig. A-1: Execution of SCB/ICB



COMMAND COMPLETION

ASC Interrupts Host with ICMB offset
Host Processes Returned Information
After Response Prepared, if any, the
Host Acknowledges the Interrupt

Note: ICB fields shown apply in most cases. Refer to the specific commands for exact definitions.

Fig. A-2: Completion of SCB/ICB

A.4. Host Status Port

This status port must be read by the host to initiate programmed I/O or a polled interrupt command completion. For future compatibility, the user should either mask off the lower nibble or ignore it.

Bit	Description	Meaning
7	Interrupt Image Flag	0 = Interrupt Inactive 1 = Interrupt Active
6	Command Port Ready	0 = Port Busy 1 = Port Ready
5	Command Port Byte Rejected	0 = Byte Accepted 1 = Illegal Cmd/Param.
4	ASC Initialized Flag	0 = Init. Required 1 = ASC Initialized
3	Reserved.	Set to "1" Mask off this bit
2	Reserved.	Set to "1" Mask off this bit
1	Reserved.	Set to "1" Mask off this bit
0	Reserved	Set to "1" Mask off this bit

Table A-9: ASC Status Byte Format

A.5. Host Interrupt Status Byte

Whenever, an the ASC asserts an interrupt, the host should read the interrupt status byte which references an ICMB. Byte 0 of the ICMB is the command status as determined by the ASC and bytes 1-3 contain the absolute address of the CDB that completed execution. In cases where this field is inapplicable, it will contain all FF's or information specific to the nature of the unsolicited interrupt.

For the diagnostic error codes, no host interrupt is generated but this register will contain a valid code when the ASC first asserts the READY status following an ASC reset.

76543210	Definition
00	Power-on condition, no diagnostics executed
01	No Diagnostic Error Occured
02	RAM failed
03	FIFO R/W failed
04	SBIC reg. R/W failed
05	Initialization D-FF failed
06	Host IRQ D-FF failed
07	ROM checksum error
10NNNNNN	Outgoing mail box number 'N' is available. Used only when "Interrupt on Free OGMB" command has been issued.
11NNNNNN	Incoming mail box 'N' needs service.

NOTE: Codes 00-07 valid after READY is asserted in the status port
Codes 05-06 valid only after when ASC is first powered-up

Table A-10: Interrupt Status Byte / Diagnostic Error Code

A.6. Host Status Codes (ICMB0)

Codes 00-7FH are generally used for commands that were issued by the host and codes above 80H are used for unsolicited interrupts to the host, provided the proper interrupt mask has been enabled by the host. Each major class of command completion is given a unique status code. The interpretation of the SCB status and/or the vue is required to differentiate between the errors belonging to a given status code.

The SCSI hardware reset is included twice as it may occur during commands pending or when the ASC is idle not executing any of the host commands. If a command is under execution, the interrupt due to a SCSI hardware reset is unmaskable. If no commands are under execution, then the host may wish to avoid dealing with this interrupt by simply setting bit 4 of the unsolicited interrupt mask.

Two separate ICMB status codes have been defined to handle scan mailbox and abort/reset message command since both of them may result in not executing or deleting any of the commands from the Qs. It may also be desirable to obtain the overall command completion status, since the individual completion status is always reported in the CDB associated with a particular command.

Code	Definition / Effect
00	Reserved
01	Command complete; No errors
02	Command complete, error logged in SCB/ICB bytes 14/15
03	Scan OGMB command has been completed (bytes 1-3 = FF)
04	Command failed to complete without SCSI status, see vue
05	Cmd terminated; bus reset by external device during SCB
06	SBIC or hardware failure, requires host reset (1-3 = FF)
07	SCSI soft reset cmd completed (bytes 1-3 = FF)
08->7F	Reserved
80	Unexpected Reselection byte 1 = VUE ; byte 2 = ID+LUN
81	Unexpected Selection byte 1 = VUE ; byte 2 = ID+LUN
82	Abort command message byte 1 = VUE ; byte 2 = ID+LUN
83	SCSI reset msg received byte 1 = VUE ; byte 2 = ID+LUN
84	SCSI hardware reset with no cmds in Q (bytes 1-3 = FF)
85->FF	Reserved

Table A-11: ICMB Return Status Values

A.7. Vendor Unique Error Codes (vue)

These errors have been regrouped mainly for ease of use by the operator or a field technician. Errors have been classified to fall under three basic categories: hardware related failures (1-31), illegal command parameter specification (32-63), and SCSI protocol related failures (64-88). This last category may include a command status instead of an SCSI failure. Accordingly some of the codes have been reserved. These error codes apply across any ICMB status. New groupings or codes may be added to list as the need arises in the future. All error codes listed are in hex.

*Diagnostic error codes reported in ICB byte 15 or on power-up in the interrupt status byte register. Normally checked before the initialization command. Replace faulty ASC component.

Code	Brief Description
00	power-up condition
01	no errors occured
02	RAM failed
03	FIFO R/W failure
04	SBIC failure
05	Initialization D-FF failure
06	Host IRQ D-FF failure
07	ROM checksum error
18	DMA error during host system memory diagnostics

*reported when SBIC interrupt response to an information phase is incorrect; target is left connected. The exact cause of error is only intended to aid host driver debugging and/or a service technician. Suggested recovery: reset ASC and re-issue all unfinished commands.

08	unexpected SBIC response during data out phase
09	unexpected SBIC response during data in phase
0A	unexpected SBIC response during command out phase
0B	unexpected SBIC response during status in phase
0C	unspecified SCSI out phase
0D	unspecified SCSI in phase
0E	unexpected SBIC response message out phase
0F	unexpected SBIC response message in phase
10	unexpected SBIC interrupt status code

*illegal command parameters specified - host may re-issue command after correction.

20	command issued with OGMB marked empty
21	illegal command parameter
22	command cannot be executed at the present time
23	target data direction contary to the one specified
24	set/read execution parameters index out of range
25	set/read execution parameters count out of range
26	host has a previous cmd in progress for same SCSI target+lun

*SCSI protocol related status - may or may not be treated as errors by the host drivers.

40	target send less data than the allocation length (bytes 16-18)
41	target wants to send more data than the allocation length
42	target wants to transfer data but host allocation length = 0
43	reserved
44	specified tag (SCSI ID+LUN) command aborted
45	specified target received reset message & cmd aborted
4D	timeout occured during target selection / reselection
4E	target did not send ID msg or invalid source ID

4F	no record of the SCB to be executed

*reported when target disconnects suddenly or retries have been exhausted (scsi bus parity errors) or target does not switch to MOP in response to ATN when an error is detected - suggested recovery: re-issue the command. Once again the exact cause is only intended for trouble shooting purposes.

50	unrecov/disc - ID or selection phase
51	unrecov/disc - command phase
52	unrecov/disc - message in phase
53	unrecov/disc - message out phase
54	unrecov/disc - ID or reslection phase
55	unrecov/disc - data transfer phase
56	unrecov/disc - status phase
57	unrecov/disc - command completion phase
FF	not applicable in current field

*additional error codes unique to the target mode of operation. In case of overlap the initiator mode, the previously defined codes (as shown above) are used. Note that the ICMB format is different in some cases where a valid host CDB pointer does not exist. Briefly, ICMB0 = ASC cmd completion status, ICMB1 = VUE, ICMB2 = ID+LUN, and ICMB3 = 0FFH or invalid. The host must then use the individual command phase commands (i.e. ICBs) to complete the operation manually. To avoid host interrupt handling, the data & status phases may be combined. In case of errors, the host must prepare the appropriate sense data block before sending the SCSI command status to the initiator.

80	scsi hard reset during a solicited selection
81	valid cmd bytes in scratchpad buffer
82	no cmd scratchpad buffer opened
83	illegal CDB transfer length (less than 6 or greater than 12)
84	no inbound data buffer opened or wrong id or unqualified lun
85	no sense data buffer opened for the specified lun
86	buffer opened but next command was not request sense
87	no inquiry buffer opened as qualified by lav/iav
88	no outbound data buffer opened or wrong id or unqualified lun

A.8. Set/Read Execution Parameter Data Block

25 data bytes can be specified with this command so that various internal parameters can be changed if desired by simply using one command instead of several individual commands. Bits that are toggled often by the host can also be set/reset by command port commands: op-codes 02 - 04.

Normally the host should read these parameters and only change those values that pertain to the host requirements. All other default values should be left unaltered by the host. These have mainly been provided for in house testing and quality assurance purposes.

Notes: All default values are shown in the parenthesis under the byte #.
 * Separate host commands through the cmd port available for these functions as well.

Byte #	Definition
0,2,4,6, 8,10,12,14 (40 or C0)	even byte values set by the host, bits 6-0, determine the SCSI synchronous data rates, with the same definition as the SBIC data sheets. Recommended sync rate value = 45H. Bit 7 is set after host writes a new value to these locations & a cmd has been issued to the target. The target ID is implied by position of the word, even or odd bytes 0-15.
1,3,5,7, 9,11,13,15 (40)	odd byte values reflect the agreed rate between the target and the initiator. These bytes are reported for information purposes only and should never be altered by the host.
16 (00)	RESERVED
17 (0C)	ram copy of the SBIC control reg. From the host's point of view only bit 0 may be set to enable SCSI bus parity. All other bits must not be altered.
18 (19)	ram copy of timeout reg. Default about 250 mS. see SBIC data sheet for description.
19 (00)	RESERVED
20 (00)	RESERVED
21 (80)	Source ID reg.
bit 7	enable reselection: target can disconnect
bit 6	enable selection: allows mult initiators
22 (00)	user flags: 1 = enable, 0 = disable
bit 2	return the number of data bytes transferred in bytes 16-18 of the SCB. The maximum specified transfer count value will be lost.
* bit 1	interrupt on free OGMB
* bit 0	enable unsolicited interrupt mask bit
23 (00)	unsolicited interrupt mask. Bits 0-4 generate the interrupts 80-84. For bit 5 the ICMB0=81 with ICMB1=vue=85. When any combination of these bits are set, the ASC will generate an extra interrupt with ICMB format as defined above.
bit 0	unexpected reselection bit mask

	bit 1	unexpected selection bit mask
	bit 2	abort msg received bit mask
	bit 3	reset msg received bit mask
	bit 4	SCSI hardware reset with no cmds in the Q
	bit 5	unexpected request sense command received (ICMB0=81)
24	(02)	number of retries on parity/cmd errors. This is mainly included for diagnostic purposes. It is not recommended for the host driver to alter this value.
25	(00)	Reserved for in-house purposes to control dma transfer.

A.9. I/O Address Space

In the tables below, writing a "1" normally sets the action denoted. The bar "-" after the function name is used to denote the complement of the above e.g. FIFO Reset-. In cases where a dual function is intended, a "0" implies the complement of the function stated, else it implies that the function is turned off.

Addr	I/O Function Selected	Dir	Data Bits
80	SBIC Address reg.	R/W	D7-D0
81	SBIC Task File regs.	R/W	D7-D0
40	Time On Bus (125 ns per count)	W	D7-D0
41	Time Off Bus (125 ns per count)	W	D7-D0
42	DMA Address High Byte	W	D7-D0
43	DMA Address Mid Byte	W	D7-D0
44	DMA Address Low Byte	W	D7-D0
45	FIFO Control reg. Byte	W	see below
	Start FIFO Transfer	W	D7
	End FIFO Transfer if Empty/Full	W	D6
	Spare	W	D5
	Transfer Last Byte Left in FIFO	W	D4
	FIFO Reset- (write 0, then 1)	W	D3
	Set FIFO Direction (1 = ASC to AT)	W	D2
	Start Arbitration when Empty/Full	W	D1
	Transfer On an Odd Address	W	D0
46	Read or Write FIFO Data Byte	R/W	D7-D0
47	Spare	W	

Table A-12: I/O Address Space

Addr	I/O Function Selected	Dir	Data Bits
20	Write Host Interrupt (IRQ) reg.	W	D0
21	Host Interrupt Status Byte	W	D7-D0
22	Set Host Command Port Ready	W	Strobe
23	Set Host Status reg. bits	W	see below
	Command Rejected (via cmd. port)	W	D5
	ASC Initialized	W	D4
24	Spare	W	
25	Spare	W	
26	SBIC Reset reg. (write 1, then 0)	W	D0
27	Diagnostic LED (1 = Test Failed)	W	D0
20	Read Host Command reg.	R	D7-D0
21	ASC Status	R	see below
	FIFO Full-	R	D7
	FIFO Empty-	R	D6
	Last Byte Left- (in staging regs)	R	D5
	Last Word Left- (in staging regs)	R	D4
	Host Cmd Port Not Ready (BUSY)	R	D3
	SBIC Interrupt Request (INTRQ)	R	D2
	ASC Channel Initialized	R	D1
	Host Interrupt Request (IRQ)	R	D0
22-27	Spares	R	

Table A-13: I/O Address Space Continued

APPENDIX B

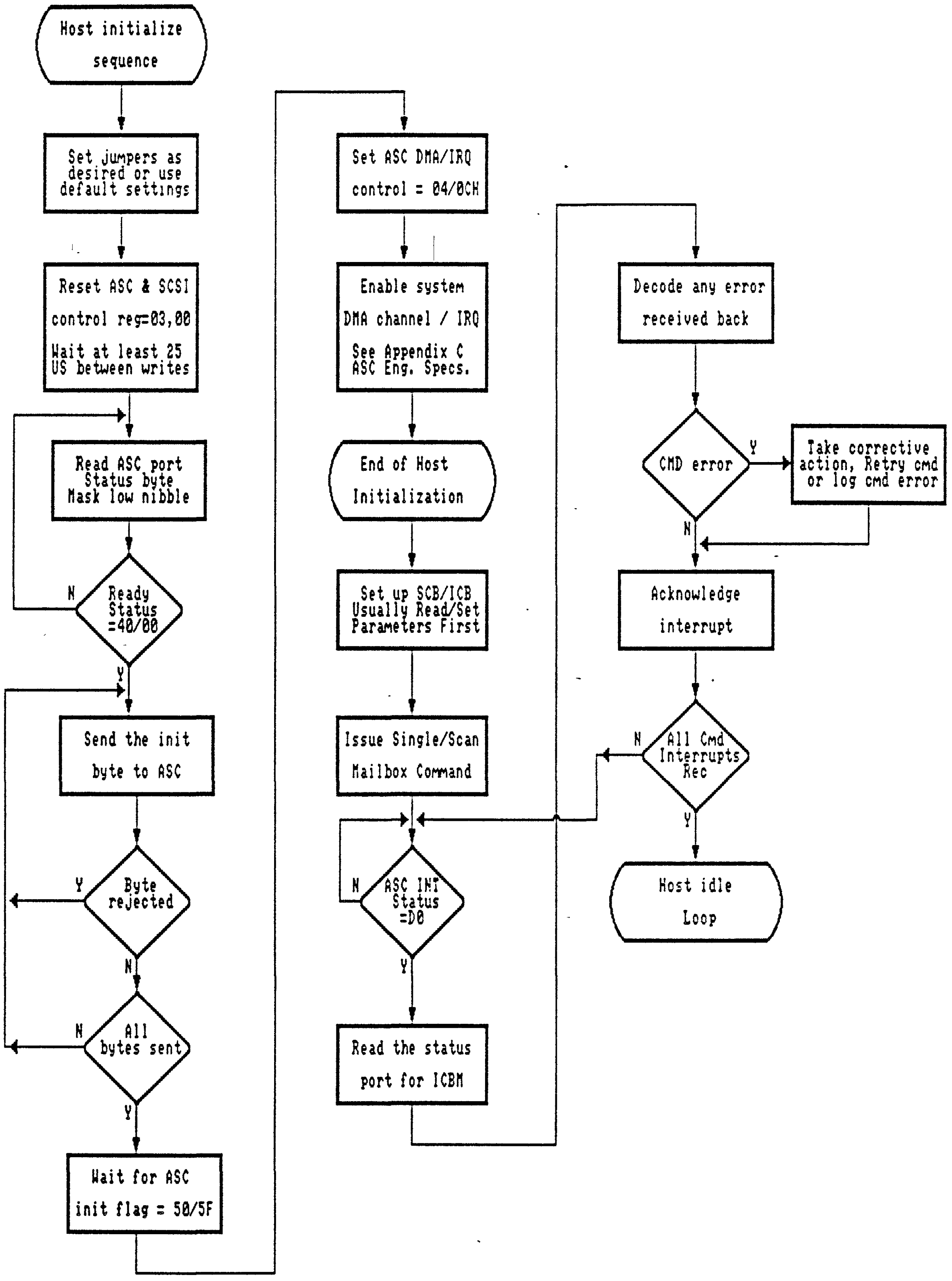

Host initialize sequence
Set jumpers as desired or use default settings
Reset ASC & SCSI control reg=03,00 Wait at least 25 US between writes
Read ASC port Status byte Mask low nibble
Ready Status =40/00
N
Y
Send the init byte to ASC
Byte rejected
Y
N
All bytes sent
N
Y
Wait for ASC init flag = 50/5F
Set ASC DMA/IRQ control = 04/0CH
Enable system DMA channel / IRQ See Appendix C ASC Eng. Specs.
End of Host Initialization
Set up SCB/ICB Usually Read/Set Parameters First
Issue Single/Scan Mailbox Command
ASC INT Status =D0
N
Y
Read the status port for ICBM
Decode any error received back
CMD error
Y
N
Take corrective action, Retry cmd or log cmd error
Acknowledge interrupt
All Cmd Interrupts Rec
N
Y
Host idle Loop

Fig. B-1: Host Initialization / Command Sequence

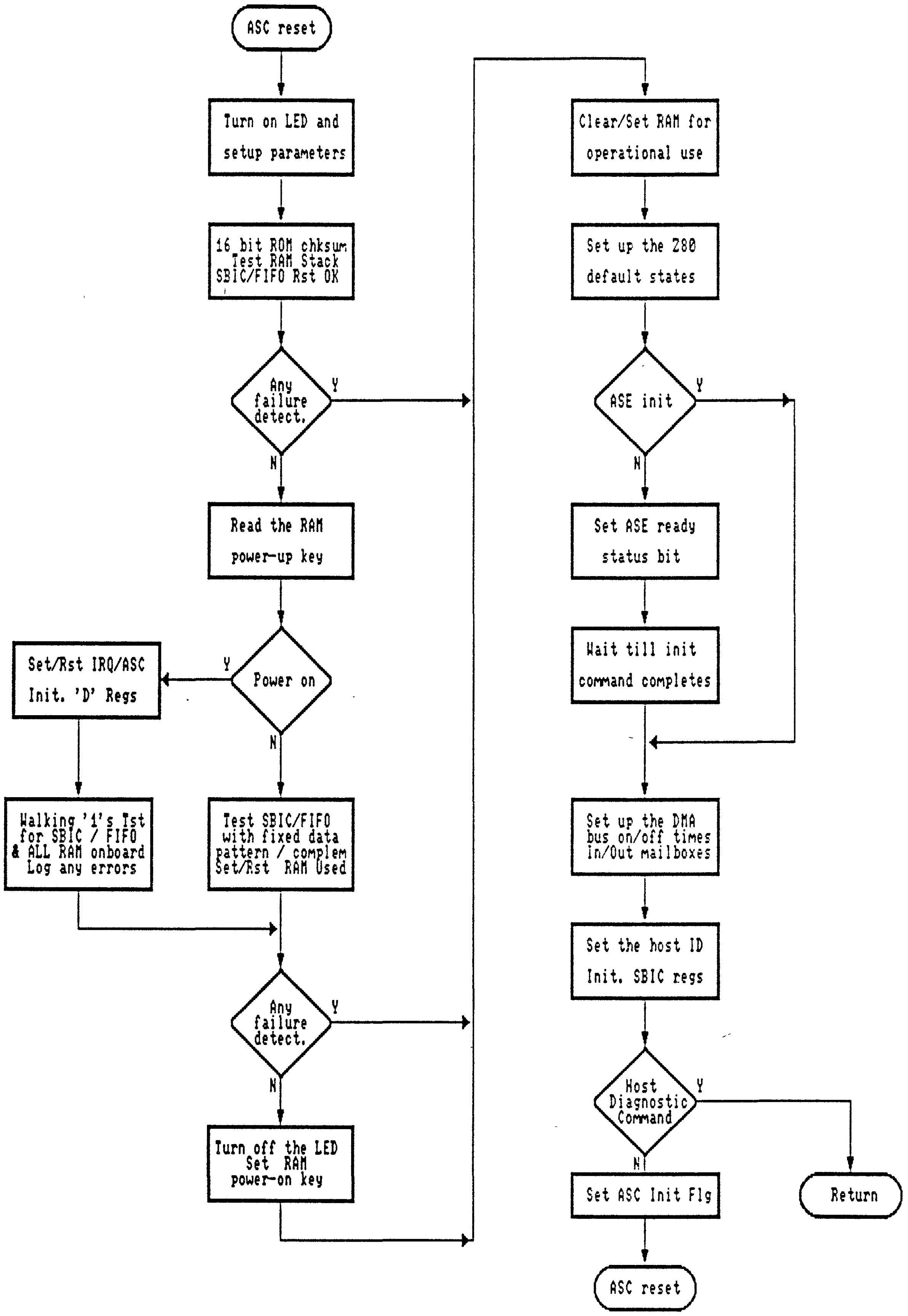

ASC reset
Turn on LED and setup parameters
16 bit ROM chksum Test RAM Stack SBIC/FIFO Rst OK
Any failure detect.
Y
N
Read the RAM power-up key
Power on
Y
N
Set/Rst IRQ/ASC Init. 'D' Regs
Walking '1's Tst for SBIC / FIFO & ALL RAM onboard Log any errors
Test SBIC/FIFO with fixed data pattern / complem Set/Rst RAM Used
Any failure detect.
Y
N
Turn off the LED Set RAM power-on key
Clear/Set RAM for operational use
Set up the Z80 default states
ASE init
Y
N
Set ASE ready status bit
Wait till init command completes
Set up the DMA bus on/off times In/Out mailboxes
Set the host ID Init. SBIC regs
Host Diagnostic Command
Y
N
Set ASC Init Flg
Return
ASC reset

Fig. B-2: ASC Reset/Diagnostic Flow

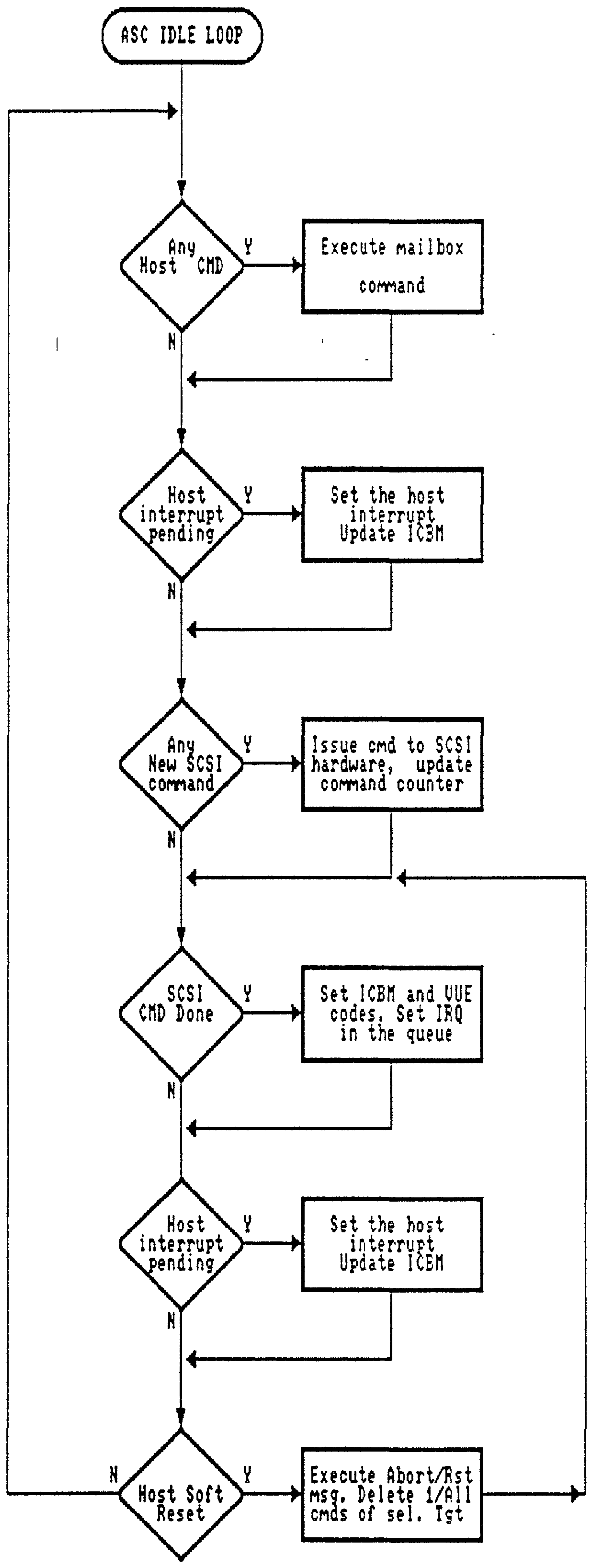

ASC IDLE LOOP
Any Host CMD
Y
N
Execute mailbox command
Host interrupt pending
Y
N
Set the host interrupt Update ICBM
Any New SCSI command
Y
N
Issue cmd to SCSI hardware, update command counter
SCSI CMD Done
Y
N
Set ICBM and VUE codes, Set IRQ in the queue
Host interrupt pending
Y
N
Set the host interrupt Update ICBM
N
Host Soft Reset
Y
Execute Abort/Rst msg. Delete 1/All cmds of sel. Tgt

Fig. B-3: ASC Idle Loop

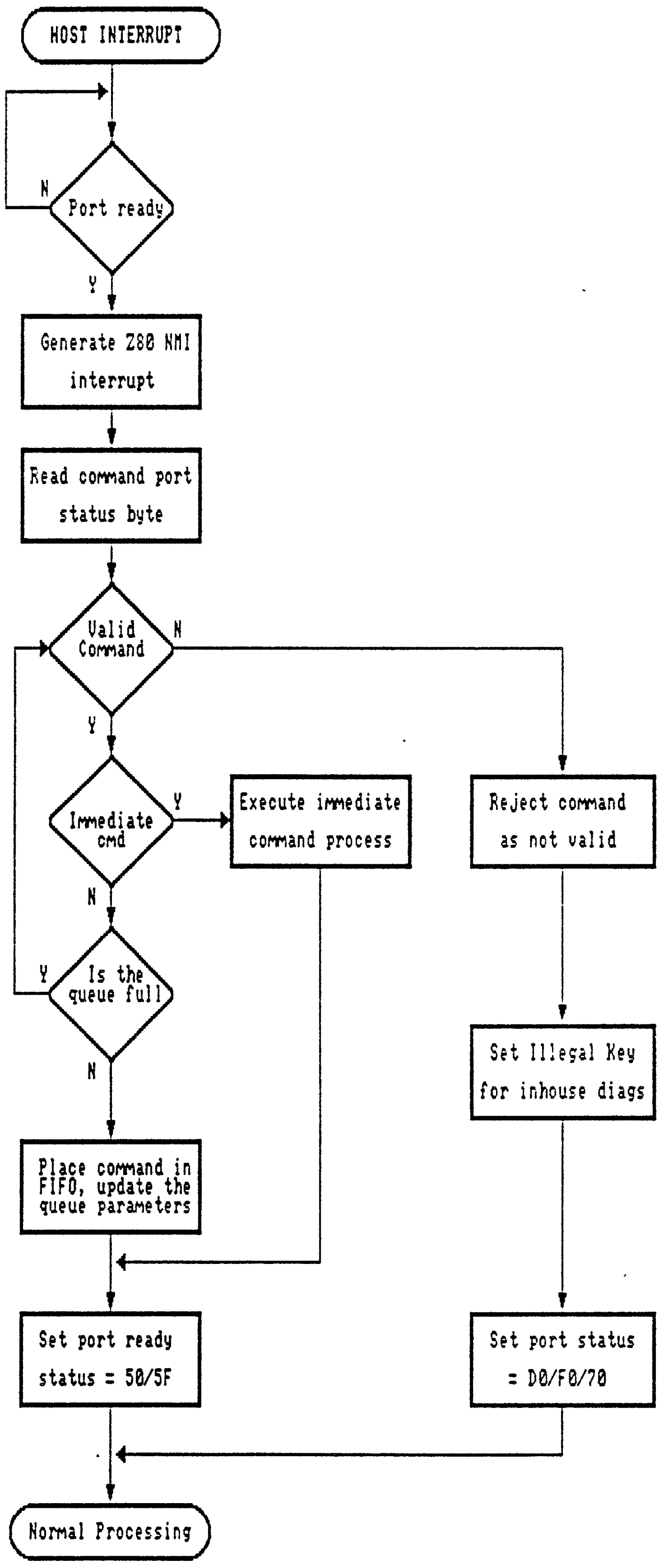

HOST INTERRUPT
Port ready
N
Y
Generate Z80 NMI interrupt
Read command port status byte
Valid Command
N
Y
Immediate cmd
Y
Execute immediate command process
Reject command as not valid
N
Is the queue full
Y
N
Set Illegal Key for inhouse diags
Place command in FIFO, update the queue parameters
Set port ready status = 50/5F
Set port status = D0/F0/70
Normal Processing

Fig. B-4: Host Command Interrupt

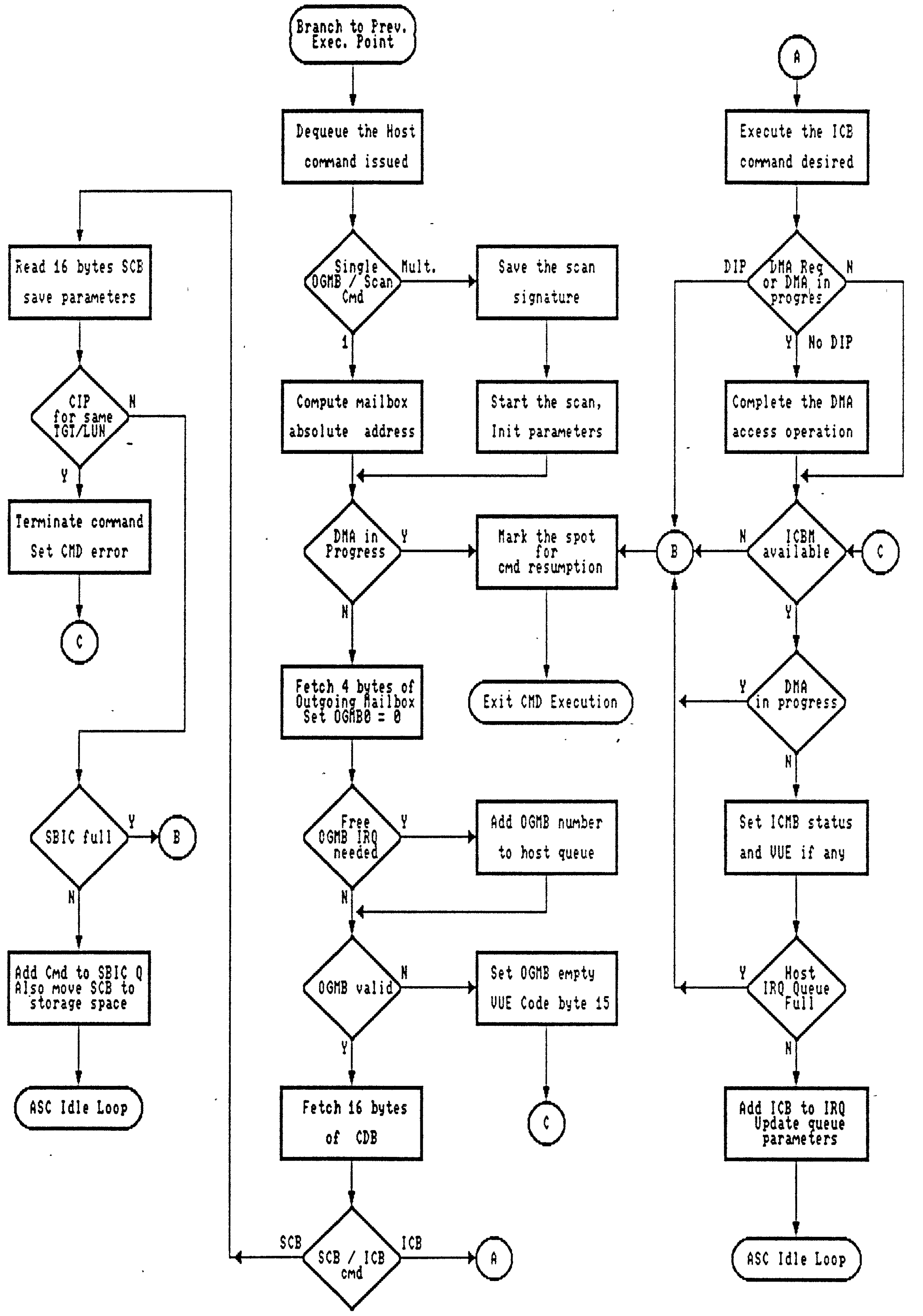
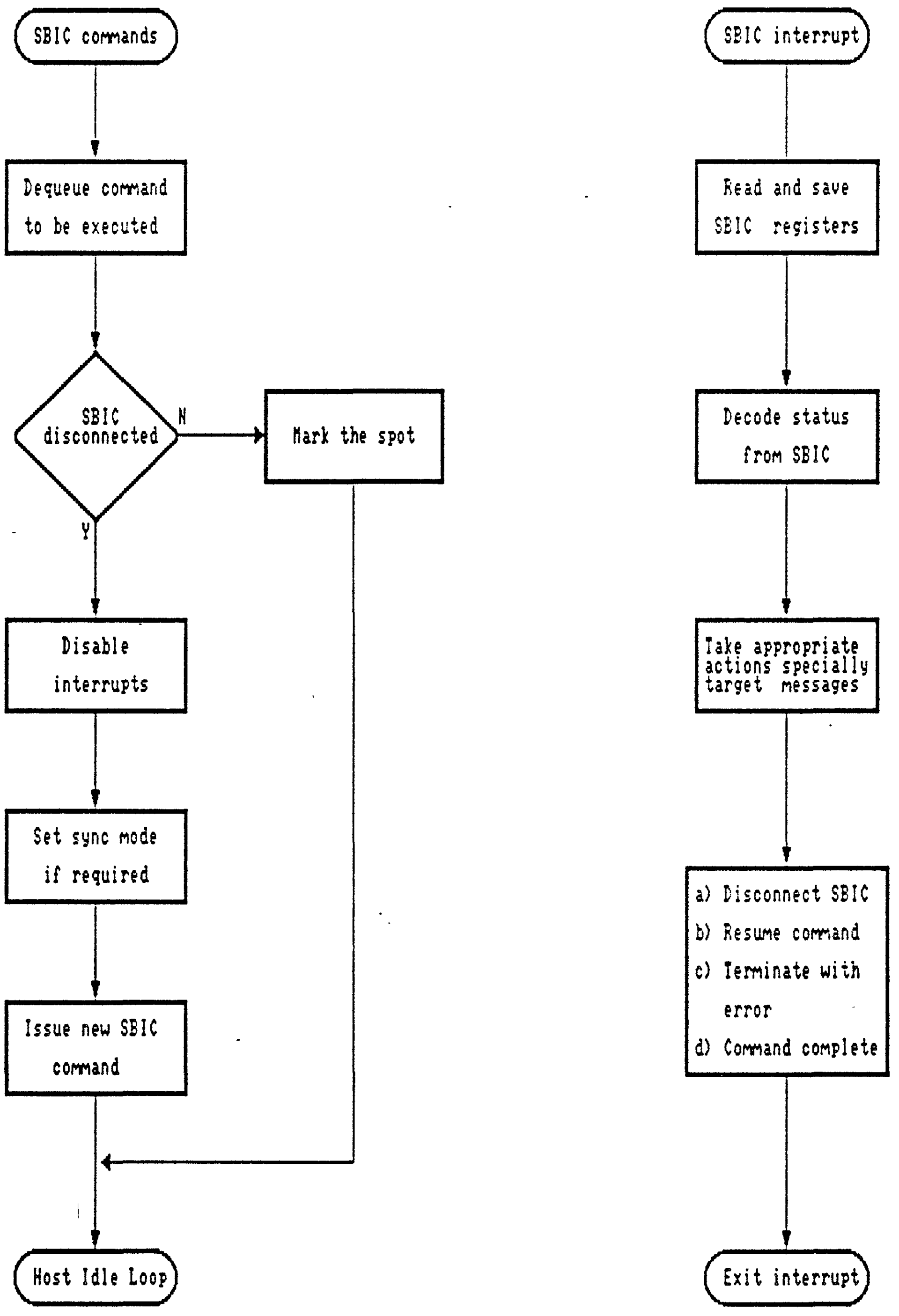

Branch to Prev. Exec. Point
Dequeue the Host command issued
Single OGMB / Scan Cmd
Mult.
1
Save the scan signature
Start the scan, Init parameters
Compute mailbox absolute address
DMA in Progress
Y
N
Mark the spot for cmd resumption
Exit CMD Execution
Fetch 4 bytes of Outgoing Mailbox Set OGMB0 = 0
Free OGMB IRQ needed
Add OGMB number to host queue
OGMB valid
Set OGMB empty VUE Code byte 15
Fetch 16 bytes of CDB
SCB / ICB cmd
SCB
ICB
Read 16 bytes SCB save parameters
CIP for same TGI/LUN
Terminate command Set CMD error
SBIC full
Add Cmd to SBIC Q Also move SCB to storage space
ASC Idle Loop
Execute the ICB command desired
DMA Req or DMA in progres
DIP
No DIP
Complete the DMA access operation
ICBM available
DMA in progress
Set ICMB status and VUE if any
Host IRQ Queue Full
Add ICB to IRQ Update queue parameters
ASC Idle Loop
A
B
C

Fig. B-5: HOST COMMAND EXECUTION SCB / ICB



(flow a) (flow b)

Fig. B-6: SCSI Command Execution Initiator mode

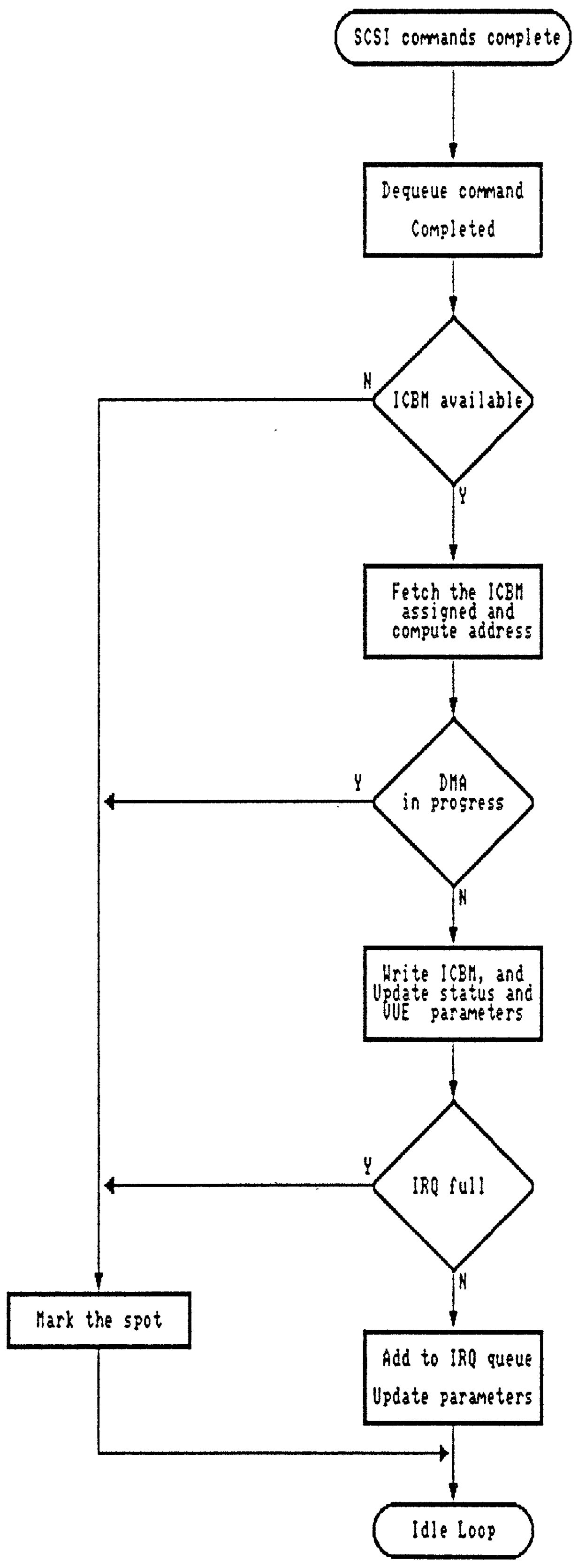

SCSI commands complete
Dequeue command Completed
ICBM available
N
Y
Fetch the ICBM assigned and compute address
DMA in progress
Y
N
Write ICBM, and Update status and QUE parameters
IRQ full
Y
N
Mark the spot
Add to IRQ queue Update parameters
Idle Loop

Fig. B-7: SCSI Command Completion

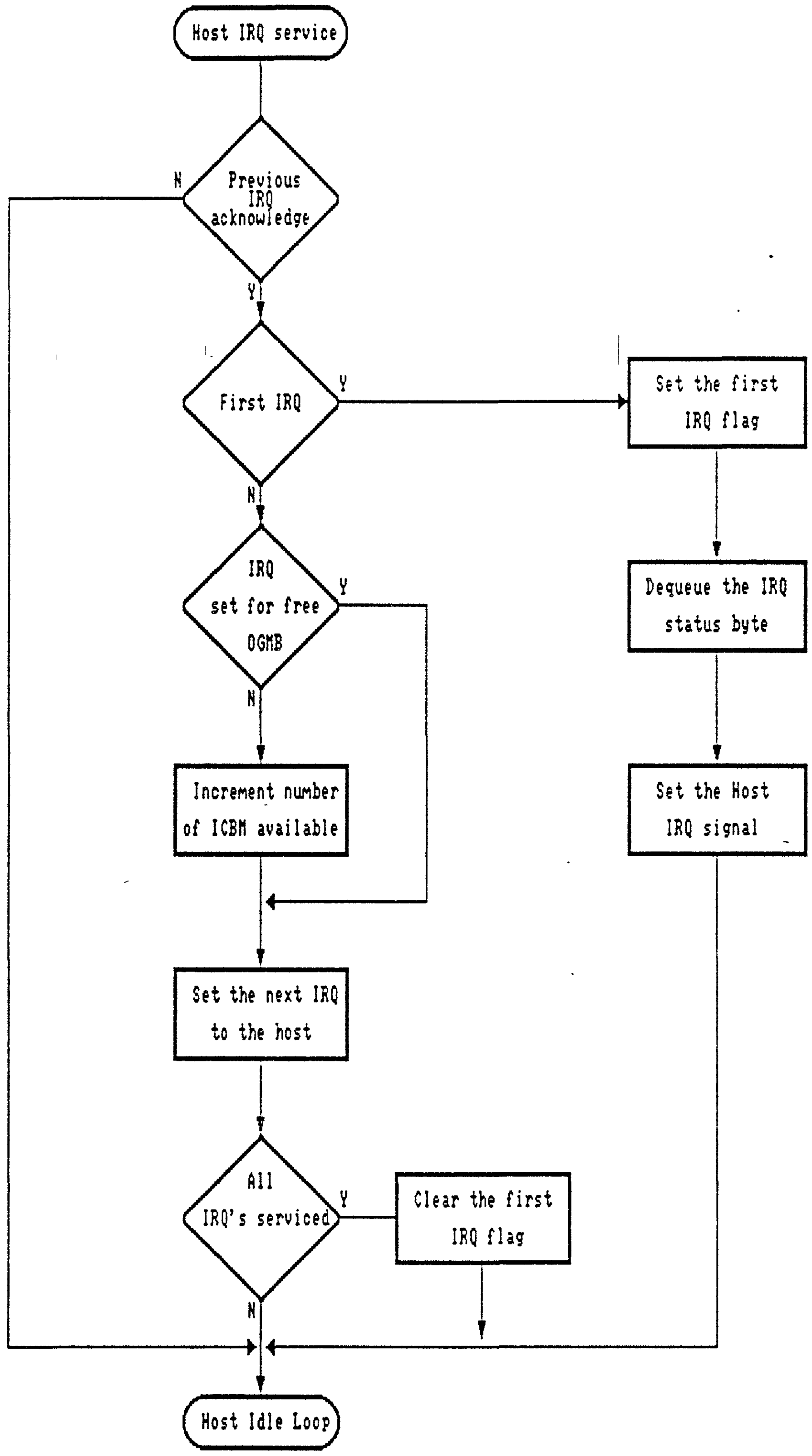


Fig. B-8: Host Interrupt Service Sequence

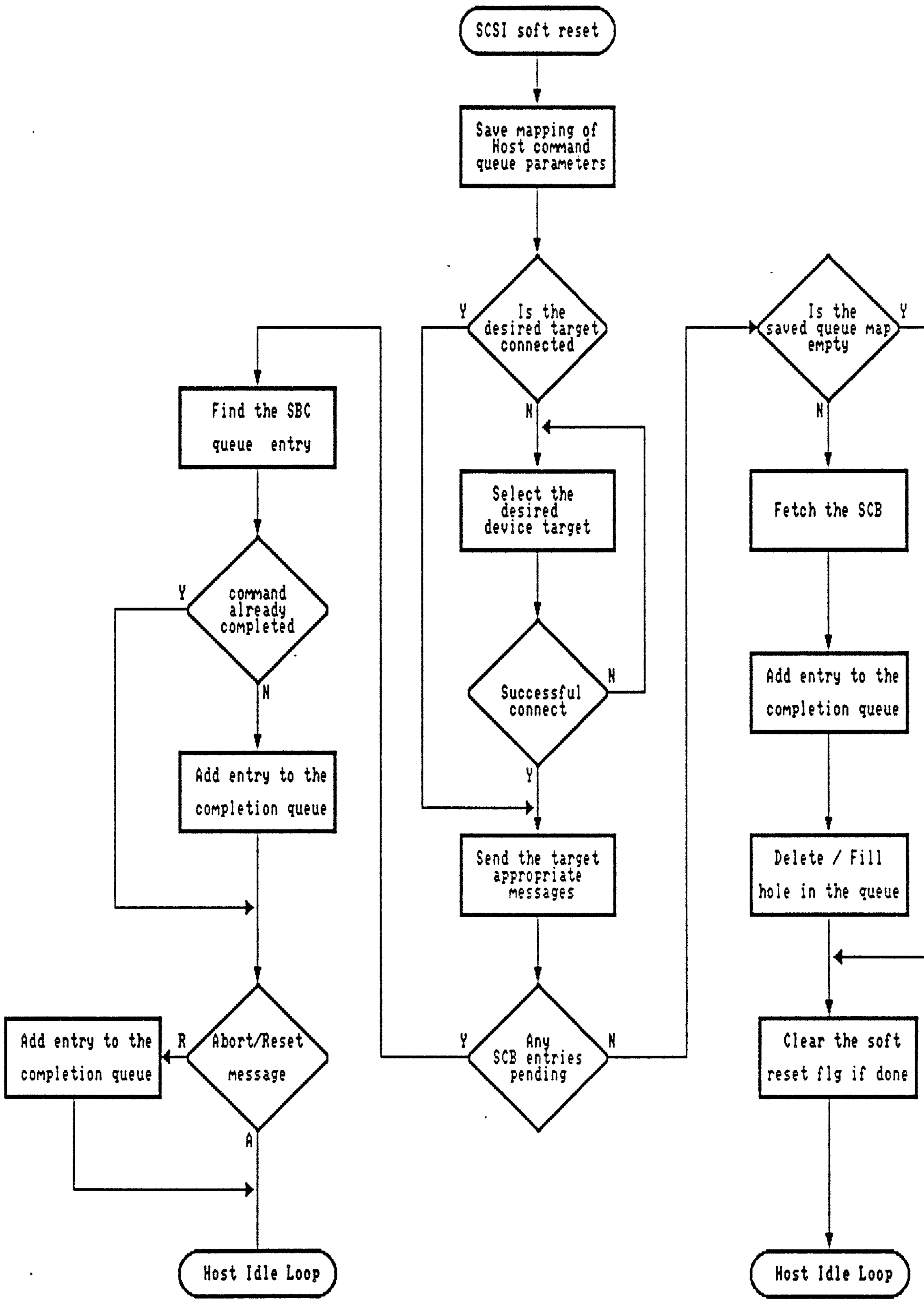

SCSI soft reset
Save mapping of Host command queue parameters
Is the desired target connected
Y
N
Select the desired device target
Successful connect
N
Y
Send the target appropriate messages
Any SCB entries pending
Y
N
Find the SBC queue entry
command already completed
Y
N
Add entry to the completion queue
Abort/Reset message
R
A
Add entry to the completion queue
Host Idle Loop
Is the saved queue map empty
Y
N
Fetch the SCB
Add entry to the completion queue
Delete / Fill hole in the queue
Clear the soft reset flg if done
Host Idle Loop

Fig. B-9: SCSI Soft Reset

APPENDIX C

C. WD7000-ASC Programming Notes

This document is intended to give WD7000-ASC Host Adapter programmers help solving a few of the more unusual programming problems associated with this device.

There are undoubtedly, other approaches to solving these problems. However, the methods presented here have been used successfully, and if used as a guide, may save the reader some time (and probably also some pain and suffering).

This document is directed toward the experienced programmer. The reader is assumed to have studied the available documentation for the 7000_ASC, and have some understanding of the function and capabilities of the host adapter and IBM PC/AT.

Program examples are shown at the end of this document.

PROBLEM #1 - First Party DMA

The 7000-ASC communicates with the host processor (AT) via DMA (Direct Memory Access) transfers. Unlike other IBM-PC DMA devices you may be familiar with, this device uses a "First Party" DMA method.

Memory addressing and transfer count are handled within the 7000-ASC, and not in the AT's on board DMA chip (8237). The AT's DMA chip (8237) does, however, handle the Processor/DMA arbitration. The I/O channel DACn and DRQn lines are tied to the AT DMA chip.

In order to use "First Party" DMA the AT DMA chip must have the selected channel placed in "Cascade" mode. This will disable all internal addressing and transfer count operation.

To Program cascade mode (see fig 1) set bits 6 and 7, along with the channel BCD representation (0 thru 3) in bits 0 and 1, of the 8237 Mode Register. Other bits in the Mode Register are not relevant and should remain 0.

```
+---+---+---+---+---+---+---+---+
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |    Mode Register
+---+---+---+---+---+---+---+---+
  |   |                   |   |
  |   |                   +---+----->  Channel Select
  +---+----------------------------->  Mode (1,1 = Cascade)
```

figure 1

The DRQn line on the I/O port is "high active", and will cause the AT DMA chip to arbitrate DMA transfers with the DRQn line open or in a "tri-state" condition. Since the 7000-ASC leaves this line in a "tri-state" condition whenever DMA is disabled (Host Control Register bit 2 not set), it is imperative that the DMA channel in the AT DMA chip be masked (disabled) before enabling or disabling the host DMA. Failure to do so is likely to cause the system to "hang".

Set the AT DMA chip channel mask by placing a 1 in bit 2 and the channel number BCD representation in bits 0 and 1 of the 8237 Mask Register (see fig 2). To enable the AT channel, clear the AT DMA chip mask by clearing bit 2 with the channel number in bits 0 and 1.

```
+---+---+---+---+---+---+---+---+
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     Mask Register
+---+---+---+---+---+---+---+---+
                      |   |   |
                      |   +---+-----> Channel number
                      +-------------> Set/Clear Mask (1/0)
```

figure 2

Unfortunately, it seems Intel has not thought prior conditions to be of importance, and made these registers "write-only". Since only DOS knows what conditions these were, and it won't tell, it is next to impossible to restore these registers upon exiting a routine.

It seems to be acceptable to simply set the DMA mask and leave it at that.

Listing 1 shows a Microsoft C implementation for enabling and disabling DMA operations.

PROBLEM # 2 - Interrupts

The 7000-ASC uses interrupts to signal command completions and other significant conditions. While the interrupt image is available at bit 7 of the ASC status port, most programs would not be efficiently served by polling this status. System interrupts are necessary for speed and efficiency.

The AT I/O channel brings interrupt requests 3 through 15 (IRQ3 thru IRQ15) on board through two 8259 Interrupt Controller chips. Since virtually all interrupts have been assigned by DOS, it is necessary to take one from an unused device or share the interrupt with a less used one.

As stated earlier, there are several ways to implement interrupt handling. This example uses DOS calls (INT 21h).

The current interrupt vector and interrupt mask status should be saved so they can be restored when your program terminates. First, retrieve the current vector using function 35h of DOS interrupt 21h (see listings 2 and 3 for an example). The 8259 Interrupt Mask register can then be read to save the mask bit for the interrupt line.

Next set the interrupt mask bit for that line in order to prevent un-wanted interrupts. Then, using function 25h of DOS interrupt 21h, set the new vector.

A word about programming languages is probably in order at this point. Many and varied facilities are available through different programming languages for the IBM PC and PC/AT. Most are extremely inadequate for interrupt handling. Some things to look for when evaluating a language for interrupt use are:

A) Interrupt Return.

The interrupt routine must send an EOI (bit 5, end of interrupt) to it's respective 8259 chip OCW (operation control word, address 0). This translates to outputting 20h to either port 20h or A0h.

An IRET instruction must be executed to return from interrupts. Use of the typical RET instruction will cause serious problems.

B) Stack.

High level languages generally have strict control over stack usage. Microsoft C, for example, does stack probes at each function entry. The problem arises when program control is passed to the interrupt handler with the stack pointer (SP) in an unknown state. A stack probe at this point is likely to cause a fatal error.

While stack probes can often be disabled, there is still the problem of pointing to a foreign stack area.

C) Segments.

High level language functions or subroutines usually make assumptions upon entry about the condition of segment registers. For example, DS, ES and SS may be assumed to point to the same segment. The routine may make use of this by blindly using these registers to access data or arguments - without setting these segments to a known state. (Microsoft C likes to point to an area named DGROUP. All functions assume this condition. This can be worked around by using an assembly language routine to set up the registers and call the C language interrupt handler. However, the handler can not call other functions or library routines.)

Listings 3 and 4 show an interrupt handler implemented in Microsoft C and assembly language. The assembly language routine "-asc_int" saves all registers, sets a local stack, points DS to DGROUP, and accesses the 7000-ASC interrupt status register. The C language interrupt handler "_int_hand" is then called and passed the interrupt status data.

Note that particular care is taken not to use library functions or call other routines in the interrupt handler. By pointing DS to DGROUP the interrupt handler has access to all global variables. Following a check of the interrupt status data, pertinent flags and switches are set accordingly.

"_int_hand" returns to the assembly language routine "_asc_int" where the 7000- ASC interrupt is acknowledged (write to Interrupt Ack port), EOI is sent to the 8259, all registers are restored and an IRET returns control to the interrupted program.

A program may have occasion to disable and enable the 7000-ASC interrupt (refer to listing 2). Like the DRQn line, the IRQn line is "high active" and will cause spurious interrupts when open or driven at a "tri-state" level (the condition of the 7000-ASC line when disabled). Mask the interrupt by setting the mask bit for the line in the 8259 OCW1 register (port 21h or A1h) before enabling or disabling the 7000-ASC interrupt (bit 3, Host Control port).

Listing 1

```
/*******************************************************/
/*                                                     */
/*            Enable/Disable 7000-ASC DMA              */
/*                                                     */
/*******************************************************/

/**********  Global Variables and  definitions  ***********/

#define EN_DMA          0x04            /* Control port enable DMA bit */
#define CASCADE         0xc0            /* 8237 cascade mode */
#define DMA_CH          0x02            /* DMA channel 6 (2 on this chip) */
#define S_DMA_MSK       0x04            /* set DMA mask bit */

extern int cont_img;                    /* ASC control port image */
extern int cont_p;                      /* ASC control port address (0x322)*/
extern int dma_mode_reg;                /* 8237 mode register address (0xd6)*/
extern int dma_mask_reg;                /* 8237 mask register address (0xd4)*/


en_dma()                /***  Enable DMA Subroutine  ***/
{

   cont_img |= EN_DMA;                  /* set enable ASC DMA image */
   outp (cont_p,cont_img);              /* enable ASC DMA before removing mask
   outp (dma_mode_reg,(DMA_CH | CASCADE));  /* disable native DMA controller *
   outp (dma_mask_reg,DMA_CH);          /* clear DMA channel mask */
   return;

}


dis_dma()               /***  Disable DMA Subroutine  ***/
{

   outp (dma_mask_reg,DMA_CH | S_DMA_MSK); /* set DMA channel mask first */
   cont_img &= ~EN_DMA;                 /* clear enable ASC DMA image */
   outp (cont_p,cont_img);              /* clear ASC DMA last */
   return;
}
```

Listing 2

```
/***********************************************/
/*                                             */
/*          Setup Interrupt Vector             */
/*                                             */
/*   This subroutine calls the assembly        */
/*   language subroutine _set_vec to do the    */
/*   actual vector setup.                      */
/*                                             */
/***********************************************/

vect()
{
   int a,b;

   a = (irq <= 7) ? (irq+8) : (irq+0x68);   /* determine int number value */
   b = (irq <= 7) ? 0x20 : 0xA0;            /* get 8259 OCW2 address */
   set_vec(a,b);                            /* call assembly lang. routine */
   return;
}


/***********************************************/
/*                                             */
/*          Enable/Disable Interrupts          */
/*                                             */
/***********************************************/

/****  Global Variables and Definitions  ****/

#define NO_INT          0                 /* interrupt flag value */
#define EN_INT          0x08              /* ASC enable interrupt bit */
#define PC_IMR          0x21              /* 8259 mask register address */
#define AT_IMR          0xA1              /* AT 8259 mask reg address */

extern int int_flg;                       /* interrupt flag */
extern int int_cnt;                       /* count for interrupts */
extern int cont_img;                      /* ASC control port image */
extern int cont_p;                        /* ASC control port address (0x322)*/
extern int irq;                           /* interrupt line used */
extern int imask[16];                     /* quick conversion for bit mask */
```

```
ei()                            /**  Enable Interrupts  **/
{
   int m;
   int imr_p;

   int_flg = NO_INT;                        /* start with clean flag */
   int_cnt = 0;                             /* init the interrupt count */
   imr_p = (irq<=7) ? PC_IMR : AT_IMR; /* find the right port */
   cont_img |= EN_INT;                      /* enable ASC interrupt */
   outp(cont_p,cont_img);
   m = (inp(imr_p)) & ~imask[irq];
   outp (imr_p,m);                          /* clear 8259 mask bit */
   return;
}


di()                            /**  Disable Interrupts  **/
{
   int m;
   int imr_p;

   imr_p = (irq<=7) ? PC_IMR : AT_IMR; /* find the right port */
   m = (inp(imr_p)) | imask[irq];
   outp (imr_p,m);                          /* set 8259 mask bit */
   cont_img &= ~EN_INT;                     /* disable ASC interrupt */
   outp (cont_p,cont_img);
   return;
}
```

Listing 3

```
/*******************************************************/
/*                                                     */
/*              ASC Interrupt Handler                  */
/*                                                     */
/*   The assembly language routine "_asc_int"          */
/*   intercepts the interrupt and passes control       */
/*   to this routine. Control is once more returned    */
/*   to "_asc_int" following the completion of this    */
/*   routine on order to do a proper interrupt return. */
/*                                                     */
/*******************************************************/

/*************************************************************************
       Caution: This program segment is for documentation
                purposes only, and is meant only to be repersentitive
                of a C language interrupt handler. Compilation of
                this routine requires additional files not shown.
*************************************************************************/


#pragma check_stack-           /* disable stack probes */

int_hand(istat)
int istat;
{
   int a,x,box;


   int_cnt++;                          /* log the interrupt */
   int_log[ilptr++] = istat;           /* log the interrupt status */
   if (istat & BIT7){                  /* see if command related */
      box = istat & BOX_MSK;           /* get the box pointer */
      if (istat & BIT6){               /* check for ICMB service request */
                                       /* service the incoming mailbox */
         switch (MB.IN[box][0]){       /* check the mailbox status */

            case CMD_DONE:
                  a = COMPLETE;        /* show task complete */
                  break;

            case CMD_ERR:
            case CMD_TO:
                  a = TSKERR;          /* show task error */
                  break;

            case CMD_ABORT:
            case SFT_RST:
                  a = ABORT;           /* show task error */
                  break;

            case HW_FAIL:
                  int_flg = HWFAIL_INT;        /* show task error */
                return;

            case SCAN_CMPL:
                if (scan_flg){                  /* see if scan in progress */
                   scan_flg--;                  /* reset the flag if set */
                   int_flg = INT_PROCESSED + box;    /* flag good interrupt
                }
                else{
                   int_flg = INVALID_INT + box;  /* show invalid interrupt *
                }
                return;

            case X_RSEL:
            case X_SEL:
            case BUS_RST:
            case HRD_RST:
                int_flg = INVALID_INT + box; /* show invalid interrupt */
                return;

            default:
                int_flg = INVALID_INT + box; /* show invalid interrupt */
                return;
         }
         for (x=0; x<TASK_MAX; x++){            /* .. and task pointer */
            if (tasks[x].abl == MB.IN[box][3]
                && tasks[x].abm == MB.IN[box][2]
                && tasks[x].abh == MB.IN[box][1]){
                break;
            }
         }
         if (x == TASK_MAX){                    /* interrupt error if no task *
            int_flg = INT_ERR + box;
            return;
         }
         tasks[x].cs = istat;                   /* save the command status */
         tasks[x].ib = box;                     /* show box used */
         if (tasks[x].ts != ACTIVE){
            int_flg = INT_ERR + box;   /* interrupt error if task not active *
            return;
         }
         tasks[x].ts = a;                       /* show new task status */
         int_flg = INT_PROCESSED + box;         /* and interrupt processed */
      }
      else{
         MB.BOXALL[box] = INACTIVE;             /* show mailbox free */
         ob_cnt++;                              /* bump the box count */
         int_flg = INT_PROCESSED + FREE_BOX;    /* valid interrupt processed */
      }
   }


   else{
      if (istat >= 1 && istat <= 7){            /* see if diag related */
         diag_stat = istat;                     /* set the diag status */
         int_flg = DIAG_INT + istat;
```

```
        }
        else{
           int_flg = UNDEFINED_INT + (istat & 0xff); /* show invalid interrupt *
        }
     }
     return;
}

#pragma check_stack+            /* restore stack probing */
```

Listing 4

```
;*******************************************************
;*                                                     *
;*              Assembly Language Routines             *
;*              for Interrupt Processing               *
;*                                                     *
;*******************************************************
;
;
;   PARAMETERS DEFINED IN C PROGRAM
;
       EXTRN    _int_hand:FAR            ;interrupt handling routine

       EXTRN    _istat_p:WORD            ;interrupt status port address
       EXTRN    _iack_p:WORD             ;interrupt ack port address
       EXTRN    _cont_p:WORD             ;control port address
       EXTRN    _cont_img:WORD           ;control port image

;
;

IH_TEXT         SEGMENT BYTE PUBLIC 'CODE'
IH_TEXT         ENDS
_DATA  SEGMENT WORD PUBLIC 'DATA'
_DATA  ENDS
CONST  SEGMENT WORD PUBLIC 'CONST'
CONST  ENDS
_BSS   SEGMENT WORD PUBLIC 'BSS'
_BSS   ENDS

DGROUP  GROUP   CONST, _BSS, _DATA
       ASSUME   CS:IH_TEXT, DS:DGROUP, SS:DGROUP, ES:DGROUP


IH_TEXT         SEGMENT
;
;   ENTRY POINT(s)
;

       PUBLIC   _set_vec
       PUBLIC   _rest_vec
       PUBLIC   _asc_int
```

```
    PASSED PARAMETER STRUCTURE

PRMS    STRUC
OLD_ES  DW      ?
OLD_DS  DW      ?
OLD_BP  DW      ?
RETN    DW      ?
RETSEG  DW      ?
ARG1    DW      ?
ARG2    DW      ?
ARG3    DW      ?
ARG4    DW      ?
ARG5    DW      ?
PRMS    ENDS
;
;

STK:           DW      128 DUP (?)               ;local stack
STKTOP:        DW      ?


VSTOR1:        DW      0                 ;vector seg storage
VSTOR2:        DW      0                 ;vector offset storage
VSTOR3:        DW      0                 ;interrupt number storage
VSTOR4:        DW      0                 ;8259 OCW2 address storage

;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;       Setup the 7000-ASC interrupt.
;
;       Call:
;               set_vec ( intno,ocw2 );
;
;               int intno;      /* interrupt number */
;               int ocw2;       /* 8259 OCW2 address */
;

_set_vec PROC  FAR                       ;far for large memory model
       PUSH    BP                        ;save regs in proper order
       PUSH    ES
       PUSH    DS
       MOV     BP,SP
       MOV     AH,35H                    ;get the old int vector
       MOV     AL,0                      ;... with DOS function 35
       INT     21H                       ;... DOS interrupt 21
       MOV     WORD PTR VSTOR1,ES        ;save old vector for later
       MOV     WORD PTR VSTOR2,BX
       MOV     BX,[BP].ARG2              ;get OCW2 address
       MOV     WORD PTR VSTOR4,BX        ;save it
       MOV     BX,[BP].ARG1              ;get the int number
       MOV     WORD PTR VSTOR3,BX        ;save it too
       MOV     DX,OFFSET _asc_int        ;get new handler address
       MOV     AX,SEG _asc_int           ;... and segment
       MOV     DS,AX
       MOV     AH,25H                          ;set up vector with DOS function 25
       MOV     AL,BL
       INT     21H                             ;...DOS interrupt 21
       POP     DS                              ;restore the regs
       POP     ES
       POP     BP
       RET                                     ;exit
_set_vec ENDP

;----------------------------------------------------------------
       PAGE
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;      Restore the old interrupt.
;
;      Call:
;              rest_vec ();
;
;

_rest_vec PROC FAR
       PUSH    BP                              ;save regs in proper order
       PUSH    ES
       PUSH    DS
       MOV     BX,WORD PTR VSTOR3              ;get old int no
       MOV     DX,WORD PTR VSTOR2              ;get old handler address
       MOV     AX,WORD PTR VSTOR1              ;... and segment
       MOV     DS,AX
       MOV     AH,25H                          ;set up vector function
       MOV     AL,BL                           ;point to vector no
       INT     21H                             ;set old vector thru DOS
       POP     DS                              ;restore the regs
       POP     ES
       POP     BP
       RET                                     ;exit
_rest_vec ENDP

;----------------------------------------------------------------
       PAGE
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;      7000-ASC Interrupt Handler
;
;      The ASC interrupt is intercepted by this routine.
;      Control is passed to the C language routine "int_hand".
;
;

_asc_int PROC  FAR
       PUSH    AX                              ;AX to system stack
       PUSH    BX                              ;BX to system stack
       PUSH    CX
       PUSH    DX
       PUSH    BP                              ;BP to system stack
       PUSH    DI                              ;save registers
```

To: Distribution

From: Chandru Sippy

Re: WD7000-ASC Target Mode : Ref Doc 96-000494 Rev X3

Date: 8-12-88

The WD7000-ASC Engineering Specification has been updated to include all of the target mode commands. As of the present writing the following operations do not function:

1. Target Synchronous mode of operation
2. Receive Command (81)
3. Set Bus On/Off times (91)
4. Open Outbound Data Buffer (90)

It is expected that items 3 and 4 will be completed by 8-16-88 and both 1 and 2 by around 9-2-88 depending upon the severity of the debugging problems encountered.

Any corrections in the specifications can be reported directly to me and these will be reflected in the next release.

Sincerely,

Chandru M. Sippy