~~SMD CONTROLLER~~ S100 - ST506 MSC 9056
MSC hard disk controller
2 DRIVES

SMD → 300 MB / 100 MB 1440 8"

1.0 INTRODUCTION

The Disk Controller is based upon the MSC-9056 module to provide the control, data buffering and error correction functions for the disk drive. Other circuitry on the controller provides the host and disk interface functions.

1.1 Features

The following features are provided by the disk controller.

- Host Interface - S100 Bus
- Disk Drive - Seagate Technology ST-506
- Number of Drives Supported - 4
- Number of Bits per Data Transfer - 8
- Alternate Sectoring on Each Track
- Variable Interleave
- 11-Bit Burst Error Correction
- 22-Bit Burst Error Detection
- Data Buffer - 512 Bytes
- Data Separator

1.2 Physical Characteristics

- Size 7.25 x 10 inches
- Vertical Clearance Required - .75 inch

1.3 Operating Characteristics

1.3.1 Power Requirements

9.5 ±1.5 VDC unregulated at 3 AMPS Maximum

-10 ±3 VDC at 300m AMPS Maximum (normally used -12 VDC)

1.3.2 Environmental Limits

	Operating		Storage	
	Min	Max	Min	Max
Temperature Range				
Fahrenheit	32	122	-40	167
Celsius	0	50	-40	75
Relative Humidity	10%	95%	10%	95%
40°F max. wet bulb temperature, no condensation				
Altitude Range				
Feet	Sea level	10,000	Sea level	15,000
Meters	Sea level	3,048	Sea level	4,572

2.0 CONFIGURATION

A number of controller variables must be configured before the controller can be operated (see Figure 2-1).

2.1 I/O Device Address

The DIP SWITCH at location 8F sets the base I/O address. I/O mapped addresses from "04" Hex to "FC" Hex can be selected. DIP SWITCH 1 is the low order address selection, and DIP SWITCH 6 is the high order. See Table 2-1 for address selection guide.

ADDRESS SELECTION GUIDE

Base Address Hex	DIP SWITCH					
	1	2	3	4	5	6
Ø4	Off	On	On	On	On	On
Ø8	On	Off	On	On	On	On
1Ø	On	On	Off	On	On	On
2Ø	On	On	On	Off	On	On
4Ø	On	On	On	On	Off	On
8Ø	On	On	On	On	On	Off
FC	Off	Off	Off	Off	Off	Off

Note: Some Dip Switches are labeled "OPEN", in place of "ON".

Open = Off
Closed = On

TABLE 2-I

2.2 I/O Interrupt Priority

Interrupt priority is selectable from VØ (highest) to V7 (lowest). A strap from the center post V to the appropriate line sets the priority (0-7 = VØ-V7)

Priority Line	S100 Bus Pin No.
0	P1-4
1	P1-5
2	P1-6
3	P1-7
4	P1-8
5	P1-9
6	P1-10
7	P1-11

2.3 Reset Select

Two possible resets signals are available for use.

POC — Power on clear, specified as occurring only on power up.

SLAVE CLR — Reset for all slave devices.

Jumper W8 — Select the reset line.

W8 A to B selects SLAVE CLEAR.

W8 A to C selects POC.

This jumper is set to POC (W8 A to C) at the factory.

2.4 I/O Write Enable

The I/O write output into the controller is selectable for enable with only output status (SOUT, P1-45), or enable with both SOUT and Write Strobe (PWR, P1-77) and together.

W7 selects the write enable

W7 A-B PWR and SOUT

W7 A-C SOUT only.

W7 is set for A-C at factory.

2.5 I/O Read Enable

The I/O read input from controller is selectable for enable with only input status (SINP, P1-46), or enable with both SINP and Read Strobe (pDBIN) and together.

W9 selects the read enable.

W9 installed SINP and pCBIN.

W9 removed SINP only.

W9 is NOT installed at the factory.

2.6 Bytes Per Sector Strap

This strap selects if the disk is to be used with 256 or 512 bytes per sector. At 256, there are 32 sectors per track. At 512, there are 17 sectors per track. W6 installed for 256 bytes and removed for 512 bytes.

W6 is installed at the factory.

2.7 Test Strap

These straps are used for factory testing, and should not be removed.

W5, W4.

These straps should not be installed.

SD1, SD2, SD3.

2.8 Drive Connection

The ST506 is electrically connected to the controller with 2 cables, 34 conductor control cable, and a 20 conductor data cable.

2.8.1 Control Cable

The control cable connects to J1 on the controller, and P1 on the drive. In dual drive configuration, this cable is daisy chained from drive to drive. Only the last drive on the string should have a terminator. Also, each drive must have an individual device address. Refer to the ST506 manual for drive preparation and connections.

2.8.2 Data Cable

Each drive has a separate data cable and must be connected to its proper connector on the controller.

Drive 1 to J2

Drive 2 to J3

Drive 3 to J4

Drive 4 to J5

Refer to Figure 2-1 for connector location and pin numbering.

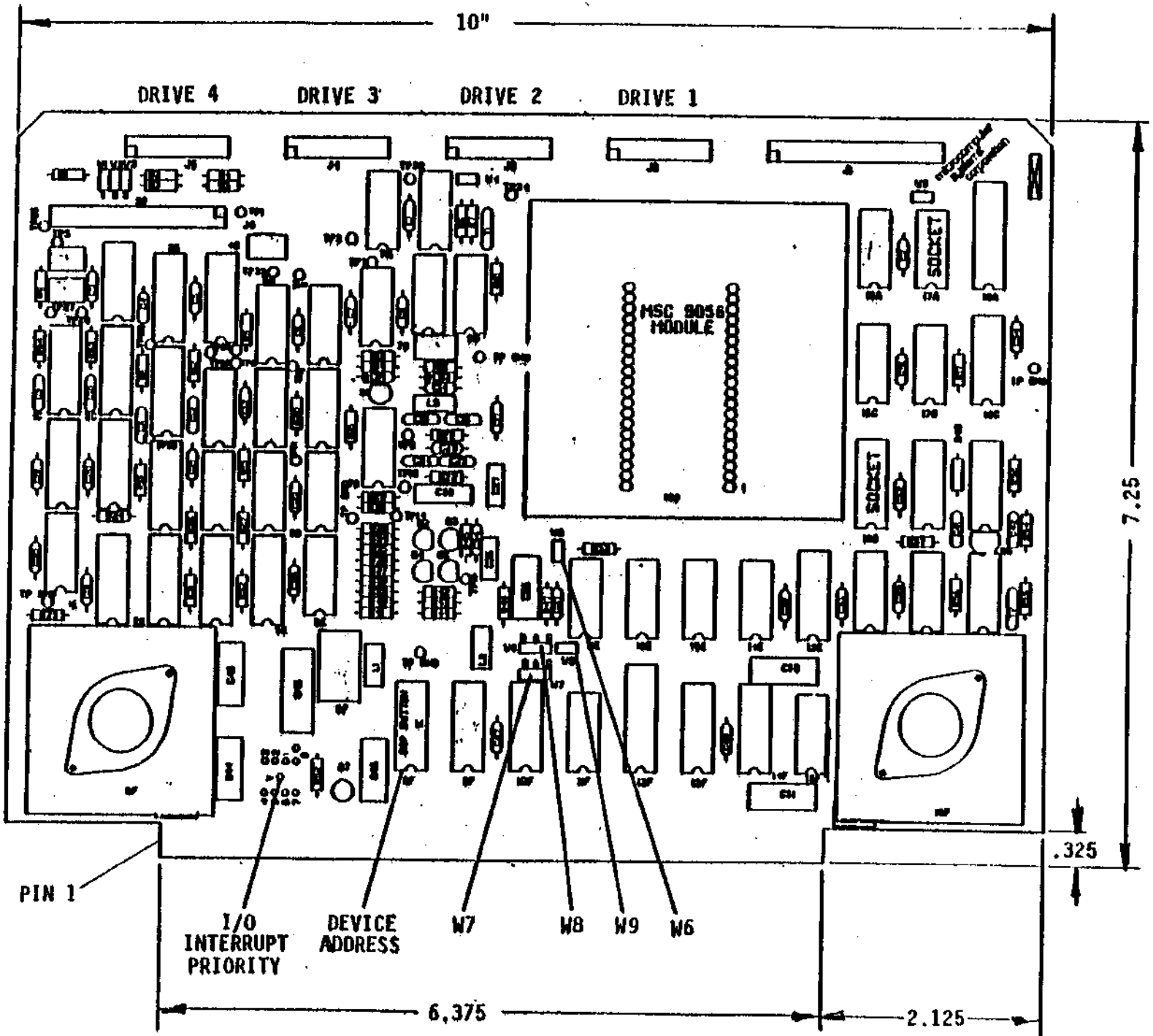

10"
DRIVE 4
DRIVE 3
DRIVE 2
DRIVE 1
MSC 8056
MODULE
SOCKET
SOCKET
7.25
.325
PIN 1
I/O
INTERRUPT
PRIORITY
DEVICE
ADDRESS
W7
W8
W9
W6
6.375
2.125

FIG 2

3.0 HOST COMMUNICATION PROCOTOL

The controller communicates with the host system via 8 bit input and output bytes at selectable I/O addresses. The address consists of a base address plus up to three additional addresses. The three additional addresses specify if the information is a command, control, status, or data byte.

The controller has two major states; Busy and Not Busy. When Not Busy, the controller can accept a command. Giving the controller a command consists of multiple bytes to fully describe the command. After getting the full command sequence, the controller will go to the Busy state and remain there until the command is completely done or an error is encountered. During the Busy state, the disk data is transferred between the host and the controllers buffer. At the end of the operation, a Status Report command sequence can be performed to see the results of the operation.

The controller can be enabled to issue an interrupt whenever BUSY is reset, or whenever DATA IN READY is active. Interrupts are cleared by clearing the interrupt enables.

3.1 Read Cycle (DATA IN)

The Read cycle uses two addresses to input the disk data (disk read), Disk status, or controller status.

3.1.1 IN ADR +0

Inputting this address provides disk data or disk status. Disk data is 256 or 512 eight bit bytes (depending on the option selected -see section 2.4).

Disk status is one byte containing the results of the previous operation. See Table 5-1.

3.1.2 IN ADR +1

Inputting this address provides controller status signals which should be used as "handshake" signals to communicate with the controller.

Bit 0 DATA OUT READY This signal signifies that the controller is ready to receive a

data byte. The host then outputs (Writes) a byte to the controller.

Bit 1 DATA IN READY — This signal signifies that the controller is ready to send a data byte. The controller then can input (Read) one data byte from controller. If DATA IN INTR is enabled, then this signal also issues an interrupt. This can be used during read drive operations to indicate that the read data is ready to send.

Bit 7 BUSY — This signal high signifies that the controller is BUSY performing a command.

This signal low indicates the controler is idle and ready to receive a command. If the DONE interrupt is enabled, then BUSY low issues an interrupt to the host.

3.2 Write Cycle (DATA OUT)

The write cycle uses three addresses to output disk data (disk write), command sequences, or control signals to the controller.

3.2.1 OUT ADR +0

Writing this address sends disk data or command information to the controller. The disk data is 256 or 512 data bytes (depending on the option selected - see section 2.4). The command information is 8 bytes as specified in Table 3.1. Whenever a command does not require 8 bytes to define it, pad bytes (any value) must be sent to the controller. See Table 3-2.

3.2.2 OUT ADR +1

Writing this address causes the controller to start a command sequence. This should only be performed when the controller is NOT BUSY. Upon seeing the command sequence activated, the controller will request the 8 bytes of command information. Thereafter, the controller will go BUSY performing the command and transferring the required data.

3.2.3 OUT ADR +2

Writing this address causes a reset of the controller. The controller will also get reset on a power up cycle. If interleave is used, a new set interleave command must be executed after this command. Interrupts are disabled after a reset.

3.2.4 OUT ADR +3

Writing this address enables or disables the interrupts.

BIT 0	True enables DONE INTR.
BIT 1	True enable DATA IN INTR.

3.3 Automatic Retries

Upon execition of any disk related command, the controller will do four automatic retries on the occurrence of certain errors. The following errors will envoke the retry:

- Position Verification (Seeking)
- Target Sector Verification
- Hard ECC Error on Read Sector
- Write Alternate Sector

The Task Bytes are represented as follows:

Byte	Task Bytes	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	Command Code	0	0	0	0	2^3	2^2	2^1	2_0
2	Drive Number Select	0	0	0	0	D_4	D_3	D_2	D_1
3	Cylinder Address (MSB)	C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8
4	Cylinder Address (LSB)	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
5	Head Address	0	0	0	0	H_3	H_2	H_1	H_0
6	Sector Address	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
7	Zero Byte	0	0	0	0	0	0	0	0
8	One Byte	0	0	0	0	0	0	0	0

If a command does not require the full 8 Bytes, then Pad Bytes must be included to maintain the 8 Byte message length. After the task is sent to the Module, data will be transferred (in our out) when the task is executed.

COMMAND SEQUENCE BYTES

Table 3-1

	CMD CODE	DRV NBR	CYL MSB	CYL LSB	HD ADR	SCT ADR	Zero Bytes	Zero Bytes	Data Bytes Number of Bytes	Data Bytes In/Out of Module
SEEK	0	X	X	X	-	-	0	0	-	-
READ	1	X	X	X	X	X	0	0	N	OUT
WRITE	2	X	X	X	X	X	0	0	N	IN
FORMAT	3	X	X	X	X	X	0	0	-	-
RECALIBRATE	4	X	-	-	-	-	0	0	-	-
STATUS	5	-	-	-	-	-	0	0	1	OUT
READ LONG	6	X	X	X	X	X	0	0	N + 4	OUT
WRITE LONG	7	X	X	X	X	X	0	0	N + 4	IN
WRITE ALT.	8	X	X	X	X	X	0	0	N	IN
SET INTERLEAVE	9	-	-	-	-	-	0	0	S	IN
WRITE CHECK	A	X	X	X	X	X	0	0	-	-
DIAGNOSTIC	B	-	-	-	-	-	0	0	-	-

(X) represents a task byte is required. (-) represents a Pad Byte is required.

(N) is the number of Data bytes per Sector which can be selected to be 256 or 512.

(S) is the logical to Physical interleave table with the number of bytes equal to the number of Sectors per track.

Byte	Interleave Table Contents
0	Physical location of logical Sector 0
1	Physical location of logical Sector 1
.	.
.	.
16	Physical location of logical Sector 16
.	.
.	.
31	Physical location of logical Sector 31

COMMUNICATION SEQUENCE SUMMARY
TABLE 3-2

4.0 COMMAND SET

The controller supports the following set of commands:

4.1 Seek

The seek command moves the heads to the specified absolute cylinder. The disk must be formatted.

4.2 Read One Sector

The read one sector command transfers one sector of data from the disk to the host system. During this command, the Module positions the heads (implied seek), verifies the ID field, transfers the sector data to the module, and checks and corrects data errors prior to transferring the data to the host.

4.3 Write One Sector

The write one sector command transfers one sector of data from the host to the disk. During this command, the Module transfers one sector of data from the host to the internal buffer, positions the heads (implied seek), verifies the ID field and if proper, writes the data to the disk.

4.4 Format One Track

The format one track command initializes all ID and data fields for the specified track. The data field is initialized with a preset pattern of B6D9. The head parameter is used to select the head of the disk prior to formatting, the cylinder parameter is used to write the ID field. This command will cause address marks to be written on the track to divide it into fixed sectors.

4.5 Recalibrate

The recalibrate command positions the heads at track 00 on the disk.

4.6 Status

The status command sends one byte of status infromation to the host. The status represents the results of the previous task.

4.7 Read Long

The read long command transfers one sector plus the four ECC (Error Correction Code) Bytes from the disk to the host. During this command, the module positions the heads (implied seek), verifies the ID field, transfers the sector data to the module, and then transfers the sector plus ECC Bytes to the host. The module does not try to correct the data field if an ECC error occurs during the read. This command can be used to verify the ECC function of the module.

4.8 Write Long

The write long command transfers one sector plus four ECC Bytes from the host to the disk. During this command, the module transfers one sector plus the ECC Bytes from the host to the internal buffer, positions the head (implied seek), verifies the ID field, and writes the data and ECC to the disk. This command can be used to verify the ECC function of the module. The appended ECC polynomial should be the same as read with the READ LONG Command.

4.9 Write Alternate Sector

This command is used to reassign a defective sector to the alternate location on the track. During this command, the Module reformats the track and formats the ID fields to signify the defective and alternate assignment.

4.10 Set Interleave

This command transfers a block of data into the Module which represent the interleave table of logical to physical assignments. After a clear or diagnostic command the Module defaults to no interleave until this command is executed.

4.11 Write Check

The command is identical to the READ Sector command, without any data transferred. It can be used to verify the previously written data can be read without ECC errors.

4.12 Diagnostic

This command causes the Module to execute a self-test; checking its internal processors, data buffers, ECC circuitry, and Program memory. A STATUS command can then be performed which will give the results. If this command does not complete within 30 seconds, the Module has a defect which prevents communication. This command will leave the module in a clear state.

5.0 CONTROLLER STATUS

The results of a controller command can be determined by examining the controller status which is shown in Table 5-1.

STATUS CODE TABLE

CODE	MEANING
00	No Error
01	Invalid Command
02	Drive Not Ready
03	Seek Timeout (2 Seconds)
04	Invalid Track 00 indication from disk drive
05	All ID Fields Bad on Track
06	Sector Not Found
08	Position Error
09	Data Transfer Start Error
OA	Write Fault Error
0B	Index/Sector Timeout
0C	Command Parameter Error
0D	Uncorrectable ECC Error
1X	Correctable ECC Error
20	Write Alternate Error
21	Alternate Sector is Defective
22	Alternate Already Assigned
23	Direct Access to Alternate Sector
24	Defective Processor
25	Defective Buffer Memory
26	Defective ECC Circuitry
27	Defective Program Memory
28	Illegal Sector Pulse during diagnostic
29	Illegal Interleave Table Parameter

NOTE: X is the length of the burst error which can be 1 to B to note if the correction span was 1 to 11 bits.

TABLE 5-1

Copies: To go w- 9391

DATA

Compact controller can run any Winchester disk drive

Small universal module with minimum external circuitry handles error correction, data buffering, and address verification at low cost

by Don Sumner, *Microcomputer Systems Corp., Sunnyvale, Calif.*

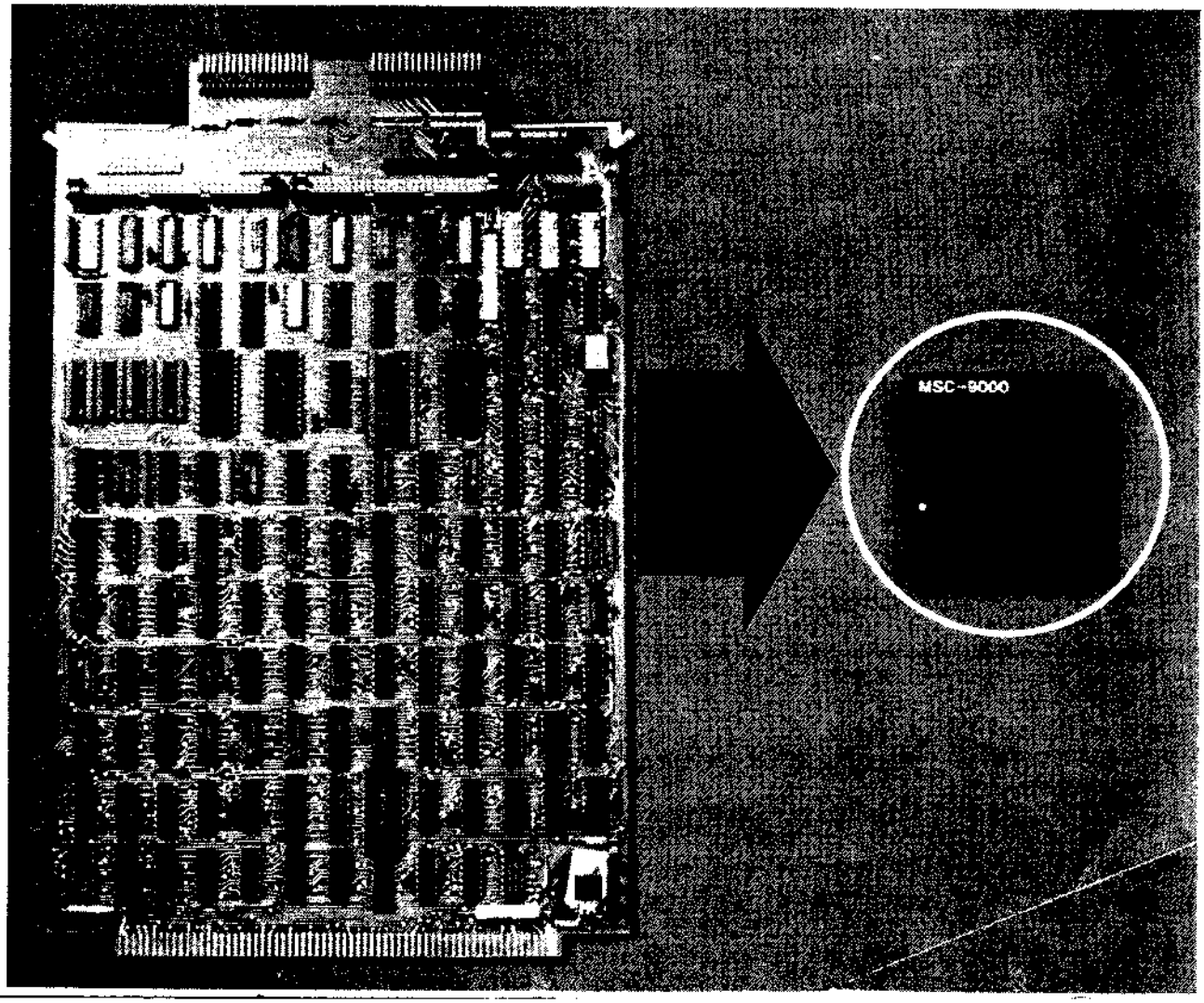
☐ Small rigid-disk drives are beginning to give floppy disks a run for their money. Aimed at the small-computer market, which is increasing by more than 30% annually, they incorporate sophisticated Winchester technology at a comparatively low cost per function.

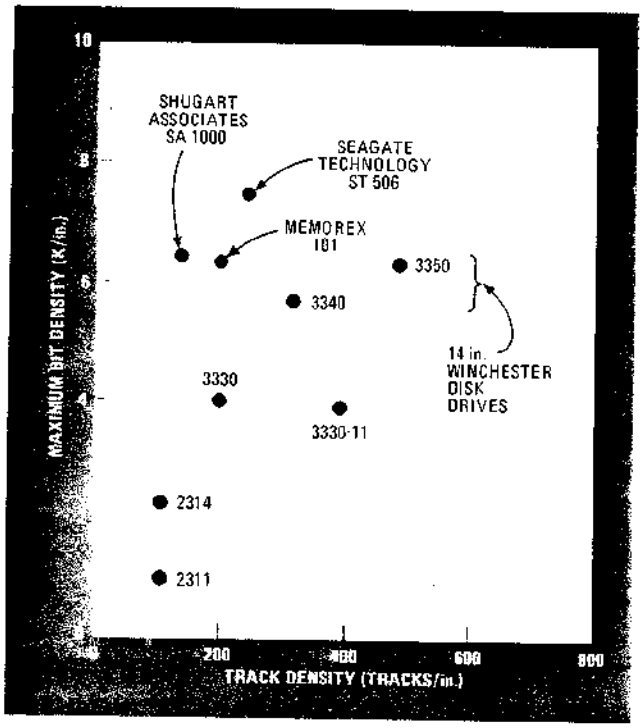
But the interfacing of the disks has presented a unique challenge to system designers, since the cost of compressing the available interfaces and controllers of large Winchester disk systems into limited volumes on a small-system budget is obviously prohibitive. The MSC-9000 controller module offers a solution to the problem. Truly universal and requiring only a little external custom circuitry on a small printed-circuit board, the module (Fig. 1) has all the features for controlling any small Winchester drive at a low cost of about $300 to $400.

It is the need for increased capacity that has pushed the development of low-cost, rigid-disk drives. Moreover, these disks, which measure from 5.25 to 8 inches in diameter, are more reliable than floppy disks.

The new drives borrow their technology from the

1. Disk control. The 3-by-3-inch potted module shown to the right contains over 75% of the functions needed to control most Winchester disk drives. The controller module replaces all of the small- and medium-scale ICs on the large printed-circuit board shown to the left.





2. Bit cramming. The new-generation 5.25- and 8-in. Winchester disk drives (shown by the colored dots) have a bit density equal to or exceeding that of earlier rigid-disk drives, and this density means that timing becomes extremely critical in their control.

earlier and more sophisticated 14-in. Winchester drives, which have sealed disk, head, and positioning assemblies; lubricated disk surfaces; and heads that rest on a disk surface when the drive is stopped. Like their predecessors, they have read/write heads that fly close to the disk surface, a thin magnetic oxide surface coating, and light head-loading force and weight.

The sealed—and hence clean—environment, the new head and loading designs, and the lubrication have improved reliability and reduced head crashes in the new drives. The disks have from 5 to 60 times the storage capacity of floppy disks in the same space and access data four times faster. They also weigh less, take up less space, and use less power than the earlier 14-in. drives of the same type and thus bring high-technology storage within the economic reach of the small-system designer.

The floppy-disk replacement market consists of small business and office systems, personal computers, process control systems, word processors, and intelligent terminals. The entire disk drive structure is designed for this market, from its physical size and mounting configuration to its interface protocol. Currently, the most popular Winchester drives have electrical interfaces that are very similar to those of a floppy-disk drive, with the controller supplying step pulses and a direction signal for the seek function and a head select and read/write enable for the data-transfer function.

It now remains to be seen if the popular drive manufacturers will switch to the soon-to-be-finalized interface standard of the American National Standards Institute, which will use a single 50-conductor flat cable, or if the ANSI standard is doomed to become a standard in name only. But regardless of interface electrical standards, the selection of controller functions depends primarily on the drive technology. The interface protocol is where the similarity ends between the floppy-disk and new generations of rigid-disk drives.

Keeping it small

Because of the control complexity required, it is not possible to design a Winchester-drive controller around a single large-scale integrated chip, as in some floppy-disk applications. But 75% of the controller circuitry required universally can be packaged in a 3-by-3-by-0.65-in. module such as the MSC-9000. LSI circuits permit its disk input/output processor to reach this small size, and a single-card controller can be designed by adding customizing circuitry.

Because of the compact simplicity of these drives, a system designer might assume that floppy-disk controller features would suffice. However, astute designers are now realizing that the controller functions of the small-system models must be no less sophisticated than those of the higher-capacity Winchester disk systems. Though drive manufacturers have significantly reduced cost by simplifying the head-movement mechanisms and reducing the number and size of parts, packing densities have approached or even exceeded densities (bits per square inch) of much larger disks (Fig. 2), so that the new drives are actually more complicated.

Slowing down the data

The new Winchester disk's high density gives it a data rate of about 600 kilobytes per second typically. Since these drives will most likely be attached to small computers with low-bandwidth I/O systems, the controllers must have data buffering to present the data at a rate acceptable to most host computers. A data buffer solves the problem of I/O bandwidth compatibility, but it immediately creates another.

Nothing can stop a rotating memory from transferring data at its desired rate—and while the controller is exchanging data between its buffer and the host memory at the acceptable rate, the disk keeps spinning. Additional blocks of data pass under the heads, unable to be accepted. Waiting for the next disk revolution would result in a significant and undesirable performance loss.

This highlights another requirement, sometimes called sector interleaving, which allows transfer of multiple sectors within one disk revolution while a slow host data rate is maintained (assuming the controller has the proper data buffer). Logical sequential sectors are not contiguous and the effect on access time of the disk motion can be cut by interleaving physical sectors so that the head comes close to the next logical one.

Another obvious result of the high-density disk is the necessity for error correction. In 1974, when Control Data Corp. first introduced its storage module drive (SMD) technology—applying IBM's 3330 technology—most system architects simply said, "I know the large systems have error correction, but this is a small system. I just can't afford such controller complexity, and besides, I'm getting good disk packs." And they had so far gotten good disks, but as production stepped up and

disk packs got heavy use, the data bases turned up with unrecoverable errors.

The bit-per-inch density of the new drives demands controllers with sophisticated error correction—using, ideally, 32-bit polynomials capable of detecting burst errors up to 22 bits long and of correcting burst errors up to 11 bits long. Since the platters and heads are sealed and fixed, a bad disk pack cannot simply be thrown away. The controller, to increase yield and lower cost, should therefore allow defect skipping in the form of either spare tracks or spare sectors. The spare sector scheme is preferable, as it avoids performance losses inherent in the spare track method due to its seek delay.

In a moving-head system, the head might land on the wrong track (often called "seek error" and assigned a probability of 1 in 10^6). This error can be compensated for by having a controller generate a disk-identifying field (commonly called a header or ID field) to verify that the drive is on target before permitting the host to transfer data.

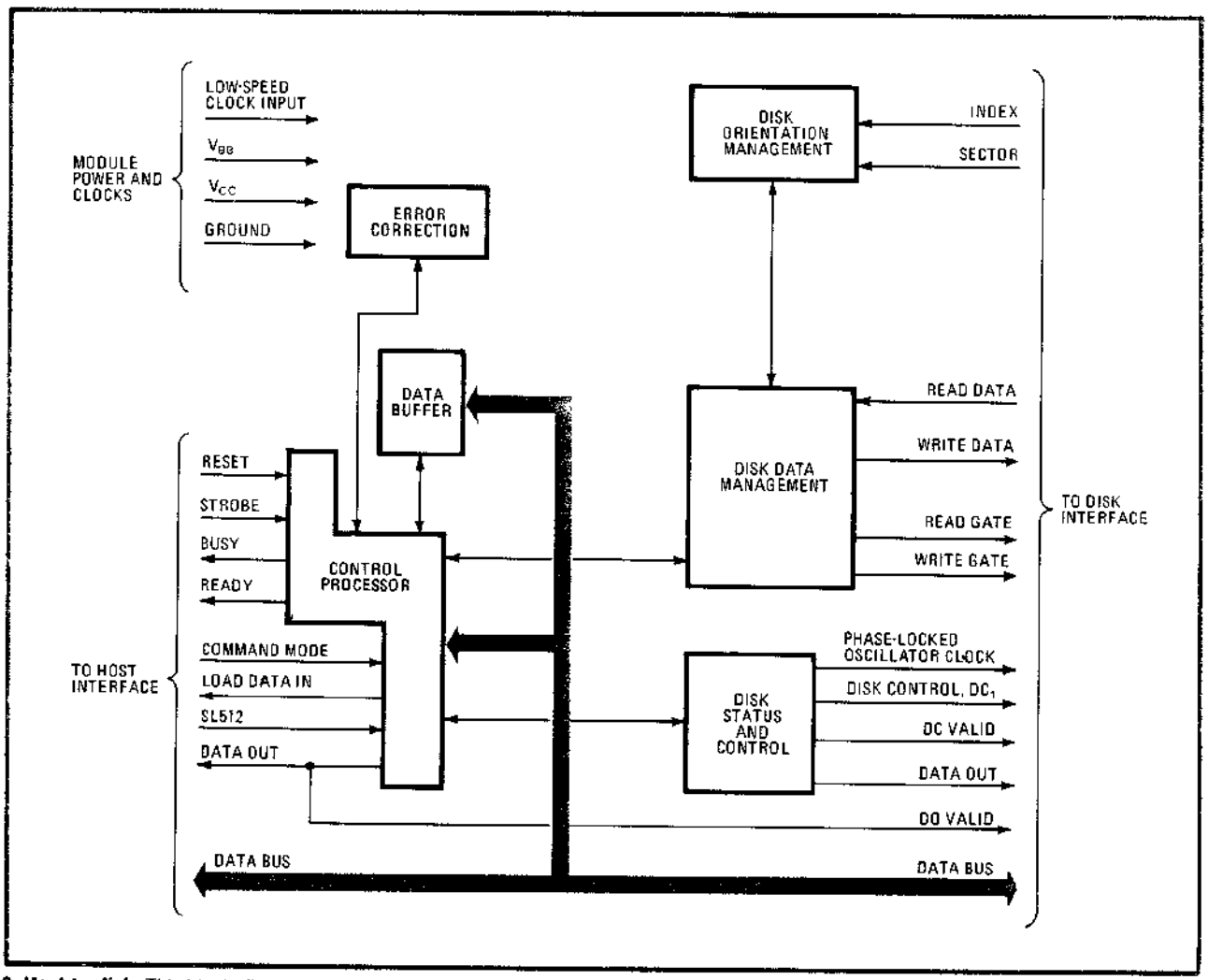
Another controller requirement is a high-level interface to allow simple communication between the host and the disk I/O processor rather than directly between the host and the device. The host computer is thus free of detailed disk control and able to handle more of its data-processing tasks, such as report generation and accounts payable and receivable. Like their mainframe predecessors, small computer systems should not be moving or selecting disk heads, calculating polynomials, making sure the one-in-a-million seek error did not occur, or even formatting the disk.

Since a cost-efficient Winchester-drive controller must have an on-board microcomputer, its excess computer power can also be used for various forms of module self-testing and diagnostics. The extra read-only memory space required for these routines is a small price to pay for improved system availability.

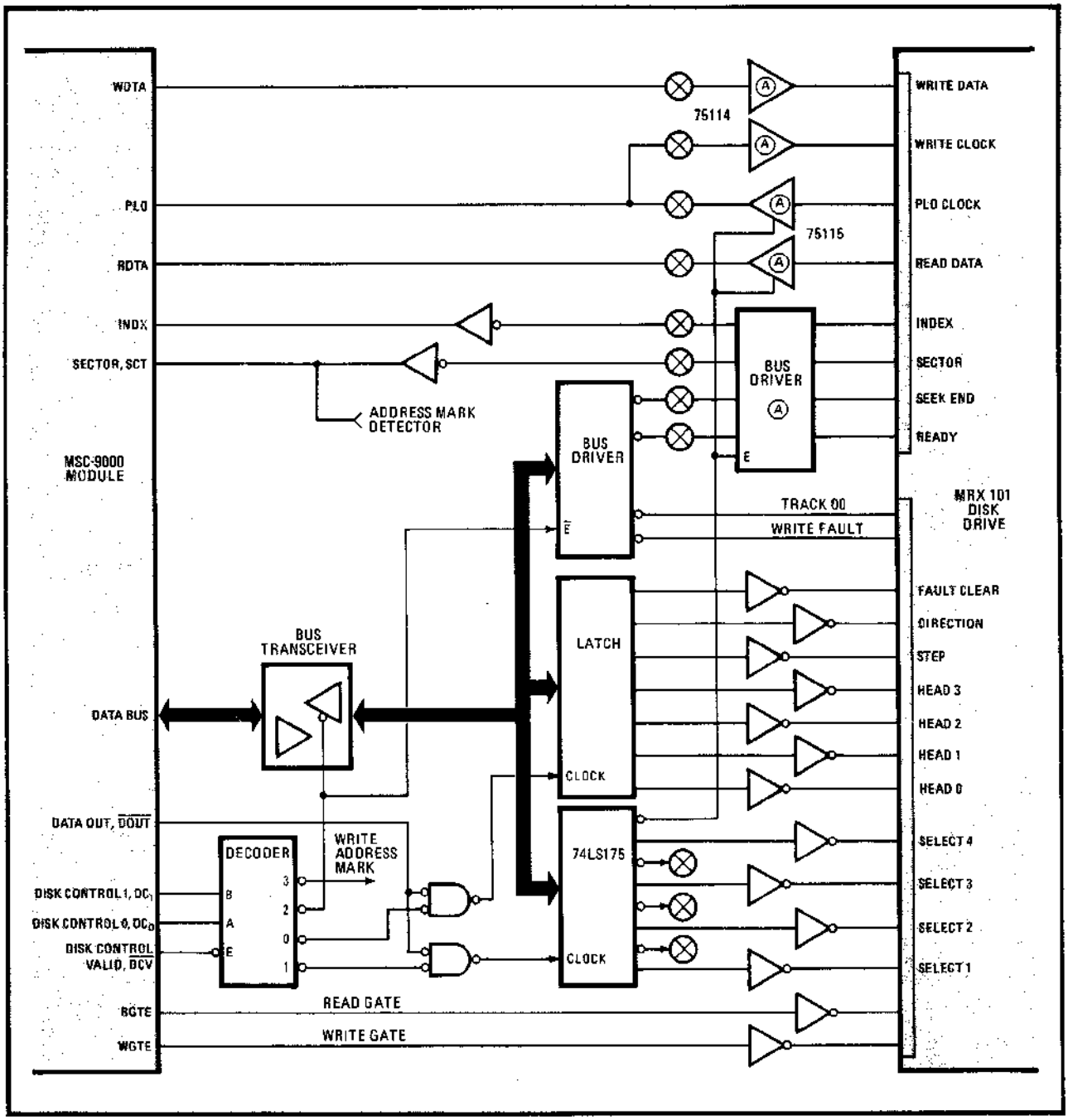
Developing a controller

Of the four ways to develop a Winchester drive controller, one might be to build the controller circuitry out of low-cost small-scale integrated logic. Unfortunately that requires the use of 300 to 400 integrated circuits—a size disadvantage.

Two approaches that are more up to date use either a high-speed bit-slice bipolar processor or an MOS micro-



3. Host to disk. This block diagram of the MSC-90000 controller shows its major components—MOS microprocessor, memory, and various LSI interface circuits. The module does most of the interfacing between the large host computer and the disk drive or drives.



4. Module to drive. Additional external circuitry is needed to interface the controller module with various types of rigid-disk drives. This interface is with a Memorex 101 disk drive. If the gates marked with an A are duplicated, it is possible to control four drives at the same time.

processor. The bit-slice technology has been applied to small-system disk controllers for the past several years. But since it requires 150 to 250 ICs (depending on host computer and system demands), it is not cost-effective for systems that use disk drives priced at even as little as $5,000 to $10,000.

Applying this type of circuitry to the new Winchester disk drives results in only minimal cost reductions, since all major functions are still absolutely necessary. The new drive's electrical interface may not require differential line drivers and receivers, but interface circuits would be the only savings. So a gross imbalance will exist—a controller that costs more than the drive and uses more room and even more power than the drive.

An MOS microprocessor used alone is undeniably a fine choice for controlling a sewing machine, a microwave oven, or a video game. However, in the real-time environment of the memory spinning at 3,600 revolutions per minute, with a new bit to contend with every 200 nanoseconds, it falls a bit short. In fact, it falls many bits short. Thus auxiliary circuitry must be introduced to augment the microprocessor with real-time, critical functions such as serial-parallel-serial conversion, polynomial generation, error detection, data synchronization,

address verification, and overall system synchronization. This requirement can result in a board or boards with the microprocessor and its associated memory and at least 130 ICs.

Only the modular approach is practical in terms of package size and cost, and the MSC-9000 module represents a "best of both worlds" solution. An MOS processor and an LSI implementation of the added real-time functions can be combined on a single board, with the further addition of circuitry for interfacing with specific disk drives.

An example of such additional circuitry would be that needed to multiplex signals between multiple disk units and the controller module. In another case, the disk drive being controlled might require nonreturn-to-zero (NRZ) data. Then, a data separator to convert modified-frequency-modulation data into an NRZ format can be implemented on the controller board or in the disk drive.

The module itself features error correction, data buffering, address verification, and even automatic retry when the module detects an error, as well as the small size required for packaging in the new compact drives.

Keeping the host free

The host processor and the module communicate via an interprocessor communication link. The module goes busy until the task is completed and then the resulting status can be obtained. The command software set is organized to keep commands at a level of simplicity so that the host need not be involved in the disk mechanism characteristics.

For example, to read or write a block of data, the host simply sends the command and address. The module will locate the address, performing a seek if required, and wait for the proper rotation position before transferring data. On a read, the controller verifies accuracy and automatically corrects errors (up to 11 bits) before passing data to the host. Other commands, such as seek, track format, recalibrate, and write alternate sector, are included for utility and error management. Commands such as module self-test, write long, and read long ensure that the controller can be easily maintained.

A functional block diagram of the MSC-9000 disk I/O processor is shown in Fig. 3. The module has an 8-bit bidirectional data bus and some associated control signals for all communications with either the host or the disk drive or drives.

One disk interface

Figure 4 shows all external circuitry required to interface the control module with a single Memorex 101 disk drive. The decoder controls bus steering and defines what information passes over the module data bus. External latches strobe the drive select, head select, and motion control signals. Circuitry can be expanded to four disk drives by duplicating the A circuits (upper right in the figure) and enabling them from the select signals of the 74LS175. The Memorex 101 drive contains a data separator, but the function can also be included on the controller board.

To accommodate disk drives without sector pulses, the module generates a write AM (address mark) signal via the external decoder. External circuitry is needed to use this signal to generate address marks. When detecting an address mark, this circuitry routes the AM detect signal into the sector pulse input to the module. During the format operation, the module segments the disk into fixed sectors by using the address marks in place of sector pulses.

The module receives data in NRZ format synchronized with the PLO (phase-locked oscillator) clock input. During a write, the module generates NRZ data on its WDTA (write data) line, also synchronized with the PLO clock.

The host interface

The controller module can be thought of as a disk I/O processor. It can be given a task and, except for transferring the actual disk data, requires no further interaction with the host until it completes the task. Three groups of module signals provide the interface facility between the module and host computer.

The first group contains a busy signal and a command request signal. The busy signal signifies when the module begins performing a task; it remains active until a task is completed. The command signal can be given by the host (whenever the module is not busy) to cause the module to accept a new command.

The second group of signals initiates the transfer of information such as commands, drive status, or actual disk data between the module and the host. This group consists of:

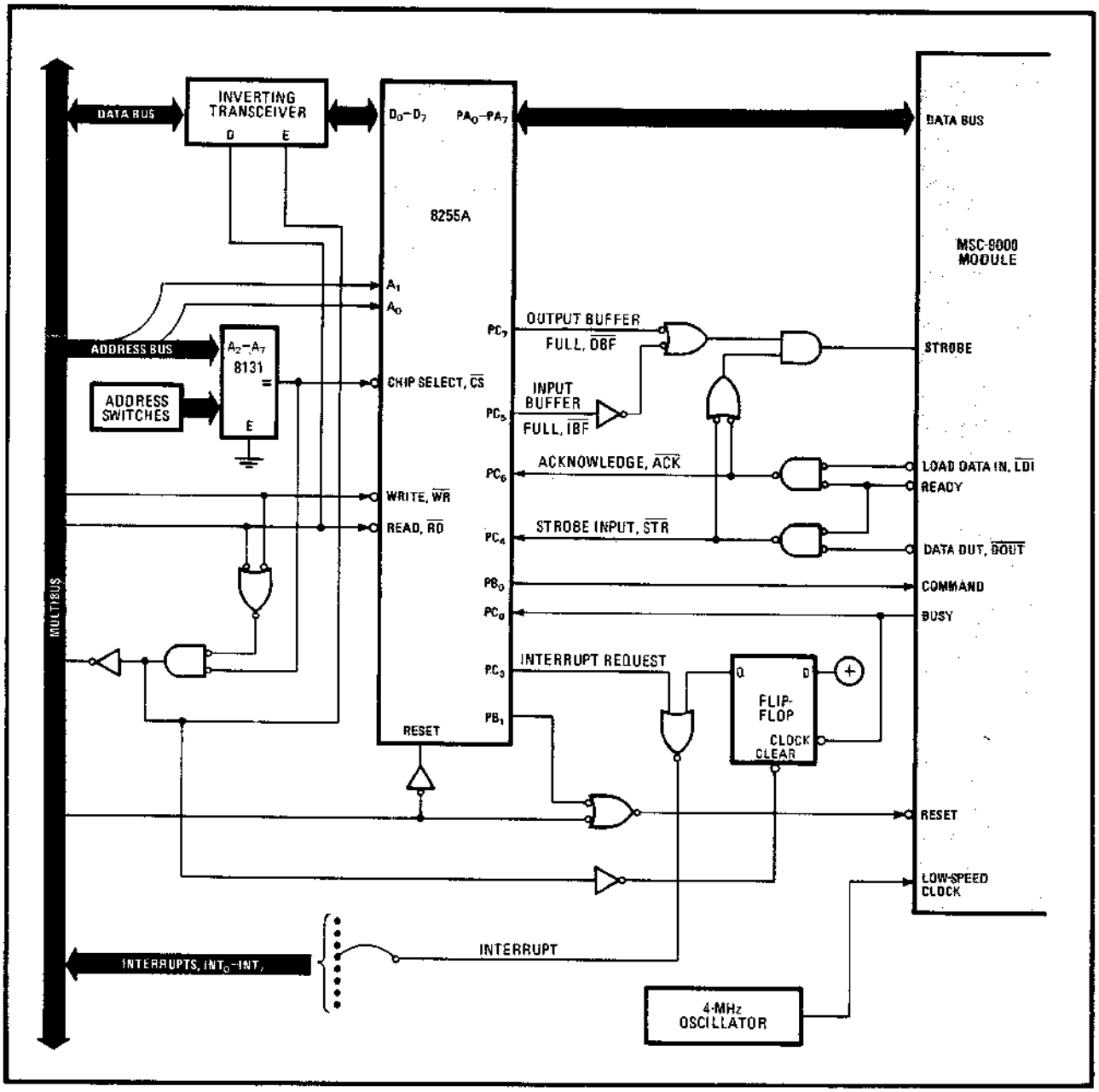
- Signals over an 8-bit, bidirectional data bus.
- Ready: a signal from the module that it is ready to transfer information to the host.
- Load data in: a signal from the module to be used with ready to enable the host interface to place information on the data bus for input to the module.
- Data out: a signal from the module to be used with ready to let the host interface receive information from the module.
- Strobe: a signal from the host interface to be put into the module whenever information is transferred with it.

The last group of signals, for general support of the module, consists of the following:

- Reset: a general reset input signal to the module.
- LCLK: a low-speed clock input to the module at 4 megahertz.
- Power: a +5-volt dc power source.

Figure 5 shows an interfacing example between the MSC-9000 module and a host system with a Multibus. The interface uses an Intel 8255A programmable peripheral interface (PPI) with three ports for communication between the host and the module. Port A is used to send data to or to receive data from the module. Port B is always an output port to generate control signals such as PB0, to request the module to accept another command, or PB1, a pulse to allow the host program to reset the module. Port C is a multifunctional port that handles various control signals, like the busy signal, PC0, or PC3, an interrupt signal.

Of course, by adding a direct-memory-access controller such as Intel's 8237, higher performance with less software overhead can be achieved. A floppy-disk controller IC can also be added on the controller board along



5. Bused disks. As illustrated, the controller module is interfaced with a host using the Multibus. An external 8255A programmable peripheral interface must be added to the controller card to make three ports for communicating data and control signals between host and module.

with the MSC-9000, creating a single controller for both Winchester and floppy disks.

Software for the Multibus application will talk mainly to the ports of the PPI rather than to the inputs to the module. Software requirements in other systems depend somewhat on the actual host-to-controller interface protocol. However, at a high level the structure is basically the same. The disk driver is entered whenever the system requires a transfer of disk data. The driver then invokes the command function and communicates the 6-byte command description to the module. After that, the DMA can be started and, upon leaving the disk driver, the system can proceed with its normal processing chores. An interrupt can easily be configured at the end of the module's busy cycle. Finally, the driver can be reentered to examine the status and signify a successful operation or ask for appropriate error-recovery procedures.

In some system architectures (notably single-task, single-user), the additional DMA circuitry may not be needed, since average seek delays account for the biggest portion of the data-access time. (Regarding average seek delays to the typical record, even non-DMA transfers of 5 microseconds per byte will contribute only a few percent.) So if the system has nothing else to do while waiting for transfer of the disk data, the software may as well perform the transfer in the driver. In either implementation (with or without DMA), software requirements to support the module stay compact and simple. □