ADAPTEC, INC.

CONTROLLER/SYSTEM INTERFACE SPECIFICATION

- SCSI -

8/7/82

ADAPTEC CONTROLLER/SYSTEM INTERFACE SPECIFICATION

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

PART C: FUNCTIONAL SPECIFICATION

TABLE OF CONTENTS (Continued)

FIGURES AND TABLES

ADAPTEC CONTROLLER/SYSTEM INTERFACE SPECIFICATION

INTRODUCTION

This specification describes a Local I/O Bus Controller Interface which can be operated asynchronously at data rates of up to 1.5 megabytes per second.

This interface is designed to provide host computers with device independence such that, within a class of peripherals (e.g. disk drives, tape drives, communication devices, or printers), devices can be interchanged freely without requiring modifications to the host system hardware or software. Additionally, this interface allows the compatible operation of hosts and devices with a wide range of capabilities by utilizing "logical" rather than "physical" addressing for all data structures. Blocks are addressed logically (up to the maximum number of blocks in a device) and each device can be interrogated to determine the number of blocks it contains.

The controller interface protocol provides for the connection of multiple INITIATORS (bus devices capable of initiating an operation) and TARGETS (bus devices capable of responding to an operational request). Arbitration (i.e., bus-contention utilizing a logical priority system) is included in the architecture of this interface.

Although the architecture is more general, this specification addresses rotating memory devices (disk drives) in particular and is presented in three parts: the Physical Specification, the Message Specification, and the Functional Specification.

Part A, the Physical Specification, defines the physical components of the interface (including the function of each line) and contains all of the specifications associated with obtaining and maintaining control of the bus.

Part B, the Message Specification, defines the message protocol which links the bus devices to each other and controls the physical path between them.

Part C, the Functional Specification, defines the operation of the interface in detail (which in turn defines the operation of the bus devices -- controllers or hosts).

ADAPTEC, INC.

CONTROLLER-SYSTEM INTERFACE SPECIFICATION

PART A

PHYSICAL SPECIFICATION

5/3/82

1. GENERAL DESCRIPTION

This system interface provides an efficient method of communication between computers and peripheral I/O devices. The eight-port, daisey-chained bus defined by this specification supports the following features:

* Single or multiple host system.
* Multiple peripheral devices and device types.
* Bus contention resolution through arbitration on a prioritized basis.
* Asynchronous data transfer at up to 1.5 MBytes/sec.
* Disconnected operations.
* Host-to-host communication.

Communication on the bus is allowed between two bus ports at a time. A maximum of eight (8) bus ports are allowed. Each port is attached to a device (e.g. controller or host adapter).

When two devices communicate with each other on the bus, one acts as an INITIATOR and the other acts as a TARGET. The TARGET (typically a controller) executes the operation. A device will usually have a fixed role as an INITIATOR or TARGET, but some devices may be able to assume either role.

An INITIATOR may address up to eight (8) peripheral I/O devices that are connected to a TARGET. The TARGET will provide a "virtual controller" for each of these devices, appearing to the system as up to eight separate controller/device pairs.

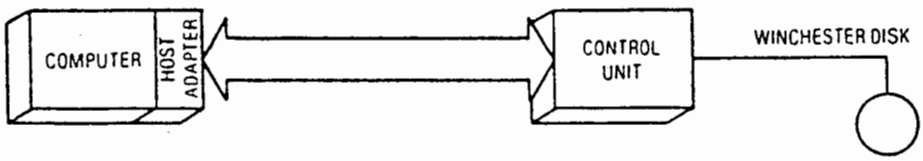
Certain bus functions are assigned to the INITIATOR and certain bus functions are assigned to the TARGET. The INITIATOR may arbitrate for the bus and select a particular TARGET. The TARGET may request the transfer of COMMAND, DATA, STATUS or other information on the bus, and in some cases, may arbitrate for the bus and reselect an INITIATOR for the purpose of continuing an operation.

Data transfers on the bus are asynchronous and follow a defined REQUEST/ACKNOWLEDGE HANDSHAKE protocol. One eight-bit byte of information may be transferred with each handshake.

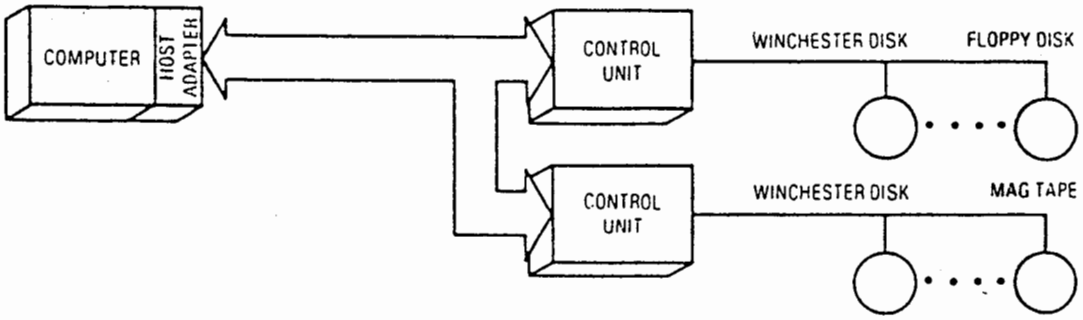
To realize the full capability of this physical interface, a level of logical path control is required. This level, termed the Message Level, is described in Part B of this specification. Additionally, the practical application of this Physical Specification requires a higher level software command set as described in Part C.

Figure 1: SAMPLE SYSTEM CONFIGURATIONS

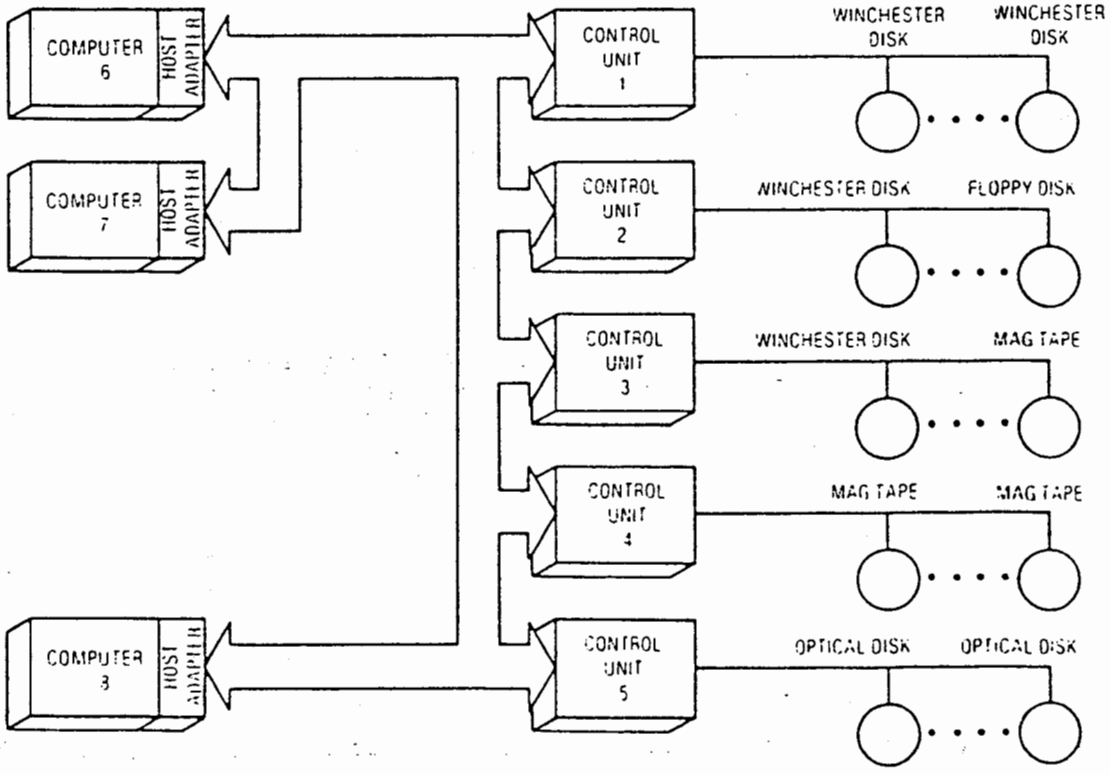
SIMPLE SYSTEM



BASIC TWO-CONTROL UNIT SYSTEM



COMPLEX SYSTEM

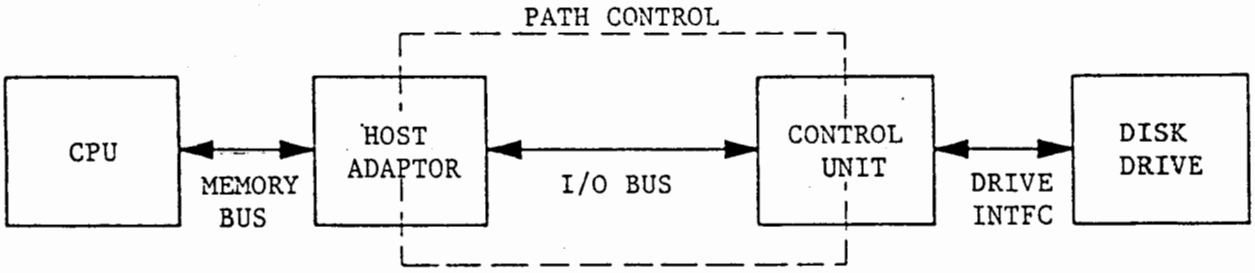


Up to 8 devices can be supported by the CONTROLLER/SYSTEM INTERFACE bus. The devices can be any combination of host CPU's and intelligent controllers.

2. PHYSICAL PATH FUNCTIONS

Figure 2 shows an INITIATOR and TARGET communicating on the bus in order to execute a command such as READ or WRITE data. For the sake of simplicity, only one of a number of possible partitions (of the physical/functional interface) is presented for illustration.

Figure 2: PHYSICAL PATH



2.1. POINTERS

In this architecture, three "conceptual" memory address pointers reside in the INTITIATOR path control. They point to the next byte of COMMAND, DATA and STATUS to be transferred.

After the pointers are initially loaded by the INITIATOR, their movement is under the strict control of the TARGET. When the TARGET transfers a byte of information to or from a (COMMAND, DATA, or STATUS) memory area, the corresponding pointer is incremented.

2.2. TYPICAL FUNCTIONS (External to Path Control)

Listed below are some typical functions that affect the physical path but originate outside its boundary. Although these "commands" rarely pass across such boundaries in practice, they aid in describing the actual sequence of events within the I/O subsystem.

2.2.1. Establish Path

This function enables the INITIATOR to establish the physical path (the physical and logical connection between the INITIATOR and a peripheral device) in order to execute a command and may involve arbitration to gain control of the bus.

ESTABLISH PATH requires the peripheral device address (i.e., the TARGET bus address and LUN within that address) and the three pointers to the COMMAND, DATA and STATUS areas. A saved copy of these pointers may also be required (see 2.2.8. Reestablish Path and 2.2.11. Restore State).

2.2.2. Get Command

This function enables a TARGET to get a COMMAND from the memory area designated by the COMMAND pointer.

2.2.3. Get Data and Send Data

These functions enable the TARGET to transfer data to or from the memory area designated by the DATA pointer.

2.2.4. Send Status

This function enables the TARGET to send STATUS information for a command to the memory area designated by the STATUS pointer.

2.2.5. End of Command

This function enables the TARGET to signal the INITIATOR that the current command has terminated and valid status has been sent.

Since the current command may be linked to another command, END OF COMMAND does not imply the end of an operation.

2.2.6. End Path

The TARGET invokes this function to enable the INITIATOR to clear the physical path to the currently attached peripheral device.

END PATH implies the end of an operation.

2.2.7. Break Path

This function enables the TARGET to temporarily break the physical path and release control of the bus.

2.2.8. Reestablish Path

This function enables the TARGET to reconnect a physical path that was temporarily broken by the "Break Path" function.

The INITIATOR address and the peripheral device address are required. Bus arbitration is also required.

The INITIATOR must restore the COMMAND, DATA and STATUS pointers to their last saved values.

2.2.9. End of Link

This function is invoked by the TARGET to indicate the termination of the current command (because the current command was linked to another command, the physical path connection is still needed).

2.2.10. Save Data Pointer

The TARGET invokes this function to enable the INITIATOR to save a copy of the current Data pointer.

2.2.11. Restore Pointers

The TARGET invokes this function to enable the INITIATOR to load the current (active) COMMAND, DATA and STATUS pointers with the last saved values.

3. BUS SIGNALS

The 9 control signals and 9 data signals (including parity), are described below:

3.1. BUSY (BSY)

BSY is an "or-tied" signal which indicates that the bus is in use.

3.2. SELECT (SEL)

SEL is an "or-tied" signal used by an INITIATOR to select a TARGET or by a TARGET to reselect an INITIATOR.

3.3. CONTROL/DATA (C/D)

C/D is a TARGET-driven signal to indicate whether CONTROL or DATA information is on the data bus. Assertion indicates CONTROL.

3.4. INPUT/OUTPUT (I/O)

I/O is a TARGET-driven signal which controls the direction of data movement on the data bus relative to an INITIATOR. Assertion indicates INPUT to the INITIATOR.

3.5. MESSAGE (MSG)

MSG is a TARGET-driven signal indicating the MESSAGE phase.

3.6. REQUEST (REQ)

REQ is a TARGET-driven signal indicating a request for a REQ/ACK data transfer handshake.

3.7. ACKNOWLEDGE (ACK)

ACK is an INITIATOR-driven signal indicating acknowledgment of a REQ/ACK data transfer handshake.

3.8. ATTENTION (ATN)

ATN is an INITIATOR-driven signal indicating the ATTENTION condition.

3.9. RESET (RST)

RST is an "or-tied" signal indicating the RESET condition.

3.10. DATA BUS (DB: 7-0,P)

Eight data bit signals, plus a parity bit signal comprise the DATA BUS. DB(7) is the most significant bit and has the highest priority during arbitration. Significance and priority decrease with decreasing bit number.

Data parity DB(P) is odd. The use of parity is a system option (i.e., either all devices on the bus generate parity and have parity detection enabled, or all devices have parity detection disabled). Parity is not valid during arbitration.

Each of the eight data signals DB(7) through DB(0) is uniquely assigned as a TARGET or INITIATOR bus address (i.e., DEVICE I.D.) which is normally assigned and "strapped" in the device during system configuration. In order to obtain the bus during arbitration, a device asserts its assigned data bit (DEVICE I.D.) and leaves the other data bits in the passive (non-driven) state.

4. BUS PHASES

The bus has eight (8) distinct operational phases and cannot be in more than one phase simultaneously.

4.1. BUS FREE PHASE

The BUS FREE phase, indicating that the bus is available for use, is invoked by the deassertion and passive release of all bus signals. All active devices must deassert and passively release all bus signals (within a BUS CLEAR DELAY) after deassertion of BSY and SEL.

Devices sense BUS FREE when both SEL and BSY are not asserted (simulataneously within a DESKEW DELAY) and the RESET condition is not active.

4.2. ARBITRATION PHASE

The ARBITRATION phase, a system option, enables a device to gain control of the bus (systems without ARBITRATION can have only one INITIATOR). This phase is required for systems which use RESELECTION. A TARGET must not disconnect from an INITIATOR that does not support these functions.

After detecting BUS FREE, a device must wait a minimum of BUS FREE DELAY and a maximum of BUS SET DELAY to assert BSY and its own DEVICE I.D. on the bus. The time required to detect the BUS FREE phase is included in the wait delay.

The DEVICE I.D. is asserted on the DATA BIT signal that corresponds to the BUS ADDRESS for the device. All other DATA BUS drivers must be passive. Data parity is not valid during arbitration.

On detecting SEL, a device must clear itself from arbitration (within a BUS CLEAR delay time) by deasserting its BSY and I.D. signals.

After an ARBITRATION DELAY (timed from the assertion of BSY) the device examines the DATA bus. If a higher priority DEVICE I.D. is on the bus (DB(7) = highest), the device clears itself from arbitration. On obtaining the bus, the device asserts SEL (after the assertion of SEL the device must wait a minimum of two BUS SETTLE DELAYS before changing any bus signals).

4.3. SELECTION PHASE

The SELECTION phase allows an INITIATOR to select a TARGET. In order to distinguish this phase from the RESELECTION phase, the I/O signal is not asserted.

In systems without arbitration, the INITIATOR waits a minimum of BUS SETTLE DELAY (after detecting BUS FREE) before driving the DATA bus with the TARGET I.D. and (optionally) its own I.D. After two DESKEW DELAYS, the INITIATOR can assert SEL.

In systems with arbitration, the BSY and SEL signals will have been asserted by the INITIATOR following arbitration. After a minimum of two BUS SETTLE DELAYS, the INITIATOR can assert the TARGET I.D. and its own I.D. on the DATA bus. The INITIATOR then waits at least two DESKEW DELAYS before de-asserting BSY and a BUS SETTLE DELAY before examining the bus for a TARGET response.

4.3. SELECTION PHASE (Continued)

On detecting the simultaneous condition (within one DESKEW DELAY) of SEL, its own I.D. asserted, BSY not asserted, and I/O not asserted, the selected TARGET examines the DATA bus for the INITIATOR I.D. and responds by asserting BSY. In systems with parity implemented, the TARGET will not respond to its DEVICE I.D. if an error is indicated or if more than two I.D.'s are on the bus.

After a minimum of two DESKEW DELAYS (following the detection of BSY from the TARGET), the INITIATOR deasserts SEL and may change the DATA signals.

The INITIATOR may "time out" the SELECTION phase by deasserting the I.D. bits on the bus. If after 100 microseconds, BSY has not been asserted, SEL may be deasserted. The TARGET must drive BSY within 90 microseconds of detecting SEL and its own I.D.

4.4. RESELECTION PHASE

The RESELECTION phase allows a TARGET to reconnect to an INITIATOR to continue an operation that was previously interrupted by the TARGET (i.e., the TARGET disconnected by allowing a BUS FREE phase to occur before the operation was completed).

RESELECTION (used only in systems with ARBITRATION) is a TARGET function which requires a query to the INITIATOR to determine its capability.

On obtaining the bus, the TARGET will assert BSY, assert SEL, wait a minimum of two BUS SETTLE DELAYS, then assert I/O together with the INITIATOR I.D. and its own I.D. The TARGET then waits a minimum of two DESKEW DELAYS and deasserts BSY. The TARGET then waits a BUS SETTLE DELAY before examining the bus for an INITIATOR response.

On detecting the simultaneous condition (within a DESKEW DELAY) of SEL, I/O, its own I.D. asserted, and BSY not asserted, the reselected INITIATOR samples the DATA BUS to determine the TARGET I.D. and responds by asserting BSY.

In systems with parity implemented, the INITIATOR will not respond to a DEVICE I.D. that has bad parity; nor will it respond if more than two I.D.'s are on the bus.

On detecting BSY from the INITIATOR, the TARGET will also assert BSY (and continue the assertion for the duration of the operation), wait a minimum of two DESKEW DELAYS, then deassert SEL and (possibly) change the I/O and DATA signals.

On detecting the deassertion of SEL, the INITIATOR releases its assertion of BSY.

4.5. INFORMATION TRANSFER PHASES

The COMMAND, DATA, STATUS and MESSAGE phases are all used to transfer data or control information through the DATA bus. The actual contents of the information is beyond the scope of this section.

The C/D, I/O and MSG signals are used to differentiate the various INFORMATION TRANSFER phases. See Table 1.

TABLE 1: INFORMATION TRANSFER PHASES

SIGNAL				DIRECTION OF
MSG	C/D	I/O	PHASE NAME	INFORMATION XFER
Ø	Ø	Ø	DATA OUT PHASE	(INIT to TARG)
Ø	Ø	1	DATA IN PHASE	(INIT from TARG)
Ø	1	Ø	COMMAND PHASE	(INIT to TARG)
Ø	1	1	STATUS PHASE	(INIT from TARG)
1	Ø	Ø	* Not Used	
1	Ø	1	* Not Used	
1	1	Ø	MSG OUT PHASE	(INIT to TARG)
1	1	1	MSG IN PHASE	(INIT from TARG)

Notes: Ø = SIGNAL DEASSERTION
1 = SIGNAL ASSERTION
INIT = INITIATOR
TARG = TARGET

The INFORMATION TRANSFER phases use the REQ/ACK handshake to control data transfer. Each REQ/ACK allows the transfer of one byte of data. The handshake starts with the TARGET asserting the REQ signal. The INITIATOR responds by asserting the ACK signal. The TARGET then deasserts the REQ signal and the INITIATOR responds by deasserting the ACK signal.

With I/O signal asserted, data will be input to the INITIATOR from the TARGET. The TARGET must ensure that valid data is available on the bus (at the INITIATOR port) before the assertion of REQ at the INITIATOR port. The data remains valid until the assertion of ACK by the INITIATOR. The TARGET should compensate for cable skew and the skew of its own drivers.

4.5. INFORMATION TRANSFER PHASES (Continued)

With the I/O signal not asserted, data will be output from the INITIATOR to the TARGET. The INITIATOR must ensure valid data on the bus (at the TARGET port) before the assertion of ACK on the bus. The INITIATOR should compensate for cable skew and the skew of its own drivers. Valid data remains on the bus until the TARGET deasserts REQ.

During each INFORMATION TRANSFER phase, the BSY line remains asserted, the SEL line remains deasserted, and the TARGET will continuously envelop the REQ/ACK handshake(s) with the C/D, I/O and MSG signals in such a manner that these control signals are valid for a BUS SETTLE DELAY before the REQ of the first handshake and remain valid until the deassertion of ACK at the end of the last handshake.

4.5.1. COMMAND PHASE

The COMMAND phase allows the TARGET to obtain command information from the INITIATOR.

The TARGET asserts the C/D signal and deasserts the I/O and MSG signals during the REQ/ACK handshake(s) of this phase.

4.5.2. DATA PHASES (DATA IN/DATA OUT)

The DATA phase includes both the DATA IN phase and the DATA OUT phase.

The DATA IN phase allows the TARGET to INPUT data to the INITIATOR. The TARGET asserts the I/O signal and deasserts the C/D and MSG signals during the REQ/ACK handshake(s) of this phase.

The DATA OUT phase allows the TARGET to obtain OUTPUT data from the INITIATOR. The TARGET deasserts the C/D, I/O and MSG signals during the REQ/ACK handshake(s) of this phase.

4.5.3. STATUS PHASE

The STATUS phase allows the TARGET to send status information to the INITIATOR.

The TARGET asserts C/D and I/O and it deasserts the MSG signal during the REQ/ACK handshake(s) of this phase.

4.5.4. MESSAGE PHASES (MESSAGE IN/MESSAGE OUT)

The MESSAGE phase includes the MESSAGE IN and MESSAGE OUT phases.

The MESSAGE IN phase allows the TARGET to INPUT a message to the INITIATOR. The TARGET asserts C/D, I/O and MSG during the REQ/ACK handshake(s) of this phase.

The MESSAGE OUT phase allows the TARGET to obtain a message from the INITIATOR. The TARGET may invoke this phase only in response to the ATTENTION condition created by the INITIATOR. In response to the ATTENTION condition, the TARGET asserts C/D and MSG and deasserts the I/O signal during the REQ/ACK handshake(s) of this phase. See 5.1. ATTENTION.

4.6. SIGNAL RESTRICTIONS BETWEEN PHASES

When the BUS is between phases, the following restrictions apply to the bus signals:

The BSY, SEL, REQ and ACK signals may not change.

The C/D, I/O, MSG and DATA signals may change.

The ATN and RST signals may change as defined under the descriptions for the ATTENTION and RESET conditions.

5. BUS CONDITIONS

The bus has two asynchronous conditions: the ATTENTION Condition and and the RESET Condition. These conditions cause certain BUS DEVICE actions and can alter the bus phase sequence.

5.1. ATTENTION CONDITION

ATTENTION allows the INITIATOR to signal the TARGET of a waiting MESSAGE. The TARGET may access the messsage by invoking a MESSAGE OUT phase.

The INITIATOR creates the ATTENTION condition by asserting ATN at any time except during the ARBITRATION or BUS FREE phases.

The TARGET responds when ready with the MESSAGE OUT phase.

The INITIATOR keeps ATN asserted if more than one byte is to be transferred.

The INITIATOR can deassert the ATN signal during the RESET condition, during a BUS FREE phase, or while the REQ signal is asserted and before the ACK signal is asserted during the last REQ/ACK handshake of a MESSAGE OUT phase.

5.2. RESET CONDITION

The RESET condition, created by the assertion of RST, is used to immediately clear all devices from the bus and to reset these devices and their associated equipment as defined in the controller specification.

RESET can occur at any time and takes precedence over all other phases and conditions. Any device (whether active or not) can invoke the RESET condition. On RESET, all devices will immediately (within a BUS CLEAR DELAY) deassert and passively release all bus signals except RST itself. TARGETS capable of continuing an I/O operation after being interrupted by RESET will clear any I/O operation that has not been established (see IDENTIFY for clarification).

The RESET condition stays on for at least one RESET HOLD TIME. During the RESET condition, no bus signal except RST can be assumed valid.

Regardless of the prior bus phase, the bus resets to a BUS FREE phase (and then starts a normal phase sequence) following a RESET condition.

6. PHASE SEQUENCING

Phases are used on the bus in a prescribed sequence. In all systems, the RESET condition can interrupt any phase and is always followed by the BUS FREE phase. (Any other phase can also be followed by the BUS FREE phase.)

In systems without ARBITRATION, the normal progression is from BUS FREE to SELECTION, and from SELECTION to one or more of the INFORMATION TRANSFER phases (COMMAND, DATA, STATUS or MESSAGE). In systems with ARBITRATION, the normal progression is from BUS FREE to ARBITRATION, from ARBITRATION to SELECTION (or RESELECTION), and from SELECTION (or RESELECTION) to one or more of the INFORMATION TRANSFER phases (COMMAND, DATA, STATUS or MESSAGE).

There are no restrictions on the sequencing between INFORMATION TRANSFER phases. A phase may even follow itself (e.g., a DATA phase may be followed by another DATA phase).

7. TIMING

A timing chart is provided in Figure 3. Unless otherwise indicated, the delay time measurements for each device are calculated from signal conditions existing at the device BUS PORT. Delays in the bus cable need not be considered for these measurements.

7.1. ARBITRATION DELAY: 1.7 microseconds (minimum)

The minimum delay required after asserting BSY for arbitration until the data bus can be examined for the result of arbitration. No maximum time.

7.2. BUS CLEAR DELAY: 350 nanoseconds (maximum)

The maximum time allowed for a device to stop driving all bus signals after: (1) the release of BSY when going to BUS FREE, or (2) another device asserts SEL during ARBITRATION.

7.3. BUS FREE DELAY: 100 nanoseconds (minimum)

The minimum delay required between detection of BUS FREE and assertion of BSY during ARBITRATION.

7.4. BUS SETTLE DELAY: 450 nanoseconds (minimum)

7.5. DESKEW DELAY: 45 nanoseconds (minimum)

7.6. REQ RESPONSE TIMEOUT: 250 milliseconds (maximum)

The maximum delay allowed between assertion of REQ by the TARGET and time out (due to lack of ACK from the INITIATOR).

7.7. RESELECT RESPONSE TIMEOUT: 250 milliseconds (maximum)

The maximum delay allowed for a BSY response from an INITIATOR before time out during RESELECTION.

7.8. RESET HOLD TIME: 25 microseconds (minimum)

The minimum time during which RST is asserted. No maximum.

7.9. SELECTION RESPONSE TIMEOUT: system dependent (maximum)

The maximum delay allowed for a BSY response from a TARGET before time out during SELECTION.

8. PHYSICAL DESCRIPTION

Devices are daisy-chained together using a common cable. Both ends of the cable are terminated. All signals are common between all devices.

8.1. CABLE REQUIREMENTS FOR SINGLE-ENDED OPTION

Single-ended drivers and receivers allow a maximum cable length of 6.0 meters (for in-cabinet interconnection). A 50-conductor flat cable (or twisted pair flat cable) is required.

At each bus port, the stub length of any conductor must be less than 0.1 meter as measured from the bus cable.

Bus termination may be internal to the bus device(s).

A 50-conductor flat cable connector consisting of two rows of 25 pins on 100 mil centers is required. The 3M "Scotchflex" #3425-3000 cable connector satisfies this requirement.

Table 2: PIN ASSIGNMENTS FOR SINGLE-ENDED OPTION

SIGNAL	PIN NUMBER	NOTES
-DB(0)	2	
-DB(1)	4	
-DB(2)	6	
-DB(3)	8	
-DB(4)	10	
-DB(5)	12	
-DB(6)	14	
-DB(7)	16	
-DB(P)	18	
-	20	for future use
-	22	for future use
-	24	for future use
-	26	for future use
-	28	for future use
-	30	for future use
-ATN	32	
-SPARE	34	for future use (terminate as a signal line)
-BSY	36	
-ACK	38	
-RST	40	
-MSG	42	
-SEL	44	
-C/D	46	
-REQ	48	
-I/O	50	

Note: All signals are low true. All odd pins are grounded.

8.2. CABLE REQUIREMENTS FOR DIFFERENTIAL OPTION

Differential drivers and receivers allow a maximum cable length of 15.0 meters (for interconnection outside of a cabinet). A 50-conductor flat cable or twisted pair cable is required.

At each bus port, the stub length of any conductor must be less than 0.2 meter as measured from the bus cable.

Bus termination may be internal to the bus device(s).

A 50-conductor flat cable connector (or equivalent for round cable) consisting of two rows of 25 pins on 100 mil centers is required.

Table 3: PIN ASSIGNMENTS FOR DIFFERENTIAL OPTION

SIGNAL	PIN NUMBER		SIGNAL
*SHIELD GROUND	1	2	GROUND
+DB(0)	3	4	-DB(0)
+DB(1)	5	6	-DB(1)
+DB(2)	7	8	-DB(2)
+DB(3)	9	10	-DB(3)
+DB(4)	11	12	-DB(4)
+DB(5)	13	14	-DB(5)
+DB(6)	15	16	-DB(6)
+DB(7)	17	18	-DB(7)
+DB(P)	19	20	-DB(P)
for future use	21	22	for future use
for future use	23	24	for future use
for future use	25	26	for future use
for future use	27	28	for future use
+ATN	29	30	-ATN
+SPARE	31	32	-SPARE
+BSY	33	34	-BSY
+ACK	35	36	-ACK
+RST	37	38	-RST
+MSG	39	40	-MSG
+DRL	41	42	-SEL
+C/D.	43	44	-C/D
+REQ	45	46	-REQ
+I/O	47	48	-I/O
GROUND	49	50	GROUND

* Optional shield ground on some cables.

9. ELECTRICAL DESCRIPTION

For the following measurements, bus termination is assumed external to the device. A typical device provides optional internal termination.

9.1. SINGLE-ENDED OPTION

All signals are low true and use open collector or three-state drivers (as noted in Section 9) terminated with 220 ohms to +5 volts (nominal) and 330 ohms to ground at each end of the cable.

Each signal driven by a device will have the following output characteristics at the bus port connection:

True = Signal Assertion = 0.0 to 0.5 VDC @ 48 ma

False = Signal Non-assertion = 2.5 to 5.25 VDC

The 7438 open collector driver satisfies this requirement.

Each signal received by a DEVICE will have the following input characteristics at the BUS PORT connection:

True = Signal Assertion = 0.0 to 0.8 VDC @ 0.4 ma (max)

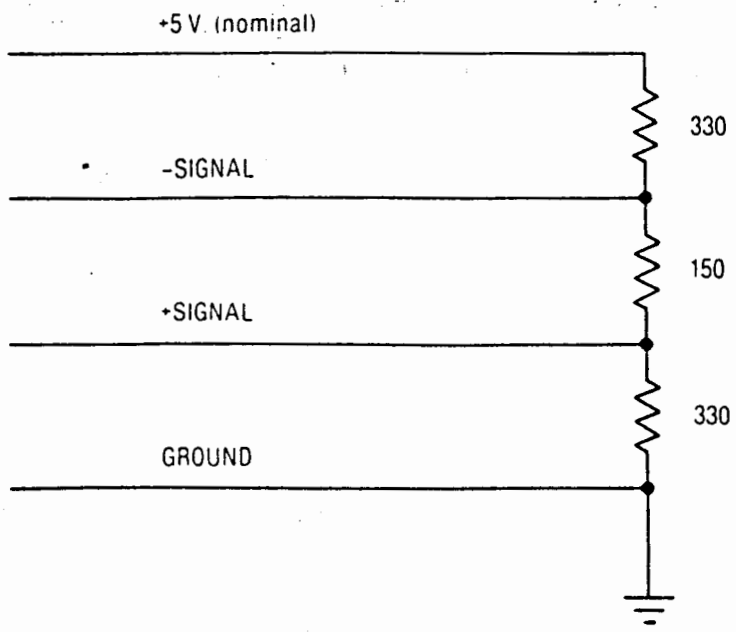
False = Signal Non-assertion = 2.0 to 5.25 VDC

The 74LS14 receiver with hysteresis meets this requirement.

9.2. DIFFERENTIAL OPTION

All bus signals consist of two lines denoted SIGNAL (+) and SIGNAL (-). A signal is ASSERTED when SIGNAL (+) is more positive than SIGNAL (-) and DEASSERTED when SIGNAL(-) is more positive than SIGNAL (+). All assigned signals are terminated at each end of the cable as shown in Figure 6, below.

Figure 4: TERMINATION FOR DIFFERENTIAL OPTION



PART B

MESSAGE SPECIFICATION

8/7/82

1. MESSAGE SYSTEM

The message system allows communication between an INITIATOR and TARGET for purposes of physical path management. This section defines the messages and lists their assigned codes (in HEX).

Normally, the first message sent by the INITIATOR after the SELECTION phase is IDENTIFY (to establish the physical path). After reselection, the TARGET's first message is also IDENTIFY. Under certain conditions, an INITIATOR may send SELECTIVE RESET or BUS DEVICE RESET as the first message.

1.1. SINGLE BYTE MESSAGES

1.1.1. Command Complete (ØØH)

This code is sent from the TARGET, at the completion of command execution (or at the end of a series of linked commands), to direct the INITIATOR to indicate COMMAND COMPLETE to the host.

This message does not imply good ending status; STATUS must be checked to determine end conditions.

1.1.2. Save Data Pointer (Ø2H)

This code is sent from the TARGET to direct the INITIATOR to save a copy of the present active data pointer for the currently attached LUN.

1.1.3. Restore Pointers (Ø3H)

This code is sent from the TARGET to restore the most recently saved pointers (for the currently attached LUN) to the active state.

Pointers to the COMMAND, DATA, and STATUS locations for the LUN will be restored to the active pointers. COMMAND and STATUS pointers will be restored at the beginning of the present operation. The DATA pointer will be restored at the beginning of the operation or at the point at which the last SAVE DATA POINTER message occurred.

1.1.4. Disconnect (Ø4H)

This code is sent from the TARGET to indicate that the current physical path will be broken (the TARGET will disconnect by releasing BSY) and that a later reconnect will be required to complete the current operation.

Error status should be stored if BSY is released in response to any message except ØØ or Ø4 (or RST). By not sending DISCONNECT or COMMAND COMPLETE before going to BUS FREE (except in response to RESET), the TARGET indicates a catastrophic error.

1.1.5. INITIATOR-detected Error (Ø5H)

This code is sent from the INITIATOR to indicate an INITIATOR-detected retryable error (since the last SAVE DATA POINTER).

1.1.6. Selective Reset (Ø6H)

This code is sent from the INITIATOR to direct the TARGET to reset the currently selected LUN and any pending I/O from that LUN for the selecting INITIATOR.

If no LUN has been selected, then all pending I/O operations from the selected TARGET to the selecting INITIATOR will be cleared.

1.1.7. Message Reject (Ø7H)

This code is sent from the INITIATOR or TARGET if the message received was inappropriate or not implemented.

1.1.8. No Operation (Ø8H)

This code is sent from the INITIATOR in response to a request for a message when no valid message exists.

1.1.9. Message Parity (Ø9H)

This code is sent from either the INITIATOR or TARGET to indicate a parity error in the last message received.

To ensure that the current message is rejected by the TARGET, the INITIATOR, before indicating rejection, must assert the ATN signal prior to its release of ACK for the REQ/ACK handshake of the faulty message.

1.1.1Ø. Linked Command Complete (ØAH, ØBH)

This code is sent from the TARGET to indicate completion of the current command.

In response to ØAH, the INITIATOR will update the pointers to next command. In response to ØBH, the pointers will be updated and the system will be signalled that the operation is complete to this point. Status will be stored in either case.

1.1.11. Bus Device Reset (ØCH)

This code is sent from the INITIATOR to the TARGET to reset all I/O operations to all INITIATORS.

1.1.12. Identify (80 TO FF)

This code is sent by either the INITIATOR or TARGET to establish the physical path connection between the INITIATOR and TARGET for a particular LUN. An I/O operation is not established until a valid IDENTIFY message has been received following the intitial selection.

Bit-7 is always set to identify this message.

Bit-6 is set by the INITIATOR to indicate its capability to accommodate disconnection and reconnection.

Bits-5, 4, and 3 are reserved.

Bits-2, 1, and 0 specify a LUN address in a TARGET.

1.2 EXTENDED MESSAGES

1.2.1. Extended Message Follows (01H)

This code is sent from either an INITIATOR or TARGET to indicate that a multiple byte message will follow.

The first byte is a length indicator for the number of bytes to follow. A value of zero indicates 256 bytes.

1.2.2. Modify Data Pointer (01H)

This message is sent from the TARGET to the INITIATOR to request that the following four byte signed value be added (twos complement) to the present value of the DATA pointer. The format of this message is:

BYTE	CODE	FUNCTION
01	01	Extended Message
02	X	Length
03	01	Modify Data Pointer Code
04-07	N	Argument

2. MESSAGE PROTOCOL

All systems are required to implement the COMMAND COMPLETE message. A functional system can be constructed without using any of the other messages if some alternate means is provided for specifying the LUN (e.g., some bits in the command).

If the INITIATOR sends the LUN as a message, the TARGET should ignore the LUN in the command. If the INITIATOR does not support IDENTIFY, the TARGET uses the LUN in the command.

2. MESSAGE PROTOCOL (Continued)

Devices indicate their capability to accommodate more than the COMMAND COMPLETE message by using the ATTENTION signal. The INITIATOR indicates this by creating the ATTENTION condition before it releases BUSY when going through the SELECTION phase. The TARGET indicates its capability by responding to the ATTENTION condition with the MESSAGE OUT phase after going through the SELECTION phase.

The first message sent by the INITIATOR after the SELECTION phase is IDENTIFY. This allows completion of the physical path to an LUN specified by the INITIATOR. After RESELECTION, the first message from the TARGET is also IDENTIFY. This allows the physical path to be established for the TARGET-specified LUN.

IF a physical path is established in an INITIATOR capable of accommodating disconnection and reconnection, the INITIATOR must ensure that the active state of the physical path is equal to the saved copy for that LUN.

PART C

FUNCTIONAL SPECIFICATION

8/7/82

1. GENERAL DESCRIPTION

This section of the Controller/System Interface Specification provides a functional definition of the interface and includes the software command set and the specific status information related to the commands. The logical structure of the I/O subsystem, the functions available from the I/O subsystem, and method by which to request these functions are described.

By defining a fixed block structure using a simple, logical address scheme, the I/O interface can support device independence. In addition, by including the address as a component of the command structure, physical requirements (such as SEEK) can be imbedded within the basic READ and WRITE requests.

This interface, despite its simplicity, is capable of providing the high level of performance required in multi-host/multi-task environments. Powerful functions, such as search and chaining, are included to enhance random access applications, and single-command, multiple-block transfers are included to simplify sequential operations.

2. FUNCTIONAL CONTROL

This section describes the process of obtaining commands and storing results.

2.1. SINGLE COMMAND

A typical operation on this interface includes a single READ command to the I/O subsystem. This operation will be described below starting with a request from the system to the INITIATOR path control logic.

The INITIATOR path control logic includes an active state and a set of stored states for active disconnected devices (systems without disconnect capability do not require stored states). On receiving ESTABLISH PATH from the system, the INITIATOR path control logic establishes the active state for the operation, arbitrates for the bus, and selects the LUN. Control of the operation is then assumed by the TARGET function control logic (a component of the LUN).

Next, the TARGET obtains the command from the INITIATOR (i.e. READ) via the GET COMMAND request to the TARGET path control. The TARGET reads the data from the peripheral device and sends it to the INITIATOR using the SEND DATA function. On completion of the READ, the TARGET stores the completion status in the INITIATOR using the SEND STATUS function. To end the operation, and to acknowledge completion, the TARGET path control logic is given the END PATH function.

2.2. DISCONNECT

In the previous READ example, the time required to obtain data can be considerable due to a time-consuming physical seek. In order to improve system throughput, the TARGET can disconnect from the INITIATOR, freeing the bus to enable requests to other LUN's. To allow disconnect, the INITIATOR path control logic must be reselectable and capable of restoring the device state upon reconnection, and the TARGET path control logic must be capable of arbitrating for the data bus and reselecting the INITIATOR.

After the TARGET has received the READ command (and determined that there will be a delay), it will direct the INITIATOR to save the present state, and the TARGET path control logic to disconnect.

When data is ready to be transferred, the TARGET will direct the path control logic to reconnect the INITIATOR. On reconnection, the INITIATOR path control logic will restore the "saved" state and the TARGET will resume the operation (as in the single command example). At END OF PATH, the INITIATOR path control logic signals the system.

When a TARGET disconnects without first saving the latest state (as might occur if an error were detected while transferring data to the INITIATOR), the operation may be repeated by either restoring the previous state or by disconnecting without saving the present state. When reconnection is completed, the previous state will be restored.

2.3. LINK

The LINK function defines a relationship between commands which allows previous operations to modify commands. LINK makes high performance I/O functions possible by providing a relative addressing capability and allowing multiple command execution without invoking the functional component of the INITIATOR and without requiring reselection.

If a specific data address (e.g. in a READ operation) is unknown, but a string of bytes (key) in the data field is known, then by simply linking the READ to a SEARCH EQUAL command, this data can be found and transferred to the INITIATOR.

The LINKED COMMAND COMPLETE function, sent from the TARGET to acknowledge current command completion, must be sent prior to requesting the next command. The INITATOR updates the stored state so that subsequent requests from the TARGET will reference the next command of the chain. Other than the LINKED COMMAND COMPLETE function and the address modification of linked commands, command processing of linked and single commands is identical.

3. COMMAND AND STATUS STRUCTURE

3.1. COMMAND DESCRIPTION BLOCK (CDB)

An I/O request to a device is made by passing a Command Description Block (CDB) to the TARGET. The first byte of the CDB is the command class and operation code. The remaining bytes specify the Logical Unit Number (LUN), block starting address, control byte, and the number of blocks to transfer. Commands are categorized into eight classes as follows:

Class Ø: 6-Byte commands (CONTROL, DATA TRANSFER, STATUS)

Class 1: 1Ø-Byte commands.

Classes 2, 3 & 4: Not Used.

Class 5: 12-Byte commands.

Class 6: Controller-Specific commands.

Class 7: Controller-Specific Diagnostic commands.

Figures 4A - 4D show the command descriptor block formats.

Table 4A: CLASS ØØ COMMANDS (6-BYTE COMMANDS)

BYTE \ BIT	7	6	5	4	3	2	1	Ø
ØØ	Class Code			Opcode				
Ø1	Logical Unit Number			(MSB) Logical Block Address				
Ø2	Logical Block Address							
Ø3	Logical Block Address							(LSB)
Ø4	Number of Blocks							
Ø5*	Reserved						FlagRq	Link

* Control Byte

Table 4B: FORMAT COMMAND

BYTE	BIT 7	6	5	4	3	2	1	0
00	Class Code					Opcode		
01	Logical Unit Number			Data	Cmplt	Reserved		
02	Reserved							
03	Interleave							
04	Interleave							
05*	Reserved						FlagRq	Link

* Control Byte

Table 4C: CLASS 01 COMMANDS (10 BYTE EXTENDED BLOCK ADDRESS)

BYTE	BIT 7	6	5	4	3	2	1	0
00	Class Code					Opcode		
01	Logical Unit Number				Reserved			Span
02	(MSB) Logical Block Address							
03	Logical Block Address							
04	Logical Block Address							
05	Logical Block Address							(LSB)
06	Reserved							
07	Number of Blocks							
08	Number of Blocks							
09*	Reserved						FlagRq	Link

*Control Byte

Table 4D: CLASS 05 COMMANDS (12-BYTE COMMANDS)

BYTE \ BIT	7	6	5	4	3	2	1	0
00	Class Code			Opcode				
01	Logical Unit Number			Reserved			Rd Inh	Wr Inh
02	(MSB) Logical Block Address							
03	Logical Block Address							
04	Logical Block Address							
05	Logical Block Address							(LSB)
06	(MSB) 2nd Logical Block Address							
07	2nd Logical Block Address							
08	2nd Logical Block Address							
09	2nd Logical Block Address							(LSB)
10	Reserved							
11*	Reserved						Flag Rq	Link

*Control Byte

3.2. CLASS CODE

The class code can be 0 to 7.

3.3. OPERATION CODE

The operation code for each class allows 32 commands (0 to 31).

3.4. LOGICAL UNIT NUMBER

Logical unit numbers allow 8 devices per TARGET (0 to 7).

This method of device addressing is designed for low-end systems that do not accommodate separate paths for command and address functions. It is not recommended for use in the more sophisticated applications.

3.5. LOGICAL BLOCK ADDRESS

Six and ten byte commands contain 21 bit starting block addresses. Extended address commands contain 32 bit starting block addresses.

The "block" concept implies that the INITIATOR and TARGET have "preset" the number of bytes of data to be transferred.

3.6. NUMBER OF BLOCKS

A variable number of blocks may be transferred under a single command. Class ØØ commands may transfer up to 256 blocks, while class Ø1 commands may transfer up to 64K blocks.

3.7. CONTROL BYTE (Last Byte in All Commands)

Bit 7-2 Not used.

Bit 1 = 1 This bit is meaningful when Bit Ø is set (indicating that an interrupt is requested for each command in a group of linked commands).

Bit Ø = 1 This bit indicates an automatic link to the next command upon completion of the current command. STATUS is sent for each command executed.

4. COMMAND DESCRIPTIONS

4.1. CLASS ØØ COMMAND DESCRIPTIONS

4.1.1. TEST UNIT READY (ØØH)

This command returns zero status if the requested unit is powered on and ready.

This is not a request for self-test. A fast response is expected.

4.1.2. REZERO UNIT (Ø1H)

This command sets the unit to a specified state such as REWIND or RECALIBRATE.

4.1.3. REQUEST SENSE (Ø3H)

This command returns unit sense.

The sense data will be valid for the check condition (status) sent to the INITIATOR and must be preserved. Sense data will be cleared on receiving a subsequent command from the INITIATOR that received the check condition (See Section 2.1Ø for sense data format).

4.1.3. REQUEST SENSE (Continued)

The number-of-blocks byte specifies the number of bytes allocated by the host for returned SENSE. For compatibility with present applications, a value of Ø - 4 in this field will transfer 4 bytes of sense data. CHECK STATUS must not be sent in response to this command.

4.1.4. FORMAT UNIT (Ø4H)

This command formats the media (See Table 4B). Bit 4 of Byte 1 indicates that the following data is available:

2 Bytes - Reserved
2 Bytes - Length of defect list (number of bytes)
N Bytes - Defect List (4 bytes per defect)

The block size used with the defect list is determined by the previous format. For unformatted units, the block size is specified by the MODE SELECT command. The defect list must be in ascending order. (Bit 3 of Byte 1 indicates that this data alone is used for formatting).

4.1.5. READ (Ø8H)

This command transfers (to the INITIATOR) the specified number of blocks starting at the specified logical starting block address. See SEARCH (4.2.5. - 4.2.7.) for the linked modification of this command.

4.1.6. WRITE (ØAH)

This command transfers (to the TARGET) the specified number of blocks starting at the specified logical starting block address. See SEARCH (4.2.5. - 4.2.7.) for the linked modification of this command.

4.1.7. SEEK (ØBH)

This command directs the unit to ready itself to transfer data from the specified address in a minimum time.

4.1.8. INQUIRY (12H)

This command requests the transfer of unit and TARGET parameters to the INITIATOR. Byte four indicates the maximum number of data bytes that may be transferred by the TARGET.

1 Byte Code:
Ø - Direct access device
1 - Sequential access device
2 - Output only device
3 - Processor

4.1.8. INQUIRY (Continued)

1 Byte Qualifier:
Bit 7 - On if removable media
Bits 6-Ø - User defined (Zero if no code)

1 Byte: Length of additional bytes
n Bytes: Additional parameters

The qualifier byte allows users to define classes of devices within a system (useful in systems with multiple removable media drives).

4.1.9. MODE SELECT (15H)

This command specifies the recording format mode for removable media drives.

Byte 5 specifies the number of data bytes. The recording code byte is set to zero for default conditions. Non-zero values select optional (such as FM or MFM) recording formats. Three bytes are not used; the next four bytes define the block size (again, zero selects the default conditions).

Byte Ø - Recording code
Bytes 1-3 - Not used
Bytes 1-4 - Block size

4.1.1Ø. RESERVE UNIT (16H)

This command reserves the unit for use by the requesting INITIATOR until a RELEASE UNIT COMMAND is received.

4.1.11. RELEASE UNIT (17H)

This command releases the unit from the INITIATOR.

This INITIATOR must have issued the RESERVE UNIT command.

4.1.12. READ DIAGNOSTIC RESULT (1CH)

This command sends analysis data to INITIATOR after completion of a SEND DIAGNOSTIC command.

Bytes 3 and 4 designate the size of the available buffer (in bytes).

4.1.13. SEND DIAGNOSTIC (1DH)

This command sends data to the TARGET to specify diagnostic tests for TARGET and peripheral units.

Bytes 3 and 4 specify the length of the data to be sent.

Table 5: SEND/RECEIVE DIAGNOSTIC FORMAT

BYTE	BIT 7	6	5	4	3	2	1	0
00				1	1	1	0	X
01	Logical Unit Number			Reserved			DOF	UOF
02	Reserved							
03	MSB Data Length							
04	Data Length							LSB
05*	Reserved						FlagRq	Link

* Control Byte

UOF (Unit offline: Byte 1, Bit 0) enables write and positioning operations on user media.

DOF (Device offline: Byte 1, Bit 1) enables execution of diagnostic commands that may adversely affect I/O operations to other LUN's on the same controller.

Table 6: CLASS 00 COMMAND CODES

OP CODE	COMMAND	OP CODE	COMMAND
00	TEST UNIT READY	0B	SEEK
01	REZERO UNIT	12	INQUIRY
03	REQUEST SENSE	16	RESERVE UNIT
04	FORMAT UNIT	17	RELEASE UNIT
08	READ	1C	READ DIAGNOSTIC
0A	WRITE	1D	WRITE DIAGNOSTIC

4.2. CLASS ØØ COMMAND DESCRIPTIONS

4.2.1. READ CAPACITY (Ø5H)

If byte 8 is ØØH, this command will return the address of the last block on the unit.

If byte 8 is Ø1H, this command will return the address of the block (after the specified address) at which a substantial delay in data transfer will be encountered (e.g., a cylinder boundary).

In both cases, the block size is defined by the last two bytes of the 8-byte data field returned as a result:

4 Bytes - Block Address
4 Bytes - Block Size

4.2.2. WRITE AND VERIFY (ØEH)

This command writes and verifies the data for the specified number of blocks. See SEARCH (4.1.14. - 4.1.16.) for the linked modification of this command.

The VERIFY is an ECC check of the written data.

4.2.3. VERIFY (ØFH)

This command verifies data for the specified number of blocks (performs an ECC check of the referenced data). No data is transferred with this command. See SEARCH (4.2.5. - 4.2.7.) for the linked modification of this command.

4.2.4. SEARCH DATA EQUAL (11H)

This command specifies a block address and a number of blocks to search.

The data transferred to the TARGET defines the area of the block to search and includes the search argument. If a command is linked to the SEARCH command and the search is successful, then the next command is fetched and executed. In this case, the address portion of the command is used as a two's complement displacement from the address at which the SEARCH was satisfied. If the search was not satisfied, the link is broken and END STATUS is presented.

If the address of a block that has satisfied the SEARCH is desired, a SENSE command must be linked to the SEARCH or issued directly after the SEARCH. The SEARCH argument is as follows:

4.2.4. SEARCH DATA EQUAL (Continued)

Bytes Ø-7: Reserved (for record search).

Bytes 8-9: Length of following search argument.

Repeated n Times:

4 byte displacement (from start of block) of field to compare
2 byte length (m) of field to compare
m byte data to compare.

4.2.5. SEARCH DATA HIGH (1ØH)

This command performs the same function as the SEARCH DATA EQUAL command, but is satisfied if the data compared is higher than or equal to the search argument. A WRITE command may not be linked to this command.

4.2.6. SEARCH DATA LOW (12H)

This command performs the same function as the SEARCH DATA EQUAL command, but is satisfied if the data compared is lower than or equal to the search argument. A WRITE command may not be linked to this command.

Table 7: CLASS Ø1 COMMAND CODES

OP CODE	COMMAND	OP CODE	COMMAND
Ø5	READ CAPACITY	ØF	VERIFY
Ø8	READ	1Ø	SEARCH DATA HIGH
ØA	WRITE	11	SEARCH DATA EQUAL
ØE	WRITE AND VERIFY	12	SEARCH DATA LOW

4.3. CLASS Ø5 COMMAND DESCRIPTIONS

4.3.1. SET LIMITS (Ø9H)

This command defines the addresses within which subsequent linked commands may operate. Only one SET LIMITS command may be linked to a chain of commands. The two low order bits of byte 8 define the legal operations within the limits of the specified addresses. Bit Ø and bit 1 indicate write and read inhibit, respectively.

5. COMPLETION STATUS BYTE (Bytes ØØ, Ø1)

Status must always be sent at the end of a command or set of linked commands. Intermediate status is sent at the completion of a linked command. Any abnormal conditions encountered during command execution causes command termination and ending status.

Bits Ø, 5 & 6: Reserved.

Bit 1: Check condition. Sense is available (See 2.1).

Bit 2: Equal. Set when any SEARCH is satisfied.

Bit 3: Busy. Device is busy or reserved. Busy status will be sent whenever a target is unable to accept a command from an INITIATOR. This condition occurs when an INITIATOR that does not allow reconnection requests an operation from a reserved or busy device.

Bit 4: Intermediate status sent. This bit is set for any intermediate status sent during a series of linked commands. This bit will not be set (regardless of the interrupt request bit) in any ending status.

Bit 7: Extended status. Indicates valid second status byte.

Byte Ø1, Bit Ø: Host adapter detected error. Reserved for the host adapter to flag the host for errors not detected by the TARGET.

Byte Ø1, Bit 7: Extended Status. Indicates valid third status byte.

Table 8: COMPLETION STATUS BYTE

BYTE \ BIT	7	6	5	4	3	2	1	Ø
ØØ	ExStat	Reserved		InStat	Busy	Equal	Check	Reserv
Ø1	ExStat	Spare						HA Err

6. SENSE BYTES: Table 9

BYTE	BIT 7	6	5	4	3	2	1	Ø
ØØ	AdrVal	Error Class			Error Code (See Tbls 11-14)			
Ø1	Reserved			(MSB)	Logical Block Address			
Ø2	Logical Block Address							
Ø3	Logical Block Address							(LSB)

6.1. EXTENDED SENSE BYTES: Table 1Ø

BYTE	BIT 7	6	5	4	3	2	1	Ø
ØØ	AdrVal	Class Ø7			Code ØØ			
Ø1	Reserved				Spare			
Ø2	Fil Mk	EOM	Reserved		Sense Key			
Ø3	(MSB)	Logical Block Address						
Ø4	Logical Block Address							
Ø5	Logical Block Address							
Ø6	Logical Block Address							(LSB)
Ø7	Additional Sense Length							
Ø8-nn	Additional Predefined Sense Bytes							

6.2. SENSE KEYS

The Sense Key is a device independent code designed to aid the system in resolving the following Sense Data:

ØØ = No Sense
Ø1 = Recoverable Error
Ø2 = Not Ready
Ø3 = Media Error (Non Recoverable)
Ø4 = Hardware Error (Non Recoverable)
Ø5 = Illegal Request
Ø6 = Media Change (Must be reported to all INITIATORS)
Ø7 = Write Protect
Ø8 = Diagnostic Unique
Ø9 = Vendor Unique
ØA = Power Up Failed
ØB = Aborted Command

Table 11: CLASS ØØ ERROR CODES IN SENSE BYTE (DRIVE ERRORS)

CODE	ERROR
ØØ	NO SENSE
Ø1	NO INDEX SIGNAL
Ø2	NO SEEK COMPLETE
Ø3	WRITE FAULT
Ø4	DRIVE NOT READY
Ø5	EQUIPMENT CHECK
Ø6	NO TRACK ØØ
Ø7	MULTIPLE DRIVES SELECTED
Ø8	NOT ASSIGNED
Ø9	MEDIA NOT LOADED
ØA - ØF	NOT ASSIGNED

Table 12: CLASS Ø1 ERROR CODES IN SENSE BYTE (TARGET ERRORS)

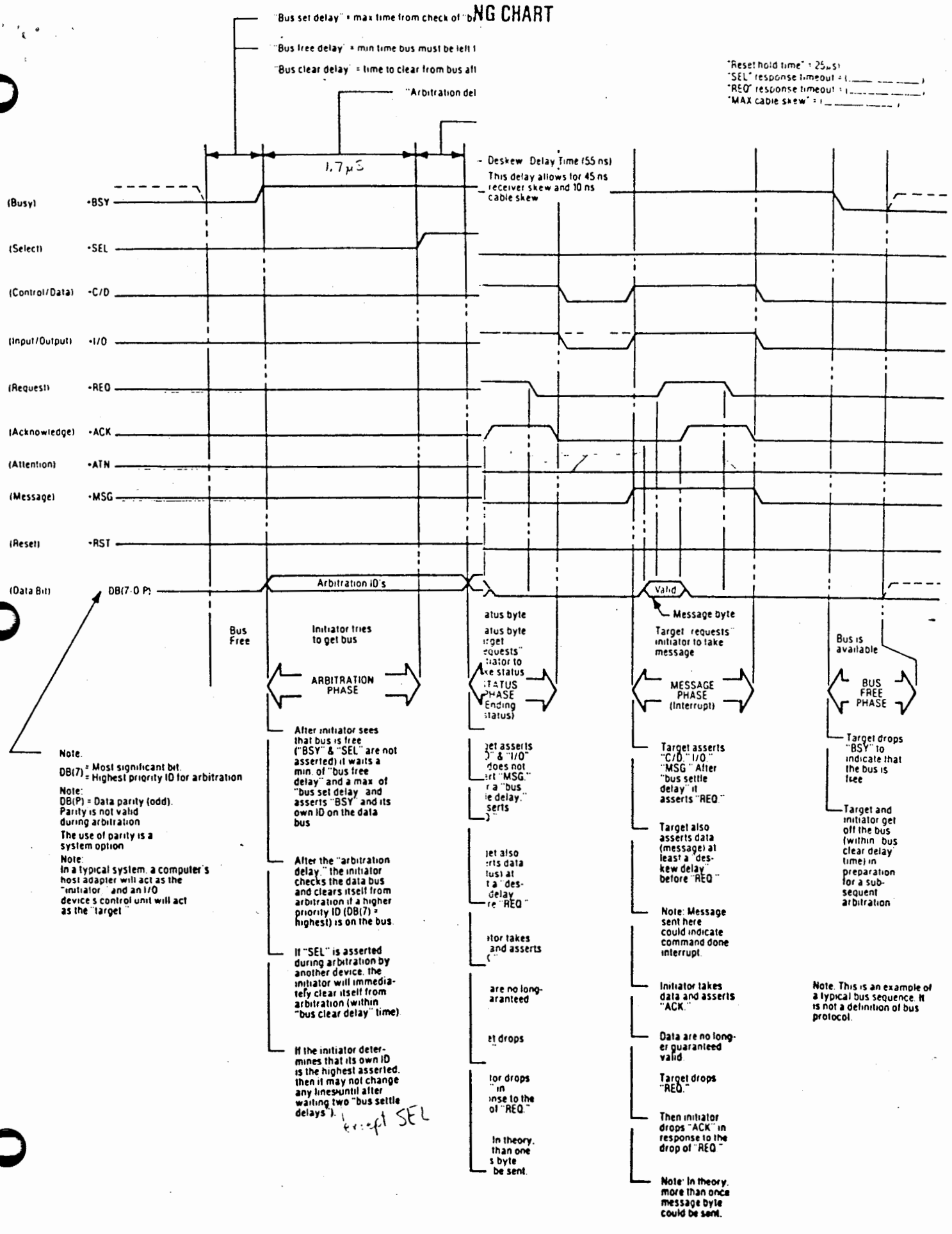
CODE	ERROR
ØØ	I.D. ERROR
Ø1	UNCORRECTABLE DATA ERROR
Ø2	I.D. SYNC BYTE NOT FOUND
Ø3	DATA SYNC BYTE NOT FOUND
Ø4	RECORD NOT FOUND
Ø5	SEEK ERROR
Ø6	NOT ASSIGNED
Ø7	WRITE PROTECTED
Ø8	CORRECTABLE DATA CHECK
Ø9	NOT ASSIGNED
ØA	INTERLEAVE ERROR
ØB	DATA TRANSFER INCOMPLETE
ØC	UNFORMATTED OR BAD FORMAT ON DRIVE
OD	SELF TEST FAILED
ØE	DEFECTIVE TRACK (MEDIA ERRORS)
ØF	NOT ASSIGNED

Table 13: CLASS Ø2 ERROR CODES (SYSTEM-RELATED ERRORS)

CODE	ERROR
ØØ	INVALID COMMAND
Ø1	ILLEGAL BLOCK ADDRESS
Ø2	ABORTED
Ø3	VOLUME OVERFLOW
Ø4 - ØF	NOT ASSIGNED

Table 14: CLASS 07 ERROR CODES (EXTENDED SENSE ERRORS)

CODE	ERROR
00	EXTENDED SENSE
01 - 0F	NOT ASSIGNED


"Bus set delay" = max time from check of "bu
"Bus free delay" = min time bus must be left f
"Bus clear delay" = time to clear from bus aft
"Arbitration del
"Reset hold time" = 25μs)
"SEL" response timeout = (
"REQ" response timeout = (
"MAX cable skew" = (
1.7μs
Deskew Delay Time (55 ns)
This delay allows for 45 ns receiver skew and 10 ns cable skew
(Busy) •BSY
(Select) •SEL
(Control/Data) •C/D
(Input/Output) •I/O
(Request) •REQ
(Acknowledge) •ACK
(Attention) •ATN
(Message) •MSG
(Reset) •RST
(Data Bit) DB(7-0 P)
Arbitration ID's
Valid
Message byte
Bus Free
Initiator tries to get bus
ARBITRATION PHASE
atus byte
atus byte
iget equests" itiator to e status
ITATUS PHASE Ending status)
Target requests" initiator to take message
MESSAGE PHASE (Interrupt)
Bus is available
BUS FREE PHASE
Note.
DB(7) = Most significant bit. = Highest priority ID for arbitration
Note:
DB(P) = Data parity (odd). Parity is not valid during arbitration
The use of parity is a system option
Note:
In a typical system, a computer's host adapter will act as the "initiator" and an I/O device's control unit will act as the "target"
After initiator sees that bus is free ("BSY" & "SEL" are not asserted) it waits a min. of "bus free delay" and a max. of "bus set delay" and asserts "BSY" and its own ID on the data bus
After the "arbitration delay," the initiator checks the data bus and clears itself from arbitration if a higher priority ID (DB(7) = highest) is on the bus.
If "SEL" is asserted during arbitration by another device, the initiator will immediately clear itself from arbitration (within "bus clear delay" time).
If the initiator determines that its own ID is the highest asserted, then it may not change any lines until after waiting two "bus settle delays").
except SEL
jet asserts)" & "I/O" does not ert "MSG." r a "bus le delay." serts)"
jet also erts data tus) at t a "des- delay re "REQ"
itor takes and asserts ("
are no longaranteed
et drops
tor drops "in nse to the of "REQ."
In theory, than one s byte be sent.
Target asserts "C/D," "I/O," "MSG." After "bus settle delay" it asserts "REQ."
Target also asserts data (message) at least a "deskew delay" before "REQ."
Note: Message sent here could indicate command done interrupt.
Initiator takes data and asserts "ACK."
Data are no longer guaranteed valid.
Target drops "REQ."
Then initiator drops "ACK" in response to the drop of "REQ."
Note: In theory, more than once message byte could be sent.
Target drops "BSY" to indicate that the bus is free
Target and initiator get off the bus (within "bus clear delay" time) in preparation for a subsequent arbitration
Note: This is an example of a typical bus sequence. It is not a definition of bus protocol.