DUAL-MODE DISK CONTROLLER BOARD

Revision 2.0

ENGINEERING DOCUMENTATION

Revision A

February 1, 1981

NOTICE: This manual contains information proprietary to Vector Graphic, Inc. It is not to be copied nor distributed without the expressed consent of the Director of Research and Development.

Copyright 1981 Vector Graphic Inc.
Copyright 1981 by Vector Graphic Inc.
All rights reserved.

Disclaimer
Vector Graphic makes no representations or warranties with respect to the contents of this manual itself, whether or not the product it describes is covered by a warranty or repair agreement. Further, Vector Graphic reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Vector Graphic to notify any person of such revision or changes, except when an agreement to the contrary exists.

Revisions
The date and revision of each page herein appears at the bottom of each page. The revision letter such as A or B changes if the MANUAL has been improved but the PRODUCT itself has not been significantly modified. The date and revision on the Title Page corresponds to that of the page most recently revised. When the product itself is modified significantly, the product will get a new revision number, as shown on the manual's title page, and the manual will revert to revision A, as if it were treating a brand new product. EACH MANUAL SHOULD ONLY BE USED WITH THE PRODUCT IDENTIFIED ON THE TITLE PAGE.
REPAIR AGREEMENT

The Dual-Mode Controller Board sold hereunder is sold "as is", with all faults and without any warranty, either expressed or implied, including any implied warranty of fitness for intended use or merchantability. However, the above notwithstanding, VECTOR GRAPHIC, INC., will, for a period of ninety (90) days following delivery to customer, repair or replace any Dual-Mode Controller Board that is found to contain defects in materials or workmanship, provided:

1. Such defect in material or workmanship existed at the time the Dual-Mode Controller Board left the VECTOR GRAPHIC, INC., factory;
2. VECTOR GRAPHIC, INC., is given notice of the precise defect claimed within ten (10) days after its discovery;
3. The Dual-Mode Controller Board is promptly returned to VECTOR GRAPHIC, INC., at customer's expense, for examination by VECTOR GRAPHIC, INC., to confirm the alleged defect, and for subsequent repair or replacement if found to be in order.

Repair, replacement or correction of any defects in material or workmanship which are discovered after expiration of the period set forth above will be performed by VECTOR GRAPHIC, INC., at Buyer's expense, provided the Dual-Mode Controller Board is returned, also at Buyer's expense, to VECTOR GRAPHIC, INC., for such repair, replacement or correction. In performing any repair, replacement or correction after expiration of the period set forth above, Buyer will be charged in addition to the cost of parts the then-current VECTOR GRAPHIC, INC., repair rate. At the present time the applicable rate is $35.00 for the first hour, and $18.00 per hour for every hour of work required thereafter. Prior to commencing any repair, replacement or correction of defects in material or workmanship discovered after expiration of the period for no-cost-to-Buyer repairs, VECTOR GRAPHIC, INC., will submit to Buyer a written estimate of the expected charges, and VECTOR GRAPHIC, INC., will not commence repair until such time as the written estimate of charges has been returned by Buyer to VECTOR GRAPHIC, INC., signed by duly authorized representative authorizing VECTOR GRAPHIC, INC., to commence with the repair work involved. VECTOR GRAPHIC, INC., shall have no obligation to repair, replace or correct any Dual-Mode Controller Board until the written estimate has been returned with approval to proceed, and VECTOR GRAPHIC, INC., may at its option also require prepayment of the estimated repair charges prior to commencing work.

Repair Agreement void if the enclosed card is not returned to VECTOR GRAPHIC, INC. within ten (10) days of end consumer purchase.

Rev. 2.0-A 2/1/81
FOREWORD

Audience

This manual is intended for experienced assembly language programmers, who have an in depth knowledge of disk systems and disk controllers.

Scope

It will describe how to implement a disk driver to use the Vector Graphic Dual-Mode Controller Subsystem in other S-100 systems.

Organization

Each section is written at a uniform level of technical depth. Each section contains specific information about the disks and controller. Latter sections of the manual build on the beginning sections. To write a successful driver it is imperative that the manual be read in its entirety.
## Vector Graphic Dual-Mode Controller Board

### TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Introduction</td>
<td>1-1</td>
</tr>
<tr>
<td>1.2 Organization</td>
<td>1-1</td>
</tr>
<tr>
<td>1.3 Hardware Fundamentals</td>
<td>1-1</td>
</tr>
<tr>
<td>1.4 Controller Support Capacity</td>
<td>1-3</td>
</tr>
<tr>
<td>1.4.1 Hard disk configuration</td>
<td>1-3</td>
</tr>
<tr>
<td>1.4.2 Floppy disk configuration</td>
<td>1-3</td>
</tr>
<tr>
<td>1.4.3 H/D format</td>
<td>1-3</td>
</tr>
<tr>
<td>1.4.4 F/D format</td>
<td>1-3</td>
</tr>
<tr>
<td>1.5 Controller Port Description</td>
<td>1-4</td>
</tr>
<tr>
<td>1.5.1 Status and control ports</td>
<td>1-4</td>
</tr>
<tr>
<td>1.5.2 Data port</td>
<td>1-4</td>
</tr>
<tr>
<td>1.5.3 Reset/Start port</td>
<td>1-4</td>
</tr>
<tr>
<td>1.6 Sector Data Format</td>
<td>1-5</td>
</tr>
<tr>
<td>1.6.1 H/D sector format</td>
<td>1-5</td>
</tr>
<tr>
<td>1.6.2 F/D sector format</td>
<td>1-6</td>
</tr>
<tr>
<td>1.7 Port Format</td>
<td>1-8</td>
</tr>
<tr>
<td>1.7.1 Control bits</td>
<td>1-8</td>
</tr>
<tr>
<td>1.7.2 Status bits</td>
<td>1-10</td>
</tr>
<tr>
<td>1.7.3 Data bits</td>
<td>1-12</td>
</tr>
<tr>
<td>1.7.4 Reset/Start port format</td>
<td>1-12</td>
</tr>
<tr>
<td>1.8 Operation</td>
<td>1-12</td>
</tr>
<tr>
<td>1.8.1 Calibration</td>
<td>1-13</td>
</tr>
<tr>
<td>1.8.2 Drive selection</td>
<td>1-14</td>
</tr>
<tr>
<td>1.8.3 Seek a track</td>
<td>1-15</td>
</tr>
<tr>
<td>1.8.4 Write</td>
<td>1-16</td>
</tr>
<tr>
<td>1.8.5 Read</td>
<td>1-17</td>
</tr>
<tr>
<td>1.9 Error Handling</td>
<td>1-18</td>
</tr>
<tr>
<td>1.9.1 Read operations</td>
<td>1-18</td>
</tr>
<tr>
<td>1.9.2 Write operations</td>
<td>1-19</td>
</tr>
<tr>
<td>1.10 Disk Driver</td>
<td>1-20</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>A</td>
</tr>
<tr>
<td>FIGURES</td>
<td></td>
</tr>
<tr>
<td>1-1</td>
<td></td>
</tr>
<tr>
<td>Subsystem Block Diagram</td>
<td>1-2</td>
</tr>
</tbody>
</table>

Rev. 2.0-A 2/1/81
1.1 INTRODUCTION

This section describes the Vector Dual-Mode Controller Subsystem (VEDMCS) in sufficient detail to enable the experienced assembly language programmer to implement a disk driver.

1.2 ORGANIZATION

This manual will start with each titled section divided into pairs of subsections. The hard disk (H/D) will be discussed first, followed by the same discussion as it relates to the floppy disk (F/D). This is to separate the specific information for each type of disk. As we progress into writing the disk driver the discussion will move to combining the two types of drives into the same routines.

1.3 HARDWARE FUNDAMENTALS

The Vector dual mode controller is a self contained unit on a single board. On-board memory, which is accessed through a control port, provides the high speed loading and unloading of data to and from the controller. Control and status each have two 8-bit registers accessed through data ports. After the controller receives the control signals and data in the correct format, it will automatically sequence the physical reading or writing of the disk. These features save valuable memory locations through reduced software, and spare the programmer from tedious timing requirements.

Other features include a built in hardware generated Error Correction Code (ECC). If the single control bit "ECC on/off" is on, the ECC will be generated and inserted immediately following the data during a disk write. During a read a new ECC is calculated and compared with the one already written. If an error is detected a correction code is inserted in the ECC block immediately following the data. In the off mode ECC is transparent.

Write pre-compensation is another single control bit feature. The programmer need only determine the track number above which pre-compensation is necessary, and turn it on or off.

Figure 1.1 is a block diagram of the controller subsystem. It is provided as a reference. All specifics will be provided in the text and tables following.
1.4 CONTROLLER SUPPORT CAPACITY

1.4.1 H/D Configuration

The VEDMCS has the capacity to multiplex up to a total of four drives. However, it can support only ONE hard disk at a time. The remaining slots can be filled with anywhere from zero to three floppy disk drives in a daisy-chain configuration.

1.4.2 F/D Configuration

One to four floppy disk drives may be attached in parallel, with the fourth disk drive filling in the address normally used by the hard disk drive.

1.4.3 H/D Format

Following are the specifications in the hard disk configuration.

- HEADS or SURFACES (data) 4
- TRACKS or CYLINDERS (ea. surface) 153
- TYPE of SECTORING HARD
- SECTORS 32
- BYTES/SECTOR (formatted) 256
- TYPE of RECORDING MFM

1.4.4 F/D Format

- HEADS or SURFACES (data) 2
- TRACKS or CYLINDERS (ea. surface) 77
- TYPE of SECTORING HARD
- SECTORS 16
- BYTES/SECTOR (formatted) 256
- TYPE of RECORDING MFM
1.5 CONTROLLER PORT DESCRIPTION

Both the hard and floppy disks are operated through the same controller ports. As the user, you, must take care in the management of the controller to insure the correct format at each access. This section is an introduction to the theory of operation for which specific breakdowns will be provided later.

1.5.1 Status and Control Ports

There are two status and control ports addressed at COH (HEX) and C1H. Each port provides 8 bits IN and 8 bits OUT. To these ports you will OUTPUT information such as drive, head, sector, and step select to the controller. Some examples of the information that you will INPUT include, write protect, ready, and track 0.

1.5.2 Data Port

The data port addressed at C2H contains 8 bits of IN/OUT data information. The buss is tied to a bi-directional 512 byte RAM on board the controller. The memory is filled or read back sequentially from address zero, with the address counter incrementing automatically after each access until the desired address is reached. All the information is contained in approximately the first 400 bytes of memory with the remainder of RAM not used. Loading the address counter to zero is handled by the reset/start port.

1.5.3 Reset/Start Port

The reset/start port is a control only port and there are no data lines associated with it. By performing an INPUT command from port C3H the controller will automatically RESET the memory address register to all zeroes. A RESET should be accomplished before each START READ/WRITE, or INPUT/OUTPUT DATA operation. A START is accomplished by issuing an OUTPUT command to port C3H with the control bit "Read/Write" (Port COH) set to the appropriate position. The controller will then automatically READ from the disk into the controller RAM or WRITE on the disk the contents of the controller RAM. The controller will NOT operate correctly unless the Status and Control Ports are serviced properly before issuing a Start command.
1.6 SECTOR DATA FORMAT

1.6.1 H/D Sector format

The recommended sector format is illustrated below.

<table>
<thead>
<tr>
<th>PREAMBLE</th>
<th>SYNC</th>
<th>HEAD</th>
<th>TRACK</th>
<th>SECTOR</th>
<th>DATA</th>
<th>ECC</th>
<th>POSTAMBLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>4</td>
<td>to end of</td>
</tr>
</tbody>
</table>

A hard disk sector consists of the following fields:

1) **PREAMBLE**: The preamble is a field of 30 bytes of 00H data. It provides a tolerance for mechanical and electronic deviations against the sector pulse, and a known data pattern for synchronization of the read data decoder. The preamble is the first thing that you must place in the controller memory for a write operation, although it will not appear in the memory during a read.

2) **SYNC**: The sync byte is a byte of 0FFH data and is used in the disk controller to determine the beginning of useful data. During a read, the sync byte is the first data byte to appear in the controller memory. Also, it should follow the preamble during a write operation.

3) **HEAD**: The head byte is a single byte of data in the range of 0 to 3 written into all sectors. Its value should be equal to the head number of any given surface and also equal to the value sent on the 'Head Select' lines to the control port. The head byte should be used to verify that the proper surface is being accessed. It also follows the sync byte in I/O operations.

4) **TRACK**: The track byte is a single byte of data in the range of 0 to 152. It follows the head byte and should be used to verify that the correct track is being accessed during a disk I/O.

5) **SECTOR**: The sector byte is a single byte of data in the range of 0 to 31. It follows the track byte in I/O operations, and should also be equal to the value sent on the 'Sector Select' lines to the control port.
6) **DATA:** The data field consists of 256 bytes of user data, and follows the sector byte in I/O operations.

7) **ECC:** The Error Correction Code is 4 hardware generated bytes and is automatically inserted after the last byte of the data field in this format. When performing a read operation, the controller calculates a new ECC, compares it with the ECC that was written, and returns with 4 bytes of OOH if the two numbers match. If it returns with anything other than OOH you will need the ECC software to make any corrections. The ECC field in the controller memory MUST be zeroed prior to a write operation.

8) **POSTAMBLE:** The rest of the sector from end of the ECC to the next sector pulse should be filled with zeroes. We recommend a minimum of 128 bytes of OOH data be filled into the memory to cover the worst case conditions. The controller itself will use only as many as is required for that particular sector.

1.6.2 **F/D Sector Format**

The recommended sector format is illustrated below.

<table>
<thead>
<tr>
<th>PREAMBLE</th>
<th>SYNC</th>
<th>TRACK</th>
<th>SECTOR</th>
<th>FILLER</th>
<th>DATA</th>
<th>CK/SUM</th>
<th>ECC</th>
<th>ECC/VALID</th>
<th>POSTAMBLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>256</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>to end of</td>
</tr>
<tr>
<td>bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>sector</td>
</tr>
</tbody>
</table>

A floppy disk sector consists of the following fields:

1) **PREAMBLE:** The preamble is a field of 40 bytes of OOH data, and is otherwise identical to the hard disk preamble.

2) **SYNC:** The sync byte is identical to the hard disk sync byte.

3) **TRACK:** The track byte is a single byte of data in the range of 0 to 76. It follows the sync byte in I/O operations, and should be used to verify that the correct track is being accessed during disk I/O.

4) **SECTOR:** The sector byte is a single byte of data in the range of 0 to 15. It is otherwise identical to the hard disk sector byte.
5) **FILLER**: The filler is 10 bytes of OOH data, and is used to keep the placement of the data field at a 'standard' position within the sector. It follows the sector byte in all disk I/O operations.

6) **DATA**: The data field consists of 256 bytes of user data, and follows the filler in I/O operations.

7) **CHECKSUM**: The checksum is 1 byte of data resulting from the software add with carry instruction of all the bytes from the track byte through the last byte of the data field. It's placement should be immediately following the last byte of the data field in disk I/O.

8) **ECC**: The use of the Error Correction Code is identical to that in the hard disk format. It's placement here is immediately following the checksum.

9) **ECC VALID**: The ECC valid byte is a single byte of data. If it has a value of OAAH, ECC is valid. Any other value indicates that ECC is not being used. We have defined this byte for software purposes so that your disk driver can compensate and make your 'old' disks upwardly compatible. ECC valid follows the ECC byte in disk I/O operations.

10) **POSTAMBLE**: The postamble follows the ECC valid byte, and is otherwise identical to that of the hard disk format. You should also maintain the minimum recommended 128 bytes of OOH data.
1.7 PORT FORMAT

1.7.1 Control Bits

Control is accomplished by outputting to the appropriate port the following bits:

<table>
<thead>
<tr>
<th>PORT</th>
<th>BIT</th>
<th>NAME</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>COH 0</td>
<td>0</td>
<td>DRIVE SELECT 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>DRIVE SELECT 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>HEAD SELECT 0</td>
<td>F/D HEAD SEL = 0 TO 1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>HEAD SELECT 1</td>
<td>H/D = 0 TO 3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>HEAD SELECT 2</td>
<td>RESERVED FOR 10 M/BYTE</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>STEP A TRACK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>DIRECTION IN/OUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>LOW CURRENT</td>
<td>H/D TRACK &gt; 127</td>
</tr>
<tr>
<td>C1H 0</td>
<td>0</td>
<td>SECTOR 0</td>
<td>F/D SECTOR = 0 TO 15</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>SECTOR 1</td>
<td>H/D = 0 TO 31</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>SECTOR 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>SECTOR 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>SECTOR 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>READ/WRITE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>ECC ENABLE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>WRITE PRE-COMPENSATION</td>
<td>H/D TRACK &gt; 63</td>
</tr>
</tbody>
</table>

The bit description is as follows:

<table>
<thead>
<tr>
<th>PORT</th>
<th>BITS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>COH 0-1</td>
<td>DRIVE SELECT: These two bits define the address of the drive to be used. The hard disk is always addressed as drive 0. Be careful to change the address jumper on the floppy drive, to 1, 2, or 3 as more drives are added.</td>
<td></td>
</tr>
</tbody>
</table>
Vector Graphic Dual-Mode Controller Board

COH 2-4 HEAD SELECT: Head select 0 and 1 are used to access the four surfaces of the current S/T disk drive. Head select 2 is reserved for the additional surfaces of the 10 megabyte version. For floppy operation, set these bits to zero to maintain compatibility with your older single sided disks so as not to read or write to the wrong side of the disk.

5 STEP A TRACK: Toggling the step bit, from 0 to 1 and back to 0 will cause the drive identified by the drive select bits to change the head position +1 track in the direction specified by the direction bit.

6 DIRECTION IN/OUT: The direction bit specifies whether to move the head in (1) toward the center hub (increasing the track number), or out (0) toward the outside edge of the disk (decreasing the track number).

7 LOW CURRENT: The low current bit should be set (1) only when performing a hard disk write when the track number is 128 or greater.

ClH 0-4 SECTOR: The sector bits specify which sector is to be accessed in a disk I/O. Sector 0-4 is used to address the 32 hard disk sectors, and Sector 0-3 is used to access the 16 floppy disk sectors.

5 READ/WRITE: To read the disk this bit is set to 1. To perform a write operation it should be set to 0. Remember, this and all other control lines should be set up correctly before performing a START.

6 ECC ENABLE: If the ECC Enable bit is set, the controller hardware will automatically insert the ECC into each sector as it is written. It also calculates, compares, and returns with zero or ECC in the read sector. Remember to zero the ECC field in the controller memory prior to a write operation.

7 WRITE PRE-COMPENSATION: Write pre-compensation should be set (1), only when performing a hard disk write with a track address of 64 or greater. It should be set to 0 at all other times.
### 1.7.2 Status Bits

Status is received by inputting from the appropriate port the following bits:

<table>
<thead>
<tr>
<th>PORT</th>
<th>BIT</th>
<th>NAME</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>COH</td>
<td>0</td>
<td>WRITE PROTECT</td>
<td>F/D</td>
</tr>
<tr>
<td>&quot;</td>
<td>1</td>
<td>READY</td>
<td>H/D</td>
</tr>
<tr>
<td>&quot;</td>
<td>2</td>
<td>TRACK 0</td>
<td>H/D &amp; F/D</td>
</tr>
<tr>
<td>&quot;</td>
<td>3</td>
<td>WRITE FAULT</td>
<td>H/D</td>
</tr>
<tr>
<td>&quot;</td>
<td>4</td>
<td>SEEK COMPLETE</td>
<td>H/D</td>
</tr>
<tr>
<td>&quot;</td>
<td>5</td>
<td>LOSS OF SYNC</td>
<td>H/D</td>
</tr>
<tr>
<td>&quot;</td>
<td>6-7</td>
<td>RESERVED</td>
<td>PULLED UP</td>
</tr>
<tr>
<td>CLH</td>
<td>0</td>
<td>FLOPPY DISK SELECTED</td>
<td></td>
</tr>
<tr>
<td>&quot;</td>
<td>1</td>
<td>CONTROLLER BUSY</td>
<td>R/W</td>
</tr>
<tr>
<td>&quot;</td>
<td>2</td>
<td>MOTOR ON</td>
<td>F/D</td>
</tr>
<tr>
<td>&quot;</td>
<td>3</td>
<td>TYPE OF HARD DISK</td>
<td>0=5MEG 1=10MEG</td>
</tr>
<tr>
<td>&quot;</td>
<td>4-7</td>
<td>RESERVED</td>
<td>PULLED UP</td>
</tr>
</tbody>
</table>

The bit description is as follows:

- **PORT COH 0 WRITE PROTECT**: The write protect is an active high signal, indicating that a write protect tab is installed on the floppy disk which has been selected by the drive select lines. You should check this bit, and when it is set, disable all write operations to this particular drive through software. There is no provision to write protect the hard disk.

- **PORT CLH 0**: The ready line is a hard disk only interface signal. When it is true, together with Seek Complete, indicates that the drive is ready to read or write, and that all I/O signals are valid. When this line is low all writing and seeking operations are inhibited. Ready time after power on is approximately 15 seconds.
COH 2 TRACK 0: When the active high track 0 signal is set to a one, the drive indicated by the drive select lines has its heads positioned at track zero.

" 3 WRITE FAULT: This hard disk only interface signal is used to indicate that a condition exists in the drive that can cause improper writing of the disk. This active high signal (1) indicates a fault, and inhibits further writing until the condition is corrected. The three conditions detected are as follows:

A. Write current exists in a head without a write gate signal or no write current exists in the head with a write gate and drive select signal.

B. Multiple heads selected.

C. DC voltages are grossly out of tolerance.

" 4 SEEK COMPLETE: This hard disk only signal is high only when the heads have settled on the final track desired after a series of step and delay instructions. Seek complete will be low during a power on recalibration which has been issued by the drive logic at power on because the heads are not over track zero. This signal must be high prior to performing any R/W operation.

" 5 LOSS OF SYNC: This hard disk active low signal (0) indicates that the phase lock loop sector counter has not achieved synchronization.

" 6-7 RESERVED: Reserved for future features. They are pulled up at the current time.

CLH 0 FLOPPY DISK SELECTED: This signal is high whenever the drive select lines decode a floppy drive address.

" 1 CONTROLLER BUSY: This signal is high only when a Start command has been issued, and a read or write operation is in progress.

" 2 MOTOR ON: This floppy disk only signal is high only when the motor of the drive selected has not timed out from a previous drive select operation.

" 3 TYPE OF HARD DISK: This is a hardwired signal indicating the type of hard disk selected, 0=5megabyte version, and 1=10megabyte version.

" 4-7 RESERVED: Reserved for future use. (Pulled up)
1.7.3 Data Bits

The data port addressed at C2H contains 8 bits of IN/OUT data information. These are the DI and DO bits 0-7 of the S-100 bus. The bus is tied through tri-state driver/receivers to a bi-directional 512 byte RAM on-board the controller. The memory is filled or read back sequentially from address zero. The address counter increments automatically after each access until the desired address is reached. All the information is contained in approximately the first 400 bytes of memory with the remainder of RAM not used. Before performing any I/O or R/W operation you must set up the control bits via the two control ports, check the appropriate status bits on the two status ports, and load the address counter to zero by issuing a RESET through the reset/start port. When outputting to the controller in preparation for a write operation, you must load, starting at address zero, the preamble through postamble. When inputting from the controller after a read operation, the first byte read will be the sync byte followed by the remainder of the bytes through the postamble, in the appropriate disk format.

1.7.4 Reset/Start Port Format

The reset/start port is a control only port, and there are no data lines associated with it. By performing an INPUT command from port C3H the controller will automatically RESET the memory address register to all zeroes. A RESET should be accomplished before each START READ/WRITE, or INPUT/OUTPUT DATA operation. A START is accomplished by issuing an OUTPUT command to port C3H with the control bit "Read/Write" (Port COH) set to the appropriate position. The controller will then automatically READ from the disk into the controller RAM or WRITE on the disk the contents of the controller RAM. The controller will NOT operate correctly unless the Status and Control Ports are serviced properly before issuing a Start command. Remember that a reset only sets the controller memory address counter to zero, and that a start only initiates and completes a transfer of data between the disk and controller memory as specified by the control bits.

1.8 OPERATION

This section will discuss only the basic functions of calibrate, select, seek, read, write, and use of the status and control bits. In the next section we will discuss some of the features that you might want to add to your driver, such as functions and error detection. At this time we are going to recombine the hard and floppy disk sections as an aid to help you use the same routines for both types of drives.
1.8.1 Calibration

Although the hard disk drives themselves provide a power on calibration, it is possible to be behind track 0. You may also at some time lose track of the drive head position and want to perform a recalibration cycle. Note the use of the counter to prevent a loop of excessive attempts. The suggested flowchart is as follows:

1. CALIBRATE
2. SELECT DRIVE
3. Y HARD DISK N
4. CLEAR COUNTER
5. LOAD COUNTER TO 80
6. STEP IN 4 TRACKS
7. STEP OUT 1 TRACK
8. TRACK = 0 ?
   Y
   THIS IS TRACK 0
   RESET SOFTWARE TRACK COUNTERS FOR THIS DRIVE
   EXIT
8. N
   INCREMENT COUNTER
   IS COUNTER >= 160 ?
   Y
   PRINT ERROR MESSAGE "TRACK 0 NOT FOUND"
   EXIT

Rev. 2.0-A 2/1/81
1.8.2 Drive Selection

The physical selection of a drive is the result of the drive select bits arriving at the controller via an output command to port COH. The result of this is that the hard disk then immediately responds with a ready signal and its other status bits. In the case of floppy disks, it starts all the motors. Checking the status bit Motor On indicates that all floppy drive motors are running. The suggested format is:

SELECT DRIVE

GET STATUS OF MOTOR ON

SEND TO CONTROL PORTS, DRIVE, HEAD, SECTOR, R/W, AND ECC ON.

Y HARD DISK?

N IN SYNC?

Y READY?

N WRITE FAULT?

PRINT ERROR

Y DELAY 250MSEC. FOR SPEED TO COME UP

N EXIT

Y WAS MOTOR ON?

N EXIT

Y PRINT ERROR

N EXIT

Y EXIT

N EXIT
1.8.3 Seek a Track

Seeking is simply a series of step commands (Toggle Bit 5, Control Port COH) with each step followed by a 3 millisecond delay to allow the mechanism to react. Additional head settling time is necessary after the final track is reached. For the floppy disk it is 15 milliseconds, and for the hard disk it is automatically provided in the status bit, Seek Complete. The direction bit must be set 100 nsec. prior to the step bit. Separate instructions are recommended.
1.8.4 Write

When writing the disk be sure to load the correct data format for the type of disk in use. The controller memory can be read or written at the rate that I/O instructions can be issued. One method is:

```
WRITE
```

```
CMAR=CONTROLLER MEMORY
ADDRESS REGISTER
```

```
SELECT DRIVE
```

```
Y HARD DISK
N
```

```
FORMAT HARD DISK BUFFER
FORMAT FLOPPY DISK BUFFER
```

```
RESET CMAR
```

```
LOAD CONTROLLER MEMORY
CALCULATE CHECKSUM AND POKE IN IF FLOPPY DISK
```

```
LOAD IN CONTROL BITS WITH WRITE
```

```
RESET CMAR
```

```
START WRITE
```

```
Y CONTROLLER BUSY
N
```

```
EXIT
```
1.8.5 Read

Reading is approximately the reverse process of writing. The difference is that you check the header for the correct head, track, and sector, and check the trailer for ECC and checksum. The suggested flowchart follows:

```
READ
SELECT DRIVE
LOAD IN CONTROL BITS WITH READ
RESET CMAR
START READ

CONTROLLER BUSY?
Y
N
RESET CMAR
INPUT FROM CONTROLLER MEMORY

HEADER AND TRAILER CORRECT?
N
PRINT ERROR
Y
EXIT
```
1.9 ERROR HANDLING

An important consideration, which may not be ignored in the design of a flexible disk driver, is the handling of errors which occur. Magnetic storage devices, in general, are subject to errors. Flexible disks are subject to damage or contamination due to handling, making error detection particularly important. Although contamination is not the problem with the Winchester hard disk technology, should a hard error occur, it must be dealt with for successful system operation. Most errors are of a temporary nature and will be invisible to the system with a properly designed disk driver. The following discussion leaves it up to the user to implement flexible error routines to handle H/D and F/D combination systems. Examples of basic and specific error codes and types can be found in the sample disk driver at the end of the manual.

Most errors can be attributed to one or more of the following sources:

1) Transient Electrical Noise

2) Media Contamination - Particles of foreign substances may become lodged between the head and the recording surface of the disk and cause data errors.

3) Head Positioning - The read write head may be positioned to the wrong track if the specified step rate is exceeded or may be marginally positioned if a drive is misadjusted.

4) Disk Centering - Due to the way a flexible disk is constructed, or in the event the disk is damaged or distorted due to mis-handling, it is possible that a diskette may be improperly clamped to the spindle in the disk drive.

The following are the suggested procedures to perform proper error handling in disk read/write operations:

1.9.1 Read Operations

1) Step the positioner to the desired track.

2) Perform a read operation as described in Section 1.6.5. If a header or checksum error occurs, re-read the sector up to 6 times. If an ECC error occurs, re-read the sector up to 6 times checking for 2 consecutive read operations in which the Error Correction Code is identical. At that time it would be safe to say that the ECC is good and to go to the software ECC in your system for the correction.
3) If the six retries were unsuccessful, step the positioner off one track and then back to the desired track. Repeat Step 2. If still unsuccessful, step the positioner off one track in the other direction and then back. Repeat Step 2.

4) Perform the restep procedure given in Step 3 up to 4 times. If still unsuccessful, deselect the unit and delay about 200msec. Reselect the unit, restore to track 0, and re-seek to the desired track. Repeat Steps 2 and 3. Perform this reselect function up to 3 times. If still unsuccessful, abort the operation with a permanent I/O error.

1.9.2 Write Operations

1) Step the positioner to the desired track.

2) Read the sector immediately preceding the desired sector. Any errors which occur should be handled in the manner described for normal read operations. This ensures that the correct head and track have been selected and that the sector counter is synchronized with the disk.

3) Write the desired sector as described in Section 1.6.4.

4) Read the sector just written to check that the data was recorded properly. If an error occurs, repeat Steps 2, 3, and 4 up to five times.

5) If unsuccessful, perform the restep operation as described for the read operation and repeat Steps 2, 3, and 4.

6) If 4 restep operations are unsuccessful, perform the reselect operation as described for the read operation up to 3 times. If still unsuccessful, abort the operation with a permanent I/O error.

If a permanent I/O error occurs, there may be a defect in or damage to the recording surface of the disk, the disk may be improperly centered, or the disk may have been written on a marginal drive.

The restep procedure takes advantage of the friction in the positioner system causing the head position to deviate slightly from the nominal track position. This position will be different when the head is stepped to a track from different positions. In normal operations this position difference has no effect, but it can possibly recover data that was written on a marginally aligned drive.

The reselect procedure serves to dislodge any foreign particles and to recalibrate the positioner, should it be positioned to the wrong track.
1.10 DISK DRIVER

As an example of all the principles in this manual, a sample disk driver is presented following this section. This driver provides the facilities to seek to a track, seek and read a sector, seek and write a sector, seek and verify a sector, initialize the disk driver, and perform a write-protect detect test.

The power-on recalibration is transparent. The driver maintains a table containing the current track address of each drive connected to the controller. The user's power-on initialize software must set the entries in the table to OFFH. The first time a drive is accessed, the driver will recognize this flag and recalibrate the positioner on the drive before performing the specified operation.

When the driver is called, a register pair must point to a parameter block referred to as the Disk Control Block or DCB which specifies the operation to be performed. When the driver returns, the condition code will reflect the status of the operation. (See the listing for details.)

The DCB is structured as follows:

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>BIT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTION CODE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCB + 0</td>
<td>0</td>
<td>SEEK TRACK ONLY</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>SEEK AND READ SECTOR</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>SEEK AND WRITE SECTOR</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>SEEK AND VERIFY SECTOR</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>INITIALIZE DISK DRIVER</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>WRITE PROTECT DETECT TEST</td>
</tr>
<tr>
<td>CONTROL FLAGS/UNIT SELECT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCB + 1</td>
<td>0-3</td>
<td>UNIT ADDRESS</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>INTERRUPT SAVE</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>WRITE PROTECT DETECT</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>COMPARE VERIFY AND WRITE CHECKSUM</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>PRE-WRITE ID CHECK CONTROL</td>
</tr>
<tr>
<td>DCB + 2</td>
<td></td>
<td>SECTOR ADDRESS (0 TO 15)</td>
</tr>
<tr>
<td>DCB + 3</td>
<td></td>
<td>TRACK ADDRESS (LSB) (0 TO 153)</td>
</tr>
<tr>
<td>DCB + 4</td>
<td></td>
<td>TRACK ADDRESS (MSB) (0 TO 153)</td>
</tr>
</tbody>
</table>

(cont.)
BUFFER ADDRESS (LSB)
BUFFER ADDRESS (MSB)

BUFFER ADDRESS IS THE START ADDRESS OF THE 268 BYTE READ/WRITE BUFFER TO BE USED IN PERFORMING THE OPERATION. EXAMPLES OF THE BUFFER ORGANIZATION FOR THE FLOPPY AND HARD DISK CAN BE FOUND IN THE SAMPLE DRIVER FOLLOWING THIS SECTION.

To perform a write operation, move the data to the read/write buffer, set up the DCB, and call the driver.

To perform a read operation, set up the DCB and call the driver. When the operation is complete, the data from the desired sector will be in the read buffer.
TITLE FLOPPY DISK DRIVER ROUTINES

COPYRIGHT (C) 1980 VECTOR GRAPHIC INC.
31364 VIA COLINAS
WESTLAKE VILLAGE, CA. 91361

FLOPPY DISK DRIVER ROUTINES FOR 5 1/4" FLOPPIES
VECTOR'S FD/HD CONTROLLER
AND CP/M 2.21
WRITTEN BY: JAY RASMUSSEN
LAST REVISION DATE: 01/26/81

THIS MODULE CONTAINS THE PHYSICAL DISK DRIVER ROUTINES.
IT IS NORMALLY ACCESSED THROUGH THE BIOS DISKCMD ROUTINE.
ALL PARAMETERS ARE PASSED IN A DEVICE CONTROL BLOCK (DCB).
THE DCB IS POINTED TO BY THE IX REGISTER.

DESCRIPTION OF THE DCB:

DCB+0 FUNCTION CODE
0 SEEK TRACK ONLY
1 SEEK AND READ SECTOR
2 SEEK AND WRITE SECTOR
3 SEEK AND VERIFY SECTOR
4 INITIALIZE DISK DRIVER
5 DRIVE READY TEST

DCB+1 CONTROL FLAGS/UNIT SELECT
BIT FUNCTION
0-3 UNIT ADDRESS
4 INTERRUPT SAVE
5 WRITE PROTECT DETECT
0=INHIBIT 1=PERFORM
6 COMPARE VERIFY AND WRITE CHECKSUM
0=INHIBIT 1=PERFORM
7 PRE-WRITE ID CHECK CONTROL
0=PERFORM 1=INHIBIT

DCB+2 SECTOR ADDRESS (0 TO 15)
DCB+3&4 TRACK ADDRESS (0 TO 153)
DCB+5&6 BUFFER ADDRESS
BUFF ADDRESS IS THE START ADDRESS OF THE
READ/WRITE BUFFER TO BE USED IN PERFORMING
THE OPERATION.

ALL OPERATIONS REQUIRE A 268 BYTE BUFFER ORGANIZED AS:
FLOPPY:
BYTE 0 TRACK ID
BYTE 1 SECTOR ID
BYTES 2-11 FILLER BYTES
BYTES 12-267 DATA

HARD DISK:
BYTE 0   HEAD ID
BYTE 1   TRACK ID
BYTE 2   SECTOR ID
BYTES 3-258 DATA
BYTES 259-267 NOT USED

BYTES 0, 1 & 2 ARE FILLED IN AS NECESSARY BY THE DRIVER

THE DISK I/O DRIVER RETURNS WITH THE CONDITION CODE SET TO 'Z' ON SUCCESS AND 'NZ' ON FAILURE OR CONDITION DETECTION.

THE ERROR CODES FOLLOW THE FOLLOWING SCHEME:

<table>
<thead>
<tr>
<th>VALUE</th>
<th>TYPE OF ERROR</th>
<th>CAUSES BDOS ERR #</th>
</tr>
</thead>
<tbody>
<tr>
<td>00H</td>
<td>NO ERROR</td>
<td>00</td>
</tr>
<tr>
<td>1XH</td>
<td>DRIVE ERRORS</td>
<td>05</td>
</tr>
<tr>
<td>2XH</td>
<td>TIMEOUT ERRORS</td>
<td>05</td>
</tr>
<tr>
<td>3XH</td>
<td>PARAMETER ERRORS</td>
<td>01</td>
</tr>
<tr>
<td>4XH</td>
<td>VERIFY ERRORS</td>
<td>01</td>
</tr>
<tr>
<td>5XH</td>
<td>ECC AND CHKSUM ERRORS</td>
<td>01</td>
</tr>
<tr>
<td>6XH</td>
<td>SYSTEM ERRORS</td>
<td>01</td>
</tr>
<tr>
<td>7XH</td>
<td>OVERLAY ERRORS</td>
<td>01</td>
</tr>
<tr>
<td>8XH</td>
<td>RESERVED FOR ASSIGNMENT</td>
<td>??</td>
</tr>
<tr>
<td>9XH</td>
<td>REALLOCATION ERRORS</td>
<td>06</td>
</tr>
<tr>
<td>AXH</td>
<td>RESERVED FOR ASSIGNMENT</td>
<td>??</td>
</tr>
<tr>
<td>BXH</td>
<td>RESERVED FOR ASSIGNMENT</td>
<td>??</td>
</tr>
<tr>
<td>CXH</td>
<td>RESERVED FOR ASSIGNMENT</td>
<td>??</td>
</tr>
<tr>
<td>DXH</td>
<td>RESERVED FOR ASSIGNMENT</td>
<td>??</td>
</tr>
<tr>
<td>EXH</td>
<td>SPECIAL TIMESHARE ERRORS</td>
<td>80</td>
</tr>
<tr>
<td>FXH</td>
<td>WARNINGS</td>
<td>07 (FOR 0F1H CODE)</td>
</tr>
</tbody>
</table>

SOME SPECIFIC CODES ARE:

<table>
<thead>
<tr>
<th>VALUE</th>
<th>TYPE OF ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>NO ERROR</td>
</tr>
<tr>
<td>11</td>
<td>DRIVE NOT READY</td>
</tr>
<tr>
<td>12</td>
<td>WRITE PROTECTED</td>
</tr>
<tr>
<td>13</td>
<td>WRITE FAULT</td>
</tr>
<tr>
<td>14</td>
<td>TRACK 0 NOT FOUND</td>
</tr>
<tr>
<td>15</td>
<td>CONTROLLER NOT JUMPERED FOR FLOPPY ONLY SYSTEM</td>
</tr>
<tr>
<td>21</td>
<td>CONTROLLER BUSY TIMEOUT</td>
</tr>
<tr>
<td>22</td>
<td>SEEK COMPLETE TIMEOUT</td>
</tr>
<tr>
<td>23</td>
<td>LOSS OF SYNC TIMEOUT</td>
</tr>
<tr>
<td>31</td>
<td>BAD COMMAND CODE</td>
</tr>
<tr>
<td>32</td>
<td>BAD UNIT VALUE</td>
</tr>
<tr>
<td>33</td>
<td>BAD SECTOR VALUE</td>
</tr>
<tr>
<td>34</td>
<td>BAD TRACK VALUE</td>
</tr>
<tr>
<td>41</td>
<td>TRACKS DON'T MATCH</td>
</tr>
<tr>
<td>42</td>
<td>SECTORS DON'T MATCH</td>
</tr>
<tr>
<td>43</td>
<td>HEADS DON'T MATCH</td>
</tr>
<tr>
<td>44</td>
<td>DATA DOESN'T MATCH</td>
</tr>
</tbody>
</table>
* 51 READ CHKSUMS DON'T MATCH
* 52 READ/WRITE CHKSUMS DON'T MATCH
* 53 ECC ERROR
* 54 UNCORRECTABLE ECC ERROR
* 55 UNCORRECTABLE SPASM ERROR
* 56 SYNC BYTE ERROR
* 91 NO ROOM LEFT IN TABLE
* E1 RESET DISK FUNCTION ATTEMPT
* E2 DISABLED AND PROTECTED
* E3 DISABLED AND BUSY
* F1 NOT MUCH ROOM LEFT IN TABLE
* TIMESHARE REQ 'TIMESHARE BIOS ? (0=NO; 1=YES) :
* HARDISK REQ 'INCLUDE HARD DISK ? (0=NO; 1=YES) :
* IFT HARDISK
* DUALLOG REQ 'SINGLE OR DUAL LOGICAL HARD DRIVES (0=SNG; 1=DL) :
* ENDF
*
* PHYSDRVR PUSH B ;SAVE REGISTERS
* PUSH D
* PUSH H
* PUSH Y
* LXI Y,RETRY ;SET RETRY POINTER
* SSPD ENTRystack ;SAVE THE STACK POINTER
* MOV A,DCBCCOM(X) ;GET THE COMMAND CODE
* CPI INITCOM
* JZ SYSINIT ;GO INITIALIZE DISK SYSTEM
* CPI WDCOM+1
* MVI A,31H ;BAD PARAMETER-COMMAND
* JRNC EREXIT1 ;GO REPORT PARAMETER ERROR
*
* WAITAVAL: WAIT FOR CONTROLLER TO BECOME AVAILABLE
*
* IFF TIMESHARE
*
* WAITAVAL DI
* LXI H,BUSY ;GET THE BUSY FLAG
* MOV A,M
* ORA A
* JRZ WAITAVAL10
* BIT 4,DCBUNT(X)
* JRZ WAITAVAL10
* EI
* JR WAITAVAL
* MVI M,OFFH ;SET CONTROLLER BUSY
*
* ENDF
*
* IFT TIMESHARE ;NEEDED FOR MULTUSER
; IS THERE AN ACTIVE PROTECTION?

; JUMP IF NOT ACTIVE

; GET THIS USER'S CONSOLE #

; PROTECT FROM THIS USER?

; NO, THIS IS PROTECT OWNER

; TEST INTERRUPT STATUS

; DISABLED AND PROTECTED ERROR

; GET CONTROLLER BUSY FLAG

; GO TAKE CONTROL IF NOT BUSY

; ELSE TEST INTERRUPT STATUS

; DISABLED AND BUSY ERROR

; SET CONTROLLER BUSY

; GET THE DCB POINTER

; INTO HL

; MAKE ECC VALID HERE

; COMMAND CHECKED ALREADY

; CHECK THE UNIT

; BAD PARAMETER- UNIT

; GETFLAGS

; HST.FLOP

; MOV A,E
* IFF HARDDISK

  CPI 2
  JRNC VALIDATE03 ;JUMP IF DRIVE C OR D
  XRI 1 ;ELSE SWAP A & B

  ENDFI

* IFT HARDDISK
IFT DUALLOG

* ORA A
JRZ VALIDATE03
DCR A

ENDIF ENDFI

INVALIDATE03 STA DRIVE TO USE
MOV A,DCBCOM(X) ;GET COMMAND AGAIN
CPI WPDCOM
JRNZ VALIDATE05
CALL CHKWRPROT ;DO COMMAND 5
JMP EXIT

INVALIDATE05 EQU $

* IFF HARDDISK

INX H
MOV A,M
STA SECTOR TO USE ;SAVE SECTOR TO USE
CPI 16 ;MUST BE 0 TO 15
JRC VALIDATE10
MVI A,33H ;BAD PARAMETER- SECTOR
JMP EXIT

EREXIT1

INVALIDATE10 INX H ;POINT TO LOW HALF OF TRACK
MVI A,F,TRKMAX ;GET THE NUMBER OF TRACKS
MOV C,A ;SAVE IT FOR LATER TEST
MVI B,HEAD0 ;DECIDE WHICH HEAD NOW
SRLR A ;NUMBER OF TRACKS/2
CMP M ;USE HEAD0 MASK IF
JRC VALIDATE20 ;TRACK IS HALF OR MORE
MOV A,M
JR VALIDATE30

INVALIDATE20 MOV B,A
LDA HST.SNG ;IS THIS A SINGLE SIDED DRIVE?
ORA A
JRNZ VALIDATE40 ;TRACK IS TOO LARGE
MOV A,M
SUB B
MVI B,HEAD1 ;ELSE USE HEAD1 MASK

INVALIDATE30 STA TRACK TO USE ;SAVE ACTUAL TRACK VALUE TO SEEK
MOV A,B
STA SLCT TO USE ;SAVE CHosen HEAD MASK
MOV A,C ;GET THE NUMBER OF TRACKS AGAIN
CMP M                  ;MUST BE =< F.TRMX
JRNC VALIDATE50

VALIDATE40
MVI A,34H               ;BAD PARAMETER- TRACK
JR EREXIT1

VALIDATE50
INX H                  ;NOW HIGH HALF OF TRACK
MOV A,M
ORA A
JRNZ VALIDATE40
LDA DRIVE TO USE       ;GET THE UNIT
LXI H,SLCT TO USE      ;POINT TO HEAD MASK
ORA M
MOV M,A                 ;SAVE HEAD AND UNIT INFO

* ENDIF *

* IFT HARDDISK *

LDA HST.FLOP            ;GET FLAG FOR DISK TYPE
ORA A
PUSH PSW
INX H
MOV A,M
STA SECTOR TO USE       ;SAVE SECTOR TO USE
JRNZ VALIDATE07         ;JUMP IF FLOPPY
CPI 32
JR VALIDATE08

VALIDATE07
CPI 16                  ;MUST BE 0 TO 15
JR VALIDATE08

VALIDATE08
MVI A,33H                ;BAD PARAMETER- SECTOR

EREXIT1
JMP EXIT

VALIDATE10
POP PSW
JRNZ VALIDATE20         ;JUMP IF FLOPPY
LXI H,H5.DISK           ;POINT TO 5MEG PARAMS
IN STATUS1
ANI HDISKMASK
JRZ VALIDATE30          ;JUMP IF 5 MEG HARD
INX H
INX H
JR VALIDATE30

VALIDATE20
LXI H,FS.DISK           ;POINT TO SNG/SIDED FLOPPY PARAMS
LDA HST.SNG
ORA A
JRNZ VALIDATE30         ;JUMP IF SNG/SIDED FLOPPY
INX H
INX H
JR VALIDATE30

VALIDATE30
MOV B,M                 ;GET HEADS PER DRIVE
INX H
MOV L,M                 ;GET TRACKS PER HEAD
MVI H,0
MOV C,H                 ;CLEAR HEAD BYTE
PUSH B
PUSH H
MOV E,L
MOV D,H
DCB B
JRZ VALIDATE50
DAD D
DJNZ VALIDATE40

DCX H ; TRACK MAXIMUM NOW IN 'HL'
MOV E, DCBTRKL(X)
MOV D, DCBTRKH(X) ; TRACK TO CHECK NOW IN 'DE'
ORA A ; CLEAR CARRY
DSBC D
MVI A, 34H ; BAD PARAMETER- TRACK
JM EXIT ; REPORT THE ERROR
POP H ; GET TRACKS PER HEAD
POP B ; GET HEADS PER LOGICAL DRIVE
XCHG

ORA A ; CLEAR CARRY
DSBC D
JM VALIDATE70
INR C ; COUNT THE HEADS
JR VALIDATE60

DAD D
MOV A, L
STA TRACK TO_USE ; SAVE THE TRACK
LDA HST, FLOP
ORA A
MOV A, C ; GET HEAD COUNT
JRNZ VALIDATE90 ; JUMP IF FLOPPY

* IFT DIALOG *

MOV A, DCBUNT(X)
ANI 0FH ; MASK OUT THE DRIVE
MOV A, C ; JUMP TO USE HEAD AS IS
JRZ VALIDATE80
MOV A, B ; ELSE ADD IN THE HEAD
ADD C ; OFFSET FOR 1 LOGICAL DRIVE

* ENDIF *

STA HEAD TO USE ; SHIFT HEAD BITS UP TO
VALIDATE80 ADD A ; PROPER LOCATION
ADD A ; SAVE THE HEAD MASK
MOV B, A ; SAVE THE HEAD TO_USE
LDA DRIVE TO_USE

VALIDATE90 ORA B ; PUT HEAD AND UNIT TOGETHER
STA SLCT TO_USE ; AND SAVE IT.

* ENDIF *

* ENSURE DRIVE IS READY *

CALL SLCT ; SELECT THE DRIVE

* SEEK TO DESIRED TRACK *

CALL SEEK ; SEEK THE TRACK

MOV A, DCBCOM(X) ; GET THE COMMAND CODE
**PERFORM READ, WRITE OR VERIFY FUNCTIONS**

**RETRY CONTROL STRUCTURE FOR READ, WRITE AND VERIFY OPERATIONS:**

**LEVEL1**

PRIMITIVE RETRIES—DEPENDS ON ROUTINE, BUT A MINIMUM

OF 5 RETRYS OF THE OFFENDING OPERATION WILL BE

PERFORMED.

**LEVEL2**

IF THE LEVEL 1 RETRIES ARE NOT SUCCESSFUL, THEN THE

POSITIONER WILL BE STEPPED OFF THE TRACK AND BACK.

THE LEVEL 1 RETRIES WILL BE PERFORMED AGAIN. THE LEVEL

2 RETRIES WILL BE PERFORMED UP TO 4 TIMES.

**LEVEL3**

IF THE LEVEL 2 RETRY PROCEDURE IS NOT SUCCESSFUL, THE

UNIT WILL BE DESELECTED TO UNLOAD THE HEAD, THEN THE

UNIT WILL BE RESELECTED, THE POSITIONER WILL BE

RECALIBRATED AND MOVED BACK TO THE DESIRED TRACK. THE

LEVEL 1 AND 2 RETRY PROCEDURES WILL BE PERFORMED AGAIN.

THIS WILL BE DONE UP TO 3 TIMES. IF NOT SUCCESSFUL, A

PERMANANT I/O ERROR WILL RESULT.

**LEVEL0**

MVI L3TRY(Y),3 ;PRESET THE RETRY COUNTERS

MVI L2TRY(Y),4

**SELECT THE DESIRED FUNCTION AND PERFORM**

**PHYS10**

MVI RETRY1(Y),5

MOV A,DCBCOM(X) ;GET COMMAND CODE

DCR A

JRNZ PHYS20 ;JUMP IF NOT READ COMMAND

**READ THE SECTOR**

**READSEC**

CALL READFUNC

JR PHYS30 ;GO CHECK FOR ERRORS

**PHYS20**

DCR A ;CHECK COMMAND

JRNZ VERIFYSEC ;JUMP IF VERIFY COMMAND

**WRITE THE SECTOR**

**WRITESEC**

CALL WRITEFUNC

JR PHYS30 ;GO CHECK FOR ERRORS

**VERIFY THE SECTOR**

**VERIFYSEC**

XRA A ;RESET READ-AFTER-WRITE FLAG

CALL VERIFYFUNC

**PHYS30**

ORA A ;TEST RETURNED ERROR CODE

JRZ EXIT ;NO ERRORS, SO EXIT

STA TEMPErr
CALL RESTEP
JNZ L2RTRY(Y) ; PERFORM UP TO 4 TIMES
JNZ PHYS10
LDA TEMPERR
CPI 30H
JNC EXIT ; END EARLY ON DRIVE/TIMEOUT
CALL RESLCT
JNZ L3RTRY(Y) ; PERFORM UP TO 3 TIMES
JNZ PHYS00
LDA TEMPERR

* EXIT
PUSH PSW
XRA A
STA BUSY ; RELEASE THE CONTROLLER
POP PSW
LSPD ENTRystack ; RESTORE STACK POINTER
POP Y
POP H
POP D
POP B
RET

* VERIFYFUNC
MVI C,0 ; NO DATA TO BE MOVED
JR READFUNC10 ; 'A' HOLDS RAW STATUS

* READFUNC
MVI C,0FFH ; DATA TO BE MOVED
XRA A
READFUNC10 STA RAWFLAG ; SAVE READ-AFTER-WRITE STATUS
MOV A,C
STA MOVEFLAG ; SAVE MOVEFLAG
MVI RETRY0(Y),6 ; SET A RETRY COUNTER
LXI H,LASTECC ; CLEAR FOUR BYTES
MVI B,4
READFUNC20 MVI M,0
INX H
DINZ READFUNC20
READFUNC30 CALL PHYSREAD
JRNZ READFUNC40 ; ANY ERRORS ?
LDA VALIDECC
CPI 0A9H
JRNZ READFUNC60 ; ECC BYTES ARE NOT VALID
CALL ECCTest ; TEST ECC BYTES FOR ZERO
JNZ READFUNC60 ; JUMP IF GOOD ECC CHECK
CALL CMPECC ; COMPARE CURRENT AND LAST ECC'S
JNZ READFUNC50 ; JUMP IF THEY ARE THE SAME
MVI A,55H ; ECC SPASM ERROR CODE
READFUNC40 DCR RETRY0(Y) ; PERFORM UP TO 6 TIMES
JRNZ READFUNC30
ORA A
RET
READFUNC50 CALL ECCFIX ; CORRECT THE MEMORY IMAGE
JRNZ READFUNC40 ; UNCORRECTABLE ECC ERROR
LDA DBL_SN
BIT 7,A
JNZ READFUNC60
LXI H,SOFTERR
READFUNC60: CALL BIOSPRINT ; REPORT THE SOFT ERROR
          CALL CHKVALUES
          JRNZ READFUNC40
          LDA RAWFLAG ; IS THIS A READ-AFTER-WRITE?
          ORA A
          JRZ READFUNC70
          LDA RAWERR
          ORA A
          MVI A, 44H ; BAD DATA ERROR CODE
          JRNZ READFUNC40

READFUNC70: EQU $

*          IFT HARDDISK

*          LDA HST.FLOP ; TEST FOR FLOPPY
          ORA A
          JRZ READFUNC70 ; RETURN IF HARD DISK (NO ERROR)

*          ENDF

*          BIT CMPBIT, DCBUNT(X)
          JRZ READFUNC80
          LXI H, CHKSUMHOLD ; POINT TO CHKSUM READ FROM DISK
          LDA CHKSUMR ; GET COMPUTED CHKSUM
          SUB M
          JRZ READFUNC70 ; RETURN IF NO ERRORS
          MVI A, 51H ; ELSE SET ERROR CODE
          JR READFUNC40 ; GO THRU RETRY SCHEME

READFUNC80: XRA A ; RETURN WITH NO ERRORS
          RET

*          WRITEFUNC BIT HCIBIT, DCBUNT(X)
          JRNZ WRITEFUNC10 ; JUMP IF NO PRE-WRITE VERIFY
          MOV A, DCBSEC(X)
          SUI 2 ; STEP BACK TWO SECTORS

*          IFT HARDDISK

*          ANI 0FH ; MODULO 16

*          ENDF

*          IFT HARDDISK

*          ANI 1FH ; MODULO 32
          MOV B, A
          LDA HST.FLOP
          ORA A
          MOV A, B
          JRZ WRITEFUNC05 ; JUMP IF HARD DISK
          ANI 0FH ; MODULO 16

*          ENDF

*          WRITEFUNC05 STA SECTOR_TO_USE
          XRA A ; RESET READ-AFTER-WRITE FLAG
CALL VERIFYFUNC ;PERFORM THE VERIFY
RNZ ;DROP OUT IF ANY ERRORS
MOV A, DCBSEC(X)
STA SECTOR TO USE

WRITEFUNC10
CALL PHYSWRITE
JRNZ WRITEFUNC25 ;TIMEOUT ERROR
BIT WPDBIT, DCBUNT(X)
JRZ WRITEFUNC20 ;DON'T TEST WRITE PROTECT
CALL CHKWRTPROT
JRZ WRITEFUNC20 ;NOT PROTECTED
MVI A, 12H ;WRITE PROTECT ERROR CODE
JMP EXIT ;ABORT- DON'T DO RETRIES

WRITEFUNC20
BIT HCIBIT, DCBUNT(X)
RNZ ;NO READ AFTER WRITE, (A=0)
MVI A, OFFH ;SET READ-AFTER-WRITE FLAG
CALL VERIFYFUNC ;READ AFTER WRITE VERIFY
JRZ WRITEFUNC30 ;OK IF NO ERRORS

WRITEFUNC25
DCR RETRY1(Y) ;ELSE RETRY WRITE AND CHECK
JRNZ WRITEFUNC10 ;UP TO 3 TIMES

WRITEFUNC30
ORA A ;RETURN IF ANY ERRORS
RNZ
BIT CMPBIT, DCBUNT(X)
RZ ;RETURN IF COMPARE NOT WANTED
LDA CHKSUMR
LXI H, CHKSUMW
SUB M
RZ ;RETURN IF GOOD COMPARE
MVI A, 52H ;R/W CHKSUMS DON'T MATCH
RET

* SYSINIT: INITIALIZE THE DISK TRACK TABLES
* SYSINIT
XRA A
CALL LDTRK10 ;POINT TO THE TRACK TABLE
MVI B, 4

SYSINIT10
MVI M, OFFH ;SET TO UNKNOWN
INX H
DJNZ SYSINIT10

EXIT1
JMP EXIT

* CHKWRTPROT: SELECT DRIVE AND TEST WRITE PROTECT STATUS
* CHKWRTPROT
LDA DRIVE TO USE ;GET THE UNIT TO TEST
OUT CNTLPRTO
IN STATUS0
ANI WPROMASK ;SETS 'A' TO 00H OR 01H
RET

* PHYSWRITE: PHYSICAL WRITE TO SECTOR
* PHYSWRITE
IN RESETPORT ;RESET THE CONTROLLER COUNTER
STA OPFLAG ;OPFLAG = OFFH ON WRITES
MVI B, 40 ;PREAMBLE IS 40 ZEROS (FLOPPY)
* IFT HARRDISK
LDA HST.FLOP
MOV D,A
BIT 0,D
JRNZ PHYSWRITE05
MVI B,HPRFAMBLE ;PREAMBLE IS 30 ZEROS (HARD)

ENDIF

PHYSWRITE05 XRA A
PHYSWRITE10 OUT DATAPORT
DJNZ PHYSWRITE10
CMA
OUT DATAPORT ;SEND OUT THE SYNC BYTE
LHLD BUFADDR ;POINT TO BUFFER & SET UP HEADER

* IFT HARDDISK

* BIT 0,D
JRNZ PHYSWRITE12 ;JUMP IF FLOPPY
LDA HEAD TO USE
MOV M,A _ _ ;POKE IN THE HEAD BYTE
INX H

* ENDIF

* PHYSWRITE12 LDA TRACK TO USE
MOV M,A _ _ ;POKE IN THE TRACK BYTE
INX H
LDA SECTOR TO USE
MOV M,A _ _ ;POKE IN THE SECTOR BYTE
INX H

* IFT HARDDISK

* BIT 0,D
JRZ PHYSWRITE17 ;JUMP IF HARD DISK

* ENDIF

* MVI B,10 ;FILLER COUNTER
PHYSWRITE15 XRA A ;NOW SEND 10 00'S
MOV M,A
INX H
DJNZ PHYSWRITE15

PHYSWRITE17 XRA A ;CLEAR CARRY
MOV E,A ;CLEAR CHECKSUM
MOV B,A ;SET COUNTER FOR 256
LHLD BUFADDR

PHYSWRITE20 CALL SENDITOUT
MVI B,12 ;12 MORE BYTES FOR FLOPPY

* IFT HARDDISK

* BIT 0,D
JRNZ PHYSWRITE22 ;JUMP IF FLOPPY
MVI B,3 ;3 MORE BYTES FOR HARD DISK
* ENDIF

PHYSWRITE22 CALL SENDITOUT
MOV A,E
STA CHKSUMW ;SAVE THE CHECKSUM
* IFT HARDDISK
* BIT 0,D
JRZ PHYSWRITE25 ;JUMP IF HARD DISK
* ENDIF

PHYSWRITE25 OUT DATAPORT ;AND SEND IT
XRA A
MVI B,4
PHYSWRITE27 OUT DATAPORT ;SEND 4 00's FOR ECC BYTES
DNZ PHYSWRITE27
MVI A,0AAH ;ECC VALID FLAG
OUT DATAPORT
XRA A
MVI B,80H ;128 ZEROS
PHYSWRITE30 OUT DATAPORT ;SEND POST-AMBLE OF ZEROS
DNZ PHYSWRITE30
JMP ACTIVATECNTLR ;'B' IS 00H (WRITEMASK)

SENDITOUT MOV A,M ;ADD IN NEXT BYTE
ADC E
MOV E,A
MOV A,M
OUT DATAPORT
INX H
DNZ SENDITOUT
RET

* PHYSREAD: PHYSICAL READ FROM SECTOR
* PHYSREAD XRA A
STA RAWERR ;NO ERRORS
LDA MOVEFLAG
MOV C,A
MVI B,READMASK
CALL ACTIVATECNTLR ;PERFORM THE READ
RNZ RESETPORT ;REPORT TIMEOUT ERROR
IN DATAPORT ;RESET CONTROLLER ERROR
IN DATAPORT
STA SYNCHOLD ;SAVE FOR LATER TEST
PHYSREAD02 XRA A ;CLEAR CARRY FLAG
MOV E,A ;AND CHKSUM BYTE
STA OZFLAG ;OPFLAG = 0 ON READS
LHLD BUFADDR ;POINT TO BUFFER
* IFT HARDDISK
* LDA HST.FLOP
MOV D,A
BIT 0,D
JNZ PHYSREAD20 ;JUMP IF FLOPPY
IN DATAPORT ;GET HEAD BYTE
STA HEADHOLD
BIT 0,C ;MOVE TO BUFFER?
JZ PHYSREAD10 ;JUMP IF NOT
MOV M,A

PHYSREAD10 ADC E
MOV E,A
INX H

* ENDI

* PHYSREAD20 IN DATAPORT ;GET TRACK BYTE
STA TRACKHOLD
BIT 0,C ;MOVE TO BUFFER?
JZ PHYSREAD30
MOV M,A

PHYSREAD30 ADC E
MOV E,A
INX H
IN DATAPORT ;GET SECTOR
STA SECTORHOLD

PHYSREAD37 BIT 0,C ;MOVE TO BUFFER?
JZ PHYSREAD40
MOV M,A

PHYSREAD40 ADC E
MOV E,A
INX H

* IFT HARD_DISK

* BIT 0,D
JZ PHYSREAD90 ;JUMP IF HARD_DISK

* ENDI

MVI B,10 ;GET FILLER

PHYSREAD70 IN DATAPORT
BIT 0,C ;SEND TO BUFFER?
JZ PHYSREAD80
MOV M,A

PHYSREAD80 ADC E
MOV E,A
INX H
DJNZ PHYSREAD70

PHYSREAD90 MVI B,0 ;SET DATA COUNTER (256 BYTES)

PHYSREAD100 IN DATAPORT
BIT 0,C ;SEND TO BUFFER?
JZ PHYSREAD110
MOV M,A
JMP PHYSREAD117

PHYSREAD110 PUSH PSW
PUSH B
MOV B,A
LDA RAWFLAG ; IS THIS A READ AFTER WRITE?
ORA A
JZ PHYSREAD115
MOV A,B
SUB M ; SAME DATA BYTES?
JZ PHYSREAD115
STA RAWERR

PHYSREAD115
POP B
POP PSW

PHYSREAD117
ADC E ; ADD TO CHKSUM
MOV E,A
INX H
DJNZ PHYSREAD100
STA CHKSUMR ; SAVE THE READ CHECKSUM

* IFT HARDDISK
*
BIT 0,D
JZ PHYSREAD120 ; JUMP IF HARD DISK

* ENDIF
*
IN DATAPORT ; ELSE GET FLOPPY CHKSUM
STA CHKSUMHOLD

PHYSREAD120
LXI H,ECCHOLD ; GET 4 ECC BYTES
MVI B,4

PHYSREAD130
IN DATAPORT ; GET THE ECC BYTES
MOV M,A
INX H
DJNZ PHYSREAD130

* IFT HARDDISK
*
BIT 0,D
JZ PHYSREAD140 ; JUMP IF HARD DISK

* ENDIF
*
IN DATAPORT
STA VALIDECC ; SAVE VALIDITY BYTE

PHYSREAD140
XRA A
RET

* CHKVALUES: VERIFY SYNC, HEAD, TRACK AND SECTOR
*
CHKVALUES
LDA SYNCHOLD
INR A
MVI A,56H ; SYNC BYTE ERROR
RNZ

CHKVALUES10
EQU $

* IFT HARDDISK
*
LDA HST,FLOP
ORA A
JRNZ CHKVALUES20 ; JUMP IF FLOPPY DISK
LXI H, HEADHOLD
LDA HEAD TO USE
SUB M
MVI A, 43H ; HEAD IN ERROR
RNZ
*

ENDIF
*

CHKVALUES20 LXI H, SECTORHOLD
LDA SECTOR TO USE
SUB M
MVI A, 42H ; SECTOR IN ERROR
RNZ

CHKVALUES30 CALL LDXRK
LDA TRACKHOLD
SUB M
RZ
MVI M, 0FFH ; SET TRACK TO UNKNOWN
MVI A, 41H ; TRACK IN ERROR
RET
*

ACTIVATECNTLR: ACTIVATE THE CONTROLLER AND WAIT UNTIL DONE
*

ACTIVATECNTLR CALL SLCT10 ; MAKE SURE WE'RE SELECTED
IN RESETPORT ; RESET CONTROLLER COUNTER
OUT STARTPORT ; START THE CONTROLLER
LXI D, 4000H ; SET A TIMEOUT COUNTER

ACTIVATE10 IN STATUS1
ANI BUSYMASK ; SEE IF DONE YET
RZ ; RETURN IF DONE
DCX D
MOV A, E
ORA D
JRNZ ACTIVATE10 ; KEEP TESTING UNTIL TIMEOUT
MVI A, 21H ; BUSY TIMEOUT ERROR
ORA A ; SET ERROR STATUS
RET
*

SLCT: SELECT AND READY THE DRIVE
*

SLCT MVI B, READMASK ; SET THE MASK FOR READS
SLCT10 LDA SECTOR TO USE ; GET THE SECTOR TO USE
ORA B ; MASK IN THE READ/WRITE FLAG
ORI ECCMASK ; ENABLE ECC LOGIC
*

IFT HARDDISK
*

MOV B, A
LDA TRACK TO USE
ANI TRK64MASK
MOV E, A ; SAVE IT TOO
JRZ SLCT12
SET 7, B
SLCT12 MOV A, B
*  
ENDIF
* OUT  CNTLPRT1
IN  STATUS1
ANI  MOTORMASK ;IS THE MOTOR ON ?
MOV  L,A  ;SAVE RESULTS OF TEST
LDA  SLCT_TO_USE

* IFT  HARDDISK

MOV  B,A
MOV  A,E
ANI  LOWCURMASK ;MUST USE LOWER CURRENT
ORA  B  ;ON WRITES PAST TRK 127

* ENDF

* OUT  CNTLPRT0  ;SEND UNIT AND HEAD INFO

* IFT  HARDDISK

LDA  HST.FLOP  ;IS THIS A FLOPPY ?
ORA  A
JRNZ  SLCT30  ;JUMP IF SO
LXI  D,0  ;SET A TIMER

SLCT14
IN  STATUS0
BIT  LOSSBIT,A
JRNZ  SLCT16
DCX  D
MOV  A,D
ORA  E
JRNZ  SLCT14
MVI  A,23H  ;LOSS OF SYNC TIMEOUT
JR  EREXIT2

SLCT16
BIT  WFLTBIT,A  ;WRITE FAULT DETECTED ?
JRZ  SLCT20
MVI  A,13H  ;WRITE FAULT ERROR
JR  EREXIT2

SLCT20
BIT  READYBIT,A  ;DRIVE READY ?
RNZ  EREXIT2
MVI  A,11H  ;DRIVE NOT READY ERROR
JR  EREXIT2

* ENDF

* SLCT30
LDA  DRIVE_TO_USE
ORA  A
JRNZ  SLCT35
IN  STATUS1
ANI  FSELMASK  ;FLOPPY ?
JRNZ  SLCT35
MVI  A,15H  ;NOT JUMPERED PROPERLY
EREXIT2
JMP  EXIT

SLCT35
LDA  LASTDRIVE
MOV  B,A
LDA  DRIVE_TO_USE
STA  LASTDRIVE
SUB B
JRZ SLCT40
MVI L,#0
SLCT40
MOV A,L
ORA A
CZ MOTORDELAY ;WAIT FOR MOTOR TO SPEED UP
RET

* SEEK: SEEK THE DESIRED TRACK *

SEEK
LDA OPFLAG
ORA A ;WAS THE LAST OPERATION A WRITE?
JRZ SEEK05 ;JUMP IF IT WAS READ
LXI D,#1 ;ELSE WAIT 1 MSEC TO MEET SPECS
CALL TIMER ;ON STEPPING AFTER A WRITE
SEEK05
CALL SLCT ;SELECT THE DRIVE
CALL LDTRK ;POINT TO DRIVE TRACK TABLE
MOV A,M ;SEE IF THE DRIVE
CPI $0FH ;HAS BEEN INITIALIZED
CZ RESTORE ;CALIBRATE IF NOT
SEEK10
LDA TRACK TO USE
MOV C,A
SUB M+ ;ALREADY AT TRACK?
RZ ;YES, SO WE'RE DONE HERE
JR JRC SEEKOUT
SEEKIN
CALL STEPIN ;STEP IN SO MANY TRACKS
DCR A
JRNZ SEEKIN
JR SEEK20
SEEKOUT
CALL STEP OUT ;STEP OUT SO MANY TRACKS
INR A
JRNZ SEEKOUT
SEEK20
CALL SETTLE ;WAIT HEAD SETTLE TIME
MOV M,C ;SAVE THE TRACK VALUE IN TABLE
RET

* STEPIN: STEP POSITIONER IN ONE TRACK *

STEPIN
MVI B,STPINMASK
JR STEPIT

*STEPOUT: STEP POSITIONER OUT ONE TRACK *

STEPOUT
MVI B,STPOUIMASK

STEPIT
PUSH PSW ;SAVE COUNTER
PUSH H ;SAVE TRACK TABLE POINTER
LDA SLCT TO USE ;GET HEAD AND UNIT INFO
ORA B ;MASK IN THE DIRECTION
OUT CNTLPORT0
SET STEPBIT,A ;SET STEP
OUT CNTLPORT0
NOP
RES STEPBIT,A ;RESET STEP
OUT CNTLPORT0
MOV A,B
STA DIRECTION ;SAVE DIRECTION OF MOVE
LXI H,STEPRATE ;GET DELAY CONSTANT
LDA DRIVE TO USE
MOV E,A
MVI D,0
DAD D
MOV A,M
ORA A
MVI E,3 ;3 MILLISECOND STEP
JR NZ STEPIT10
MVI E,30 ;30 MILLISECOND STEP
STEPIT10
CALL TIMER
POP H
POP PSW
RET

* * RESTORE: RECALIBRATE POSITIONER *
*
RESTORE MVI M,0FFH ;SET THE TRACK BYTE TO UNKNOWN
MVI A,4
RESTORE10 CALL STEPIN ;GO IN SOME TO BE SURE NOT ON 0
DCR A
JR NZ RESTORE10
MVI C,F,STEPMAX+10 ;MAXIMUM STEP COUNT
*
IPT HARDDISK
*
LDA HST.FLOP
ORA A
JR NZ RESTORE20
MVI C,W,STEPMAX+10 ;MAXIMUM STEP COUNT
*
ENDIF
*
RESTORE20 CALL STEPOUT
IN STATUS0
BIT TRK0BIT,A ;ARE WE THERE YET?
JR NZ RESTORE30 ;JUMP IF WE ARE
DCR C
JR NZ RESTORE20
MVI A,14H ;TRACK 0 NOT FOUND
JMP EXIT
RESTORE30 XRA A
MOV M,A
RET

* STEP OFF TRACK ONE, THEN BACK TO CORRECT POSSIBLE
* MARGINAL TRACK POSITION OF DRIVE WHICH WROTE THE
* DISK. IF TRACK 0, SUBSTITUTE RESTORE ROUTINE.
*
RESTEP CALL LDTTRK ;POINT TO CURRENT TRACK BYTE
MOV A,M
ORA A ;TRACK 0 ?
JR NZ RESTEP10
CALL RESTORE
JR SETTLE
RESTEP10
LDA DIRECTION ORA A JRZ RESTEP20 CALL STEPIN CALL STEPOUT JR SETTLE

RESTEP20 CALL STEPOUT CALL STEPIN JR SETTLE

* RESLCT: RETRY ROUTINE TO RESTORE TO 0, AND RESELECT

* RESLCT
LXI D,200 ;DROP OUT DELAY
CALL TIMER
CALL SLCT
CALL LDTRK
CALL RESTORE
JMP SEEK

* * MOTORDelay: WAIT 0.25 SECONDS FOR MOTOR TO REACH R/W SPEED

* MOTORDelay
LXI D,250
JR TIMER

* * SETTLE: WAIT HEAD SETTLE TIME

* SETTLE
LXI D,15

* IFT HARRDISK

* LDA HST.FLOP ORA A JRNZ TIMER ;JUMP IF FLOPPY
LXI D,4000H ;SET UP SEEK COUNTER
IN STATUS0
BIT SEEKBIT,A ;TEST SEEK COMPLETE BIT
RNZ
DCX D
MOV A,E ORA D JRNZ SETTLE10 ;CONTINUE WAITING
MVI A,22H ;SEEK COMPLETE TIMEOUT
JMP EXIT ;GO REPORT THE ERROR

* ENDIF

* * TIMER: DELAY NUMBER OF MILLISECONDS IN 'DE' REGISTER PAIR

* TIMER
PUSH B LDA SLCT TO USE ;RETRIGGER 4 SEC. TIMER
MOV B,A LDA DIRECTION ORA B OUT CNTLPORT0 MVI B,192
TIMER10
MOV A,B
 TIMER20
SUI 1
ORA A
JRNZ TIMER20
DCX D
MOV A,E
ORA D
JRNZ TIMER10
POP B
RET

* LDTRK: SET 'HL' POINTER TO CURRENT TRACK BYTE IN TRACK TABLE
*
LDTRK
MOV A,DDBUNT(X) ;GET THE UNIT IN QUESTION
ANI 0FH ;MASK IT OUT
*
IPT HARDDISK
IPT DUALLOG
*
CPI 2
JRN C LDTRK10
XRA A
*
ENDIF
ENDIF

* LDTRK10
PUSH D
MOV E,A
MVI D,0
LXI H,TRACKADDR
DAD D
POP D
RET

* SOFTErr
DB 0DH,0AH
DTZ 'SOFT ERROR--'

* THE FOLLOWING ROUTINES AND VARIABLES ARE NORMALLY INCLUDED
* IN THE LOGICAL DRIVER. THE LOGICAL DRIVER IS RESPONSIBLE
* FOR SETTING UP THE PHYSICAL DRIVER'S DCB.
*
* GETFLAGS: SET SINGLE FLAG (AND FLOPPSY IF HARD DISK SYSTEM)
* DRIVE IS IN 'A'
* RETURNS SINGLE IN 'B' (AND FLOPPSY IN 'C')
*
IFF HARDDISK
*
GETFLAGS
ANI 0FH
MOV B,A
ORA A
RZ ;DRIVE 0 ALWAYS DOUBLE SIDED
INR B
LDA DBL_SNG ;GET CONFIGURATION BYTE
GETFLAGS10
RRC
DJNZ GETFLAGS10
RNC ;B = 0 IF DOUBLE SIDED
DCR B ;B = FF IF SINGLE SIDED
RET

* ENDIF

* IFT HARDDISK

GETFLAGS ANI 0FH
MOV B,A

* IFT DUALLOG

* CPI 2

* ENDIF

* IFF DUALLOG

* CPI 1

* ENDIF

* MVI A,0
JRC GETFLAGS10
CMA
MOV C,A
RC

* IFT DUALLOG

* DCR B

* ENDIF

* GETFLAGS20
LDA DBL_SNG
RRC
DJNZ GETFLAGS20
RNC
DCR B
RET

* ENDIF

* BIOSPRINT
MOV A,M
ORA A
RZ
MOV C,A
CALL CONOUT
INX H
JMPR BIOSPRINT

* CONOUT
RET ;CONSOLE OUTPUT ROUTINE

* IFF HARDDISK

* NUMUNITS EQU 4
* ENDF

* IFD HARDDISK IFD DUALLOG

* NUMUNITS EQU 5

* ENDF

* IFF DUALLOG

* NUMUNITS EQU 4

* ENDF

* HST.SNG DB 0 ;SET IF HOST IS SINGLE SIDED
HST.FLOP DB 0 ;SET IF HOST IS A FLOPPY DRIVE
DBL SNG DB 0 ;CONFIGURATION BYTE
STEPRATE DB 0,0,0,0 ;STEPRATE TABLE FOR DRIVES

* *

TITLE FLOPPY/HARD DISK ECC ROUTINES

* COPYRIGHT (C) 1980,1981 VECTOR GRAPHIC INC.
* 31364 VIA COLINAS
* WESTLAKE VILLAGE, CA. 91361
*
* LAST REVISION DATE: 01/26/81
*
* THIS MODULE CONTAINS THE FOLLOWING ECC RELATED ROUTINES:
*
* ECCFIX - CORRECTION ROUTINE.
* ECCTEST - TESTS FOR GOOD ECC VALUE.
* CMPECC - COMPARES CURRENT ECC CODE WITH LAST ECC
* CODE THEN SETS THE LAST ECC CODE EQUAL
* TO THE CURRENT.
*
* HARD DISK FORMAT:
* 4 OVERHEAD BYTES: SYNC,HEAD,CYLINDER AND SECTOR
* 256 DATA BYTES
* 4 ECC BYTES
*
* FLOPPY DISK FORMAT:
* 3 OVERHEAD BYTES: SYNC,TRACK AND SECTOR
* 10 FILLER BYTES
* 256 DATA BYTES
* 1 CHECKSUM BYTE
* 4 ECC BYTES
*
HECCOFF EQU 4+256 ;HARD DISK ECC OFFSET
FECCOFF EQU 3+10+256+1 ;FLOPPY DISK ECC OFFSET
LDB.H EQU 3+256 ;LAST DATA BYTE (HARD DISK)
LDB.F EQU 2+10+256 ;LAST DATA BYTE (FLOPPY DISK)
HECCCON EQU HECCOFF*8+7 ;HARD DISK ECC CONSTANT
FECCCON EQU FECCOFF*8+7 ;FLOPPY DISK ECC CONSTANT

* ECCFIX: THIS ROUTINE CALCULATES AND CORRECTS DATA IF
* POSSIBLE. IF THE ERROR IS AN UNCORRECTABLE TYPE, THIS
* ROUTINE RETURNS WITH 54H IN REGISTER 'A'. IF THE ERROR
* WAS PROPERLY CORRECTED, THEN 00H IS RETURNED IN REGISTER
* 'A'. ALSO, THE 'Z' FLAG (IN PSW) WILL BE SET ACCORDING
* TO THE RETURNED STATUS OF REGISTER 'A'.
*
* ENTER WITH:
*
* NO REQUIREMENTS
*
* EXITS WITH:
*
* AF - ERROR STATUS
* REGISTERS USED: HL,DE,BC
*
* EXTERNAL VARIABLES:
*
* HST.FLOP - USED TO DETERMINE IF THE
* DRIVE IS A FLOPPY OR HARD DISK.
*
* BUFADDR - HOLDS POINTER TO CURRENT
* DISK I/O BUFFER.
*
* SYNHOLD - HOLDS CURRENT SYNC BYTE
* HEADHOLD - HOLDS CURRENT HEAD BYTE
* TRACKHOLD - HOLDS CURRENT TRACK BYTE
* SECTORHOLD - HOLDS CURRENT SECTOR BYTE
* CHKSUMHOLD - HOLDS CURRENT CHECKSUM
* ECCHOLD - HOLDS CURRENT ECC DATA.
* MOVEFLAG - BUFFER DATA VALID ?
*
* CORRECTION ROUTINE. SEE PAGE 12 OF REPORT BY NEAL GLOVER.
*
* INITIALIZE PSEUDO SHIFT REGISTERS AND SHIFT COUNT (J).
*
ECCFIX
LXI H,ECCHOLD ;SET HL POINTER TO ECC BYTES
MOV B,M ;SYNDROME BITS X0-X7
INX
MOV C,M ;SYNDROME BITS X8-X15
INX
MOV D,M ;SYNDROME BITS X16-X23
INX
MOV E,M ;SYNDROME BITS X24-X31
IFT   HARDDISK

*  
LDA   HST.FLOP   ;GET FLOPPY TEST STATUS
ORA   A
LXI   H,HECCCON  ;LOAD J WITH CONSTANT K1
JRZ   CALGN      ;JUMPS IF HARD DISK

*  
ENDIF

*  
LXI   H,FECCCON  ;LOAD J WITH CONSTANT K1

*  CLEAR ALGN-FLAG

*  
CALGN  XRA  A    ;CLEAR A
STA    ALGNFLG  ;CLEAR ALGN-FLAG

*  LEFT JUSTIFY FIRST NON-ZERO SYNDROME BYTE IN 'B'

*  
JUST  ORA  B
JRNZ  SHIFT
MOV   A,L
ADI   8
MOV   L,A
JRNC  JUST10
INR   H

JUST10  MOV   B,C
MOV   C,D
MOV   D,E
MVI   E,0
JR    JUST

*  SHIFT PSEUDO SHIFT REG UNTIL CORRECTABLE PATTERN FOUND

*  
SHIFT  SRLR  B    ;SHIFT
PARR  C
PARR  D
PARR  E
JRNC  SHIFT10  ;BRANCH IF NO BIT SHIFTED
MOV   A,E
XRI   0COH
MOV   E,A
MOV   A,D
XRI   64H
MOV   D,A
MOV   A,C
XRI   82H
MOV   C,A
XRI   64H
MOV   D,A
MOV   A,C
XRI   80H
MOV   B,A
MOV   A,B
ORA   A
JRZ   PTRNINST
SHIFT20  XRA  A
ORA   L
JRNZ  SHIFT30
ORA  H
JZ  UNCRR

SHIFT30
DCX  H
JR  SHIFT

;UNCORRECTABLE

DEC DECREMENT SHIFT COUNT ('J')

* TEST FOR CORRECTABLE PATTERN *

PTRNTST
LDA  ALGNFLG
ORA  A
JRNZ  PTRNTST5
MOV  A,D
ADD  E
JRNZ  SHIFT20
MOV  A,C
ANI  7
JRNZ  SHIFT20
MVI  A,1
STA  ALGNFLG

;LOAD ALGN-FLAG
;BRANCH IF ALGN-FLAG NON ZERO
;BRANCH IF CORRECTABLE
;PATTERN NOT YET FOUND
;BRANCH SAME AS ABOVE
;SET ALGN-FLAG NON-ZERO

;TEST 'J' MODULO 8
;JMP IF BYTE ALIGN NOT COMPLETE

* CORRECT BYTES IN ERROR *

CORRECT
MOV  B,C
MOV  C,D
MVI  A,3

;MOVE PATTERN

CORR10
SRLR  H
RARR  L
DCR  A
JRNZ  CORR10
XCHG

;DIVIDE BIT BISPLACEMENT BY 8
;TO GET BYTE DISPLACEMENT

* IFT  HARDDISK *

LDA  HST.FL0P
ORA  A
JRNZ  CORR30
MOV  A,D
ORA  A
JRNZ  CORR20
MOV  A,E
ORA  A
JRZ  CORR90
CPI  1
JRZ  CORR110
CPI  2
JRZ  CORR70
CPI  3
JRZ  CORR80

;JUMP IF FLOPPY
;ERROR UP HIGH
;SYNC/HEAD ERROR
;HEAD/TRACK ERROR
;TRACK/SECTOR ERROR
;SECTOR/DATA ERROR

CORR20
LXI  H,LDB,H
ORA  A
DSBC  D
JRZ  CORR105
JRC  CORR50
XRA  A

;LAST DATA BYTE ERROR
;DATA FIELD ERROR
* RET
  ENDF
*

CORR30
  MOV A, D
  ORA A
  JRNZ CORR40 ; ERROR UP HIGH
  MOV A, E
  ORA A
  JRZ CORR90 ; SYNC/TRACK ERROR
  CPI 1
  JRZ CORR70 ; TRACK/SECTOR ERROR
  CPI 2
  JRZ CORR80 ; SECTOR/DATA ERROR

CORR40
  LXI H, LDB.F
  ORA A
  DSBC D
  JRZ CORR100 ; LAST DATA BYTE ERROR
  JRNC CORR50 ; DATA FIELD ERROR
  XRA A
  RET

CORR50
  LDA MOVEFLAG
  ORA A
  RZ
  LHLD BUFADDR
  DCX H
  DAD D

CORR60
  MOV A, M
  XRA B
  MOV M, A
  INX H

CORR65
  MOV A, M
  XRA C
  MOV M, A
  XRA A
  RET

CORR70
  LXI H, TRACKHOLD ; TRACK/SECTOR ERROR
  CALL CORR60
  JR CORR50

CORR80
  LXI H, SECTORHOLD ; SECTOR/DATA ERROR
  MOV A, M
  XRA B
  MOV M, A
  JR CORR50

CORR90
  LXI H, SYNCHOLD ; SYNC/HEAD-TRACK ERROR
  CALL CORR60
  LDA MOVEFLAG
  ORA A
  RZ
  LHLD BUFADDR
  JR CORR65

CORR100
  LXI H, CHKSUMHOLD ; LAST DATA BYTE ERROR
  MOV A, M
  XRA C
  MOV M, A

CORR105
  LDA MOVEFLAG
ORA A
RZ
LHLD BUFADDR
DCX H
DAD D
MOV A,M
XRA B
MOV M,A
XRA A
RET

* IFT HARD_DISK
*
CORR110 LXI H,HEAD_HOLD ; HEAD/TRACK ERROR
CALL CORR60
JR CORR50
*
ENDIF
*
* UNCORRECTABLE ERROR EXIT
*
UNCORR MVI A,54H ; SET ERROR RETURN BYTE
ECCEEXIT ORA A ; SET '2' FLAG FOR ERROR CODE
RET
*
ALGNFLG DS 1
*
* EXCTEST: THIS ROUTINE TESTS THE ECC BYTES AFTER A READ OPERATION
* TO SEE IF THEY ARE ALL ZERO. IT RETURNS WITH REGISTER 'A' EQUAL
* TO 00H AND '2' FLAG SET IF ALL THE ECC BYTES WERE ZERO. OTHERWISE,
* IT RETURNS A 52H IN REGISTER 'A' AND THE '2' FLAG RESET.
*
* ENTER WITH:
* *
* EXITS WITH:
* *
* AF - ERROR STATUS
* REGISTERS USED: HL, B
*
* EXTERNAL VARIABLES:
* *
* ECCHOLD - HOLDS CURRENT ECC DATA.
*
ECCTEST LXI H,ECCHOLD
MVI B,4 ; SET ECC BYTE COUNTER
XRA A
ECCTEST10 ORA M ; TEST A BYTE
INX H
JRNZ ECCTEST20 ; JUMP IF NOT ZERO
DJNZ ECCTEST10 ; ELSE CONTINUE TESTING
JR ECCTEST30
ECCTEST20 MVI A,53H ; REPORT THE ERROR
ECCTEST30 ORA A
RET

* CMPECC: THIS ROUTINE COMPARES THE LAST ECC CODE TO THE CURRENT
* ECC CODE TO DETERMINE IF A CONSISTENT ERROR HAS OCCURRED. IT ALSO
* SETS THE LAST ECC CODE EQUAL TO THE CURRENT ECC CODE BEFORE EXITING.
* REGISTER 'A' WILL BE 00H AND THE 'Z' FLAG WILL BE SET IF THE TWO
* CODES WERE THE SAME. OTHERWISE, REGISTER 'A' WILL BE 01H AND THE
* 'Z' FLAG WILL BE RESET.
* * ENTER WITH:
* * NO REQUIREMENTS
* * EXITS WITH:
* * AF - ERROR STATUS
* * REGISTERS USED: HL, DE, BC
* * EXTERNAL VARIABLES:
* * ECCHOLD - HOLDS CURRENT ECC DATA.
* * LASTECC - FOUR BYTE VARIABLE THAT
* * CONTAINS THE PREVIOUS ECC BYTES FOR
* * A PARTICULAR READ OPERATION.
* * CMPECC
LXI  H, ECCHOLD       ;SET POINTER TO ECC BYTES
MVI  C, 0             ;CLEAR 'BAD COMPARE' FLAG
LXI  D, LASTECC       ;SET POINTER TO LASTECC BYTES
MVI  B, 4             ;SET A COUNTER
CMPECC10
LDAX  D                ;GET A LASTECC BYTE
CMP   M                ;COMPARE IT TO AN ECC BYTE
JRZ   CMPECC20         ;JUMP IF GOOD COMPARE
MVI   C, 1             ;SET 'BAD COMPARE' FLAG
CMPECC20
MOV   A, B,M           ;SET LASTECC = THIS ECC
STAX  D                ;STEP THE POINTERS
INX H
INX D
DJNZ  CMPECC10         ;DO ALL FOUR BYTES
MOV   A, C             ;PUT ERROR CODE INTO 'A'
ORA  A
;SET RETURN STATUS
RET

* * * * *
* PHYSICAL DRIVER EQUATES AND STORAGE
* * *
* COMMAND CODES
SEEKCOM  EQU  0
READCOM  EQU  1
WRITCOM  EQU  2
VERCOM   EQU  3
INITCOM  EQU  4
WPDCOM   EQU  5
* DDB OFFSETS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDBCOM</td>
<td>EQU 0</td>
</tr>
<tr>
<td>DDBNUM</td>
<td>EQU 1</td>
</tr>
<tr>
<td>DDBSEC</td>
<td>EQU 2</td>
</tr>
<tr>
<td>DDBTRKL</td>
<td>EQU 3</td>
</tr>
<tr>
<td>DDBTRKH</td>
<td>EQU 4</td>
</tr>
<tr>
<td>DCBBUFL</td>
<td>EQU 5</td>
</tr>
<tr>
<td>DCBBUFH</td>
<td>EQU 6</td>
</tr>
</tbody>
</table>

* RETRY OFFSETS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETRY0</td>
<td>EQU 0</td>
</tr>
<tr>
<td>RETRY1</td>
<td>EQU 1</td>
</tr>
<tr>
<td>L2RTRY</td>
<td>EQU 2</td>
</tr>
<tr>
<td>L3RTRY</td>
<td>EQU 3</td>
</tr>
</tbody>
</table>

* BUFFER OFFSETS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARDDATA</td>
<td>EQU 0004H</td>
</tr>
<tr>
<td>FLOPDATA</td>
<td>EQU 000DH</td>
</tr>
<tr>
<td>FLOPCS</td>
<td>EQU 010DH</td>
</tr>
</tbody>
</table>

* OPTION BIT NUMBERS (OPTCONF)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRLBIT</td>
<td>EQU 0</td>
</tr>
<tr>
<td>DIRHBIT</td>
<td>EQU 1</td>
</tr>
<tr>
<td>NOTUSED</td>
<td>EQU 2</td>
</tr>
<tr>
<td>WPDBIT</td>
<td>EQU 5</td>
</tr>
<tr>
<td>CMPBIT</td>
<td>EQU 6</td>
</tr>
<tr>
<td>HCIBIT</td>
<td>EQU 7</td>
</tr>
</tbody>
</table>

* CONTROLLER BIT NUMBERS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPROBIT</td>
<td>EQU 0</td>
</tr>
<tr>
<td>FSELBIT</td>
<td>EQU 0</td>
</tr>
<tr>
<td>READYBIT</td>
<td>EQU 1</td>
</tr>
<tr>
<td>BUSYBIT</td>
<td>EQU 1</td>
</tr>
<tr>
<td>TRKOBIT</td>
<td>EQU 2</td>
</tr>
<tr>
<td>MTBIBIT</td>
<td>EQU 2</td>
</tr>
<tr>
<td>WPLIBIT</td>
<td>EQU 3</td>
</tr>
<tr>
<td>SEEKBIT</td>
<td>EQU 4</td>
</tr>
<tr>
<td>STEPBIT</td>
<td>EQU 5</td>
</tr>
<tr>
<td>LOSSBIT</td>
<td>EQU 5</td>
</tr>
<tr>
<td>R WB1T</td>
<td>EQU 5</td>
</tr>
<tr>
<td>DIRBIT</td>
<td>EQU 6</td>
</tr>
<tr>
<td>ECCBIT</td>
<td>EQU 6</td>
</tr>
<tr>
<td>LCURBIT</td>
<td>EQU 7</td>
</tr>
</tbody>
</table>

* CONTROLLER BIT MASKS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEAD0</td>
<td>EQU 00H</td>
</tr>
<tr>
<td>HEAD1</td>
<td>EQU 04H</td>
</tr>
<tr>
<td>WRITEMASK</td>
<td>EQU 00H</td>
</tr>
<tr>
<td>READMASK</td>
<td>EQU 20H</td>
</tr>
<tr>
<td>STPOUTMASK</td>
<td>EQU 00H</td>
</tr>
<tr>
<td>STPINMASK</td>
<td>EQU 40H</td>
</tr>
<tr>
<td>FSELMAK</td>
<td>EQU 01H</td>
</tr>
<tr>
<td>WPROMASK</td>
<td>EQU 01H</td>
</tr>
<tr>
<td>BUSYMASK</td>
<td>EQU 02H</td>
</tr>
<tr>
<td>MOTORMASK</td>
<td>EQU 04H</td>
</tr>
</tbody>
</table>
HDISKMASK EQU 08H
ECCMASK EQU 40H
LOWCURMASK EQU 80H
TRK64MASK EQU 0COH

* CONTROLLER PORT NAMES
CNTLPORT0 EQU 0COH
CNTLPORT1 EQU 0C1H
STATUS0 EQU 0COH
STATUS1 EQU 0C1H
DATAPORT EQU 0C2H
STARTPORT EQU 0C3H
RESETPORT EQU 0C3H

* F.TRKMAX EQU 154 ;NUMBER OF TRACKS
F.STEPMAX EQU 77
W.STEPMAX EQU 153
TRACKADDR EQU OFF54H ;LOCATION OF TRACK TABLE
BUSY EQU OFF10H
HPREAMBLE EQU 30

* SPECIAL TIMESHARE EQUATES
  IPT TIMESHARE

  GETCONNUM EQU 0E00FH
  SWAP EQU 0E018H
  PROTECTED EQU OFF0FH

  ENDFI

* STORAGE

  IFF HARDDISK

  ENTRystack DW 0
  BUFADDR DW 0
  RETRY DB 0,0,0,0
  TEMPER DB 0
  MOVEFLAG DB 0
  CHKSUMR DB 0
  CHKSUMW DB 0
  DRIVE TO USE DB 0
  TRACK TO USE DB 0
  SECTOR TO USE DB 0
  SLCT TO USE DB 0
  LASTDRIVE DB 0
  LASTBECC DB 0,0,0,0
  VALIDECC DB 0
  SYNCHOLD DB 0
  TRACKHOLD DB 0
  SECTORHOLD DB 0
  CHKSUMHOLD DB 0
  ECCHOLD DB 0,0,0,0
  DIRECTION DB 0
  OPFLAG DB 0
RAWFLAG    DB  0
RAWERR     DB  0

*         
ENDIF

*         
IFT      HDDISK

*         
ENTRYSTACK DW  0
BUFAADDR  DW  0
RETRY     DB  0,0,0,0
TEMPERR   DB  0
MOVEFLAG  DB  0
CHKSUMR   DB  0
CHKSUMW   DB  0
DRIVE TO USE DB  0
HEAD TO USE DB  0
TRACK TO USE DB  0
SECTOR TO USE DB  0
SLCT TO USE DB  0
LASTDRIVE DB  0
LASTECC   DB  0,0,0,0
VALIDECC  DB  0
SYNCHOLD  DB  0
HEADHOLD  DB  0
TRACKHOLD DB  0
SECTORHOLD DB  0
CHKSUMHOLD DB  0
ECHOLD    DB  0,0,0,0
DIRECTION DB  0
OPFLAG    DB  0
RAWFLAG   DB  0
RAWERR    DB  0

*         
INFOIBL   EQU $ ;DISK INFORMATION TABLE

*         
IFT      DUALLOG

*         
FS.DISK   DB  1 ;HEADS PER LOGICAL UNIT
           DB  77 ;TRACKS PER HEAD
FD.DISK   DB  2 ;HEADS PER LOGICAL UNIT
           DB  77 ;TRACKS PER HEAD
H5.DISK   DB  2 ;HEADS PER LOGICAL UNIT
           DB  153 ;TRACKS PER HEAD
H10.DISK  DB  3 ;HEADS PER LOGICAL UNIT
           DB  153 ;TRACKS PER HEAD

*         
ENDIF

*         
IFF      DUALLOG

*         
FS.DISK   DB  1 ;HEADS PER LOGICAL UNIT
           DB  77 ;HEADS PER TRACK
FD.DISK   DB  2 ;HEADS PER LOGICAL UNIT
           DB  77 ;TRACKS PER HEAD
H5.DISK   DB  4 ;HEADS PER LOGICAL UNIT
           DB  153 ;TRACKS PER HEAD
H10.Disk  DB  6  ; HEADS PER LOGICAL UNIT
           DB  153 ; TRACKS PER HEAD
*    ENDIF
    ENDIF
*