MORROW DESIGNS

DISCUS M10--DISCUS M20--DISCUS M26

WINCHESTER DISK SYSTEMS

USER'S MANUAL

Preliminary

Introduction. 1
The System Bootstrap. 3
  Implementing the Bootstrap. 3
  Bootstrap Software. 4
System I/O Requirements 6
  I/O Driver Jump Table 7
  Installing I/O Drivers. 8
  I/O Driver Specifications 8
Programming Specifications. 10
  The Dispatch Jump Table 10
  Disk Utility Subroutines. 12
Hardware Level Registers. 19
  I/O Addressing. 19
  I/O Register Map. 20
  Readable Registers. 20
  Read/Write Registers. 23
  Write Only Registers. 23
  Controller Initialization 28
  Interrupts. 29
  Software Drivers
  Warranty
  Schematics

INTRODUCTION

The Discus M26, M20, and M10 are complete mass storage subsystems each consisting of three main components: a winchester type hard disk drive with a formatted storage capacity of 26 million bytes (M26) 20 million (M20) or 10 million bytes (M10) respectively, universal power supply which provides the necessary DC voltages to power the drive, and a controller which establishes a data channel between the disk drive and any S-100 based computer.

The drive and power supply are housed in a cabinet which has been designed for either table top or rack-mount operation. The only extra hardware required to mount the M26 drive in a standard 19" RETMA equipment rack is a pair of rack-mount slides which attach directly to the cabinet. The 8 inch M10 and M20 system is currently supplied in a desk top unit only. The physical dimensions are: 9.06 inches wide, 22.55 inches long and 5.05 inches high.

The controller plugs into any slot of the mother board of an S-100 system and is connected to the drive through a pair of flat cables. One of the cables carries clock and data information and the other carries control and status information. The controller can accommodate up to four drives, so the M26 system can be expanded to a total capacity of 104 million bytes, M20 to over 80 million bytes and the M10 to over 40 million.

The Disk Operating System software supplied with the Discus systems is the popular CP/M version 2.2. It was chosen because of the readily available application software that runs under CP/M. With CP/M, the owner of a Discus hard disk system has access to virtually any software that has been written for an 8080, 8085, or Z80 microcomputer.

A user can communicate with the Discus system on any of three levels: through CP/M, through low level software drivers which are included on the disk, or through direct commands to the Disk Jockey Winchester controller. The protocol for using the system through CP/M are covered in detail by the CP/M manual furnished under separate cover. The details of using the system through the low level software drivers and for using the hardware directly are covered in this manual.

A Winchester type sealed disk drive such as the one with the Discus M26 M20 and M10 has many advantages and also some limitations. Large capacity, low maintenance, and high reliability are the most obvious advantages. The major disadvantage is the fixed nature of the magnetic media. The media of a floppy disk drive can be removed and placed in a secure environment. This is not the case with the present generation of winchester disk drives. The media is sealed inside the unit and cannot be removed. Thus, as long as the drive is connected to a computer, the data written on the magnetic surface of the disk is never really secure.

The present solution to this data security problem is to backup the data periodically on some storage device which has removable media. The most practical device for this job at the present time is a floppy disk system. By using the facilities provided by CP/M, it is quite easy to transfer files from the Discus Hard drive to a floppy disk system such as the Discus 2D or the Discus 2+2. In the near future it will be possible to backup data on a hard disk by using small high density cassette tape drives. Presently the prices of these units are such that a backup system using them costs almost as much as the hard disk system itself. However, manufacturers are designing low cost tape drives that will allow backup tape systems to sell for approximately half the price of a hard disk system - a much better state of affairs. Morrow Designs plans to introduce a cassette tape backup system in 1981.

INTRODUCTION

The Disk Jockey/Winchester controller contains all the logic necessary to control the drive and to transfer data to and from the disk. It appears to the system as four I/O devices and has no memory in the address space of the CPU. Therefore some other element of the system has the responsibility for the initial load of the operating system and/or utility software from the disk. This initial loading operation is referred to as the bootstrap operation. The purpose of this section is to discuss several ways that the bootstrap operation can be implemented and to

IMPLEMENTING THE BOOTSTRAP

The code necessary to perform a bootstrap is roughly 100 bytes long. The least expensive but most troublesome method to bring up the system is to enter this code into memory and execute it. Fortunately there are several alternatives to this process which Morrow's will supply at nominal cost:

1. The bootstrap code can be programmed into any of the more common EPROMs: 1702, 5204, 2708, or 2716. Any of these parts are acceptable if supplied by the customer. If the part is to be supplied by Morrow Designs, it will be a 2708. The starting address of the program can be anywhere but must be supplied along with the order.

2. The program can be supplied on a CP/M compatible floppy diskette as a command file. In this case the program has a starting address of 100H (Hex) in accordance with the system requirements of CP/M.

3. When CP/M 2.2 is supplied with a Discus I or Discus 2D system, the CBIOS software of the CP/M contains a module which knows that a Discus M26 M20 or M10 system may be attached to the computer and treats it as drives E, F, and G (M26) and (M20) or drives E and F (M10). Drives A, B, C, and D are reserved for floppys.

THE BOOTSTRAP SOFTWARE

The program listed below will load the disk system software into memory. There is an option which will either load CP/M or only the utility low lever disk drivers. The program starts at location 100H (Hex) but can be easily changed to start at the beginning of any page by altering the third byte of the jump (and conditional jump) instructions to the value of the desired page.

```
                                1     *HARD DISK BOOTSTRAP SOFTWARE
001:000 303 000 003             2             JMP   START
                                3
001:003 001:375                 4             DS    509       room for boot
                                5
003:000 076 374                 6     START   MVI   A,DRIVEA  select
003:002 323 122                 7             OUT   FUNCTN    -drive A
003:004 076 005                 8             MVI   A,DRENBL  turn on drive
003:006 323 120                 9             OUT   CONTRL    -command register
003:010 333 120                 10    RLOOP   IN    STATUS    test for
003:012 346 040                 11            ANI   READY     drive A ready
003:014 302 010 003             12            JNZ   RLOOP
003:017 076 007                 13            MVI   A,DSKRUN  enable the
003:021 323 120                 14            OUT   CONTRL    -controller
003:023 333 120                 15    WAITZ   IN    STATUS    test for heads
003:025 037                     16            RAR   .         -at track zero
003:026 322 053 003             17            JNC   SDONE
003:031 076 370                 18            MVI   A,STEPO   execute
003:033 323 122                 19            OUT   FUNCTN    -the
003:035 076 374                 20            MVI   A,DRIVEA  -step out
003:037 323 122                 21            OUT   FUNCTN    -command
003:041 333 120                 22    WAITC   IN    STATUS    wait for
003:043 346 004                 23            ANI   COMPLT    -the seek
003:045 312 041 003             24            JZ    WAITC     -to complete
003:050 303 023 003             25            JMP   WAITZ
003:053 333 120                 26    SDONE   IN    STATUS    get an image
003:055 117                     27            MOV   C,A       -of the status reg
003:056 333 120                 28    IWAIT1  IN    STATUS    wait for
003:060 221                     29            SUB   C         -the index pulse
003:061 312 056 003             30            JZ    IWAIT1    -to arrive
003:064 333 120                 31    IWAIT2  IN    STATUS    wait for the
003:066 221                     32            SUB   C         -next index pulse
003:067 302 064 003             33            JNZ   IWAIT2    test for head settle
003:072 076 010                 34            MVI   A,HEADER  reset the
003:074 323 121                 35            OUT   COMMD     -buffer pointer
003:076 257                     36            XRA   A         -to header area
003:077 323 123                 37            OUT   DATA      head 0
003:101 323 123                 38            OUT   DATA      track 0
003:103 076 001                 39            MVI   A,SECTOR  1 for CP/M
003:105 323 123                 40            OUT   DATA      -or 30 for drivers
003:107 076 200                 41            MVI   A,SYSTEM  system key
003:111 323 123                 42            OUT   DATA
003:113 076 001                 43            MVI   A,DREAD   issue a
003:115 323 121                 44            OUT   COMMD     -read command
003:117 333 120                 45    WAITD   IN    STATUS    wait for command
003:121 346 002                 46            ANI   OPDONE    -to complete
003:123 312 117 003             47            JZ    WAITD
003:126 333 123                 48            IN    DATA      low order byte of
003:130 157                     49            MOV   L,A       -bootstrap address
003:131 137                     50            MOV   E,A
003:132 333 123                 51            IN    DATA      high order byte of
003:134 147                     52            MOV   H,A       -bootstrap address
003:135 127                     53            MOV   D,A
003:136 333 123                 54    LLOOP   IN    DATA      load
003:140 022                     55            STAX  D         -the
003:141 034                     56            INR   E         -bootstrap
003:142 302 136 003             57            JNZ   LLOOP
003:145 351                     58            PCHL  .         branch there
```

```
                                1   *HARD DISK BOOTSTRAP SOFTWARE
0100    C3 00 03                2           JMP  START
                                3
0103    01FD                    4           DS   509        room for boot
                                5
0300    3E FC                   6   START   MVI  A,DRIVEA  select
0302    D3 52                   7           OUT  FUNCTN    -drive A
0304    3E 05                   8           MVI  A,DRENBL  turn on drive
0306    D3 50                   9           OUT  CONTRL    -command register
0308    DB 50                  10   RLOOP   IN   STATUS    test for
030A    E6 20                  11           ANI  READY     drive A ready
030C    C2 08 03               12           JNZ  RLOOP
030F    3E 07                  13           MVI  A,DSKRUN  enable the
0311    D3 50                  14           OUT  CONTRL    -controller
0313    DB 50                  15   WAITZ   IN   STATUS    test for heads
0315    1F                     16           RAR  .         -at track zero
0316    D2 2B 03               17           JNC  SDONE
0319    3E F8                  18           MVI  A,STEPO   execute
031B    D3 52                  19           OUT  FUNCTN    -the
031D    3E FC                  20           MVI  A,DRIVEA  -step out
031F    D3 52                  21           OUT  FUNCTN    -command
0321    DB 50                  22   WAITC   IN   STATUS    wait for
0323    E6 04                  23           ANI  COMPLT    -the seek
0325    CA 21 03               24           JZ   WAITC     -to complete
0328    C3 13 03               25           JMP  WAITZ
032B    DB 50                  26   SDONE   IN   STATUS    get an image
032D    4F                     27           MOV  C,A       -of the status reg
032E    DB 50                  28   IWAIT1  IN   STATUS    wait for
0330    91                     29           SUB  C         -the index pulse
0331    CA 2E 03               30           JZ   IWAIT1    -to arrive
0334    DB 50                  31   IWAIT2  IN   STATUS    wait for the
0336    91                     32           SUB  C         -next index pulse
0337    C2 34 03               33           JNZ  IWAIT2    test for head settle
033A    3E 08                  34           MVI  A,HEADER  reset the
033C    D3 51                  35           OUT  COMMD     -buffer pointer
033E    AF                     36           XRA  A         -to header area
033F    D3 53                  37           OUT  DATA      head 0
0341    D3 53                  38           OUT  DATA      track 0
0343    3E 01                  39           MVI  A,SECTOR  1 for CP/M
0345    D3 53                  40           OUT  DATA      -or 30 for drivers
0347    3E 80                  41           MVI  A,SYSTEM  system key
0349    D3 53                  42           OUT  DATA
034B    3E 01                  43           MVI  A,DREAD   issue a
034D    D3 51                  44           OUT  COMMD     -read command
034F    DB 50                  45   WAITD   IN   STATUS    wait for command
0351    E6 02                  46           ANI  OPDONE    -to complete
0353    CA 4F 03               47           JZ   WAITD
0356    DB 53                  48           IN   DATA      low order byte of
0358    6F                     49           MOV  L,A       -bootstrap address
0359    5F                     50           MOV  E,A
035A    DB 53                  51           IN   DATA      high order byte of
035C    67                     52           MOV  H,A       -bootstrap address
035D    57                     53           MOV  D,A
035E    DB 53                  54   LLOOP   IN   DATA      load
0360    12                     55           STAX D         -the
0361    1C                     56           INR  E         -bootstrap
0362    C2 5E 03               57           JNZ  LLOOP
0365    E9                     58           PCHL .         branch there
```

INTRODUCTION

To do useful work, the operating system must communicate with at least one other I/O device besides the disk. This device is called the SYSTEM CONSOLE and generally consists of a video display and a keyboard. This device allows the user and the operating system to communicate with each other. Depending on overall requirements it may be necessary to attach other I/O devices such as a printer or a modem to the system.

I/O devices vary greatly in their electrical and mechanical characteristics. To communicate with them, allowances must be made for these variances. In order that the operating system be flexible, it must remain aloof to the peculiarities of the devices that it communicates with. Thus, each time a new type of I/O device is connected to the system a communications problem is created. It is resolved by a software module called a "driver". The driver must accept commands from the operating system and translate them into a form that the I/O device will accept. The I/O device is usually connected to the system through an interface which translates the computer's logical signals into the proper electrical signals the device can understand. Generally this interface is a circuit board that plugs into the bus of the computer and is connected to the device through a cable.

In order to create a driver, the user needs to have detailed knowledge concerning (1) how information is passed back and forth between the device and the interface, (2) how the interface and the computer communicate, and (3) how the operating system passes information back and forth to the driver.

Drivers vary a great deal in length and complexity. However, irrespective of their size or intricacy, there is an important observation to be made about the creation of a driver: someone who does not possess a clear and detailed understanding of how BOTH the device and its interface function should never attempt to write the software to drive it. This type of task also presupposes considerable skill in assembly language programming. Faulty or incomplete knowledge of the device or its interface usually results in a program that doesn't work or, worse yet, only partially works. A great deal of time can be wasted and equipment may be damaged in trying to make a bad driver function. If there is any question in the user's mind about how a driver should be written, he should consult with the personnel at the computer store where the system was purchased. In many cases, the driver software will be a stock item and can usually be installed before the system leaves the store.

In many areas there are groups of computer enthusiasts who have joined together in clubs to share knowledge and exchange information. These organizations usually meet at regular intervals and provide an ideal environment to gain experience and sharpen skills in all areas of computing. The user who wishes to learn the art of creating device drivers will benefit from joining one of these groups because of the wide range of skills its members possess.

This section of the manual discusses how information is passed back and forth between the operating system and the driver. Also presented are several sample drivers for the system console which interface to some of the more common S-100 computers.

THE I/O DRIVER JUMP TABLE

CP/M maintains a table of jump instructions in high memory. Each entry in this table points to an I/O driver that the operating system may have occasion to use. There are 17 entries in this table. Only 6 of these are of concern to the user since the others point to disk drivers. The table, as it appears in memory, is presented below. The entries marked with an asterisk are the ones of interest to the user and will be discussed in detail.

```
 JMP BOOT      ;arrive here from a cold start load
 JMP WBOOT     ;arrive here for warm start
*JMP CONST     ;check for console character ready
*JMP CONIN     ;read console character in
*JMP CONOUT    ;write console character out
*JMP LIST      ;write listing character out
*JMP PUNCH     ;write character to punch device
*JMP READER    ;read the reader device
 JMP HOME      ;move to track 0 on selected disk
 JMP SELDSK    ;select disk drive
 JMP SETTRK    ;set track number
 JMP SETSEC    ;set sector number
 JMP SETDMA    ;set data transfer address
 JMP READ      ;read selected sector
 JMP WRITE     ;write selected sector
 JMP LISTST    ;return list status
 JMP SECTRAN   ;sector translate subroutine
```

At location 0 in memory CP/M maintains a jump instruction to the second entry of this table - JMP TABLE+3 (to WARM BOOT). The value in location 2 is the page number that table starts on. In almost all operating systems, the jump table starts at the beginning of a page which means the value of location 1 will be 3.

The DISCUS M26 M20 and M10 disk systems are shipped with two types of I/O configurations:

1. Preinstalled drivers which interface to
   a. DISCUS 2D floppy disk controller
   b. SWITCHBOARD I/O controller board
   c. SOL computer
   d. EXIDY computer

2. No installed drivers - each entry in the table marked with an asterisk is a "jump to self" instruction

INSTALLING I/O DRIVERS

In systems which have no installed drivers, there is a 512 byte section of memory immediately following the jump table reserved for drivers user needs to furnish. In some instances it may not be necessary to install all six of the drivers listed above. For instance, if there is no punch device in the system, the punch device driver would consist of a single RET(urn) instruction. The following procedure should be used to install the necessary drivers:

1. Study carefully the CP/M 2.2 Alteration Guide section of the CP/M documentation manual.

2. Code and assemble the drivers one by one.

3. Enter the assembled code into the reserved 512 byte memory area immediately following the jump table.

4. Alter the "jump to self" instructions so as to point to the newly installed drivers.

5. Test and debug the drivers.

6. When a driver functions correctly execute the SAVEUSER command. This command automatically alters the operating system so the driver is loaded back into memory when the system boots.

7. Save a copy of the source code for the drivers on a file on the disk. If the operating system is made larger (or smaller) the drivers can be reinstalled with a minimum of effort.

I/O DRIVER SPECIFICATIONS

All simple character I/O operations are assumed to be performed in ASCII, upper and lower case, with high order (parity bit) set to zero. An end-of-file condition for an input device is given by an ASCII control-z (1A hex). Peripheral devices are seen by CP/M as "logical" devices, and are assigned to physical devices within the CBIOS (see the CP/M documentation).

In order to operate, the system needs only the CONST, CONIN, and CONOUT drivers (LIST, PUNCH, and READER may be used by application programs, but not by the CCP). The LISTST entry is currently used only by DESPOOL, and may be a simple RET instruction.

DEVICES

CONSOLE The principal interactive console which communicates with the operator, accessed through CONST, CONIN, and CONOUT. Typically, the CONSOLE is a device such as a CRT or Teletype.

LIST The principal listing device, if connected to the system, is usually a hard copy device such as a printer or Teletype.

PUNCH The principal tape reading device, such as an optical reader or Teletype.

Note that a single peripheral can be assigned as the LIST, PUNCH, and READER device simultaneously. If no peripheral device is assigned as the LIST, PUNCH, or READER device, the driver that the user creates may give an appropriate error message so that the system does not "hang" if the device is accessed by PIP or some other user program. Alternately, the PUNCH and LIST drivers can just simply return, and the READER driver can return with a 1A (hex) in register A to indicate an immediate end-of-file.

DRIVERS

CONST Sample the status of the currently assigned console device and return 0FFH in register A if a character is ready to read, and 00H in register A if no console characters are ready.

CONIN Read the next console character into register A, and set the parity bit (high order bit) to zero. If no console character is ready, wait until a character is typed before returning.

CONOUT Send the character from register C in the console output device. The character is in ASCII, with high order parity bit set to zero. You may want to include a time out on a line feed or carriage return if your console device requires some time interval at the end of the line (such as a TI Silent 700 terminal). You can, if you wish, filter out control characters which cause your console device to react in a strange way (a control-z causes the Lear Seigler terminal to clear the screen, for example).

LIST Send the character from register C to the currently assigned listing device. The character is in ASCII with zero parity.

PUNCH Send the character from register C to the currently assigned punch device. The character is in ASCII with zero parity.

READER Read the next character from the currently assigned reader device into register A with zero parity (high order bit must be zero), and end-of-file condition is reported by returning an ASCII control-z (1AH).

INTRODUCTION

There are applications which require a more intimate degree of control over the disk than is possible through the operating system. On the other hand, it may not be desirable or necessary to communicate directly with the controller through hardware registers. A set of low level software drivers is included with the system to meet the foregoing needs. These drivers can be loaded directly from the system tracks through an option in the bootstrap loader. The source code has also been provided in a file called HDFIRM.ASM. This file can be accessed through the operating system if the user needs to modify it. The code has been assembled to run starting at 370:000 Octal (0F800 Hex). A change in the "origin" statement of the file will allow the code to be assembled for other starting address.

THE DISPATCH JUMP TABLE

To use the low level drivers, the user should branch to the appropriate address in a jump table in the first few words of the driver software. Since each subroutine ends with a RET(urn) instruction a CALL instruction should be used to branch to it.

The jump table contains jump instructions that point to the true address of a utility routine within the software. Having a jump table allows the individual routines to be updated and moved without having to change software that calls the routines. Let A represent the address of word 0 of the driver software. In the standard version, A = 370:000Q (F800H). The addresses to call for the utility routines are then:

ADDRESS	STANDARD VALUE		SYMBOLIC VALUE	FUNCTION
	Octal	Hex		
A	370:000	F800	TRKZRO	Recalibrate (seek to track 0)
A+3	370:003	F803	TRKSET	Seek to a track
A+6	370:006	F806	SETSEC	Select a sector
A+9	370:011	F809	SETDMA	Set the data transfer address
A+12	370:014	F80C	DREAD	Read a sector of disk data
A+15	370:017	F80F	DWRITE	Write a sector of disk data
A+18	370:022	F812	SELDRV	Select a disk drive
A+21	370:025	F815	DMAST	Read data transfer address
A+24	370:030	F818	STATUS	Disk status input
A+27	370:033	F81B	STHEAD	Select a read/write head
A+30	370:036	F81E	SETKEY	Set the "key" for a sector

The specific function of each subroutine is described below.

A subroutine completes by executing a RET instruction. If the routine completes normally, it returns with the carry flag cleared. If an error is detected, the carry flag is set. A map of the error conditions is placed in the CPU's A register. A program should always test the carry flag after a return from a disk utility subroutine and branch to an appropriate error handling routine if the carry flag is set.

DISK I/O

To understand the significance of the disk utility software, it is necessary to say a few words about how data is organized on the disk.

Information on the disk is organized into concentric tracks. The number of tracks varies depending on the DIscus System you have ordered. The M26 has 202 tracks while both the M20 and M10 contain 244 per platter. The disk read/write heads can be moved to any track by a series of step in or step out commands. A step in command moves the read/write heads one track towards the center of the disk. A step out command moves the heads one track away from the center of the disk. The numbering of the tracks is arranged so that track zero is farthest from the center.

There are two rotating platters inside the drive and both the upper and lower sides of the platter are used to store magnetic information Floating over each surface is a pair of read/write heads mounted side by side and spaced so that when the outer head is positioned over its inner most track, it is still farther away from the center of the platter than the inner head is when positioned over its outer most track. That is, track zero for the inside head is closer to the center of the platter than track 201 (M26) or 243 (M10\M20) is for the outside head. There are a total of eight heads on the M26 two for each surface. The M20 system contains a total of 8 heads per drive one for each surface. The M10 system contains a total of four heads one for each surface.

Once the read/write heads have been moved to the desired track, the rotation of the disk will move a circle of magnetic material beneath (or above) each of the heads. Within these eight circles data is recorded in distinct regions called sectors. The sector is the smallest amount of information that can be separately read from or written to the disk and each one contains 512 bytes of data.

In the header field which precedes the data field of a sector, the track number, the head number, the sector number, and data security information are recorded. During read or write commands, this header is read before data transfers take place.

The disk drive has a sensor that reports when the read/write heads are physically positioned at track zero. A series of step out commands must be issued by the controller until this status line becomes active. This operation will always position the head to the same physical track. The seek to track zero command is often called a recalibrate command and is a standard utility subroutine supplied with the disk software. Whenever the heads are moved to another track, the disk drivers must account for this change in position so that when read or write commands are issued, correct track information is passed to the controller.

Transferring a sector of disk data between memory and the disk involves the following steps (each corresponding to a subroutine call to the disk utility software, with the exception of error checking):

1. Specify the track number the read/write heads should be positioned over during subsequent data transfers between the disk and memory. There are a total of 202 tracks, numbered 0 through 201 on the M26. There are a total of 244 tracks, numbered 0 through 243 on the M10\M20.

2. Check for error conditions.

3. Specify a head to be selected during subsequent read or write operations. There are a total of eight and are numbered 0 through 7 on the M26/M10 The M10 has a total of 4 heads which are numbered 0 to 3.

4. Specify a sector number that will be involved in subsequent data transfers between the disk and memory. There are a total of 32 sectors numbered 1 through 32 for the M26. There are a total of 21 sectors numbered 1 through 21 for the M10\M20.

5. Check for error conditions.

6. Specify the starting memory address of the block of data to be transferred to or from the disk.

7. Perform the read or write operation.

8. Check for errors.

DISK UTILITY SUBROUTINES

TRKZRO - This subroutine positions the read/write heads to the outermost track of the disk platters: track 00. The track zero sensor is used to determine this positioning. Except for track 00, the drive has no way to know where the read/ write heads are positioned. It is one of the responsibilities of the disk utility software to always know over what track the heads are positioned. In general, when a drive is first selected, the track position of the heads is not known. Thus, the TRKZRO routine should be called. In fact, if there is ever any doubt about the position of the heads this routine should be called.

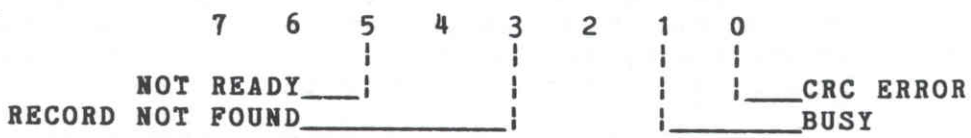
TRKSET - This routine will issue the proper commands to the drive to position the read/write heads over the track which is specified by the CPU's C register. The value in the C register should be between 0 and 201 (decimal) for the M26 and 0 to 243 for the M10\M20.. A value outside of these bounds will cause the routine to abort with the carry flag set and bit 6 of the A register set. A test is performed to make sure the controller is not busy processing data transfer command. Also, the status of the most recently selected drive is tested. If the controller is busy, or the drive is not ready, the carry bit is set and the routine aborts. As before, the A register will indicate the type of error that was encountered: if bit 1 is set, the controller was busy. If bit 5 is set, the drive was not ready. If there are no error conditions, the routine issues a series of step pulses to the drive so to move the read/write heads to the proper track. This series of step commands is issued much faster than the heads can move. This does not pose a problem however, since the drive has the ability to buffer and collect pulses that arrive too rapidly. This ability enhances the performance of a multiple drive system: after a series of step commands are issued, it is possible to deselect the drive and select another. In this way, one drive can be moving its heads to a new track while another is transferring data. This type of operation is called overlapped seek. The logic of the TRKSET routine has been designed to allow as many overlapped seeks to occur as is practical. Care has been taken so that waits encountered for head settle times are shared whenever possible.

SETSEC - This routine allows the user to specify what sector will be involved in the next data transfer operation between the disk and memory. The sector number is passed by the C register of the CPU. It should be between 1 and 32 (decimal) for the M26 and 1 and 21 (decimal) for the M10\M20. If the value in C is outside these bounds, the carry flag is set and the routine aborts.

SETDMA - During disk transfer operations, blocks of data move to and from the disk. These blocks are 512 bytes long. The starting address (in memory) of a data block that will be involved in the next disk transfer operation is specified by the contents of the B-C register pair when the SETDMA routine is called. The high order byte of the address is in the B register and the low order byte is in the C register. This routine cannot produce an error.

DREAD - This subroutine transfers information from the disk to memory. If the controller is busy or if the drive is not ready, the routine aborts with the carry flag set. Error information is detailed below. The drive involved in the operation is one specified by the most recent call to the SELDVR routine (see below). The position of the read/write heads is determined by the latest call to the TRKSET routine which involved the presently selected drive. The head number and sector number is given by the most recent calls to the STHEAD routine (see below) and SETSEC routine respectively. The starting memory address where the transfer will occur is specified by the most recent call to the SETDMA routine. If the drive is ready and the controller is not busy, DREAD issues a series of commands to the controller which will cause it to transfer information from the proper sector of the disk to its internal buffer. If any errors have occured, the carry flag is set and the routine aborts without transferring data from the controller to memory. If the transfer is free of errors, the data is moved from the controller's buffer into memory starting at the address specified by the last call to SETDMA.

"DREAD" REGISTER A ERROR BITS

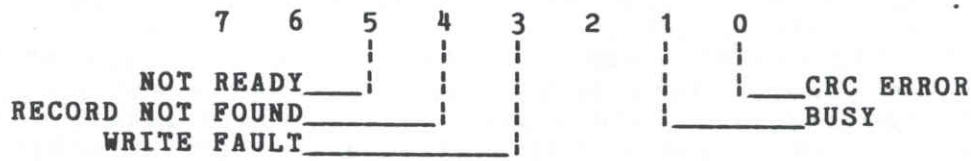


The "RECORD NOT FOUND" bit indicates that the external software has not selected the proper key to access the sector in question. If the "RECORD NOT FOUND" bit is set, the "CRC ERROR" bit should also be set. On rare occasions, error bits 0 and 3 will indicate that the header of the sector contains bad data or that a flaw exists in the magnetic media at this area of the sector. The bad data can be corrected or the media flaw can be detected through the use of diagnostic software covered in the next section of this manual. If CRC ERROR is 1 and the others bits are 0, an error was made in reading the data in the "data" area of the sector. When data is written to the disk, a binary polynominal is created from the serial stream as it is transferred to the disk. This polynominal is divided by a fixed prime polynominal of order 16 until a remainder of less than 16 is produced. The data bits of this remainder are appended to the end of the data

field. When a sector is read back from the disk, the same polynominal is recreated by the serial stream except that the remainder polynominal at the end is appended to the stream. Thus when the original prime polynominal is now divided into the new one, there will be a zero remainder if there have been no read errors. If there were read errors, the division will produce a remainder. If this remainder is non zero, the CRC error bit in the A register is set. The hardware in the controller which implements the CRC logic may not, at first glance, appear to function as described above. The hardware takes advantage of the fact that the division can be done while the polynominal is being created. Normally, when the hardware detects a CRC error, the calling software will try to re read the data. After 10 trys if the data is still bad, a hard error is reported. In this case, diagnostic software should be used to test the integrity of the magnetic media in this sector and place it in the bad sector file if necessary.

DWRITE - This subroutine transfers information from memory to the disk. If the controller is busy or if the drive is not ready, the routine aborts with the carry flag set. A map of the error bits is presented below. The drive involved in the operation is the one specified by the most recent call to the SELDRV routine (see below). The position of the read/write heads is determined by the latest call to the TRKSET routine which involved the presently selected drive. The head and sector number are given by the most recent calls to STHEAD (see below) and SETSEC routines respectively. The starting memory address where the transfer will occur is specified by the most recent call to the SETDMA routine. If the drive is ready and the controller is not busy, a block of data 512 bytes long is transferred to the controller's buffer from memory. DWRITE then issues a series of commands to the controller which cause the controller to write the data in its buffer to the proper sector on the disk. If any errors occur, the carry flag is set and the A register is loaded with the proper error bits.

"DWRITE" REGISTER A ERROR BITS

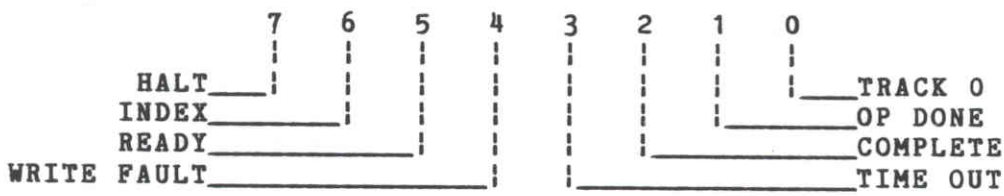


For disk write operations, the "CRC ERROR" and "RECORD NOT FOUND" bits should always be set together. This type of error condition is discussed above in the DREAD routine. The "WRITE FAULT" bit is an indication of an exceptional condition at the drive during the time the WRITE GATE signal is active. For details, see the drive manual included in the documentation. This bit should never be set if the hardware is functioning correctly and there are no faults in the cables which connecting the controller to the drive(s).

SELDRV - The value of the C register determines which one of four drives is to be selected. Only the two low order bits of register C are used for drive selection. The routine tests the "drive ready" status and delays approximately 2 minutes if the drive has not been selected before. The reason for this long wait is that it may take this long for the drive to stabilize when power is initially applied. If the drive is not ready, the routine returns with the carry flag set.

STATUS - The controller has two status registers. One is a full 8 bits wide while the other is only 2 bits wide. This routine reads the first status byte into the A register and the two bits of the second status register into the B register. The meaning of the various bits are detailed next.

THE A STATUS REGISTER



THE B STATUS REGISTER

HALT - When this bit is a 1, the controller is halted and not presently executing a command. When this bit is a zero, the controller is either preparing to execute a command or in the process of executing a command.

INDEX - The level of this bit changes whenever an index pulse is transmitted from the presently selected drive.

READY - When this bit is a 1, the presently selected drive is "ready" and can respond to commands from the controller. When this bit is a 0, the drive is not ready and will not respond to controller commands.

WRITE FAULT - When this bit is a 1, it indicates there was an exceptional condition present the last time the WRITE GATE signal to the drive was active. An example of an exceptional condition is that both READ GATE and WRITE GATE were active at the same time. It is possible that WRITE FAULT will be active when power is first applied to the drive. However, DRVSEL will always reset WRITE FAULT when the drive is initially selected. This bit indicates there is a hardware fault of some kind; either in the drive, the controller, or the connecting cables. Normally, this bit will be a 0.

TIME OUT - This bit is set to a 1 whenever the command the controller is executing takes longer than 8 revolutions of the disk. When the controller starts to execute a command, a counter is enabled which is clocked by index pulses from the drive. If the command is still in progress after 8 revolutions, the TIME OUT bit is set and the command in progress is terminated. If this bit is set after a transfer operation, it is an indication that the "key" field in the sector header on the disk does not match the key that the disk drivers have been given. When this bit is set, the RETRY bit in the B register should also be set. Normally, the "key" field of the sector header has a zero value. For the disk utility software discussed in this section of the manual the default value for the key is zero. Unless a call was made to the SET KEY special routine to change the key, the TIME OUT bit should always be zero. The one exception to this rule is if there is a hard data error in the header field of the sector. Usually this will mean there is a flaw in the magnetic media and this sector should added to the BAD SECTOR file.

COMPLETE - When this bit is a 0, there is a drive in the system which has received one or more step commands and is in the process of moving its heads from one track to another. A drive does not have to be selected to affect this bit. When this bit is a 1, all the drives in the system have completed their seeks.

OP DONE - When this bit is a 1, it indicates that the controller has completed some kind of transfer command. Unlike HALT, this bit will be reset whenever a command is issued to the controller - even a NOP command. Once reset, it will remain 0 until another transfer operation is completed.

RETRY - When the retry bit is set, a CRC error of some kind was made during the most recent transfer operation between the disk and the controller. If the CRC error was in the header area of the sector, the TIME OUT bit will also be set. If the error was in the data area of the sector, the TIME OUT bit will be zero. Once set, this bit will remain set until a transfer operation occurs in which there is not a CRC error.

SEEK DONE - This bit is set whenever the COMPLETE bit makes a transition from zero to one. As with OP DONE, it is reset by any command to the controller.

STHEAD - This routine selects one of eight read/write heads on the M26\M20 or one of four read/write heads on the M10. The least significant three bits of the C register determine which head will be selected. The heads are numbered 0 to 7 on the M26\M20 and numbered 0 to 3 on the M10 disk systems. Once a head is selected it will remain selected, even if a different drive is selected. No errors are reported by this routine.

SETKEY - This routine is used to pass a new sector access key to the disk utility software. There are six bytes in the header field of a sector. These are detailed below:

HEAD	TRACK	SECTOR	KEY	CRCHI	CRCLO

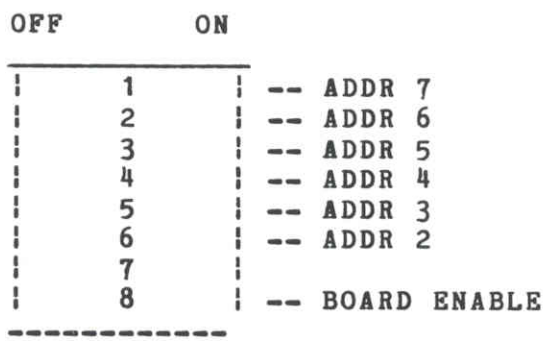
When the controller is issued a read or write command, it scans sector headers until it finds the correct one. However, it demands more than just the correct sector number. The number of the selected head must agree with the value of the first byte of the header field. The track number that the heads are positioned over must agree with the value of the second byte of the header field. The value of the fourth byte of the header field must be a zero or must agree with the what was passed in the C register during the most recent call to the SETKEY routine. Finally, the value of the two CRC error detection bytes must produce a zero remainder for the controller hardware. If all these requirements are satisfied, the controller will proceed to transfer a sector of data to or from the disk. Thus, the "key" field of the sector header allows for 256 levels of security for each sector of data. Except on track 0, this "key" field is normally a zero. By using the software described in the next section, this field can be altered and even read. The purpose of the SETKEY routine is to allow a user access to sectors which have a key different from zero.

INTRODUCTION

Users desiring a greater degree of control over the disk than afforded by the software drivers described previously may wish to refer directly to the I/O device registers on the Disk Jockey Hard Disk Controller. There are seven one byte registers. Three are read only, three are write only and one is read/write. These registers occupy four locations in the I/O address space of the system. They may appear anywhere in this space. The only restriction being that the lowest of the four addresses must be divisible by four.

I/O ADDRESSING

At location 8C on the circuit board, there is an eight position DIP switch used to determine the starting address of the controller. One of the switches is not used and another serves as a board enable. The other six are connected to a comparator which compares switch settings with I/O addresses on the bus. If there is a match and the board is enabled, I/O commands are allowed to access the controller. Below is a layout of this DIP switch.



As an example of addressing the Disk Jockey HD Controller, the following switch settings will address the board to respond to I/O addresses 120Q through 123Q (50H - 53H):

SWITCH	SETTING
1	ON
2	OFF
3	ON
4	OFF
5-8	ON

I/O REGISTER MAP

Let A represent the address of register 0 of the controller. In boards with standard addresses, A = 120Q (50H). The addresses of the controller registers are then:

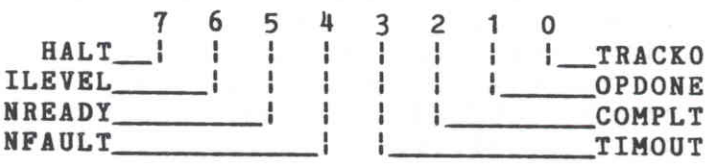
Address	Standard Value Octal	Hex	Symbolic Value	Function
A	120	50	STATUS/CONTL	Status/Control Port
A+1	121	51	COMD\RESUL	Command\Aux Status Port
A+2	122	52	FUNCTN	Drive Function Port
A+3	123	53	Data	Controller Data port

READABLE REGISTERS

Register 0 - The Main Status Port for the Controller and Drive. Location 120Q (50H) in the standard system.

This register contains bits that identify the current status of the Disk Jockey Controller and the currently selected drive. The details of this register are presented below:

CONTROLLER STATUS REGISTER



HALT - This is the "not busy" flag of the controller. When the Disk Jockey is not executing a command, this bit is a 1. When a data transfer command is strobed into the command register, the halt bit is reset to 0. At this point the controller is busy and will not respond to new commands until the HALT bit is again 1. Moreover, while HALT is 0 the CPU does not have access to the internal data bus and therefore cannot read from or write to the controller's data buffer. A program interfacing directly to the Disk Jockey controller should monitor this bit to determine when a command completes. The Main Status Register interfaces directly to the S-100 DI (Data Input) bus to allow the system to have access to the status port regardless of the state of the controller.

ILEVEL - This bit changes state with each index pulse from the currently selected drive. Drives that are not selected or not ready cannot transmit index pulses. Thus, ILEVEL only toggles when the selected drive is ready.

NREADY - This bit is a 0 only when the currently selected drive is powered up and ready to receive commands or transfer data.

NFAULT - Each drive in the system, when selected, sends a negative logic signal to the controller called WRITE FAULT. NFAULT monitors this line. WRITE FAULT is active (at a 0 level) if an illegal logic condition existed during a data transfer to the drive. An example of an illegal logic condition is READ GATE and WRITE GATE were active at the same time. This could happen, for example, if the 50 conductor cable between the controller and the drive were installed upside down at one end. There are also other conditions which could occur internal to the drive which can cause WRITE FAULT to be active (see the drive manual for details). Occasionally, WRITE FAULT is active when a drive is first powered up. The utility software, as a matter of course, resets WRITE FAULT on drives that it selects for the first time. Under normal conditions, NFAULT is 1.

TIMOUT - This bit is the latched output of a counter which is clocked by index pulses from the currently selected drive. The counter is enabled when the controller is busy. If a command is in progress after 16 revolutions of the disk, TIMOUT is set to 1 and the command is terminated. This insures that the controller will never "hang" trying to complete a command. Typically, this bit is set when the controller is asked to search for a sector header image that does not exist on the current track. TIMOUT is reset whenever a new command is sent to the controller.

COMPLT - Each drive in the system (which is ready) sends a negative logic signal to the controller called SEEK COMPLETE. This signal is present even if the drive is not selected. When a drive receives a head step command (or a burst of head step commands), logic inside the drive sets the SEEK COMPLETE line false. While the heads are moving to a new track this signal remains false. SEEK COMPLETE goes active again just as the heads have stopped (but not settled). When all the SEEK COMPLETE lines from the drives are active, COMPLT is 1. If any drive's heads are in motion, COMPLT is 0.

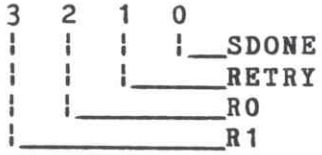
OPDONE - This bit is set to a 1 whenever the controller finishes a data transfer command. It is reset whenever any command is issued to the controller.

NTRCK0 - This bit is 0 when the heads of the currently selected drive are positioned over track zero. If the heads are over any other track, NTRK0 is 1.

Register 1 - Auxiliary Status Port for the Controller and Drive. Location 121Q (51H) in the standard system.

This register is four bits wide and contains the auxiliary status information regarding the drive and controller. The details of these bits are presented below:

AUXILIARY STATUS REGISTER



R0,R1 - These two bits are used by the controller to inform external software as to the revision level of the board. The encoding scheme for R0 and R1 is given below:

R1	R0	rev level
1	1	0
1	0	1
0	1	2
0	0	3

RETRY - This bit is set to 0 whenever a command is issued to the controller. During transfers from the disk, the serial data stream is routed to two places: the shift register where the data is assembled into 8 bit bytes for storage in the data buffer and to the CRC logic where polynominal division is performed. The last 16 bits of any transfer is the CRC error check word. These bits are not stored in the buffer. The last task of any command that transfers data from the disk to the controller is to compare the contents of the CRC register with the CRC error check word. If an error occurs, RETRY is set to 1. Thus, RETRY is the "read data" error flag. If RETRY is high after a read command AND the TIMOUT bit of the main status register is 0,the calling program should read the data again. If RETRY remains high through 10 trys, a hard error is present on the sector. When the controller is asked to search for a sector header image that does not exist on the current track, both TIMOUT and RETRY will be high. If both TIMOUT and RETRY are high then (1) the track is not formatted, (2) there is a hard error in a sector header field, or (3) the controller has an improper sector header image their header area of its data buffer (this is the most likely of the three possibilities).

SDONE - Whenever the COMPLT bit of the main status register makes a transition from 0 to 1, SDONE is set to 1. Any command to the controller will clear SDONE to 0.

READ/WRITE REGISTERS

Register 2 - The Controller Data Port.
Location 123Q (53H) in the standard system.

When the controller writes data to the disk, it is taken from an on board 1024 byte buffer. Conversely, when data is read from the disk, it is stored in this same buffer. Register 2 is the window between this buffer and the CPU. This register is used to fill the buffer before write commands are issued and to empty it after reads commands have completed.

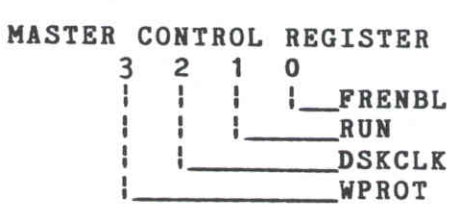
Associated with the data port is a pointer which serves to address different locations of the on board data buffer. It is incremented after references to the data port. The pointer can be reset to either half of the buffer by commands to the controller (refer to the command port). Each half of the buffer is 512 bytes long. One half is for data, the other is for sector header information. The controller uses only the first six bytes of the header half of the buffer - the remainder is available to the system.

Data is transferred to the buffer by first resetting the pointer and then initiating successive references to this register. The address pointer will automatically increment after each reference. After data is written in the buffer, it will remain stable unless a disk write command is issued or new data is written to the buffer by the CPU. Likewise, once a disk read operation has loaded the buffer, the CPU may retrieve data from it as often as desired.

WRITE ONLY REGISTERS

Register 3 - The Control Port.
Location 120Q (50H) in the standard system.

This is a four bit wide register which functions as the master control port for the board. The four high order data bits are ignored. The function of each low order bit is outlined below:



WPROT - This bit serves two purposes, depending on the state of the NFAULT bit of the main status register. When WPROT is 1, the currently selected drive will be write protected as long as there are no write fault conditions present. If NFAULT=0, WPROT will reset a write fault condition when it is brought high and then low. The drive is write enabled when WPROT is 0 and NFAULT is 1. That is - the disk will accept write commands if there are no write faults present and the write protect bit, WPROT, is low.

DSKCLK - This bit determines how the master clock of the controller will be driven. If DSKCLK is 0, the master clock signal on the board will come from PHASE2 - Pin 24 on the S-100 bus. If DSKCLK is 1, the currently selected drive will furnish the master clock to the controller. During data transfers, DSKCLK must be 1. However, if the selected drive is not ready or has encountered a write fault condition, it does not transmit any clock signals. In such a situation, the DSKCLK bit should be brought low so the controller will respond to commands.

RUN - When this bit is 0, the controller is reset and halted. It will not respond to commands - not even the buffer pointer reset commands. This bit is the master enable signal for the controller and should be set to 1 just after the first drive in the system is ready and does not have a write fault condition pending.

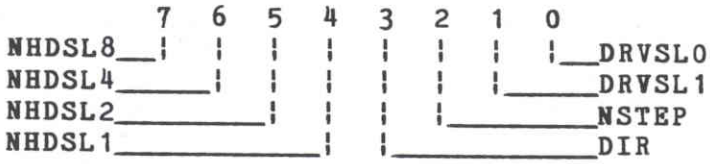
FRENBL - This bit enables the output of the drive select and drive function register. The function register outputs are TRI-STATE drivers that can be enabled or disabled. When FRENBL is 0, these drivers are disabled. When this bit is 1, the drivers are enabled.

Each bit in the control port is cleared to a zero whenever the S-100 bus signals POC (pin 99) or PRESET (pin 75) are active. Thus, the drive function register's outputs are disabled, the controller is in a reset state (with RUN false), the master clock is driven by PHASE2, and the disk is write enabled.

Register 4 - The Drive Select and Drive Function Port.
Location 122Q (52H) in the standard system.

This register selects one of four drives, one of (up to) sixteen heads, and controls the two lines which step the heads. There are eight data bits in the register and their specific functions are presented below:

DRIVE FUNCTION REGISTER



NHDSL - These are the head select bits. By using four, up to sixteen heads can be selected. On standard drives there are only eight heads. However, there is a model available which has an extra eight heads that are fixed. This allows a user to have 131K of fast access memory on the disk which is independent of the position of the eight moving heads. This model must be specially ordered and customers may have a longer wait for such systems. The relationship between the NHDSL bits and the head selected at the drive is detailed below. Heads numbered 0 through 7 move from track to track, and are present on all models of M26. The fixed heads are numbered 8 through 15, and are usually not present. Heads 8 through 15 should never be selected on a standard drive.

NHDSL8	NHDSL4	NHDSL2	NHDSL1	HEAD NO.
0	0	0	0	15
0	0	0	1	14
0	0	1	0	13
0	0	1	1	12
0	1	0	0	11
0	1	0	1	10
0	1	1	0	9
0	1	1	1	8
1	0	0	0	7
1	0	0	1	6
1	0	1	0	5
1	0	1	1	4
1	1	0	0	3
1	1	0	1	2
1	1	1	0	1
1	1	1	1	0

DIR - This bit controls the direction the read/write heads move when the NSTEP bit is pulsed. DIR must never change when NSTEP makes a transition from 0 to 1. If DIR is 0, the heads will move toward the center of the disk when step pulses are issued. Conversely, when DIR is 1, the heads will move away from the center of the disk. Track 0 is the outermost track on the disk. Track numbers increase as the heads move closer to the center of the disk.

NSTEP - This bit is used to issue step commands to the disk. Its idle state is 1. To issue a step command, NSTEP is brought low and then high. DIR must not change when NSTEP makes the transition from low to high. There are two modes of operation for stepping the heads: buffered and normal. In normal mode, the heads will move at the rate of the incoming NSTEP pulses and the minimum time between successive steps is 1 millesecond. In buffered mode, the step pulses are buffered into a counter at the drive. After the last pulse, the heads will begin stepping toward the appropriate track. The COMPLT bit (of the status register) will go true after the heads arrive at the proper track.

DRVSL - These two bits select one of four drives. The relationship between these bits and the physical drive that is selected is given by the following table:

DRVSL1	DRVSL0	DRIVE NO.
0	0	0
0	1	1
1	0	2
1	1	3

When power is first applied to the system the function register is in an unknown state. It is the user's responsibility to write a proper pattern to this port before its output drivers are enabled. In particular, NSTEP should be 1 when the outputs are first enabled.

Register 5 - The controller command port.
Location 121Q (51H) in the standard system.

This register is four bits wide and commands are transferred using the low order nibble of the data. The controller will execute six commands: reset the data buffer pointer to the first location of the data area, read a sector header, read a sector of data, write a sector of data, write a sector header, and reset the data buffer pointer to the first location of the header area. There are several more commands which the controller can accept, but these are for test purposes and should NEVER be used in a normal environment. A loss of data on the disk could occur if the user should issue any unlisted command. A detailed explanation of the standard commands is presented below. The command numbers below are also the numeric values to use when issuing commands to the controller through this port.

COMMAND 0 As explained previously, the controller has a 1024 byte data buffer which is divided into two sections: sector data and sector header information. These two sections share a pointer which is used to address the buffer. Command 0 resets this pointer to the beginning of the data area of the buffer. All commands affect bits in the status registers. Command 0 resets OPDONE, TIMOUT, and SDONE. No other status bits are changed.

COMMAND 3 Read a sector header. After receipt of this command, the HALT bit of the main status register is brought false and no further access to the internal data bus is allowed until the command terminates. OPDONE, TIMOUT, SDONE, and RETRY are reset. When the next sector pulse from the drive arrives, the controller reads the four bytes of the header into the 3rd, 4th, 5th, and 6th locations of the sector header area of the buffer. The CRC checksum bytes are compared with the contents of the CRC register. If there was an error in the data transfer, the RETRY bit in the auxiliary status port is set. After the CRC bytes have been checked, the HALT bit is brought back to its true state and the command terminates with OPDONE set.

COMMAND 1 This is the read sector command. As before, the first thing that occurs is OPDONE, TIMOUT, RESET, and SDONE are reset. The HALT bit is reset and access to the internal data bus is inhibited. As sector pulses arrive from the drive, the controller scans the sector header and compares it with the FIRST four bytes of the header area of the internal buffer. If the sector header data matches the buffer AND the CRC bytes match the contents of the CRC register, the data in the sector is read into the data area of the internal buffer. The first byte of data is written into the 3rd location of this area in the buffer. Successive bytes are written into successive locations. The next to last byte is written into the first location of the data

area and the last byte from the disk is placed in the second location. After the last byte is read, the controller compares the CRC checksum bytes with the contents of the CRC register. If there is a compare error, the RETRY bit of the auxiliary status register is set and the command terminates with OPDONE set. If the header does not match the pattern in the header area of the buffer, the controller continues to scan successive headers until a match is found (including CRC data) or until 16 index pulses have occured. If no match is found by the 16th index, the TIMOUT bit of the main status register and the RETRY bit of the auxiliary status register are set and the command terminates with OPDONE and HALT being set.

COMMAND 5 This is the write sector command. The sequence of events is exactly the same as with the read sector command up to the point of the actual data transfer. If a match has occured between the header area of the buffer and a sector header, the data area of the internal buffer is written to the sector where the header match took place. The data is transferred starting at the FIRST location of the data area of the buffer. Successive bytes are taken from successive locations. The last byte written to the disk is fetched from the last location in the data area of the buffer. If the header image in the buffer has no counterpart on the current track and head of the disk, the RETRY and TIMOUT bits are set in the status registers after 16 revolutions from the time the command was issued. As before, OPDONE and HALT are also set at the end of the command.

COMMAND 7 This command writes the first four bytes in the header area of the buffer on a sector header. The header in question is the very first that is encountered after receipt of the command. TIMOUT or RETRY cannot be set by this command. As usual, however, OPDONE and HALT are set at the command's conclusion.

COMMAND 8 This command resets the internal pointer to the first location of the header area of the controller's data buffer. The other effects of this command are identical to command 0.

Within the four bits allowed by commands, a total of sixteen are possible. However, the user is strongly cautioned NOT to use any except those listed above. If any other commands are issued to the controller, data on the disk could be lost if the WPROT bit in the control register is low. A good practice is to keep WPROT high at all times except during periods when write commands (5 & 7) are in progress.

CONTROLLER INITIALIZATION

When the system is first powered up the controller should be initialized before starting any operations that involve the disk. Outlined below is just one of several ways that this task can be accomplished.

1. Initialize the Drive function register - port 122Q. In particular NSTEP must be initialized to 1. If drive 0 is to be selected, a typical value to load this register with is 374Q (FC hex).

2. Load the Control register - port 120Q with 11Q to enable the outputs of the function register and to write protect the drive.

3. Wait for the drive to become ready and then switch the master clock to the drive clock. Also, toggle the WPROT bit to clear any write fault conditions. Finally, turn the run bit on and write protect the drive.

PREPARING THE CONTROLLER FOR DATA TRANSFER COMMANDS

Before giving the controller a read or a write command, certain tasks should be attended to. These are outlined below:

1. Select the proper drive.

2. Move the heads to the proper track.

3. Select the desired head.

4. Load the sector header area of the internal buffer with the proper header information: head number, track number, sector number, and key. This data must be written in the first four bytes of this area of the buffer.

5. If data is to be written to the disk, load the data area of the buffer with the desired information from main memory.

The user will find it useful to study carefully the utility software listings at the back of this manual to see an example of how these tasks have been carried out.

INTERRUPTS

In the lower left hand corner of the board there are two jumper holes labeled "A" and "B". "A" is driven by the OPDONE bit of the main status register and "B" is driven by the SDONE bit of the auxiliary status register. These two jumper holes can be connected to any of nine interrupt lines of the S-100 bus: VI0 - VI7 or PINT. These signals have been brought in from the bus to jumper holes to the right of the "A" and "B" holes and just above the edge connector at the bottom of the board. These jumper holes are labeled on the silk screened legend. "A" and "B" can be connected to the same line or, if OPDONE and SDONE should have different priority levels, they can be connected to different interrupt lines. When one or more of these signals are connected to the interrupt lines, interrupts can be generated by the board when transfer operations complete and/or when head motion at the disk completes. Throughput to and from the disk can be greatly enhanced by using these two signals properly.

```
                ;*****************************************************************
                ;*                                                               *
                ;* Low Level Hard Disk Drivers. The following routines are the   *
                ;* lowest level drivers for the hard disk.                       *
                ;*                                                               *
                ;* Written By Bobby Dale Gifford.                                *
                ;* 12/8/80                                                       *
                ;*                                                               *
                ;*****************************************************************

000C =          HDREV   EQU     12              ;Revision number
0004 =          MAXHD   EQU     4               ;Maximum # of Hard Disk Drives

0050 =          HDORG   EQU     50H             ;Hard Disk Controller origin
0050 =          HDSTAT  EQU     HDORG           ;Hard Disk Status
0050 =          HDCNTL  EQU     HDORG           ;Hard Disk Control
0053 =          HDDATA  EQU     HDORG+3         ;Hard Disk Data
0052 =          HDFUNC  EQU     HDORG+2         ;Hard Disk Function
0051 =          HDCMND  EQU     HDORG+1         ;Hard Disk Command
0051 =          HDRESLT EQU     HDORG+1         ;Hard Disk Result
0002 =          RETRY   EQU     2               ;Retry bit of result
0001 =          TKZERO  EQU     1               ;Track zero bit of status
0002 =          OPDONE  EQU     2               ;Operaction done bit of status
0004 =          COMPLT  EQU     4               ;Complete bit of status
0008 =          TMOUT   EQU     8               ;Time out bit of status
0010 =          WFAULT  EQU     10H             ;Write fault bit of status
0020 =          DRVRDY  EQU     20H             ;Drive ready bit of status
0040 =          INDEX   EQU     40H             ;Index bit of status
0004 =          PSTEP   EQU     4               ;Step bit of function
00FB =          NSTEP   EQU     0FBH            ;Step bit mask of function
0004 =          HDRLEN  EQU     4               ;Sector header length
0200 =          SECLEN  EQU     512             ;Sector data length
000F =          WENABL  EQU     0FH             ;Write enable
000B =          WRESET  EQU     0BH             ;Write reset of function
0005 =          SCENBL  EQU     5               ;Controller control
0007 =          DSKCLK  EQU     7               ;Disk clock for control
00F7 =          MDIR    EQU     0F7H            ;Direction mask for function
00FC =          NULL    EQU     0FCH            ;Null command
0000 =          IDBUFF  EQU     0               ;Initialize data command
0008 =          ISBUFF  EQU     8               ;Initialize header command
0001 =          RSECT   EQU     1               ;Read sector command
0005 =          WSECT   EQU     5               ;Write sector command
0015 =          SECCNT  EQU     21              ;Sectors per track
                                                ;32 for M26
                                                ;21 for M10 & M20

0100                    ORG     100H

0100 317603             LXI     SP,STACK
0103 0E00               MVI     C,0
0105 CD7603             CALL    SETDRV
0108 CD7903             CALL    HOME
010B 014401             LXI     B,BXX
010E CD8203             CALL    SETDMA
0111 0E00       LOOP    MVI     C,0
0113 CD7C03             CALL    SETTRK
```

```
0116 0E00               MVI     C,0
0118 CD8503             CALL    SETHEAD
011B 0E01               MVI     C,1
011D CD7F03             CALL    SETSEC
0120 0E80               MVI     C,80H
0122 CD8803             CALL    SETKEY
0125 CD7B04             CALL    HDREAD
0128 00                 NOP
0129 0EC8               MVI     C,200
012B CD7C03             CALL    SETTRK
012E 0E00               MVI     C,0
0130 CD8503             CALL    SETHEAD
0133 0E01               MVI     C,1
0135 CD7F03             CALL    SETSEC
0138 0E00               MVI     C,0
013A CD8803             CALL    SETKEY
013D CD7B04             CALL    HDREAD
0140 00                 NOP
0141 C31101             JMP     LOOP


0144            BXX     DS      512

0344                    DS      50
0376 =          STACK   EQU     $
```

```
0376 C39703     SETDRV  JMP     HDDRV       ;Select disk
0379 C3E203     HOME    JMP     HDHOME      ;Recalibrate
037C C31804     SETTRK  JMP     HDTRK       ;Seek to specified track
037F C36704     SETSEC  JMP     HDSEC       ;Prep for sector #
0382 C34A04     SETDMA  JMP     HDDMA       ;Prep for DMA address
0385 C37404     SETHEAD JMP     HDHEAD      ;Set head #
0388 C3DD03     SETKEY  JMP     HDKEY       ;Set the key in next transfer
038B C37B04     READ    JMP     HDREAD      ;Read one sector
038E C3B004     WRITE   JMP     HDWRITE     ;Write one sector
0391 C34B05     DMASTAT JMP     DMAHD       ;Return DMA address
0394 C35305     GETSTAT JMP     STATHD      ;Get drive status

0397 3E03       HDDRV   MVI     A,3
0399 A1                 ANA     C
039A 324705             STA     HDDISK
039D F6FC               ORI     NULL        ;Select drive
039F D352               OUT     HDFUNC
03A1 3E05               MVI     A,SCENBL    ;Enable the controller
03A3 D350               OUT     HDCNTL
03A5 0EEF               MVI     C,239       ;Wait for Disk to ready
                                            ; 2 minutes for M26
                                            ; 30 seconds for M10 & M20
03A7 210000             LXI     H,0
```

```
03AA 2B        TDELAY  DCX     H
03AB 7C                MOV     A,H
03AC B5                ORA     L
03AD CCDB03            CZ      DCRC
03B0 37                STC
03B1 C8                RZ
03B2 DB50              IN      HDSTAT          ;Test if ready yet
03B4 E620              ANI     DRVRDY
03B6 C2AA03            JNZ     TDELAY
03B9 2AFC03            LHLD    SETTLE
03BC 7C                MOV     A,H
03BD B5                ORA     L
03BE C8                RZ
03BF 210000            LXI     H,0             ;Time one revolution of the drive
03C2 0E40              MVI     C,INDEX
03C4 DB50              IN      HDSTAT
03C6 A1                ANA     C
03C7 47                MOV     B,A             ;Save current index level in B
03C8 DB50      INDX1   IN      HDSTAT
03CA A1                ANA     C
03CB B8                CMP     B               ;Loop util index level changes
03CC CAC803            JZ      INDX1
03CF 23        INDX2   INX     H
03D0 DB50              IN      HDSTAT          ;Start counting until index returns
03D2 A1                ANA     C               ;       previous state
03D3 B8                CMP     B
03D4 C2CF03            JNZ     INDX2
03D7 22FC03            SHLD    SETTLE          ;Save the Count for timeout delay
03DA C9                RET

03DB 0D        DCRC    DCR     C               ;Conditional decrement C routine
03DC C9                RET

03DD 79        HDKEY   MOV     A,C
03DE 321A05            STA     NKEY
03E1 C9                RET

03E2 CD3405    HDHOME  CALL    DRVPTR
03E5 3600              MVI     M,0             ;Set track to zero
03E7 23                INX     H               ;Point to seek flag
03E8 3601              MVI     M,1             ;Set not seeking, but must delay
03EA DB50      STEPO   IN      HDSTAT          ;Test status
03EC E601              ANI     TKZERO          ;At track zero ?
03EE C8                RZ
03EF 3E01              MVI     A,1
03F1 37                STC
03F2 CD3904            CALL    ACCOK           ;Take one step out
03F5 CD5004            CALL    WSDONE          ;Wait for previous seek to finish
03F8 C3EA03            JMP     STEPO

03FB 210000    DELAY   LXI     H,0             ;Get delay
03FC =         SETTLE  EQU     $-2
03FE 2B        DELOOP  DCX     H               ;Wait 20ms
03FF 7C                MOV     A,H
0400 B5                ORA     L
0401 23                INX     H
```

```
0402 2B              DCX     H
0403 C2FE03          JNZ     DELOOP
0406 215C05          LXI     H,DRIVES-1
0409 0605            MVI     B,MAXHD+1
040B 23        DELUP INX     H
040C 23              INX     H
040D 05              DCR     B
040E C8              RZ
040F 7E              MOV     A,M
0410 3D              DCR     A
0411 C20B04          JNZ     DELUP
0414 77              MOV     M,A
0415 C30B04          JMP     DELUP

0418 CD3405    HDTRK CALL    DRVPTR          ;Get pointer to current track
041B 5E              MOV     E,M             ;Get current track
041C E5              PUSH    H               ;Save pointer to current track
041D 1C              INR     E               ;Ever homed this drive ?
041E CCE203          CZ      HDHOME
0421 E1              POP     H               ;Restore track pointer
0422 5E              MOV     E,M             ;Get current track
0423 71              MOV     M,C             ;Update the track
0424 7B              MOV     A,E             ;Need to seek at all ?
0425 91              SUB     C
0426 C8              RZ
0427 F5              PUSH    PSW             ;Save # of steps
0428 23              INX     H               ;Point to the seek complete flag
0429 7E              MOV     A,M             ;Get current seek progress flag
042A 3C              INR     A               ;Currently seeking ?
042B E5              PUSH    H               ;Save seek flag pointer
042C CC5004          CZ      WSDONE          ;Wait if currently seeking
042F E1              POP     H
0430 36FF            MVI     M,0FFH          ;Set seek in progress flag
0432 F1              POP     PSW
0433 3F              CMC                     ;Get carry into direction
0434 DA3904          JC      ACCOK
0437 2F              CMA
0438 3C              INR     A
0439 47        ACCOK MOV     B,A             ;Prep for build
043A CD4005          CALL    BUILD
043D E6FB      SLOOP ANI     NSTEP           ;Get step pulse low
043F D352            OUT     HDFUNC          ;Output low step line
0441 F604            ORI     PSTEP           ;Set step line high
0443 D352            OUT     HDFUNC          ;Output high step line
0445 05              DCR     B               ;Update repeat count
0446 C23D04          JNZ     SLOOP           ;Keep going the required # of tracks
0449 C9              RET

044A 60        HDDMA MOV     H,B             ;Save the DMA address
044B 69              MOV     L,C
044C 229504          SHLD    HDADD
044F C9              RET

0450 DB50      WSDONE IN     HDSTAT          ;Wait for seek complete to finish
0452 E604            ANI     COMPLT
0454 CA5004          JZ      WSDONE
```

```
0457 215C05            LXI     H,DRIVES-1      ;Update all seek in progress flags
045A 0605              MVI     B,MAXHD+1
045C 23        WSUP    INX     H
045D 23                INX     H
045E 05                DCR     B
045F C8                RZ
0460 7E                MOV     A,M
0461 E601              ANI     1
0463 77                MOV     M,A
0464 C35C04            JMP     WSUP

0467 AF        HDSEC   XRA     A
0468 B1                ORA     C
0469 37                STC
046A C8                RZ
046B 3E15              MVI     A,SECCNT
046D 91                SUB     C
046E D8                RC
046F 79                MOV     A,C
0470 321605            STA     HDSECTR
0473 C9                RET

0474 79        HDHEAD  MOV     A,C
0475 E607              ANI     7               ; 7 for M26 & M20, 3 for M10
0477 324105            STA     HEAD
047A C9                RET

047B CDF904    HDREAD  CALL    HDPREP
047E D8                RC
047F AF                XRA     A
0480 D351              OUT     HDCMND
0482 2F                CMA
0483 D353              OUT     HDDATA
0485 D353              OUT     HDDATA
0487 3E01              MVI     A,RSECT         ;Read sector command
0489 D351              OUT     HDCMND
048B CDDF04            CALL    PROCESS
048E D8                RC
048F AF                XRA     A
0490 D351              OUT     HDCMND
0492 0680              MVI     B,SECLEN/4
0494 210000            LXI     H,0
0495 =         HDADD   EQU     $-2
0497 DB53              IN      HDDATA
0499 DB53              IN      HDDATA
049B DB53      RTLOOP  IN      HDDATA          ;Move four bytes
049D 77                MOV     M,A
049E 23                INX     H
049F DB53              IN      HDDATA
04A1 77                MOV     M,A
04A2 23                INX     H
04A3 DB53              IN      HDDATA
04A5 77                MOV     M,A
04A6 23                INX     H
04A7 DB53              IN      HDDATA
04A9 77                MOV     M,A
```

```
04AA 23                 INX     H
04AB 05                 DCR     B
04AC C29B04             JNZ     RTLOOP
04AF C9                 RET

04B0 CDF904     HDWRITE CALL    HDPREP          ;Prepare header
04B3 D8                 RC
04B4 AF                 XRA     A
04B5 D351               OUT     HDCMND
04B7 2A9504             LHLD    HDADD
04BA 0680               MVI     B,SECLEN/4
04BC 7E         WTLOOP  MOV     A,M             ;Move 4 bytes
04BD D353               OUT     HDDATA
04BF 23                 INX     H
04C0 7E                 MOV     A,M
04C1 D353               OUT     HDDATA
04C3 23                 INX     H
04C4 7E                 MOV     A,M
04C5 D353               OUT     HDDATA
04C7 23                 INX     H
04C8 7E                 MOV     A,M
04C9 D353               OUT     HDDATA
04CB 23                 INX     H
04CC 05                 DCR     B
04CD C2BC04             JNZ     WTLOOP
04D0 3E05               MVI     A,WSECT         ;Issue write sector command
04D2 D351               OUT     HDCMND
04D4 CDDF04             CALL    PROCESS
04D7 D8                 RC
04D8 3E10               MVI     A,WFAULT
04DA A0                 ANA     B
04DB 37                 STC
04DC C8                 RZ
04DD AF                 XRA     A
04DE C9                 RET

04DF DB50       PROCESS IN      HDSTAT          ;Wait for command to finish
04E1 47                 MOV     B,A
04E2 E602               ANI     OPDONE
04E4 CADF04             JZ      PROCESS
04E7 3E07               MVI     A,DSKCLK
04E9 D350               OUT     HDCNTL
04EB DB50               IN      HDSTAT
04ED E608               ANI     TMOUT           ;Timed out ?
04EF 37                 STC
04F0 C0                 RNZ
04F1 DB51               IN      HDRESLT
04F3 E602               ANI     RETRY           ;Any retries ?
04F5 37                 STC
04F6 C0                 RNZ
04F7 AF                 XRA     A
04F8 C9                 RET

04F9 DB50       HDPREP  IN      HDSTAT
04FB E620               ANI     DRVRDY
04FD 37                 STC
```

```
04FE C0                 RNZ
04FF 3E08               MVI     A,ISBUFF        ;Initialize pointer
0501 D351               OUT     HDCMND
0503 CD4005             CALL    BUILD
0506 F60C               ORI     0CH
0508 D352               OUT     HDFUNC
050A 3A4105             LDA     HEAD
050D D353               OUT     HDDATA          ;Form head byte
050F CD3405             CALL    DRVPTR
0512 7E                 MOV     A,M             ;Form track byte
0513 D353               OUT     HDDATA
0515 3E00               MVI     A,0             ;Form sector byte
0516 =          HDSECTR EQU     $-1
0517 D353               OUT     HDDATA
0519 3E00               MVI     A,0
051A =          NKEY    EQU     $-1
051B D353               OUT     HDDATA
051D 23                 INX     H               ;Bump to seek flag
051E 7E                 MOV     A,M
051F 3C                 INR     A               ;Update the seek in progress flag
0520 E5                 PUSH    H
0521 CC5004             CZ      WSDONE
0524 E1                 POP     H
0525 7E                 MOV     A,M
0526 3D                 DCR     A               ;Test for delay also
0527 CCFB03             CZ      DELAY
052A 3E07               MVI     A,DSKCLK
052C D350               OUT     HDCNTL
052E 3E0F               MVI     A,WENABL
0530 D350               OUT     HDCNTL
0532 AF                 XRA     A
0533 C9                 RET

0534 2A4705     DRVPTR  LHLD    HDDISK
0537 EB                 XCHG
0538 1600               MVI     D,0
053A 215D05             LXI     H,DRIVES
053D 19                 DAD     D
053E 19                 DAD     D
053F C9                 RET

0540 3E00       BUILD   MVI     A,0
0541 =          HEAD    EQU     $-1
0542 17                 RAL
0543 17                 RAL
0544 17                 RAL
0545 17                 RAL
0546 F600               ORI     0
0547 =          HDDISK  EQU     $-1
0548 EEF0               XRI     0F0H
054A C9                 RET

054B E5         DMAHD   PUSH    H
054C 2A9504             LHLD    HDADD
054F 44                 MOV     B,H
0550 4D                 MOV     C,L
```

```
0551 E1                 POP     H
0552 C9                 RET

0553 DB51       STATHD  IN      HDRESLT
0555 E603               ANI     3
0557 47                 MOV     B,A
0558 DB50               IN      HDSTAT
055A EE31               XRI     31H
055C C9                 RET

055D FFFF       DRIVES  DW      0FFFFH
055F FFFF               DW      0FFFFH
0561 FFFF               DW      0FFFFH
0563 FFFF               DW      0FFFFH

0565                    END
```

LIMITED WARRANTY

DISCUS M26tm
HARD DISK SYSTEM

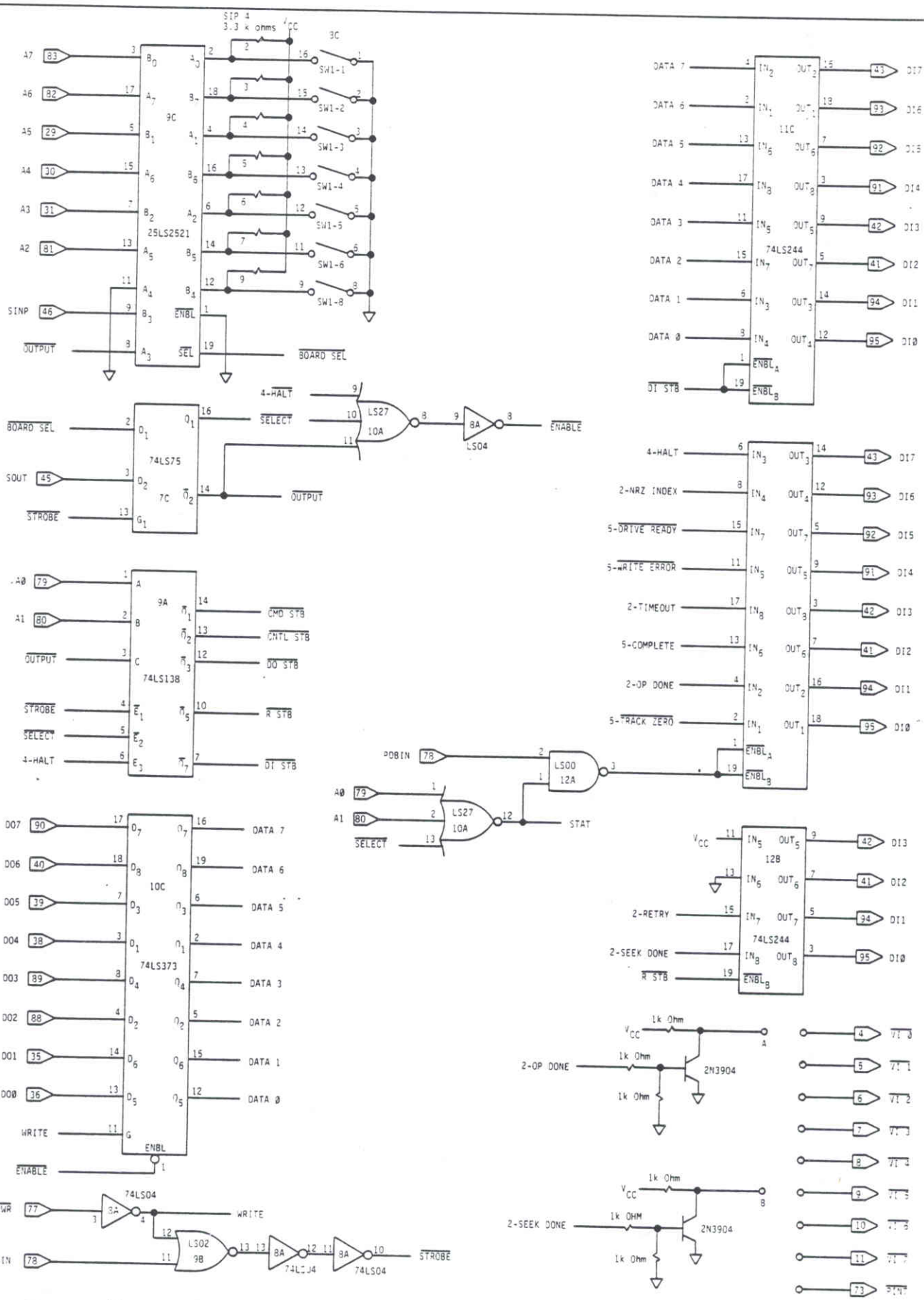
This addendum to Morrow Designs Inc. Limited Warranty applies to the Shugart Associates Model 4008 Hard Disk used in the DISCUS M26 Hard Disk system.

Parts and labor for a hard disk drive purchased from Morrow Designs Inc. are warranted for a period of forty-five (45) days from the invoice/purchase date. For a period of six (6) months from the invoice/purchase date:

> Outside the protective cover - parts, including the printed circuit boards, are warranted. Labor performed by Morrow Designs Inc. to locate the faulty part will be performed at no charge. If the unit must be returned to Shugart Assoc. for diagnosis and repair, a fixed fee of $175 will be charged.
>
> Inside the protective cover - parts and labor are warranted.

After six months, a minimum service charge of $100 will be applied. Current rates for labor and parts costs will be charged up to a maximum of $550. If the fee is to exceed $100, the customer will be notified in advance of the actual repair.

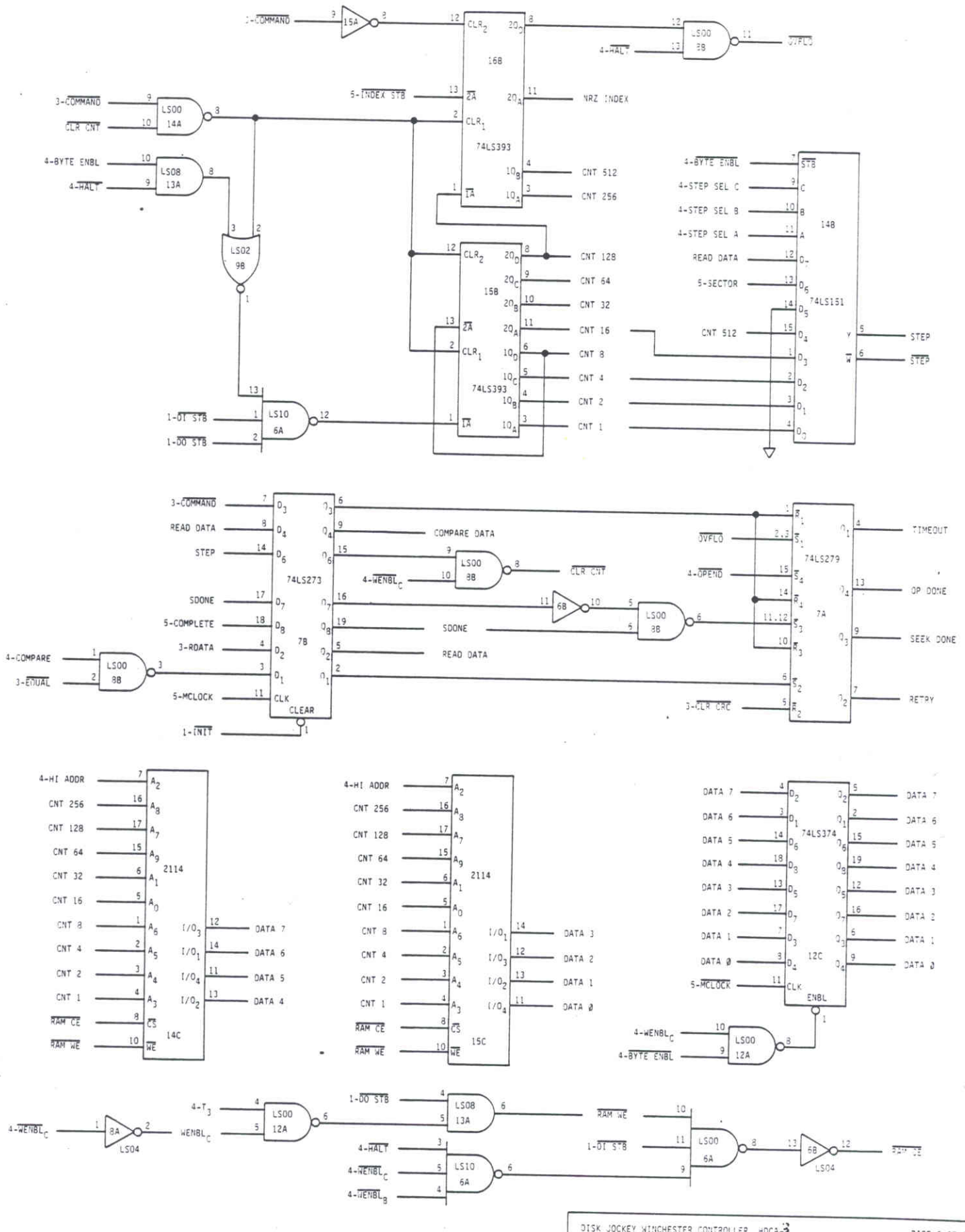
Repaired parts are warranted for a period of ninety (90) days after the shipping date or until the expiration of the six (6) months from original purchase/invoice date, whichever is longer.



DISK JOCKEY WINCHESTER CONTROLLER 40CA- 3

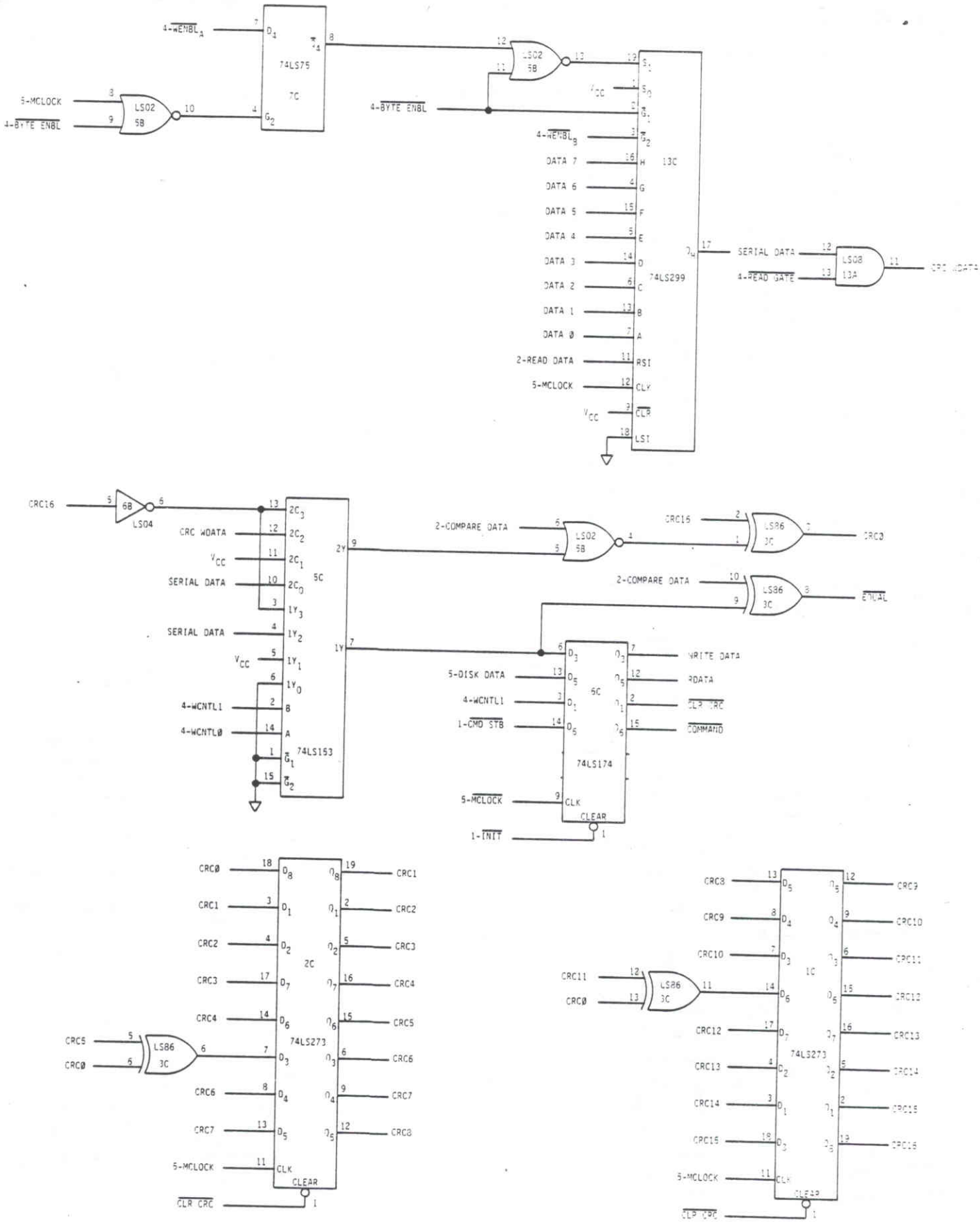
PAGE 1 OF 5

DATA BUFFERS AND BOARD SELECT LOGIC



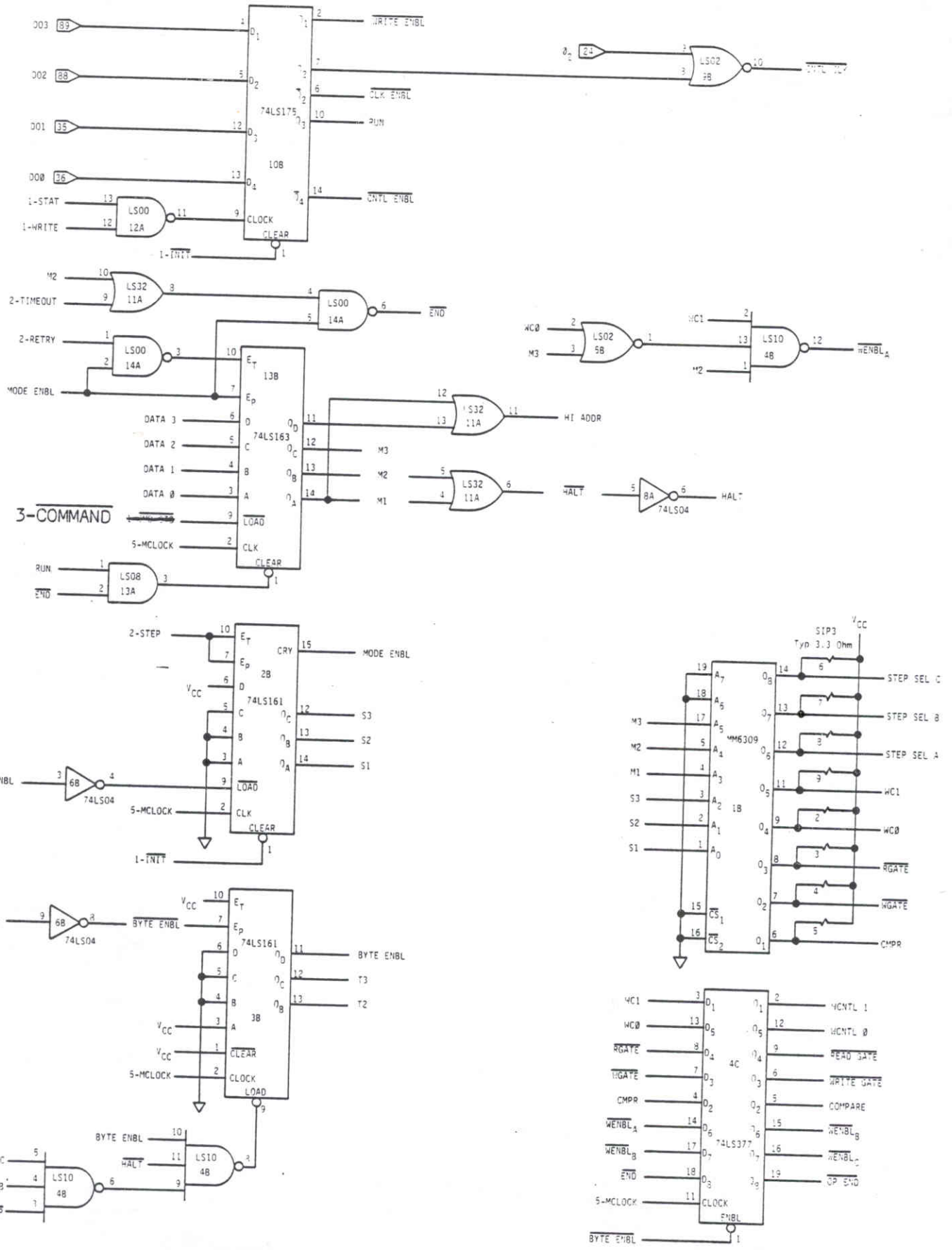
DISK JOCKEY WINCHESTER CONTROLLER HDCA-3

RAM BUFFER, STEP & COMPLETION LOGIC



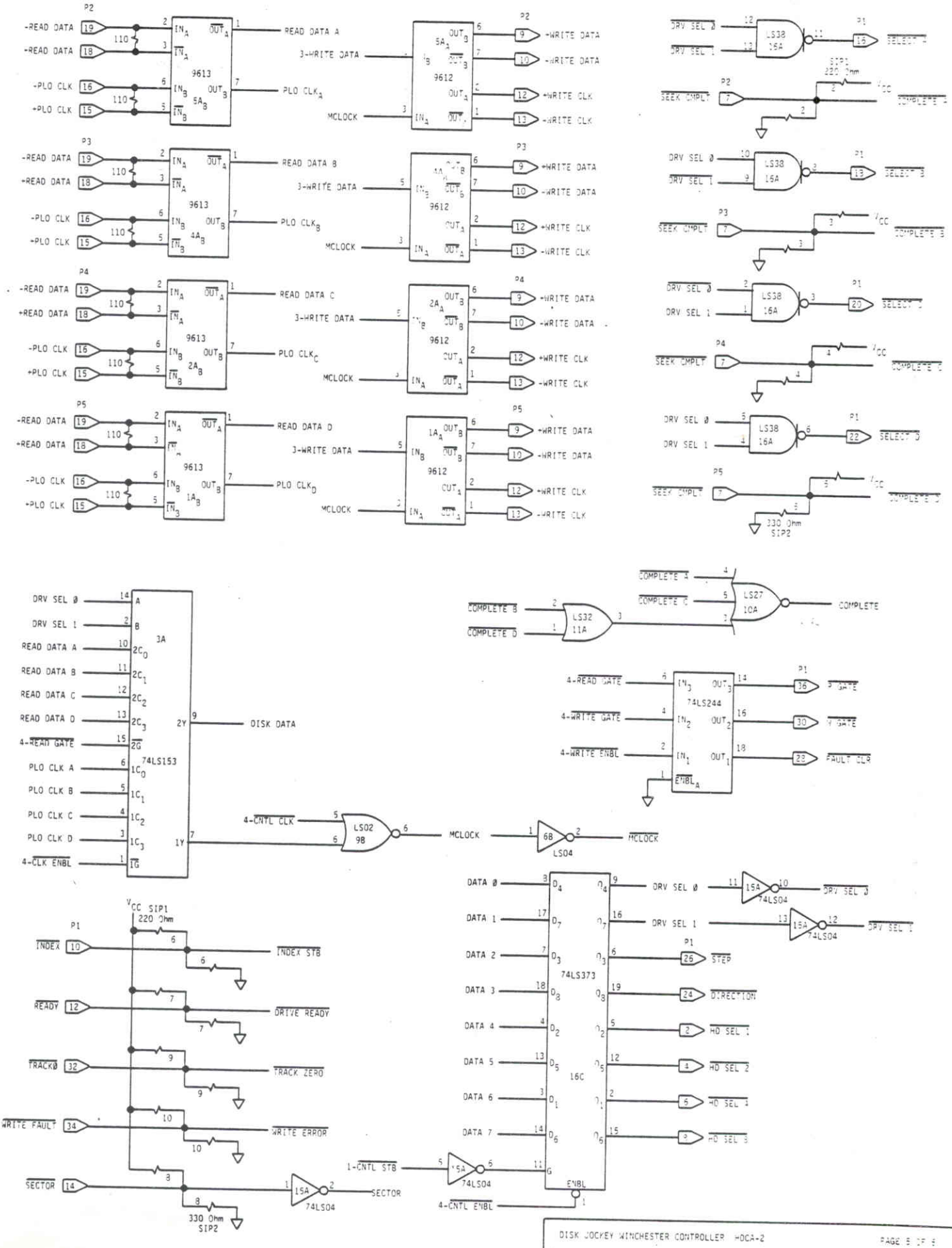
DISK JOCKEY WINCHESTER CONTROLLER HDCA-8 PAGE 3 OF 8

READ/WRITE LOGIC & CRC ERROR REGISTER



DISK JOCKEY WINCHESTER CONTROLLER HDCA-13 PAGE 4 OF 8

CONTROL SEQUENCE LOGIC



DISK JOCKEY WINCHESTER CONTROLLER HDCA-2

DISK INTERFACE BUFFERS

PAGE 5 OF 5