HARDWARE DIAGNOSTICS

for UNIFLEX T.M.

CDSBUG DIAGNOSTIC

for UNIFLEX T.M.

Important Note

Most of the enclosed diagnostics will work only with the MP-09 processor board and NOT with the SWTPC MPU-1 processor board. This is due to the lack of an allocate physical resource system call in the UniFlex operating system. If this system call is added to UniFlex in the future, all diagnostics will be modified to work with the MPU-1 board.

Diagnostic: CDSBUG

CDSBUG is a diagnostic tool designed to assist qualified technical personnel in finding and rectifying malfunctions in the SWTPC CDS Marksman disk units.

IMPORTANT NOTE

Improper use of the diagnostic program "CDSBUG" will result in loss of information contained on the CDS disk units. Southwest Technical Products Corporation specifically disclaims any responsibility or liability for any such damages incurred or generated by the "CDSBUG" diagnostic program. This program is not sold or intended for distribution to persons unfamiliar with the CDS units or the operation of diagnostic tools. CDSBUG remains the sole property of Southwest Technical Products and may not be reproduced or distributed without prior written permission.

Disclaimer

CDSBUG - MARKSMAN DIAGNOSTIC PROGRAM

Table of Contents

Table of Contents

INSTALLATION PROCEDURE

NEEDED: The distribution diskette for the CDSBUG diagnostic. This diskette is in UniFlex format, single sided, single density, and contains the following files:

cdsbug	-- CDSBUG Diagnostic Program
boots/cds	-- Boot Block for CDS Disks
mkfs	-- Make File System Program

NOTE: The diagnostic program "CDSBUG" takes over the entire UniFlex operating system. In order to terminate the "CDSBUG" program the system must be RESET. This implies that the "CDSBUG" program <u>must</u> <u>not</u> be run on a system in multi-user mode.

To install the CDSBUG diagnostic program, mount the disk containing the program "CDSBUG", and copy the CDSBUG program onto a disk with a bootable UniFlex system. It is suggested that the CDSBUG diagnostic be placed in the "/etc" directory, and that it belong to the super-user. A possible procedure for installing the CDSBUG diagnostic is shown below:

```
++ /etc/mount /dev/fd1 /usr2        -- mount the CDSBUG diskette
++ copy /usr2/cdsbug /etc           -- copy CDSBUG into "/etc"
++ copy /usr2/mkfs /etc             -- copy MKFS into "/etc"
++ owner system /etc/cdsbug         -- give ownership to "system"
++ perms o-rwx /etc/cdsbug          -- turn off non-owner perms
++ crdir /etc/boots                 -- create boot directory
++ copy /usr2/boots/* /etc/boots    -- copy boots into "/etc"
++ owner bin /etc/boots/*           -- give ownership to "bin"
```

RUNNING THE CDSBUG DIAGNOSTIC

To run the CDSBUG diagnostic program, first be certain no other users are logged into the system. Then use the "/etc/shutup" program to take the UniFlex system into Single User mode. The run the CDSBUG program by typing "/etc/cdsbug" and return:

```
    ++ /etc/cdsbug                    -- run the CDSBUG diagnostic
```

The CDSBUG program will take over the UniFlex system and the screen should look like this:

```
              -- SWTPC CDS DIAGNOSTIC - MP-09 VERSION 3.1:2 --

COMMAND:|_|           BLOCK: 00000  ADDRESS: 00000  STATUS: 00 0000
         |                                                   |  | |
         |                                                   |  | |
         |                                                   |  | |
         +-- Cursor                  Controller Status --------+  | |
                                                                 | |
                                     Diagnostic Status -----------+ |
                                                                    |
                                          Drive Status -------------+
```

"COMMAND" is followed by the cursor, indicating that the program is waiting for the next command string to be entered. Commands consists of strings of one to nineteen characters, and are terminated by the return key. Command strings are interpreted by the CDSBUG diagnostic from left to right, one character at a time. Several commands may be placed in the same command string (up to 19 characters). Spaces imbedded in the command string are ignored.

"BLOCK" is the current block number. The current block is set by several commands, and may be incremented or decremented by using the "+" or "-" commands. Commands that operate upon disk media typically use the current block number for their manipulations.

"ADDRESS" indicates the hexadecimal address of the buffer memory. The CDSBUG program uses a single 8K buffer for all transactions with the CDS disk controller. Note that the value shown is a 20-bit physical address.

"STATUS" shows the status of the CDS unit. Two sets of hexadecimal numbers are shown. The first number is the 8-bit CDS CONTROLLER status. The second number is the 16-bit combined DIAGNOSTIC and DRIVE status. The primary controller status byte provides information about the execution and completion of controller commands, and in general, is used by programs for handshake and error detection. The diagnostic and drive status bytes are used by diagnostic programs to determine the cause and extent any problems with the CDS Disk Unit.

The CONTROLLER status is an 8-bit value returned to the S/09 computer from the CDS Disk Unit Controller. Status is presented whenever a command terminates, an invalid command is issued, the controller goes busy during drive sequencing, and upon detection of any drive error condition. The individual bits are explained in detail in the section on the SWTPC Controller and summarized below:

```
                                    _______________________________
                                   |_7_|_6_|_5_|_4_|_3_|_2_|_1_|_0_|
                                     |   |   |   |   |   |   |   |
    BIT NUMBER                       |   |   |   |   |   |   |   |
                                     |   |   |   |   |   |   |   |
    7 -- DISK ON LINE AND READY -----+   |   |   |   |   |   |   |
    6 -- COMMAND END --------------------+   |   |   |   |   |   |
    5 -- CONTROLLER BUSY ---------------------+   |   |   |   |   |
    4 -- DISK WRITE PROTECTED --------------------+   |   |   |   |
    3 -- CRC ERROR -----------------------------------+   |   |   |
    2 -- NO RECORD FOUND ---------------------------------+   |   |
    1 -- SEEK INCOMPLETE -------------------------------------+   |
    0 -- COMMAND ERROR -------------------------------------------+
```

The diagnostic status byte returns a value dependant upon the particular command being executed by the CDS controller. For non-diagnostic commands, this byte is the command that resulted in status being presented. Diagnostic commands use this byte to return data to the diagnostic program. Diagnostic status bytes are documented in the detailed command descriptions at the end of this manual. Non-diagnostic command status is summarized below:

00	No Operation	60	Format Write
10	Restore	70	Sector Write
20	UnVerified Seek	80	Swap Out to Disk
30	Verified Seek	90	Swap In from Disk
50	Sector Read	A0	Set Virtual Volume

Diagnostic commands return status dependant upon their function. For those commands that do not return a value from the controller, the status consists of two four-bit digits. The first is the digit "F", specifying the diagnose command, and the second four-bit digit is the diagnostic sub-function number. Other diagnostic commands return an eight-bit value which is dependant upon the parameterization and execution of the command. The following table details returning status for those diagnose commands that return command and subfunction numbers:

F2	Format Track	FB	Write Configuration
F4	Read Sector Header	FC	Read Scratchpad
F5	Read Bad Sector	FD	Read Control ROM
F6	Write Bad Sector	FE	Disable Mapping
F9	Set Sector Spacing	FF	Marksman Reset

The DRIVE status is an eight-bit value presented to the SWTPC CDS Controller. Five of these bits come directly from the CDS Marksman drive interface, two come from front panel buttons, and the last bit is used as a Marksman RESET control line. A detailed description of these signals can be found in the section on the SWTPC Controller. These bits are summarized below:

```
                                            _______________________________
                                           | 7 | 6 | X | 4 | 3 | 2 | 1 | 0 |
                                             |   |   |   |   |   |   |   |
BIT NUMBER                                   |   |   |   |   |   |   |   |
                                             |   |   |   |   |   |   |   |
7 -- DRIVE SELECT ---------------------------+   |   |   |   |   |   |   |
6 -- WRITE ENABLE -------------------------------+   |   |   |   |   |   |
5 -- DRIVE CONTROLLER RESET -------------------------+   |   |   |   |   |
4 -- CRC CHECK ------------------------------------------+   |   |   |   |
3 -- WRITE UNSAFE -------------------------------------------+   |   |   |
2 -- STATUS PRESENTED -------------------------------------------+   |   |
1 -- COMMAND READY --------------------------------------------------+   |
0 -- DRIVE ONLINE -------------------------------------------------------+
```

CDSBUG COMMAND INTRODUCTION

The CDSBUG diagnostic provides access to the SWTPC CDS Marksman Unit at command level. With some notable exceptions, each CDSBUG command letter will cause a single command to be issued to the CDS Disk Unit and display of returning status. This section is an introduction to those commands, and details the use of the diagnostic for certain useful cases.

Since the level of access to the CDS Disk Unit is on a physical block level, it is probable that careless manipulation of the data on the media will result in the complete destruction of any UniFlex\ file system and the data that it contains. It is strongly recommended that all data be backed up off the CDS unit <u>prior</u> to running the CDSBUG program. A possible exception to this recommendation is the use of this diagnostic to clear a hard read error. This particular case will be the first example.

Example 1: Clearing a Read Error

Hard read errors on the disk media are rare, but can be caused by sudden catastrophic power failures, static discharges, or inadvertent operation of the write protect button on the front panel. The errors typically make themselves apparent while running one of the TSC file system diagnostic programs (such as "devcheck"). Hard read errors can make the file system unusable by UniFlex*, particularly if they occur on the first few cylinders of the disk.

If a read error is suspected, make sure that everyone is logged off the UniFlex* system and then take the system down. Once the system is in single user mode, you can bring up the CDSBUG diagnostic (See the section on Running CDSBUG). To locate the read error, type in the following command group:

```
    COMMAND: =0 [ S*? R? +1 ]              -- locating a read error
```

Commands in this command string are separated by spaces. Remember that the CDSBUG command interpreter ignores spaces. First, the "=0" command is used to set the current block to zero. The "[" signals the start of a repeat group. The string "S*?" causes a SEEK ("S") to the current block ("*"), followed by a test for errors ("?"). Once the seek has completed, the string "R?" reads the seeked block and checks for errors. If no errors are found, the string "+1" increments the current block number (by one), and when the "]" end of repeat group is encountered, the command sequence repeats. The net effect of this command group is to seek to block zero, and subsequently read the entire disk.

Since there are no terminating conditions on this command group, the CDSBUG program will continue to read blocks until an error is detected. If there are no read errors on the disk, the terminating error will occur when the Seek command issues a seek to a block that is past the end of the disk. In this case, the returning status will show

SEEK INCOMPLETE and COMMAND ERROR. The status display will look something like this:

```
    COMMAND:          BLOCK: 0884B  ADDRESS: 72000  STATUS: C3 30D3
```

The block number, "884B" (In Hex, remember), is one larger than the largest block accessable by the drive. In this case, this number is for an M-20. The address, "72000" (Also in hex) is the 20-bit physical address of the CDSBUG buffer area. Most informative are the status bytes. As shown by the controller status on page 4, the "C3" byte indicates disk online, command end, seek incomplete, and command error. This is the definitive indication of an invalid seek. Note that the diagnostic status byte is "30" -- indicating that the failing command was a verified seek. Finally, the drive status byte of "D3" indicates drive seleted, write enabled, command ready, and drive online.

If a hard error is located, the command group will terminate when it reaches the block containing the error. Since the read used is a non-diagnostic read, no data will be transfered from the CDS Disk unit to the computer. In particular, remember that the data in the CDSBUG buffer is <u>not</u> the data that was read from the erroneous block. The CDSBUG status display will look like this:

```
    COMMAND:          BLOCK: 0261B  ADDRESS: 72000  STATUS: C8 50D3
```

The block number, "0261B" is the block containing the error. The controller status of "C8" indicates disk online, command end, and CRC error. Note that the diagnostic status byte is "50" indicating a read command. At this point, we should at least look to see if any of the data in the block can be recovered. The following command group should be entered to attempt to read the failing block and display its data:

```
    COMMAND: S*? G<  BLOCK: 0261B  ADDRESS: 72000  STATUS: C8 F5D3

  65 6D 20 66 6F 72 20 61 63 63 65 73 73 20 74 6F    em for access to
  74 3F FF FF FF FF FF FF FF FF C0 6E 64 20 43 65    t?........@nd Ce
  6E 74 72 6F 6E 69 78 20 70 72 69 6E 74 65 72 73    ntronix printers
  2E 20 20 54 68 69 73 20 69 73 20 64 6F 6E 65 20    .  This is done
  74 68 72 6F 75 67 68 20 61 20 66 61 63 69 6C 69    through a facili
  74 79 20 63 61 6C 6C 65 64 0D 22 50 72 69 6E 74    ty called."Print
  65 72 20 53 70 6F 6F 6C 69 6E 67 22 2E 0D 49 6E    er Spooling"..In
  20 61 64 64 69 74 69 6F 6E 2C 20 74 65 78 74 20     addition, text
  66 69 6C 65 73 20 6D 61 79 20 62 65 20 74 72 61    files may be tra
  6E 73 66 65 72 72 65 64 20 62 65 74 77 65 65 6E    nsferred between
  20 74 68 65 20 46 6C 65 78 20 61 6E 64 20 55 6E     the Flex and Un
  69 46 6C 65 78 20 73 79 73 74 65 6D 0D 74 68 72    iFlex system.thr
  6F 75 67 68 20 75 73 65 20 6F 66 20 74 68 65 20    ough use of the
  70 72 6F 67 72 61 6D 73 20 22 74 6F 75 6E 69 22    programs "touni"
  20 61 6E 64 20 22 66 72 6F 6D 75 6E 69 22 2E 0D     and "fromuni"..
  41 73 20 61 20 74 65 72 6D 69 6E 61 6C 20 65 6D    As a terminal em
```

As you can see by examining the above block of data, 10 bytes of data starting at location 011 have been corrupted (In this case, a data dropout). This particular error was probably caused by a static discharge while this block was being written. You should use the CDSBUG "M" (Modify) command to repair the faulty data. If you don't know what the data should be, just fill the bytes with blanks. Once the data has been patched, writing the block back to the disk media will clear the read error. Note that in this case, the bad data was simply blanked out:

```
     COMMAND: S*? W   BLOCK: 0261B  ADDRESS: 72000  STATUS: C0 70D5

  65 6D 20 66 6F 72 20 61 63 63 65 73 73 20 74 6F    em for access to
  20 20 20 20 20 20 20 20 20 20 20 20 20 20 43 65                  Ce
  6E 74 72 6F 6E 69 78 20 70 72 69 6E 74 65 72 73    ntronix printers
  2E 20 20 54 68 69 73 20 69 73 20 64 6F 6E 65 20    .  This is done
  74 68 72 6F 75 67 68 20 61 20 66 61 63 69 6C 69    through a facili
  74 79 20 63 61 6C 6C 65 64 0D 22 50 72 69 6E 74    ty called."Print
  65 72 20 53 70 6F 6F 6C 69 6E 67 22 2E 0D 49 6E    er Spooling"..In
  20 61 64 64 69 74 69 6F 6E 2C 20 74 65 78 74 20     addition, text
  66 69 6C 65 73 20 6D 61 79 20 62 65 20 74 72 61    files may be tra
  6E 73 66 65 72 72 65 64 20 62 65 74 77 65 65 6E    nsferred between
  20 74 68 65 20 46 6C 65 78 20 61 6E 64 20 55 6E     the Flex and Un
  69 46 6C 65 78 20 73 79 73 74 65 6D 0D 74 68 72    iFlex system.thr
  6F 75 67 68 20 75 73 65 20 6F 66 20 74 68 65 20    ough use of the
  70 72 6F 67 72 61 6D 73 20 22 74 6F 75 6E 69 22    programs "touni"
  20 61 6E 64 20 22 66 72 6F 6D 75 6E 69 22 2E 0D    and "fromuni"..
  41 73 20 61 20 74 65 72 6D 69 6E 61 6C 20 65 6D    As a terminal em
```

Because of the size and density of the data held on the CDS media, it is not unusual for a drive to have a few media defects. These defects result in hard data errors that cannot be cleared. The SWTPC Controller maintains a media defect map on a configuration track, and the controller firmware insures that no data is stored on tracks with media defects. Century Data Systems guarantees the following maximum defect rate on their drives:

1. No defects on cylinder zero, head zero.

2. No more than:

6 defective tracks	on M-10
12 defective tracks	on M-20
24 defective tracks	on M-40
36 defective tracks	on M-80
48 defective tracks	on M-160

This data is based upon twelve passes over the entire disk surface using a random data pattern. Century Data Systems provides a written statement with each drive, detailing the locations of known defective tracks. When the CDS drives are burned in at the SWTPC plant, this statement is used to initialize the configuration track for each disk. If this information is lost, or if a new persistant media error appears, the CDSBUG diagnostic provides the ability to replace the configuration information.

Example 2: Entering Configuration Information

The SWTPC Controller reserves one media track to contain configuration information about the CDS drive. This track is located on the last cylinder and head on the drive, and is written into every sector on that track. When the drive is sequenced up with the select button, this information is read and used to initialize several internal tables in the controller. The configuration block contains a checksum (besides the sector CRC) to validate the data. The controller need only find one good configuration block out of the 40 or so sectors. During normal operation, programs are not permitted to read or write the configuration blocks.

If the Marksman has never been configured (configuration track is erased), you must use the CDSBUG "^" command to manually sequence up the drive. (Typically, this operation has been performed at SWTPC and need not be repeated). Since the controller always attempts to read the configuration information whenever the drive is sequenced up via the Select button, a totally unformatted drive may cause the controller to "lock up". You should power up the drive with the Select button in the non-select position. If the drive sequences up on its own, turn the power off to the drive (which will spin it down), the depress the Select button once, then power the drive back up and reset the system. Reset the system, boot UniFlex*, and run the CDSBUG program. Then use the following command sequence to bring up the drive:

```
COMMAND: N ^                    -- Sequence Up the Drive
```

Once the drive has been spun up, you must locate the start of the configuration blocks in order to place format information on them. This format information is required in order to write the configuration blocks. The following command sequences will set the current block number to that of the first configuration block:

```
COMMAND: =0 [ P*? +29 ]        -- M-10, M-20, and M-40 drives

COMMAND: =0 [ P*? +2A ]        -- M-80 drives

COMMAND: =0 [ P*? +38 ]        -- M-160 drives

COMMAND: =0 [ P*? +1 ]         -- All drives, but is slow
```

Note that the increment value is different for the various Marksman drives. By using an increment value of one, the command sequence will work correctly on all types of Marksman, but it is quite a bit slower than by using a track increment. This command sequence will terminate with a seek incomplete indication. Any other error condition indicates that the CDS disk unit is malfunctioning.

n40 = 110BF   end = 110E8

When the block number has been set to the start of the configuration track, the following command sequence should be used to write track format information on the disk media:

```
     COMMAND: 0 Q [ P*? F? +1 ]     -- Write blank format info
```

In order to write the configuration information on a CDS drive, you must run the CDSBUG program and use the "K" command. This command will obtain certain information from the CDS control ROM, and then prompt you for configuration parameters:

```
     COMMAND: K       BLOCK: 00000  ADDRESS: 72000  STATUS: C2 FBD3

   Data Sector Spacing:  11     ---- Defective Tracks (Cyl-Hd) -----
                                0066-3  0067-3  0068-3 0069-3 0070-3
   Swap Sector Spacing:   5     0071-3  0201-1

        Swap Cylinders:  12

    Defective Cylinder: 201
           Head Number:   1

       Virtual Volumes:   3
                               ---- Virtual Volume Blocks Table ----
    Volume 3 - Start: 07801    00000-11096  00000-05000  05001-07800
                 End: 11096    07801-11096

     Total Available: 11096
```

The data and swap sector spacings, as shown above, are the block skew value used to properly match the disk sector spacing with the read capabilities of UniFlex*. These values were obtained empirically by averaging an assortment of job mixes for each given sector spacing value. For any given installation, the system manager may find it advantageous to "tune" these values for his specific needs. For example, those sites running mainly basic programs may want to increase the data spacing value. The following table gives suggested values for several system configurations:

	1.0 MHz	1.5 MHz	2.0 Mhz/3509	2.0 MHz
Data:	19	15	14	11
Swap:	7	6	6	5

The number of swap cylinders defines the boundary at which the SWTPC controller changes sector spacings. When a disk is formatted, the amount of swap space assigned should match the number of cylinders assigned for swapping by the Marksman configurator. If for some reason, it is deemed desirable to use a single spacing throughout the CDS disk, the number of swap cylinders may be set to zero. The swap spacing feature is not mandatory; it is simply used to improve performance. Remember that assigning more swap space than is actually used in a system does not improve performance.

Systems with small main memories (128K) will require more swap space than systems with more main memory. A good first approximation is to supply 96K of swap space for each attached terminal. The following table lists the number of swap cylinders versus the amount of swap space for several drive configurations:

CDS	Cyls-Swap	K	Cyls-Swap	K	Cyls-Swap	K	Cyls-Swap	K
M-10	12	492K	16	656K	20	820K	28	1148K
M-20	8	656K	12	984K	16	1312K	20	1640K
M-40	6	984K	8	1312K	10	1640K	12	1968K
M-80	8	1008K	10	1260K	12	1512K	16	2016K
M-160	6	1008K	8	1344K	10	1680K	12	2016K

Defective tracks are assigned alternates from the logical end of the media. The configurator maintains a "first available alternate" track address, which is initially set the the track immediately prior to the configuration track. As defective tracks are entered, the configurator generates an entry in the alternate track map, and decrements the available alternate track address. In order to map out defective tracks, you should enter the cylinder number (in decimal), followed by the proper head number. When all defective tracks have been assigned, entering a carriage return will cause the configurator to continue.

Version 4 of the SWTPC Controller supports virtual volumes on the Marksman drive. Up to 16 virtual volumes may be defined (Volume zero is always the physical volume). Enter the number of virtual volumes desired, or just carriage return to disable virtual volume support. If selected, you must set the ending block number for each virtual volume. This block number is in hexadecimal. The starting block number is automatically set to be one more than the ending block number of the previous virtual volume. When all virtual volumes have been defined, the configurator will write the configuration blocks to the Marksman. Note that Virtual Volume Zero and One both always start at block zero. If volume one contains a UniFlex* file system, then volume zero (if mounted) will contain the identical file system, even though the uppermost block address of volume zero is higher that that of volume one.

If alternate tracks have been assigned, it will be necessary to re-format the volume. The formathd program may be used if formatting must proceed on a working UniFlex* system, otherwise it is much faster to use the CDSBUG program to create the initial format. The third example deals with formatting a configured Marksman.

Example 3: Formatting a Marksman

The SWTPC Controller provides for formatting a Marksman one track at a time. Since no provision is made for issuing diagnostic commands from UniFlex* (a good idea), the CDSBUG program must be used to format the Marksman. (A format operation may be performed from UniFlex*, but it will proceed one block at a time.) The Marksman drive should be configured (see previous example) before formatting.

First, you must issue a manual sequence-up command whether or not the drive is spinning and ready. This command sets a flag that indicates the drive is in dianostic mode, and this flag is checked by the track format routine. This check is to prevent inadvertant loss of an entire track of data. If the drive has just been configured (see previous example), it is not necessary to re-issue the sequence up command. The following command sequence will force a manual spinup and insure that the configuration block has been correctly read:

```
COMMAND: ^ J?    BLOCK: 00000  ADDRESS: 72000  STATUS: CO FAD3
```

Pay particular attention to the status bytes. If the returning status is not "CO", then the configuration blocks are unreadable and the drive should not be formatted. In this case, reconfigure the drive as per the previous example. Formatting the drive without proper configuration can cause hard seek errors later on when the drive is in use.

Once the drive has been sequenced up and the configuration blocks read, the media can be formatted. First issue a command sequence to clear the buffer and set the block number to zero:

```
COMMAND: 0 =0    BLOCK: 00000  ADDRESS: 72000  STATUS: CO FAD3
```

When this is accomplished, use one of the following command sequences to completely format the drive:

```
COMMAND: [ P*? T? +29 ]        -- M-10, M-20, and M-40 drives

COMMAND: [ P*? T? +2A ]        -- M-80 drives

COMMAND: [ P*? T? +38 ]        -- M-160 drives

COMMAND: [ P*? F? +1 ]         -- All drives, but VERY slow
```

These command sequences should terminate with a seek incomplete indication. Any other error indications in the status byte mean that the drive is not formattable. Be sure the write protect is off. Remember that this formatting process <u>does not</u> place a UniFlex* file structure on the disk; it merely prepares the media to receive data. A drive does not need to be reformatted unless it is reconfigured or unclearable data errors manifest themselves.

DETAILED DESCRIPTION OF COMMANDS

Commands to CDSBUG are specified as single letters, sometimes followed by a value designator. They are normally used as primitives in a more complicated command sequence used to test some feature of the CDS controller. A command sequence consists of a string of command letters, terminated by a carriage return. The CDSBUG program will perform each command in the sequence in the order in which they were entered. Note that by the use of the begin repeat ("[") and end repeat ("]") commands, complex loops may be generated. The repeat commands do not nest.

Command interpretation continues until either a carriage return is encountered, an end repeat character is encountered, or until the ABORT flag is set. A carriage return will simply terminate the command sequence and the CDSBUG program will wait for more keyboard input. An end repeat character will cause the interpreter to back up the the last begin repeat character. If no begin repeat character is found in the buffer, the ABORT flag is set. Several commands set the ABORT flag, see the detailed command descriptions. When the ABORT flag is set, a bell is sounded at the terminal, the current command sequence is aborted, and the interpreter waits for keyboard input. The following paragraphs detail the CDSBUG commands.

COMMAND 0:

The "0" (zero) command clears the buffer to zeros. All 8K of the buffer area is cleared. No I/O commands are sent to the CDS unit.

COMMAND 1:

The "1" (one) command fills the buffer with all ones (Hex FF). All 8K of the buffer area is set to ones. No I/O commands are sent to the CDS unit.

COMMAND 2:

The "2" (two) command fills the buffer with an address-variable pattern. All 8K of the buffer area is set to this pattern. No I/O commands are sent to the CDS unit.

COMMAND 7:

The "7" (seven) command compares the pattern in the buffer to the pattern that would have been generated by the "2" command. All 8K of the buffer area participates in the compare. No I/O commands are sent to the CDS unit. If the pattern does not match, the ABORT flag is set and the ADDRESS field will show the first byte at which the pattern did NOT match.

COMMAND A:

The "A" command performs a surface analysis on the CDS unit. Alternate track mapping is disabled and the entire surface of the disk unit is analyzed. A summary report of all defects located is generated at the end of the test. All data previously present on the drive is destroyed. The results of the surface analysis may be used to initialize the Disk Configuration Records. (See the Configuration command, "K").

COMMAND B:

-- reserved --

COMMAND C<n>:

The "C" command sends a status request command through the SWTPC controller to the Marksman controller. Status byte requests range from 0 to 15 (Decimal) and are dependant upon the particular Marksman drive. The status bytes are documented in detail in the Marksman Performance Specification, Century Data Systems document #76220-906. Returned status is passed back to the CDSBUG program as the diagnostic status byte.

COMMAND D<n>:

The "D" command command is used to select a virtual volume. If the specified volume is out of range, or if virtual volumes are not supported, the command returns Command Error. The diagnostic status byte is set to "A0" (Hex).

COMMAND E:

The "E" command is used to cause the SWTPC controller to read a sector header. No CRC indications are produced and no no error status returned. The diagnostic status byte is set to "F4" (Hex).

COMMAND F:

The "F" command performs a format write on the currently selected block. A "P" command must have been used to select the block. If the disk is write protected, if there is no seek complete, if a "P" command was not used to set the sector, or if the format write simply fails, then the ABORT flag is set. The diagnostic status byte is set to "60" (Hex).

COMMAND G:

The "G" command reads a block (possibly with bad CRC) and transfers the contents of the sector buffer to memory. No retries are attempted, and the sector buffer is always transfered to the computer's memory. This command may be used to attempt to recover data from disk blocks with bad CRC's. The diagnostic status byte is set to "F5" (Hex).

COMMAND H:

The "H" command is the "CDSBUG" help command. A menu of available CDSBUG command characters is displayed on the screen. No I/O is performed and the contents of the buffer are not changed.

COMMAND I:

The "I" command reads the SWTPC controller scratchpad into the CDSBUG buffer. For the interpretation of the scratchpad contents, you should refer to the UDISK PROGRAM LISTING. Remember that in order to purchase this information, you will be required to sign a binding non-disclosure agreement. The program listing is not included with the CDSBUG diagnostic. The diagnostic status byte is set to "FC" (Hex).

COMMAND J:

The "J" command is used to read a configuration block into the CDSBUG buffer. In addition, the contents of the block are interpreted and shown on the CDSBUG information frame. Reading the configuration block contents causes the SWTPC controller to reconfigure itself according to the data contained in the block. The diagnostic status byte is set to "FA" (Hex).

COMMAND K:

The "K" command is used to invoke the CDSBUG configurator. The user is prompted for several configuration parameters and a configuration block is constructed. This block is then written into all of the blocks on the last track of the Marksman. The diagnostic status byte is normally set to "FB" (Hex), but if for some reason not all of the blocks can be written, the block number of the failing write is returned.

COMMAND L<n>:

The "L" command is used to perform a data loopback test. The low-order eight bits of the value <n> is sent to the SWTPC controller and returned in the diagnostic status byte. This command may be used to test the controller tranceivers.

COMMAND M:

The "M" command enables the user to modify the contents of the CDSBUG buffer. The ADDRESS field shows the currently selected byte in the buffer. The contents of this byte may be modified by entering appropriate hexadecimal data characters. A carriage return will terminate the "M" command. The "M" command normally starts at the top of the buffer, and uses the four "arrow" keys to move the current byte (indicated by the cursor) through the buffer.

COMMAND N:

The "N" command sends a NO-OP sequence to the CDS controller. This command may be used to see if the computer can pass a valid command to the CDS and have status returned. The diagnostic status byte is set to "00" (Hex).

COMMAND O:

The "O" command writes the contents of the first 512 bytes of the buffer to the currently selected sector on the CDS disk unit. The written sector is tagged with a bad CRC. The "O" command is used to verify proper operation of the CRC error detection circuitry. The diagnostic status byte is set to "F6" (Hex).

COMMAND P<n>:

The "P" command is used to perform an unverified seek. The command letter is followed by a value designator (which can be either a hexadecimal number or an asterisk, indicating the current BLOCK value) and this is used as the block number. This value is also made the current BLOCK value. Certain commands must be <u>immediately</u> preceeded by an unverified seek. The diagnostic status byte is set to "20" (Hex).

COMMAND Q:

The "Q" command is used to disable track mapping and enable access to the configuration blocks. A one-to-one mapping is established between logical blocks and physical sectors. The diagnostic status byte is set to "FE" (Hex).

COMMAND R:

The "R" command reads the currently selected block into the first 512 bytes of the CDSBUG buffer. Note that the current BLOCK becomes the currently selected sector only upon the issuance of a Seek command. No indication is given of possible soft read errors. The diagnostic status byte is set to "50" (Hex).

COMMAND S<n>:

The "S" command is used to perform a verified seek. The command letter is followed by a value designator (which can be either a hexadecimal number or an asterisk, indicating the current BLOCK value) and this is used as the block number. This value is also made the current BLOCK value. The controller reads the first header block on the positioned track to verify head position. If the position verifies incorrect, up to 27 automatic restores and retries may be performed before the seek finally fails. No indication is given of possible soft seek errors. The diagnostic status byte is set to "30" (Hex).

COMMAND T:

The "T" command is used to format a track. This command must have been immediately preceeded by an unverified seek (Command "P"), the specified seek address must be a multiple of a track, and the diagnostic sequence flag must have been set in the controller. The sequence flag is normally set via a manual sequence up command, "^". Zeros are written into every sector on the track. The diagnostic status byte is set to "F2" (Hex).

COMMAND U:

The "U" command issues an invalid command to the CDS controller. The "U" command is typically used to set the error conditions in the status byte, or to test the error detection code in the SWTPC controller. The diagnostic status byte is set to "F0" (Hex).

COMMAND V:

The "V" command displays the version and revision information of the current CDS control ROM. The control ROM is read into the CDSBUG buffer and the checksum is calculated. Note that the ROM data in the buffer may be examined by use of the memory examine function "M". The diagnostic status byte is set to "40" (Hex). The following information is displayed by the "V" command:

```
        ROM Checksum is: FACE
        ROM is Version 4.0:1 (2 MHz)
        Released August 1, 1982
        Released 06/07/82
     41 sectors per track
      8 tracks per cylinder
    213 cylinders
     11 Data Sector Spacing
      5 Swap Sector Spacing
     24 Alternate Track Maps
     16 Virtual Drives
```

COMMAND W:

The "W" command writes the first 512 bytes of the buffer into the currently selected block. Note that the current BLOCK becomes the currently selected sector only upon the issuance of a Seek command. The diagnostic status byte is set to "70" (Hex).

COMMAND X:

The "X" command swaps the contents of the CDSBUG buffer out to disk at the currently selected block. All 8K of the buffer is transfered to disk unless the swap is terminated prematurely with an error condition. Note that the current BLOCK becomes the currently selected sector only upon the issuance of a Seek command, and that the swap commands CHANGE the currently selected sector in the controller. The diagnostic status is set to "80" (Hex).

COMMAND Y:

The "Y" command swaps data from the currently selected block into the CDSBUG buffer. The entire 8K buffer is transferred from disk unless the swap is terminated prematurely due to an error condition. Note that the current BLOCK becomes the currently selected sector only upon the issuance of a Seek command, and that the swap commands CHANGE the currently selected sector in the controller. The diagnostic status is set to "90" (Hex).

COMMAND Z:

The "Z" command issues a restore command to the disk. The disk goes offline, and the heads are stepped outward until a TRK000 indication is received, then the disk is brought back online with cylinder and track set to zero. This command sets Seek Incomplete. The diagnostic status is set to "10" (Hex).

COMMAND [:

The "[" command denotes the beginning of a repeat group for the CDSBUG command interpreter. No I/O commands are issued to the disk and the contents of the buffer are not changed.

COMMAND]:

The "]" command denotes the end of a repeat group for the CDSBUG command interpreter. No I/O commands are issued to the disk and the contents of the buffer are not changed. The command line is scanned from right to left to locate the matching begin repeat character, and the command interpreter continues processing commands from that point. Repeated commands may be terminated by typing carriage return on the terminal.

COMMAND .:

The "." (period) command displays the current controller and disk status. No I/O commands are issued to the disk and the contents of the buffer are not changed. The block number, address, controller, and disk status displays are updated.

COMMAND ,:

The "," (comma) command provides a 1-second delay in command sequence interpretation. This command is useful if dynamically displayed status bytes must be observed. No commands are sent to the CDS unit.

COMMAND ?:

The "?" command updates the current status display. No I/O commands are issued to the disk and the contents of the buffer are not changed. The block number, address, controller, and disk status displays are updated. The Controller status byte is tested for errors, and if any are found, the ABORT flag is set. The "?" command is rather simple minded about what it considers errors.

COMMAND <:

The "<" command displays the upper 256 bytes of the buffer, or, if buffer display is already in progress, the previous (low addresses) 256 bytes in the buffer. No I/O commands are issued to the disk and the contents of the buffer are not changed.

COMMAND >:

The ">" command displays the lower 256 bytes of the buffer, or, if buffer display is already in progress, the next (higher addresses) 256 bytes in the buffer. No I/O commands are issued to the disk and the contents of the buffer are not changed.

COMMAND +<n>:

The "+" (plus) command is used to increment the current block number. If the plus sign is followed by a hexadecimal number, then the current block number is incremented by that value. value. If no number is specified, the current block number is incremented by one. No I/O commands are issued to the disk and the contents of the buffer are not changed. Note that this command does NOT change the currently selected sector.

COMMAND -<n>:

The "-" (minus) command is used to decrement the current block number. If the minus sign is followed by a hexadecimal number, then the current block number is decremented by that value. value. If no number is specified, the current block number is decremented by one. No I/O commands are issued to the disk and the contents of the buffer are not changed. Note that this command does NOT change the currently selected sector.

COMMAND =<n>:

The "=" (equals) command is used to set the current block number. No I/O commands are issued to the disk and the contents of the buffer are not changed.

COMMAND ^:

The "^" command is used to manually sequence up the Marksman disk. Once the manual sequence command has been issued, the Marksman unit no longer obeys the Select button on the front panel. This command is used to get a virgin disk ready to be formatted. The normal sequence up process issues a read for the configuration information. If the disk has never been formatted, the controller may "lock up" by attempting to read an entirely blank track. Therefore, a new disk is brought up via the "~" command and formatted. The diagnostic status byte is set to "F7" (Hex).

COMMAND ~:

The "~" command is used to manually sequence down the Marksman disk. Once the manual sequence command has been issued, the Marksman unit no longer obeys the Select button on the front panel. The diagnostic status byte is set to "F7" (Hex).

COMMAND %:

The "%" (percent) command will reset the Nevada board (the board on top of the CDS disk drive that looks like the state of Nevada). The Nevada board can lock up after certain invalid command sequences. In this case, the "%" command can be used to free it. The diagnostic status byte is set to "FF" (Hex).

COMMAND $:

The "$" (dollar) command is used to perform a software reset on the SWTPC controller. All diagnostic indications are cleared and normal power-up sequence processing is performed. No status is returned from the software reset command.

COMMAND #<n>:

The "#" (hash) command sends a diagnostic command request through the SWTPC controller to the Marksman controller. Diagnostic requests range from 0 to 15 (Decimal) and are dependant upon the particular Marksman drive. The diagnostics are documented in detail in the Marksman Performance Specification, Century Data Systems document #76220-906. Returned status is passed back to the CDSBUG program as the diagnostic status byte. Note that not all Marksman controllers have diagnostic capability.

COMMAND @<n>:

The "@" (at sign) command is used to set the sector spacing value used for data sectors. This value may be used for timing determinations. Note that this command does not alter the sector spacing value in the configuration blocks. The diagnostic status is set to "F9" (Hex).

SUMMARY OF CDSBUG COMMANDS

"0"	Fill all 8K of buffer with zeros.
"1"	Fill all 8K of buffer with ones.
"2"	Fill all 8K of buffer with data pattern.
"7"	Compare all 8K of buffer to data pattern.
"A"	Perform detailed surface analysis and report.
"B"	(Unused at present)
"C<n>"	Perform Marksman Status request.
"D<n>"	Select a virtual volume.
"E"	Read a sector header.
"F"	Perform format write on block.
"G"	Read a block containing an error.
"H"	Get Help with CDSBUG.
"I"	Read controller scratchpad into buffer.
"J"	Read configuration information.
"K"	Configure the CDS disk unit.
"L<n>"	Perform a data loopback test.
"M"	Modify the buffer memory.
"N"	Send a NOP to the controller.
"O"	Write a block with a CRC error.
"P<n>"	Perform an unverified seek to block <n>.
"Q"	Disable alternate track mapping.
"R"	Read the currently seeked block.
"S<n>"	Perform a verified seek to block <n>.
"T"	Format a track.

Summary of CDSBUG Commands (Cont.)

"U"	Issue an invalid controller command.
"V"	Get Firmware version information.
"W"	Write the currently seeked block.
"X"	Swap buffer out to disk.
"Y"	Swap buffer in from disk.
"Z"	Restore the disk.
"["	Begin a repeat group.
"]"	End a repeat group.
". . . . . . ." . . .	Display current status information.
","	Provide a one-second delay.
"?"	Test for errors.
"<"	Display upper or previous buffer.
">"	Display lower or next buffer.
"+<n>"	Increment the current block number.
"-<n>"	Decrement the current block number.
"=<n>"	Set the current block number.
"^"	Sequence the Marksman UP.
"~"	Sequence the Marksman DOWN.
"%"	Reset the Nevada Board.
"$"	Reset the CDS Controller.
"#<n>"	Perform Marksman diagnostic <n>.
"@<n>"	Set data sector spacing to <n>.

USING MKFS TO CREATE A FILE SYSTEM

Once a CDS disk is formatted, it is suitable for containing a UniFlex* file system. However, certain tables and structures must be placed on the disk in order for UniFlex* use it. The MKFS (Make File System) program writes these structures onto a formatted but otherwise empty disk. A UniFlex* file system has the following components:

1. A Boot Block (Optional).
2. A Super Block.
3. File Descriptor Nodes (FDN's).
4. A Root Directory.
5. A Free Space List.
6. Swap Space (Optional).
7. User Directories and Files (Optional).

If the file system is to be bootable, then both swap space and a boot block are required. The boot block is the first block on the device (block zero), and contains the initial program load (IPL) for UniFlex. Remember that these IPL programs are different for various configurations of the UniFlex* system. In particular, the IPL programs supplied with Version 1 and Version 2 UniFlex* are not compatable. Boot blocks are normally contained in directory "/etc/boots". The following table details the boot block files and the configuration in which they are used:

Boot Name	System Configuration
cds	CDS Unit with MP-09 Processor
dmf2	DMF-2 Floppy with MP-09 Processor
dmf3	DMF-3 Floppy with MP-09 Processor
win1	WIN-1 5" Winchester with MP-09

In order to run the mkfs program, the CDS disk must have been configured and formatted. This process is normally performed at SWTPC before the disk units are shipped, but in the event that format or configuration is lost, refer to Example 2 and Example 3 in the previous section. Once the disk is formatted, boot UniFlex (from floppy) and run mkfs:

```
++ mkfs /dev/hd0 +b=cds +k=<swap size>
```

where <swap size> is the size in Kilobytes you want to allocate for swapping. If the configuration reserves space for swapping, the size of the swap space should match. Note that these values do not have to match; they are used only to improve performance. Also remember that assigning more swap space than is actually used in a system does not improve performance. A good rule of thumb is to assign 96K of swap space for each attached terminal.

Once the CDS disk has a file system, you can install a bootable copy of UniFlex*. In order to do this, you should mount the disk as directory "/usr2" and use the "/etc/crdisk" script provided with UniFlex*. Once this is done, you must install the copy of UniFlex* that uses the CDS disk as root and swap device. To do this, use the following sequence of UniFlex* commands:

```
++ /etc/mount /dev/hd0 /usr2          -- mount hard disk
++ /etc/crdisk                        -- use script to move stuff
++ copy /uniflex1 /usr2/uniflex       -- copy hard disk root
++ /etc/install /usr2/uniflex         -- install the UniFlex*
++ /etc/unmount /dev/hd0              -- unmount the hard disk
```

The CDS disk unit should now be bootable via the UniBug "H" command. Do not use any "putboot" program earlier than Version 2, as the internal bootstrap has been changed. The new boot blocks all reside in the directory "/etc/boots".

CDS MARKSMAN AND SWTPC CONTROLLER

The CDS Marksman is a fixed-media Winchester technogology disk drive with one or two 14-inch media platters and two movable heads per surface. For the M10, M-20, and M-40, there are 213 cylinders with two, four, or eight tracks per cylinder respectively. For the M-80 and M-160, there are six heads per cylinder, with 569 and 837 cylinders, respectively. The movable heads are controlled by the Marksman basic control board, a Motorola 6802 based control unit. This controller is responsible for sector and index pulse timing, and for the positioning of the heads over the media. Sector timing on the M-10, M-20, and M-40 is accomplished with an optical timing wheel attached to the lower disk spindle, while the M-80 and M-160 use timing information embedded in a servo surface on the disk media itself.

Attached to the CDS Marksman via a short parallel control cable is the SWTPC Marksman Controller. The SWTPC controller arbitrates the data paths between the host computer and the Marksman media. It is responsible for all data reads and writes. The SWTPC Controller to Host Computer interface contains two data paths. The first is a low-speed data path known as the letterbox. It is this path that is used to pass commands from the host to the controller, and to pass status from the controller back to the host. This data path appears to be three eight-bit registers and a two-bit flag register. Conventionally, these registers are called the command and address registers, and the tag-interrupt register.

When the host computer reads the letterbox, it appears to be four memory locations. In S/09 and UniFlex systems, these locations decode at bus addresses $FF100 to $FF103. Because of incomplete decoding, these four address locations may be repeated up to address $FF1FF, however, there are only four registers. The low order register is called the primary status byte, and returns the SWTPC controller status. The next two registers are the secondary status registers, and contain the returning command and drive status, respectively. The last register is two bits long, and indicates IRQ pending, and Controller Read Pending, respectively. The following figure depicts the letterbox registers as the S/09 would read them:

```
  $FF100      $FF101      $FF102      $FF103   <----   S/09 Bus Address

   8 bits    / - - - 16 bits - - - \  2 bits
+----------+----------+----------+---+-------+        CDS Letterbox as
|          |          |          |   |xxxxxx |        READ by the S/09
+----------+----------+----------+---+-------+
 \        / \                   / \ /
  \      /   \                 /   \________     Tag-Interrupt Register
   \    /     \               /                    (2 High-order bits)
    \  /       \             /
     \/         \___________/______________      Detail Status (16 bits)
      \
       \___________________________________      Primary Status Register
                                                        (8 bits)
```

When the host computer writes a disk command to the letterbox, four writes are required. The first three writes constitute a 24-bit command and address word, structured with the first four bits being the primary command specification, followed by 20 bits of address or data. The 20-bit data field is dependant upon the primary command to be issued. The fourth write is to the tag-interrupt register, to indicate that the controller should process a command. Note that the tag register <u>must</u> be written <u>after</u> the 24-bits of command and address have been written. The data used to write the tag register is ignored. In addition, it is an error to write the data or tag registers until bit 6 of the tag register is zero, indicating that no interrupt is pending in the controller.

Once the tag register has been written, bit 6 will enter the one state until the controller initiates service on the command request. When the controller completes service of the command, bit 7 of the tag-interrupt register will be brought to the one state, indicating that the controller is presenting an interrupt to the host computer. When an interrupt is presented to the host by the controller, the host must read <u>at least</u> the primary status register and the tag-interrupt register. Reading the tag register after reading the primary status register will cause the IRQ request in the host to be cleared. (Note that if the IRQ request is not cleared, and the IRQ Jumper on the MP-HD board is in the IRQ position, the host computer may find itself locked into an interrupt request service routine.)

Once status is read, the host must then check the primary status byte to be sure command end has been presented. If the status information presented is not ending status (i.e., the Command End bit is not set), the controller is continuing to service the command (although exception bits may have already been set in the status bytes) and another interrupt will be presented upon command end. No additional commands may be issued to the controller until ending status is presented.

The following figure depicts the letterbox registers as the S/09 would write them:

```
  $FF100       $FF101       $FF102       $FF103    <----    S/09 Bus Address

4 bits / - - - -  20 bits  - - - - \  2 bits
+-----+-----+-----------+-----------+---+--------+       CDS Letterbox as
|     |     |           |           |   | xxxxxx |        WRITTEN by S/09
+-----+-----+-----------+-----------+---+--------+
 \   / \                           / \ /
  \ /   \                         /   \__________ Tag-Interrupt Register
   \     \                       /                  (2 High-order bits)
    \     \                     /
     \     \___________________/_____________ Command Argument (20 bits)
      \
       \_____________________________________ Primary Command Specifier
                                                (4 high order bits)
```

The following example program shows one method (non-interrupt) of sending a command to the CDS unit and returning status. This routine is entered with the A register containing the disk command word with the low-order four bits zero. The X register contains the logical address of the buffer. The routine XLATE translates a logical address in the X register to a 20-bit physical address in the A and X registers. Note how the upper four bits of the address is combined with the four bit disk command. This code was taken from a diagnostic program used to test CDS buffer transfers:

```
                              *
                              .   Disk routine to send command to CDS
                              .
                              .   Enter with A = Disk Command Byte
                              .              X = Buffer Address Word
                              .    Exit with B = Primary Status
                              .
                              dodisk:  proc

 F100                         D_CMD    equ     $F100     Command Register
 F101                         D_ADR    equ     $F101     Address Register
 F102                         D_TAG    equ     $F102     Tag Register
 0040                         T_CRP    equ     $40       Controller Read Flag
 0080                         T_CSP    equ     $80       Status Presented Flag
 0040                         S_END    equ     $40       Command End Flag
                              *
                              .   get 24-bit command w/ physical address
                              .
 29FC 34 12                            pshs    a,x       save command
 29FE AD 9F FF B3                      jsr     [XLATE]   get physical address
 2A02 AA E0                            ora     0,s+      or in command byte
                              *
                              .   wait for controller to read flag
                              .
 2A04 F6 F1 02                crwait   ldb     D_TAG     read tag register
 2A07 C5 40                            bitb    0_CRP    see if read pending
 2A09 26 F9                            bne     crwait    loop until done
 2A0B B7 F1 00                         sta     D_CMD     store the command
 2A0E BF F1 01                         stx     D_ADR     store the address
 2A11 B7 F1 02                         sta     D_TAG     then tag controller
                              *
                              .   wait for controller to present status
                              .
 2A14 F6 F1 02                spwait   ldb     D_TAG     read tag register
 2A17 C5 80                            bitb    0_CSP    see if status there
 2A19 27 F9                            beq     spwait    loop if not
 2A1B F6 F1 02                cewait   ldb     D_TAG     -- This read REQUIRED
 2A1E C5 40                            bitb    0_END    see if command end
 2A20 27 F9                            beq     cewait    loop if not
 2A22 35 92                            ret     a,x

                                       end     dodisk
```

Primary Controller Status Byte

The primary status byte is an 8-bit value returned to the computer from the CDS Contoller. Status is presented whenever a command terminates, an invalid command is issued, the controller goes busy during sequencing, and upon detection of any drive error condition. It is the primary status byte that is used by UniFlex* block device drivers to perform all operations upon the CDS units. The following diagram details the primary status byte:

```
                                        ________________________________________
         Bus Read at $FF100             | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
                                          |   |   |   |   |   |   |   |
Bit: 7 -- Disk Online and Ready ----+   |   |   |   |   |   |   |
     6 -- Command End ------------------+   |   |   |   |   |   |
     5 -- Controller Busy -------------------+   |   |   |   |   |
     4 -- Write Protected -----------------------+   |   |   |   |
     3 -- CRC Error ---------------------------------+   |   |   |
     2 -- No Record Found -------------------------------+   |   |
     1 -- Seek Incomplete -----------------------------------+   |
     0 -- Command Error -----------------------------------------+
```

Disk Online and Ready -- The MARKSMAN disk unit has been sequenced up and has presented READY status to the controller. This indicates that the drive is up to speed, the controller has been configured (or has determined that it cannot be configured), and no write unsafe condition exists. It is not a write unsafe condition if the media is write protected.

Command End -- The last command passed to the CDS controller has been completed and primary status has been presented to the computer. This bit is made zero whenever a command is accepted by the controller, and returns to a one whenever the controller finishes interpreting the command and presents status.

Controller Busy -- A command was issued to the CDS controller and no completion status has been presented. If both command end and controller busy are presented, a command was incorrectly issued to the controller before the previous command had terminated.

Write Protected -- A media write command was issued to the controller and could not be completed because the disk was write protected. This is an error condition and does NOT reflect the sense of the write protect button on the front panel.

CRC Error -- A media read command was issued to the controller and the CRC read at the end of the data field did not agree with the CRC computed from the data read from the disk. When presented by non-diagnostic read commands, no buffer transfer has taken place.

No Record Found -- A media read or write command was issued to the controller and the cylinder, track, and sector numbers of the specified block did not agree with the ID field on the media. The command was terminated prematurely. In the case of a write command, no data has been placed on the media.

Seek Incomplete -- The controller is not oriented with respect to the media. Either no seek commands have been issued, an invalid seek command was issued, or a restore command was processed. This bit does not represent an error condition unless it is presented upon completion of a SEEK command.

Command Error -- A command presented to the controller was found to be invalid. For example, a write command presented with the seek incomplete bit set.

Diagnostic Status Byte

The diagnostic status byte returns a value that is dependant upon the particular command executed by the CDS controller. For non-diagnostic commands, the upper four bits are the value of the command word sent to the controller. The value returned by diagnostic commands ("Fx" Commands) is documented under the particular command.

Under normal operations, the diagnostic status byte is used to detect spurious interrupts from the CDS controller. Under certain conditions (marginal power line, faulty filter capacitor) the CDS controller will present status to the computer not associated with the completion of a command. For example, if a lightning strike causes a momentary DC Unsafe condition in the drive, the CDS controller will report status to the computer. If command execution is in progress, the controller retry mechanisms will be invoked to reprocess the command. When the command terminates, status will be presented <u>again</u>. Whenever the controller presents spurious status, the diagnostic status word is "FF" (Hex) and the command end bit is <u>not</u> set. Driver software can ignore this interrupt, or log the error condition. Returning diagnostic status for normal commands is summarized below:

00 - No Operation	60 - Format Block
10 - Restore	70 - Write Block
20 - UnVerified Seek	80 - Swap Out to Disk
30 - Verivied Seek	90 - Swap In from Disk
50 - Read Sector	A0 - Set Virtual Volume

Drive Control Status Byte

The DRIVE status is an eight-bit value presented to the SWTPC CDS Controller. Five of these bits come directly from the CDS Marksman drive interface, two come from front panel buttons, and the last bit is used as a Marksman RESET control line. These bits are summarized below:

```
                                    ________________________________
                                   |_7_|_6_|_X_|_4_|_3_|_2_|_1_|_0_|
                                     |   |   |   |   |   |   |   |
     BIT NUMBER                      |   |   |   |   |   |   |   |
                                     |   |   |   |   |   |   |   |
     7 -- Drive Select --------------+   |   |   |   |   |   |   |
     6 -- Write Enable ------------------+   |   |   |   |   |   |
     5 -- Drive Controller Reset ------------+   |   |   |   |   |
     4 -- CRC Error -----------------------------+   |   |   |   |
     3 -- Write Unsafe ------------------------------+   |   |   |
     2 -- Status Presented ------------------------------+   |   |
     1 -- Command Ready -------------------------------------+   |
     0 -- Drive Ready -------------------------------------------+
```

Drive Select -- This bit reflects the status of the front panel SELECT button. When high (one), the button indicates that the drive should be sequenced up. Note that this signal does not directly affect the ready or not ready status of the drive.

Write Enabled -- This bit reflects the status of the front panel WRITE PROT button. When high (one), the button indicates that the drive is write <u>enabled</u>. This signal is used directly to disable writing on the CDS media.

Drive Control Reset -- This bit is reflects a signal from the SWTPC controller to the Marksman controller. A low active signal (zero) sent to the Nevada board causes the drive to undergo an unconditional reset and causes the heads to be relocated to the landing zone. The pulse width of this signal must be greater than or equal to 10 microseconds. The drive will start the rezero when the bit goes back high (one). A maximum of 450 milliseconds is required to position the heads over the landing zone.

CRC Error -- This bit reflects the signal presented by the controller CRC generator circuitry. The validity of this status bit is dependant on the sequence of commands issued by the controller and does not necessarily reflect the actual condition of the CRC on any particular disk block. This bit is to be considered valid only <u>immediately</u> after a disk block read.

Write Unsafe -- This bit reflects the status of the WUSF signal from the Marksman controller to the SWTPC CDS controller. When true (one), it indicates that an unsafe write process was attempted (i.e., multiple heads selected, DC unsafe, or write on protected head). This bit is cleared when status is presented from the drive to the controller. Typically, if this bit is presented, it indicates that the CDS 24 volt power supply is faulty.

Status Presented -- This bit reflects the status of the CSTAT signal from the Marksman Controller. When true (one), it indicates that the Marksman controller is presenting drive status to the SWTPC controller.

Command Ready -- This bit reflects the status of the CRDY signal from the Marksman controller. When true (one), it indicates that the Marksman is in the input mode and ready to accept commands.

Drive Ready -- This bit reflects the status of the DRDY signal from the Marksman. When true (one), it indicates that the drive is up to speed, DC power is safe, and no drive faults have occured.

CONTROLLER COMMAND WORDS

CONTROLLER NOP

```
Bit 0                                                      Bit 23
+---------+---------+-----------------+-----------------+
| 0 0 0 0 | x x x x | x x x x x x x x | x x x x x x x x |
+---------+---------+-----------------+-----------------+
           \                                            /
 Cmd:  0    \___________  Don't Care  ________________/
```

Controller NOP (Command 0) -- The controller immediately presents ending status. Command Error and No Record Found status are reset. The diagnostic status byte is set to "00" (Hex).

RESTORE DRIVE

```
Bit 0                                                      Bit 23
+---------+---------+-----------------+-----------------+
| 0 0 0 1 | x x x x | x x x x x x x x | x x x x x x x x |
+---------+---------+-----------------+-----------------+
           \                                            /
 Cmd:  1    \___________  Don't Care  ________________/
```

RESTORE DRIVE (Command 1) -- The controller sets the Seek Incomplete status bit, then checks the state of the drive ready bit. If the drive is not ready, Command Error is set and the restore command terminates. Otherwise, a rezero command is issued to the Marksman controller and the track orientation is marked as invalid. The diagnostic status byte is set to "01" (Hex).

UNVERIFIED SEEK

```
Bit 0                                                      Bit 23
+---------+---------+-----------------+-----------------+
| 0 0 1 0 | b b b b   b b b b b b b b   b b b b b b b b |
+---------+---------+-----------------+-----------------+
           \                                            /
 Cmd:  2    \_________  20-bit Block Number ___________/
```

UNVERIFIED SEEK (Command 2) -- An unverified seek causes the head mechanism to be positioned over the proper cylinder and the correct head and sector to be selected. The sector header is not read to insure that the correct track has been reached. Unverified seeks are used for initial formatting when questionable (or no) data exists on the media. Certain commands, such as Format Write, require that an Unverified Seek <u>immediately</u> preceed them. For details on seek processing, see Command 3 - Verified Seek. The diagnostic status byte is set to "20" (Hex).

```
                             VERIFIED SEEK
     Bit 0                                                   Bit 23
     +---------+---------+-----------------+-----------------+
     | 0 0 1 1 | b b b b   b b b b b b b b   b b b b b b b b |
     +---------+---------+-----------------+-----------------+
                \                                          /
       Cmd:  3   \__________ 20-bit Block Number __________/
```

VERIFIED SEEK (Command 3) -- A seek command causes the head mechanism to be positioned over the proper cylinder and the correct head and sector to be selected. The diagnostic status byte is set to "30" (Hex). The 20-bit logical block number is based at zero, and represents the number of the block that will subsequently be processed. The media is mapped to appear as contiguous blocks, with numbers zero to the maximum allowed. The following steps are followed during seek processing:

1. The block number is biased by the currently selected virtual volume upbias. This number is then checked against the virtual volume uplimit. If the block number is invalid, or if the drive is not selected and up to speed, seek incomplete and command error are set and seek processing terminates. Note: Because of the division algorithm used for seek calculations, block numbers (including biased virtual volume blocks) must never be greater than (N_SEC / 2) * 65536 - 1. For the M-10, M-20, and M-40, this number is 147FFF (Hex) or 1,343,487 (Decimal). For the M-80, this number is 14FFFF (Hex) or 1,376,255 (Decimal). For the M-160, this number is 1CFFFF (Hex) or 1,900,543 (Decimal).

2. The block number is compared against the next block number. If a match is found, seek calculations can be bypassed. Since full seek calculation is time consuming, this heuristic results in a significant increase in performance. If a quick seek cannot be performed, the block number is divided by the number of sectors per track. The remainder of this division is the physical sector number required. The quotient is the track number, which is divided by the number of tracks per cylinder. The remainder from this division is the physical head number and the quotient is the physical cylinder number.

3. The cylinder and head numbers are compared to the current disk position to see if head movement can be bypassed. If head motion is required, the bad track map is checked for alternate track assignment. A position command is then issued to the Marksman controller to move the heads. Track orientation marked as invalid. The write unsafe status from the Marksman is checked to insure that the position command terminated normally. If any errors are detected, the Seek Incomplete bit is set and the retry sequence is initiated.

4. If verification is requested, the next record header on the selected track is read and the cylinder and head numbers compared to the requested track. If the cylinder and head numbers match, the Seek Incomplete bit is reset. The next block number is calculated for quick seek processing. If the next block would span a track boundary, the quick seek is disabled. The physical sector number is used to calculate the logical sector and previous sector numbers.

5. If a retry sequence is necessary, the diagnostic status bit is checked. If set, Seek Incomplete is set and processing terminates immediately. Otherwise, a rezero command is issued to the Marksman and the seek request is retried up to a total of 23 times. If all retries fail, Seek Incomplete is set and the seek command terminates. Note that Command Error is set only if the block number is invalid.

```
                         FIRMWARE VERSION
                                                       Bit 23
 Bit 0
+---------+---------+-----------------+-----------------+
| 0 1 0 0 | 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 v v v v v |
+---------+---------+-----------------+-----------------+
                                               \       /
   Cmd:  4               Version Byte Number ___\_____/
```

FIRMWARE VERSION (Command 4) -- This command is used to return information regarding the Version Numbers and other information about the SWTPC Controller Firmware. The Version Byte Request Number is used request specific information about the current firmware. The information is returned in the diagnostic status byte. Available information is as follows:

0 - Firmware Version
1 - Firmware Revision
2 - Firmware Modification
3 - 2-MHz Mod Flag
4 - Release Date - Month
5 - Release Date - Day
6 - Release Date - Year
7 - Sectors per Track
8 - Tracks per Cylinder
9,10 - Number of Cylinders
11 - Default Data Spacing
12 - Default Swap Spacing
13 - Number of Track Maps
14 - Number of Virtual Volumes
15 - Size of Swap Table
16 - Size of Buffer Memory

```
                              READ BLOCK
   Bit 0                                                     Bit 23
   +---------+---------+-----------------+-----------------+
   | 0 1 0 1 | e e e e | a a a a a a a a   a a a a a a a a |
   +---------+---------+-----------------+-----------------+
                       \                                   /
     Cmd:  5            \_____ 20-bit Physical Memory Address _____/
```

READ BLOCK (Command 5) -- This command reads the selected sector from disk and performs a DMA transfer operation to move the sector to the designated area of memory. The address specified must be a full 20-bit physical address, and the sector buffer must not cross a 4K segment boundary. The diagnostic status byte is set to "50" (Hex). The following steps are followed during read processing:

1. If the seek incomplete bit is set, or if the drive is not selected and up to speed, the read command is terminated with a command error. If the controller is not oriented as to track position, a sector header read is performed to assure proper orientation.

2. The current sector is compared to the number of the sector <u>previous</u> to the one designated for reading. When the proper number spins around, the read circuitry is activated to read both the header and data portion of the next (proper) sector. At the completion of the read, the status of the CRC error line is sampled and used to set the CRC Error bit in the primary status byte. The requested cylinder, track, and sector numbers are compared to those of the sector just read, and the result is used to set the No Record Found status bit. The disk read causes track orientation to be lost.

3. If the read diagnostic flag is set (See Diagnostic Read) or if neither the CRC Error or No Record Found bits are set, a DMA transfer operation is used to move the sector data area to the host computer. The DMA transfer is conducted as a burst operation and requires 512+4 machine cycles for completion. Status is then presented and the read processing terminates.

4. If the read diagnostic flag is not set, and either CRC Error or No Record Found are set, then the read is retried. Up to 32 retries may be processed before a permanent error is reported. If retry processing fails, the sector data area is not transfered to the host computer.

FORMAT BLOCK

```
Bit 0                                                    Bit 23
+---------+---------+-----------------+-----------------+
| 0 1 1 0 | e e e e | a a a a a a a a   a a a a a a a a |
+---------+---------+-----------------+-----------------+
          \                                            /
 Cmd:  6   \_____ 20-bit Physical Memory Address _____/
```

FORMAT BLOCK (Command 6) -- This command is used to write data onto media that may contain corrupted (or no) data. Format write processing proceeds through the same details as ordinary writes, except that no track headers are read (in step 3). The controller waits until the next index pulse, then counts sectors until it gets to the sector previous to the selected sector number. For this reason, the controller can perform only one format write per disk rotation. Absolute track orientation is provided at the expense of performance. Note that if the previous command was not an unverified seek, then the format write command is terminated with a command error. The diagnostic status byte is set to "60" (Hex). See the description of WRITE BLOCK.

WRITE BLOCK

```
Bit 0                                                    Bit 23
+---------+---------+-----------------+-----------------+
| 0 1 1 1 | e e e e | a a a a a a a a   a a a a a a a a |
+---------+---------+-----------------+-----------------+
          \                                            /
 Cmd:  7   \_____ 20-bit Physical Memory Address _____/
```

WRITE BLOCK (Command 7) -- This command performs a DMA transfer to obtain the specified buffer from the host computer. The buffer is then written to the selected sector on disk. The address specified must be a full 20-bit physical address, and the sector buffer must not cross a 4K segment boundary. The diagnostic status byte is set to "70" (Hex). The following steps are followed during write processing:

1. If the seek incomplete bit is set or if the disk is not selected and up to speed, then the write command is terminated with a command error. If the disk is write protected, then command error and write protect are presented to the host.

2. A DMA transfer operation is performed to move the buffer from the host computer into the disk controller. The DMA transfer is conducted in burst mode and requires 512 + 4 machine cycles for completion. The DMA transfer operation causes track orientation to be lost. The selected cylinder, track, and sector numbers are used to construct a record header in the controller buffer. If the

diagnostic mode flag is not set, then the CRC corruption word in the sector header is set to zero. (See the description of Diagnostic Write.)

3. If this is a normal write operation, the next sector header is read from disk. If this header belongs to the sector previous to the selected sector, the write data code is entered immediately (next step). Otherwise, the controller will count sectors until it gets to what it thinks is the sector two sectors previous to the selected sector. The next sector header is then read from disk. The cylinder, track, and sector numbers of this header are compared to those of the sector prior to the selected sector. If they do not match, the No Record Found status bit is set and retry processing is initiated. In this way, all non-format write commands read and verify the header of the sector immediately prior to the sector that will be written.

4. Once proper track positioning has been achieved, the DMA Write circuitry is activated to write the controller buffer (preamble, CRC corruption word, header sync word header, data buffer, and postamble) to the disk media. The CRC is automatically calculated and appended to the record. Ending status is presented and write processing is terminated. Note that the write operation causes track orientation to be lost.

```
                         SWAP OUT TO DISK
Bit 0                                                      Bit 23
+---------+---------+------------------+-----------------+
| 1 0 0 0 | e e e e | a a a a a a a a     a a a a a a a a |
+---------+---------+------------------+-----------------+
           \                                            /
  Cmd:  8   \_____ 20-bit Physical Memory Address _____/
                   Pointer to Address Space Page Table
```

SWAP OUT TO DISK (Command 8) -- This command is used to write a address space to disk starting at the selected sector. The 20-bit physical address that is passed to the Swap command is the address of the memory page table for the address space to be swapped. The page table is always 16 bytes long, and any unused pages must contain the hex value $F3 (Remember that the low order four bits of the MP-09 memory map are inverted). Each active page in the table causes 8 sectors to be written to the disk (in address sequential order), with the sector number properly incremented. At the end of the swap command, the currently selected sector will be the number of the last 512-byte buffer transfered. The diagnostic status byte is set to "80" (Hex).

The actual sequence in which the 512-byte buffers are fetched from memory and in which the sectors are written to the disk media may not be the same as the order in which these areas are arranged in memory. Because of the disk buffering algorithms, these sectors may be written nonsequentially. The CDS controller maintains logical consistancy between disk blocks and swap requests, regardless of the physical order in which data is accessed.

UniFlex* uses space reserved at the end of the logical volume for swap operations. For this reason, the CDS controller has been given the ability to use different sector spacings for data and swap operations. The use of this option can provide as much as a two-to-one speed advantage over normal sector spacing values. In order to properly use this feature, the amount of swap space reserved on the disk should match the number of cylinders configured for swapping.

```
                           SWAP IN FROM DISK
                                                              Bit 23
   Bit 0
   +--------+--------+-----------------+-----------------+
   | 1 0 0 1 | e e e e | a a a a a a a a   a a a a a a a a |
   +--------+--------+-----------------+-----------------+
             \                                          /
     Cmd:  9   \______ 20-bit Physical Memory Address ____/
                       Pointer to Address Space Page Table
```

SWAP IN FROM DISK (Command 9) -- This command is used to read an address space from disk starting at the selected sector. The 20-bit physical address that is passed to the Swap command is the address of the memory page table for the address space to be swapped. The page table is always 16 bytes long, and any unused pages must contain the hex value $F3 (Remember that the low order four bits of the MP-09 memory map are inverted). Each active page in the table causes 8 sectors to be read from the disk (in address sequential order), with the sector number properly incremented. At the end of the swap command, the currently selected sector will be the number of the last 512-byte buffer transfered. The diagnostic status byte is set to "90" (Hex).

The actual sequence in which the 512-byte buffers are fetched from memory and in which the sectors are written to the disk media may not be the same as the order in which these areas are arranged in memory. Because of the disk buffering algorithms, these sectors may be written nonsequentially. The CDS controller maintains logical consistancy between disk blocks and swap requests, regardless of the physical order in which data is accessed.

```
                          SELECT VIRTUAL VOLUME
      Bit 0                                                     Bit 23
      +---------+---------+-----------------+-----------------+
      | 1 0 1 0 | 0 0 0 0 | 0 0 0 0 0 0 0 0 | d d d d d d d d |
      +---------+---------+-----------------+-----------------+
                                             \               /
        Cmd: 10      Virtual Volume Number ___\_____________/
```

SELECT VIRTUAL VOLUME (Command 10) -- The Virtual Volume Select command changes the pointer into the Virtual Volume Table in the CDS controller. This has the effect of being a "drive select", except that the "drive" is one of the Virtual Volumes defined by the CDS Configurator (See Configuration Commands). If virtual volume support is not included in the CDS configuration, or if the selected virtual volume does not exist, the select command returns Command Error and No Record Found. If no errors are detected, all subsequent Seek commands will have their block numbers biased by the selected virtual volume upcount, and then limit checked against the virtual volume block limit. The diagnostic status byte is set to "A0" (Hex).

DIAGNOSTIC COMMAND WORDS

INVALID DIAGNOSTIC COMMAND

```
Bit 0                                                        Bit 23
+---------+---------+---------+---------+-----------------+
| 1 1 1 1 | x x x x | x x x x | 0 0 0 0 | x x x x x x x x |
+---------+---------+---------+---------+-----------------+
           \____ Ignored ____/ \ Error / \___ Ignored ___/
  Cmd: 15
```

INVALID DIAGNOSTIC COMMAND (Command 15, Function 0) -- The Command Error bit is set and the controller immediately presents ending status. No I/O requests are issued to the Marksman. The diagnostic status byte is set to "F0" (Hex). This command is used to test the command decode and error detection code in the CDS controller.

FORMAT TRACK

```
Bit 0                                                        Bit 23
+---------+---------+---------+---------+-----------------+
| 1 1 1 1 | x x x x | x x x x | 0 0 1 0 | d d d d d d d d |
+---------+---------+---------+---------+-----------------+
           \____ Ignored ____/ \ Format/ \__ Data Byte __/
  Cmd: 15                          Track
```

FORMAT TRACK (Command 15, Function 2) -- The format track command is used to write format information on an entire track of the Marksman media. This command performs the same function as the Format Block command, except that an entire track is processed (in 100 mSec.) as opposed to one block (in 25 mSec.). The blocks written on the track are filled with the data byte field. For normal formatting, a data byte of zero is recommended.

In order to prevent accidental loss of data, the format track command checks for three conditions: First, a sequence drive up command must have been issued sometime prior to the format track. Second, an unverified seek command must have been used to set the current block immediately prior to issuance of the format track. Third, the seek address specified must lie on an integral track boundary. If any of these conditions are not met, the format track will terminate with a command error. The diagnostic status byte is set to "F2" (Hex).

```
                       DATA LOOPBACK TEST
Bit 0                                                Bit 23
+---------+---------+---------+---------+----------------+
| 1 1 1 1 | x x x x | x x x x | 0 0 1 1 | d d d d d d d d |
+---------+---------+---------+---------+----------------+
           \____ Ignored ____/ \ Data / \__ Data Byte __/
  Cmd: 15                          Loop
```

DATA LOOPBACK TEST (Command 15, Function 3) -- The data byte presented by the command is returned to the computer as the diagnostic status byte. This command is normally used to check the integrety of the letterbox data path, including the CDS to Computer cable assembly.

```
                       READ SECTOR HEADER
Bit 0                                                Bit 23
+---------+---------+---------+---------+----------------+
| 1 1 1 1 | x x x x | x x x x | 0 1 0 0 | x x x x x x x x |
+---------+---------+---------+---------+----------------+
           \____ Ignored ____/ \ Read / \___ Ignored ___/
  Cmd: 15                        Header
```

READ SECTOR HEADER (Command 15, Function 4) -- The read header command causes the CDS controller to read the next sector header on the currently selected track. This causes the controller to become oriented with respect to the current track. This command is used to perform seek verification during drive checkout. The diagnostic status byte is set to "F4" (Hex).

```
                       DIAGNOSTIC READ
Bit 0                                                Bit 23
+---------+---------+---------+---------+----------------+
| 1 1 1 1 | e e e e | a a a a | 0 1 0 1 | x x x x x x x x |
+---------+---------+---------+---------+----------------+
           \                 / \ Read / \___ Ignored ___/
  Cmd: 15   \               /
             \_____________/_______ Upper 8 bits of 20-bit
                                     Physical Address. Low
                                     order bits are zero.
```

DIAGNOSTIC READ (Command 15, Function 5) -- The diagnostic read command is used to test for soft read errors, and to recover data from sectors that may have CRC errors. For details about read processing, see the documentation on READ BLOCK. Note that normal block retry procedures are suspended by a diagnostic read. The diagnostic status byte is set to "F5" (Hex).

```
                         DIAGNOSTIC WRITE
                                                       Bit 23
Bit 0
+--------+--------+--------+--------+-----------------+
| 1 1 1 1 | e e e e | a a a a | 0 1 1 0 | x x x x x x x x |
+--------+--------+--------+--------+-----------------+
          \                 / \ Write / \___ Ignored ___/
  Cmd: 15  \               /
            \_____________/_______ Upper 8 bits of 20-bit
                                    Physical Address. Low
                                     order bits are zero.
```

DIAGNOSTIC WRITE (Command 15, Function 6) -- The diagnostic write command is used to write a sector to disk with a known bad CRC. This is accomplished through the use of the CRC Corruption word in the disk preamble. When a disk write is initiated, the write circuitry primes the CRC generator with zero. The CRC generator is then clocked for each bit in the disk record preamble. During normal writes, all of these bits are zero, causing the CRC generator value to remain zero. If non-zero bits are placed in the preamble, the CRC is corrupted and will appear as invalid when encountered during read processing. The diagnostic status byte is set to "F6" (Hex).

```
                         SEQUENCE DRIVE UP
                                                       Bit 23
Bit 0
+--------+--------+--------+--------+-----------------+
| 1 1 1 1 | x x x x | x x x x | 0 1 1 1 | 0 0 0 1 0 0 0 0 |
+--------+--------+--------+--------+-----------------+
          \____ Ignored ____/ \       / \__ Sequence __/
  Cmd: 15                      \     /      Drive Up
                 Drive Request __\___/
```

SEQUENCE DRIVE UP (Command 15, Function 7) -- The sequence up command is used to spin up the Marksman drive, regardless of the state of the front panel SELECT button. A manual sequence flag is set internal to the controller which is required for operation of certain other commands. The process of sequencing consists of the following steps:

1. The drive status is checked for drive ready. If the drive is not ready, a sequence up command is sent to the Marksman and the returning status checked for errors.

2. A set sector command is issued to override the sector length switches on the Marksman. This implies that the sector length switches are effectively ignored.

3. The disk is restored to insure proper orientation of the Marksman controller. The seek incomplete bit is set in the status byte.

4. If sequencing was initiated by the front panel button, the configuration blocks are read and verified. If a manual sequence is in progress, configuration processing is bypassed. The diagnostic status byte is set to "F7" (Hex).

```
                         SEQUENCE DRIVE DOWN
Bit 0                                                       Bit 23
+---------+---------+---------+---------+-----------------+
| 1 1 1 1 | x x x x | x x x x | 0 1 1 1 | 0 0 0 1 0 0 0 1 |
+---------+---------+---------+---------+-----------------+
          \____ Ignored ____/ \         / \__ Sequence __/
  Cmd: 15                      \       /      Drive Down
                 Drive Request __\___/
```

SEQUENCE DRIVE DOWN (Command 15, Function 7) -- The sequence down command is used to spin up the Marksman drive, regardless of the state of the front panel SELECT button. A manual sequence flag is set internal to the controller which is required for operation of certain other commands. The diagnostic status byte is set to "F7" (Hex).

```
                         REQUEST DRIVE STATUS
Bit 0                                                       Bit 23
+---------+---------+---------+---------+-----------------+
| 1 1 1 1 | x x x x | x x x x | 0 1 1 1 | 0 0 0 0 s s s s |
+---------+---------+---------+---------+-----------------+
          \____ Ignored ____/ \         / \___ Status ____/
  Cmd: 15                      \       /    Request Byte
          Request Drive Status __\___/
```

REQUEST DRIVE STATUS (Command 15, Function 7) -- The request status command interrogates the Marksman controller for status. The particular status byte requested is returned as the diagnostic status byte. The meaning of this status byte is documented in the Marksman Performance Specification, Century Data Systems document #76220-906. Note that not all revisions of the Marksman controller have extended status.

```
                    REQUEST DRIVE DIAGNOSTIC
Bit 0                                                    Bit 23
+---------+---------+---------+---------+-----------------+
| 1 1 1 1 | x x x x | x x x x | 0 1 1 1 | 1 0 0 0 s s s s |
+---------+---------+---------+---------+-----------------+
          \____ Ignored ____/ \         / \_ Diagnostic _/
  Cmd: 15                      \       /     Request Byte
          Request Diagnostic __\___/
```

REQUEST DRIVE DIAGNOSTIC (Command 15, Function 7) -- The drive diagnostic request command passes the diagnostic request byte to the Marksman controller. When the Marksman controller has finished its diagnostic processing, drive status (zero) will be presented to the CDS controller. This status is returned as diagnostic status to the computer. In certain circumstances, additional drive status requests may be issued to obtain additional information about the Marksman. The meaning of this status byte is documented in the Marksman Performance Specification, Century Data Systems document #76220-906. Note that not all revisions of the Marksman controller have diagnostic capability.

```
                     RESET SWTPC CONTROLLER
Bit 0                                                    Bit 23
+---------+---------+---------+---------+-----------------+
| 1 1 1 1 | x x x x | x x x x | 1 0 0 0 | x x x x x x x x |
+---------+---------+---------+---------+-----------------+
          \____ Ignored ____/ \         / \___ Ignored ___/
  Cmd: 15                      \       /
                                \___/____ Reset SWTPC
                                           Controller
```

RESET SWTPC CONTROLLER (Command 15, Function 8) -- The Controller Reset command causes a software reset of the CDS controller. A reset indication is presented to the Marksman drive and all reset initialization is performed on the SWTPC Controller. An interrupt is returned to the computer upon completion of the reset processing, but no valid status is returned.

SET DATA SECTOR SPACING

```
Bit 0                                                      Bit 23
+---------+---------+---------+---------+-----------------+
| 1 1 1 1 | x x x x | x x x 0 | 1 0 0 1 | s s s s s s s s |
+---------+---------+---------+---------+-----------------+
           \__ Ignored __/ \___ Set ___/ \___ Spacing ___/
   Cmd: 15                    Spacing
```

SET DATA SECTOR SPACING (Command 15, Function 9) -- The set data spacing command is used to set the sector interleave used by the CDS controller. This spacing overrides the spacing definition found in the configuration block. Note that this value is <u>not</u> stored in the configuration block on disk, hence it does not permanently change the sector spacing used by the drive. This command is normally used in certain backup procedures. The diagnostic status byte is set to "F9" (Hex).

SET SWAP SECTOR SPACING

```
Bit 0                                                      Bit 23
+---------+---------+---------+---------+-----------------+
| 1 1 1 1 | x x x x | x x x 1 | 1 0 0 1 | s s s s s s s s |
+---------+---------+---------+---------+-----------------+
           \__ Ignored __/ \___ Set ___/ \___ Spacing ___/
   Cmd: 15                    Spacing
```

SET SWAP SECTOR SPACING (Command 15, Function 9) -- The set swap spacing command is used to set the sector interleave used by the CDS controller. This spacing overrides the spacing definition found in the configuration block. Note that this value is <u>not</u> stored in the configuration block on disk, hence it does not permanently change the sector spacing used by the drive. The diagnostic status byte is set to "F9" (Hex).

```
                         READ CONFIGURATION BLOCK
                                                        Bit 23
    Bit 0
    +---------+---------+---------+---------+--------------+
    | 1 1 1 1 | e e e e | a a a a | 1 0 1 0 | x x x x x x x x |
    +---------+---------+---------+---------+--------------+
               \                  / \  Read / \___ Ignored ___/
      Cmd: 15   \                /  Configuration
                 \              /
                  \____________/_______ Upper 8 bits of 20-bit
                                          Physical Address. Low
                                           order bits are zero.
```

READ CONFIGURATION BLOCK (Command 15, Function 10) -- The Read Configuration command is used to initialize certain control tables from one of the configuration blocks recorded on the last track of the disk. The diagnostic status byte is set to "F5" (Hex).

In order to obtain this data, the controller disables all alternate track mapping, then calculates the block number of the first block on the last track. Each block on the track is examined in turn. Whenever a valid configuration block is located, the controller initializes the alternate track maps, data and swap spacing, and the virtual volume table. If no valid configuration blocks are read, a no record found indication is presented. The format of the configuration block is documented under the "Write Configuration" command.

```
                         WRITE CONFIGURATION BLOCKS
                                                        Bit 23
    Bit 0
    +---------+---------+---------+---------+--------------+
    | 1 1 1 1 | e e e e | a a a a | 1 0 1 1 | x x x x x x x x |
    +---------+---------+---------+---------+--------------+
               \                  / \ Write / \___ Ignored ___/
      Cmd: 15   \                /  Configuration
                 \              /
                  \____________/_______ Upper 8 bits of 20-bit
                                          Physical Address. Low
                                           order bits are zero.
```

WRITE CONFIGURATION BLOCKS (Command 15, Function 11) -- The write configuration command is used to write the configuration blocks onto the Marksman maintainance track. This track is located on the last head of the last cylinder on the drive. The specified physical address points to the 512-byte configuration block to written. This block is written into every sector on the last track. If at lease one of these blocks is readable, the drive can be configured (See Read Configuration).

During normal operation, the track containing the configuration blocks is protected against access by the host. This track can be accessed in one of three ways: First, by issuance of a Disable Track Map command (F00E00). Second, by issuance of a Read Configuration command, and Third, by use of the Write Configuration command. The following table lists the cylinder address and block number of the first configuration block:

Model	First Block Number		Cyl	Hd	#Sec
M-10	17,425	04411	212	1	41
M-20	34,891	0884B	212	3	41
M-40	69,823	1106F	212	7	41
M-80	143,346	22FF2	568	5	42
M-160	283,864	454D8	844	5	56

The configuration blocks contain a two-byte identification code ($99BB), 508 bytes of configuration information, and a two-byte checksum that is separate from the block's CRC field. The identifier and checksum are designed to prevent inadvertant recognition of a data block as a configuration block. The format of the configuration block is as follows:

```
0000  figblok    org     0          set origin at zero

0000  F_SYN      rmb     2          SYNC word for Block
0002  F_SSI      rmb     1          Data Sector Spacing
0003  F_SPI      rmb     1          Swap Sector Spacing
0004  F_FAC      rmb     2          Fold Cylinder Value
0006  F_VVC      rmb     1          Virtual Volume Count
0007  F_ATC      rmb     1          Alternate Track Count
0008  F_FAT      rmb     3          First Alternate Track
000B             rmb     115
007E  F_VVT      rmb     6*16       Virtual Volume Table
00DE  F_ATM      rmb     6*48       Alternate Track Map
01FE  F_SUM      rmb     2          Configuration Checksum

0200             errif   *!=512     block wrong size!
```

The diagnostic status byte is normally set to "FB" (Hex), but if one of the configuration block writes fails, it's sector number (on the configuration track) is returned instead. Normally, drives shipped with SWTPC CDS Disk Units are configured at the factory. SWTPC will refuse to ship drives that are not configurable.

```
                    READ CONTROLLER SCRATCHPAD
 Bit 0                                          Bit 23
+--------+--------+--------+--------+----------------+
| 1 1 1 1 | e e e e | a a a a | 1 1 0 0 | x x x x x x x x |
+--------+--------+--------+--------+----------------+
          \                 / \ Read / \___ Ignored ___/
  Cmd: 15  \               /  Scratchpad
            \             /
             \___________/_______ Upper 8 bits of 20-bit
                                   Physical Address. Low
                                   order bits are zero.
```

READ CONTROLLER SCRATCHPAD (Command 15, Function 12) -- The read scratchpad command reads the CDS controller scratchpad RAM into the 1K block of memory specified. For the interpretation of the scratchpad contents, refer to the UDISK PROGRAM LISTING. This listing is not included with the CDSBUG diagnostic package. To obtain this proprietary information, you must submit a written non-disclosure agreement to Southwest Technical Products Corporation, 219 W. Rhapsody, San Antonio, Texas, 78216. The diagnostic status byte is set to "FC" (Hex).

```
                    READ CONTROLLER FIRMWARE
 Bit 0                                          Bit 23
+--------+--------+--------+--------+----------------+
| 1 1 1 1 | e e e e | a a a a | 1 1 0 1 | x x x x x x x x |
+--------+--------+--------+--------+----------------+
          \                 / \ Read / \___ Ignored ___/
  Cmd: 15  \               /   Firmware
            \             /
             \___________/_______ Upper 8 bits of 20-bit
                                   Physical Address. Low
                                   order bits are zero.
```

READ CONTROLLER FIRMWARE (Command 15, Function 13) -- The contents of the UDISK control ROM is read into the specified 8K buffer area. This command is used by the CDSBUG "V" command to determine the ROM checksum. For more detailed information, refer to the UDISK PROGRAM LISTING. This listing is not included with the CDSBUG diagnostic package. To obtain this proprietary information, you must submit a written non-disclosure agreement to Southwest Technical Products Corporation, 219 W. Rhapsody, San Antonio, Texas, 78216. The diagnostic status byte is set to "FD" (Hex).

```
                         DISABLE ALTERNATE TRACKS
Bit 0                                                          Bit 23
+----------+----------+----------+----------+-----------------+
| 1 1 1 1 | x x x x | x x x x | 1 1 1 0 | x x x x x x x x |
+----------+----------+----------+----------+-----------------+
           \____ Ignored ____/ \        / \___ Ignored ___/
  Cmd: 15                        \      /
                                  \___/___ Disable Alternate
                                           Track Assignments
```

DISABLE ALTERNATE TRACKS (Command 15, Function 14) -- The disable track mapping command causes the controller to establish a one-to-one correspondance between logical block numbers and physical sector numbers. In addition, the track containing the configuration blocks is made accessible to all disk commands. The alternate track map is made empty, virtual volume zero is selected, and the limit block number is set to the physical number of blocks on the media. The diagnostic status byte is set to "FE" (Hex).

```
                        RESET MARKSMAN CONTROLLER
Bit 0                                                          Bit 23
+----------+----------+----------+----------+-----------------+
| 1 1 1 1 | x x x x | x x x x | 1 1 1 0 | x x x x x x x x |
+----------+----------+----------+----------+-----------------+
           \____ Ignored ____/ \        / \___ Ignored ___/
  Cmd: 15                        \      /
                                  \___/___ Reset Marksman
                                             Controller
```

RESET MARKSMAN CONTROLLER (Command 15, Function 15) -- The reset Marksman command sends a drive reset to the Marksman controller. The seek incomplete bit is then set in the primary status byte. If the drive is spinning and ready, this command will cause the front panel "Select" light to momentarily go out. The diagnostic status byte is set to "FF" (Hex).

GENERAL SYSTEM DIAGNOSTICS

DISKTEST - DISKMEM - TIMETEST - CFTEST

General System Diagnostics
(MP-09 / UniFlex Version)

Disclaimer

DISKTEST
(UniFlex Version)

DISKTEST is a diagnostic program that can be used by the UniFlex system user or by qualified technical personnel as an aid in the testing or the repairing of a DMF2 or DMF3 Disk Controller and disk drives. The test is very useful if errors are suspected or if your system has just been installed.

DISKTEST assumes a DMF2 or DMF3 disk controller and one or two double-sided, double density eight-inch disk drives are being tested.

PROCEDURE

NEEDED: One or two formatted "test disks". Both disks must be formatted double-sided, double density by the program "formatfd". The "test disks" do not have to contain UniFlex or any other utilities.

NOTE: The system MUST BE in single user mode.

Insert disk containing the program "DISKTEST", type "disktest" and return. The message "UniFlex Disktest" should appear on the screen. At this point, you may want to remove the system disk containing the "DISKTEST" program and insert one or two "test disks". The test then asks if the formatted scratch disks have been installed -- answering "y" will cause the test to continue. The next prompt "Enter drive to test or hit return for all" asks which drive is to be tested. If both drives are to be tested, type "Return", if only drive 1 is to be tested, type "1", or if only drive 0 is to be tested, type "0".

DO NOT USE DISKTEST ON ANY 'WORKING' DISKETTE OR DATA MAY BE DESTROYED!!

Once DISKTEST is executed you will first see on the screen "Testing read/write capabilities of disk drives" and the testing will start. A "+" is returned to the terminal for each good pass of the test. Testing will continue until either "Control C" is typed (which will terminate the program and cause the system to stop) or an escape key is typed (which will interrupt the program until the escape key is typed again — this will continue the DISKTEST) or if an error is encountered an error message will be displayed (see ERROR MESSAGES) and the test will continue. DISKTEST will change the format and contents of the data on track 76 — thus any diskettes used for DISKTEST should be used for testing purposes only.

ERROR MESSAGES

The errors reported on the terminal screen by DISKTEST are those reported by the disk controller integrated circuit on the board. Both "soft errors" and "fatal errors" will be reported. Soft errors are errors that may be caused by dust particles, diskette wrinkling, motor speed fluctuation and a number of other factors. Fatal errors are very serious errors. If any fatal errors occur or if more than one soft error is found for every five passes then problems may be suspected and a technician should be contacted to have the system checked.

If an error message occurs it will be followed by the history surrounding the error. This is useful information for the technician. The error is followed by what the test was doing when the error occurred ("READING" or "WRITING"), which drive the error occurred in, the last command executed before the error occurred ("PAST") and where the error actually occurred ("PRES"). "PAST" and "PRES" are followed by information about the "DRIVE", "TRACK", "SECTOR", "TYPE" and the "STATUS" -- all of which are reported in hexadecimal numbers. "DRIVE" is equal to either "00" or "01" depending on which drive was in error. "TRACK" and "SECTOR" are the actual locations on the diskette which are in error. "TYPE" is set to either "88" (a read operation) or "A8" (a write operation). The "STATUS" tells the ending state of the disk controller after the error occurred.

STATUS

```
                          ________________________________________
                         |__7__|__6__|__5__|__4__|__3__|__2__|__1__|__0__|
                            |     |     |     |     |     |     |     |
BIT NUMBER                  |     |     |     |     |     |     |     |
                            |     |     |     |     |     |     |     |
 7 -- DRIVE NOT READY-------|     |     |     |     |     |     |     |
 6 -- WRITE PROTECTED-------------|     |     |     |     |     |     |
 5 -- RECORD TYPE/WRITE FAULT-----------|     |     |     |     |     |
 4 -- RECORD NOT FOUND------------------------|     |     |     |     |
 3 -- CRC ERROR-------------------------------------|     |     |     |
 2 -- LOST DATA-------------------------------------------|     |     |
 1 -- DATA REQUEST----------------------------------------------|     |
 0 -- BUSY------------------------------------------------------------|
```

SUMMARY OF ERROR MESSAGES

SOFT ERRORS:

CRC (Cyclic Redundancy Check) ERROR --
The CRC "checksum" read at the end of an ID or data field did not agree with the data read off the disk. One or more bits of the data read will be invalid.

RECORD NOT FOUND --
The track and sector that the disk controller was trying to read was not found. This error may occur when the disk's heads are positioned on the wrong track, when the density is wrong, when the diskette is faulty, etc.

FATAL ERRORS:

DRIVE NOT READY --
No diskette installed, door not closed, non-existent drive specified, etc.

DISK IS WRITE PROTECTED --
Writing cannot be completed because the diskette is write protected.

ERROR IN COMPARING WRITTEN SECTOR TO MEMORY --
The test pattern that was read from the disk does not agree with the pattern written initially. This can be caused by a faulty memory to disk transfer during the write or a faulty disk to memory transfer during a read.

DATE LOST IN TRANSFER --
A byte that was read from the disk by the disk controller IC was not read by the DMA portion of the DMF2.

UNABLE TO SET DENSITY ON SELECTED DRIVE --
The test was unable to read a test sector from the disk that is used for initialization (un-formatted disk, no disk installed, etc.).

DRIVE SELECT ERROR --
The drive selected could not be tested.

DISKMEM
(UniFlex Version)

The diagnostic DISKMEM may be used by the UniFlex system user or qualified technical personnel as an aid in finding errors in disk to memory and memory to disk data transfers. DISKMEM will damage the information on one sector of track 76 of the diskette, therefore a "test disk" (a diskette that contains the program "DISKMEM" and no vital information) should be used for testing purposes only. Reset should only be used when absolutely necessary.

DISKMEM assumes a DMF2 or DMF3 disk controller and at least one double-sided, double density eight-inch disk drive is being used for this test.

PROCEDURE

One "test disk," formatted double-sided, double density, containing the program "DISKMEM" is needed to run this test. Insert the "test disk" into drive 0 (this test will only test in drive 0) and type "DISKMEM".

Once DISKMEM is executed, transfers will begin between the diskette in drive 0 and memory. The message "Testing transfers - wait for test to completely terminate" will be appear on the screen. A "+" will be displayed after each pass and the test will terminate after approximately 256 passes displaying the message "Disk to memory test completed". DISKMEM may be exited any time by typing "Control C". To temporarily interrupt the test, type the escape key. To continue the test, type the escape key a second time.

ERROR MESSAGES

The errors reported on the terminal screen by DISKMEM are those reported by the disk controller integrated circuit on the board. Both "soft errors" and "fatal errors" will be reported. Soft errors are errors that may be caused by dust particles, diskette wrinkling, motor speed fluctuation and a number of other factors. Fatal errors are very serious errors. If any fatal errors occur or if more than one soft error is found for every five passes then problems may be suspected and a technician should be contacted to have the system checked.

If an error message occurs it will be followed by the history surrounding the error. This is useful information for the technician. The error is followed by what the test was doing when the error occurred ("READING" or "WRITING"), where the last command was executed before the error occurred ("PAST") and where the error actually occurred ("PRES"). "PAST" and "PRES" are followed by information about the "DRIVE", "TRACK", "SECTOR", "TYPE" and the "STATUS" -- all of which are reported in hexadecimal numbers. "DRIVE" is equal to either "00" or "01" depending on which drive was in error. "TRACK" and "SECTOR" are the actual locations on the diskette which are in error. "TYPE" is set to either "88" (a read operation) or "A8" (a write operation). The "STATUS" tells the ending state of the disk controller after the error occurred.

STATUS

```
                          ________________________________________
                         |__7__|__6__|__5__|__4__|__3__|__2__|__1__|__0__|
                            |     |     |     |     |     |     |     |
BIT NUMBER                  |     |     |     |     |     |     |     |
                            |     |     |     |     |     |     |     |
 7 -- DRIVE NOT READY-------|     |     |     |     |     |     |     |
 6 -- WRITE PROTECTED ON----------|     |     |     |     |     |     |
 5 -- RECORD TYPE/WRITE FAULT-----------|     |     |     |     |     |
 4 -- RECORD NOT FOUND------------------------|     |     |     |     |
 3 -- CRC ERROR-------------------------------------|     |     |     |
 2 -- LOST DATA-------------------------------------------|     |     |
 1 -- DATA REQUEST----------------------------------------------|     |
 0 -- BUSY------------------------------------------------------------|
```

SUMMARY OF ERROR MESSAGES

SOFT ERRORS:

CRC (Cyclic Redundancy Check) ERROR --
The CRC "checksum" read at the end of an ID or data field did not agree with the data read off the disk. One or more bits of the data read will be invalid.

RECORD NOT FOUND --
The track and sector that the disk controller was trying to read was was not found. This error can occur when the disk's heads are positioned on the wrong track, when the density is wrong, when the diskette is faulty, etc.

FATAL ERRORS:

DRIVE NOT READY --
No diskette installed, door not closed, non-existent drive specified, etc.

THE SELECTED DISK IS WRITE PROTECTED --
Writing cannot be completed because the diskette is write protected.

ERROR IN COMPARING WRITTEN SECTOR TO MEMORY --
The test pattern that was read from the disk does not agree with the pattern written initially. This can be caused by a faulty memory to disk transfer during the write or a faulty disk to memory transfer during a read.

DATA LOST IN XFER FROM 1791 TO 6844 --
A byte that was read from the disk by the disk controller IC was not read by the DMA portion of the DMF2.

MPS2TEST - MPL2TEST - MPIDPATEST

Special Interface Diagnostics
(MP-09 / UniFlex Version)

Disclaimer

MPS2TEST

MPS2TEST is a diagnostic program that can be used by qualified technical personnel as an aid in testing or repairing a SWTPC MP-S2 serial interface.

This test requires special jumper configurations and a special test cable. The board should be configured as shown on the configuration pages that follow. A diagram of the test cable is also given. Also, this test operates by taking over the computer from the operating system and should never be used in the multi user mode.

Test Procedure

After making the appropriate connections and hooking up the cables on the MP-S2, type "MPS2TEST" and return.

THIS TEST WILL NOT WORK IF THE CABLES ARE NOT HOOKED UP!!

The program will begin execution and the message "SWTPC MP-S2 Serial Interface Diagnostic" will appear on the screen, followed by the prompt:

"Input the port number [1-7] "

At this time the port number of the interface to be tested should be entered. NOTE: Be sure that the MP-S2 test cable is installed.

If a valid port number was entered, execution of the actual test will commence. If the test locks up, it is possible that the board being tested has a handshake problem which the diagnostic cannot recover from. If this happens, the RESET switch must be depressed. Execution at any other time can be terminated by typing "Control C". A "+" is returned for each good pass of the test. If there are any errors found in testing the interface, an error message (see ERROR MESSAGES) should appear. If no error mesage is returned then the interface checks out ok.

Important Note

This test operates by writing and reading variable data patterns between ports and is very good at detecting dropped bits or erroneous data transfers. This test will not always successfully detect certain interrupt problems or problems at particular data rates. This test should be used as an AID in uncovering certain hard to find transfer problems but should not be used as a complete checkout tool.

Error Messages

SERIAL INTERFACE ERRORS:

UNKNOWN SERIAL ERROR -- Error found in the keyswitch hold down latch or error in the RTS output.

DIAGNOSTIC LOOKUP ERROR -- Internal program failure caused by malfunctioning central processor, memory or disk.

INCORRECT TRANSFER FROM A PORT TO B PORT -- Error in data transfer from port A to port B.

INCORRECT TRANSFER FROM B PORT TO A PORT -- Error in data transfer from port B to port A.

FALSE STROBE CONDITION DETECTED -- Incorrect state of RTS output or hold down inputs incorrect.

ERROR FOUND IN MAKING A STROBE LOW -- RTS output incorrect on A side or hold down input incorrect on B side.

ERROR FOUND IN MAKING B STROBE LOW -- RTS output incorrect on B side or hold down input incorrect on A side.

OVERRUN OR FRAMING ERROR DETECTED (A PORT REC.) -- A character or a number of characters were received but not read from the Received Data Register prior to subsequent characters being received. A synchronization error, a faulty transmission or a break condition may also be the cause of this error message.

OVERRUN OR FRAMING ERROR DETECTED (B PORT REC.) -- A character or a number of characters were received but not read from the Received Data Register prior to subsequent characters being received. A synchronization error, a faulty transmission or a break condition may also be the cause of this error message.

```
               MP-S2 TEST CABLE AND CONFIGURATION
----------------------------------+
                                  |
                                 _|_         pin 20 -------+
     _                          | |                        |
    | |            A port (TOP) | |                        |
    | |                         | |                12 -------+
    | |                         | |
    | |                         | |                 2 ------------------------+
    | |  set top half           | |                 3 ---------------------+  |
    | |  for 9600 baud          | |                 8 ------------------+  |  |
    | |                         | |                19 ---------------+  |  |  |
    | |                         |_|                                  |  |  |  |
    |_|                          |                                   |  |  |  |
                                 |                                   |  |  |  |
                                 |            DB-25 connectors       |  |  |  |
                                 |                                   |  |  |  |
                                 |                                   |  |  |  |
                                 |                                   |  |  |  |
  _                             _|_                                  |  |  |  |
 | |                           | |          pin  8 ------------------+  |  |  |
 | |         B port (BOTTOM)   | |                19 ---------------------+  |  |
 | |                           | |                 2 ------------------------+  |
 | |                           | |                 3 ---------------------------+
 | |  set bottom half          | |
 | |  for 9600 baud            | |                12 -------+
 | |                           | |                          |
 | |                           | |                          |
 |_|                           |_|                20 -------+
                                |
                                |
____________________________    |
____________________________|---+
```

MPL2TEST

MPL2TEST is a diagnostic program that can be used by qualified technical personnel as an aid in testing or repairing a SWTPC MP-L2 parallel interface.

This test requires special jumper configurations and a special test cable. The board should be configured as shown on the configuration pages that follow. A diagram of the test cable is also given. Also, this test operates by taking over the computer from the operating system and should never be used in the multi user mode.

Test Procedure

After making the appropriate connections and hooking up the cables on the MP-L2, type "MPL2TEST" and return.

THIS TEST WILL NOT WORK IF THE CABLES ARE NOT HOOKED UP!!

The program will begin execution and the message "SWTPC MP-L2 Parallel Interface Diagnostic" will appear on the screen, followed by the prompt:

"Input the port number [1-7] "

At this time the port number of the interface to be tested should be entered. NOTE: Be sure that the MP-L2 test cable is installed.

If a valid port number was entered, execution of the actual test will commence. If the test locks up, it is possible that the board being tested has a handshake problem which the diagnostic cannot recover from. If this happens, the RESET switch must be depressed. Execution at any other time can be terminated by typing "Control C". A "+" is returned for each good pass of the test. If there are any errors found in testing the interface, an error message (see ERROR MESSAGES) should appear. If no error mesage is returned then the interface checks out ok.

Important Note

This test operates by writing and reading variable data patterns between ports and is very good at detecting dropped bits or erroneous data transfers. This test will not always successfully detect certain interrupt problems or problems at particular data rates. This test should be used as an AID in uncovering certain hard to find transfer problems but should not be used as a complete checkout tool.

Error Messages

PARALLEL INTERFACE ERRORS:

DIAGNOSTIC LOOKUP ERROR -- Internal program failure caused by malfunctioning central processor, memory or disk.

FALSE STROBE OR PIA INTERRUPT LOCKED HIGH -- False input strobe received or an internal problem exists in the 6820.

ERROR FROM A SIDE OUT1 TO CB1 -- Error in data transfer.

ERROR FROM A SIDE OUT2 TO CB2 -- Error in data transfer.

INCORRECT A TO B DATA TRANSFER -- Error in data transfer.

INCORRECT B TO A STROBE (OUT1 --> CA1) -- Error in handshake transfer.

INCORRECT B TO A STROBE (OUT2 --> CA2) -- Error in handshake transfer.

MP-L2 TEST CABLE AND CONFIGURATION

```
-----------------------+
                      _|_
                     | | |                pin 14 ---------------------------------+
        A port (TOP) | | |                    15 ------------------------------+  |
                     | | |                    16 ----------------------------+ |  |
                     | | |                    17 -------------------------+  | |  |
                     | | |                    18 -----------------------+ |  | |  |
                     | | |                    19 --------------------+  | |  | |  |
                     | | |                    20 ------------------+ |  | |  | |  |
                     | | |                    21 ---------------+  | |  | |  | |  |
                     |_|_|                    22 -------------+ |  | |  | |  | |  |
                       |                      23 -----------+ | |  | |  | |  | |  |
                       |                      24 ---------+ | | |  | |  | |  | |  |
                       |                      25 -------+ | | | |  | |  | |  | |  |
                       |                                | | | | |  | |  | |  | |  |
                       |          DB-25 connectors      | | | | |  | |  | |  | |  |
MP-L2                  |                and             | | | | |  | |  | |  | |  |
                       |          MP-L2 Test Cable      | | | | |  | |  | |  | |  |
                       |                                | | | | |  | |  | |  | |  |
                       |                pin 22 ---------+ | | | |  | |  | |  | |  |
                       |                    23 -----------+ | | |  | |  | |  | |  |
                      _|_                   24 -------------+ | |  | |  | |  | |  |
                     | | |                  25 ---------------+ |  | |  | |  | |  |
    B port (BOTTOM)  | | |                  21 -----------------+  | |  | |  | |  |
                     | | |                  20 --------------------+ |  | |  | |  |
                     | | |                  19 ----------------------+  | |  | |  |
                     | | |                  18 -------------------------+ |  | |  |
                     | | |                  17 ---------------------------+  | |  |
                     | | |                  16 ------------------------------+ |  |
                     | | |                  15 --------------------------------+  |
                     |_|_|                  14 -----------------------------------+
  jumper both sides    |
  for NO interrupts    |
                       |
                       |
________________       |
________________|------+
```

MPIDPATEST

MPIDPATEST is a diagnostic program that can be used by qualified technical personnel as an aid in testing or repairing the parallel output port on a SWTPC MP-ID board.

This test requires special jumper configurations and a special test cable. The board should be configured as shown on the configuration pages that follow. A diagram of the test cable is also given. Also, this test operates by taking over the computer from the operating system and should never be used in the multi user mode. NOTE: In order to use the test, an MP-L2 parallel interface board is required in the system.

Test Procedure

After making the appropriate connections and hooking up the cables on the MP-ID and the MP-L2, type "MPIDPATEST" and return.

THIS TEST WILL NOT WORK IF THE CABLES ARE NOT HOOKED UP!!

The program will begin execution and the message "SWTPC MP-ID Parallel Output Port Diagnostic" will appear on the screen, followed by the prompt:

"Input the port number of the MP-L2 [1-7] "

At this time the port number of the MP-L2 that will be used to check the MP-ID port should be entered.

If a valid port number was entered, execution of the actual test will commence. If the test locks up, it is possible that the board being tested has a handshake problem which the diagnostic cannot recover from. If this happens, the RESET switch must be depressed. Execution at any other time can be terminated by typing "Control C". A "+" is returned for each good pass of the test. If there are any errors found in testing the interface, an error message (see ERROR MESSAGES) should appear. If no error mesage is returned then the interface checks out ok.

Important Note

This test operates by writing and reading variable data patterns between ports and is very good at detecting dropped bits or erroneous data transfers. This test will not always successfully detect certain interrupt problems or problems at particular data rates. This test should be used as an AID in uncovering certain hard to find transfer problems but should not be used as a complete checkout tool.

Error Messages

MP-ID PARALLEL OUTPUT PORT ERRORS:

DIAGNOSTIC LOOKUP ERROR -- Internal program failure caused by a malfunctioning central processor, memory or disk.

FALSE STROBE DETECTED ON MP-ID INPUT -- The MP-ID received a signal (strobe) from the MP-L2 which should not have been sent.

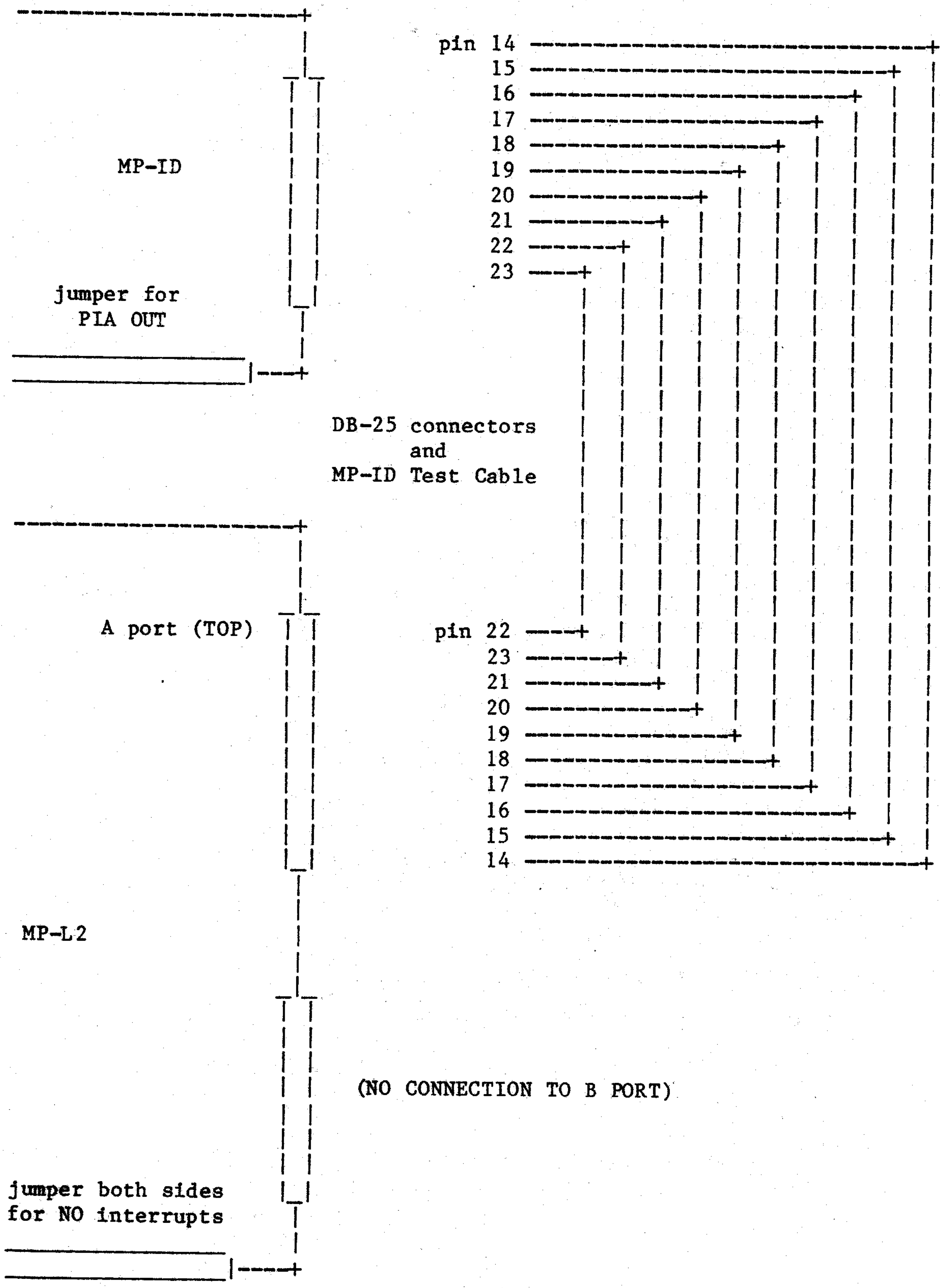
STROBE FORM MP-ID TO MP-L2 LOST -- Handshake signal output from MP-ID was never received at MP-L2.

FALSE STROBE DETECTED ON MP-L2 INPUT -- The MP-L2 received a signal (strobe) from the MP-ID which should not have been sent.

INCORRECT MP-L2 / MP-ID DATA TRANSFER -- One or more bits of data incorrectly transferred.

STROBE FROM MP-L2 TO MP-ID LOST -- Handshake signal output from MP-L2 was never received at MP-ID.

MP-ID Test Cable and Configuration


MP-ID
jumper for PIA OUT
pin 14
15
16
17
18
19
20
21
22
23
DB-25 connectors and MP-ID Test Cable
A port (TOP)
pin 22
23
21
20
19
18
17
16
15
14
MP-L2
(NO CONNECTION TO B PORT)
jumper both sides for NO interrupts

TIMETEST
(UniFlex Version)

The TIMETEST diagnostic can be used by the UniFlex system user or qualified technical personnel as an aid in testing or repairing the 6840 timer on the MP-ID board.

TIMETEST assumes a 1 or 2 MHz processor speed and stretched or non-stretched cycle memories.

PROCEDURE

NEEDED: One disk that contains the diagnostic program "TIMETEST".

NOTE: The system MUST BE in single user mode.

The diagnostic is executed by typing "timetest" and return. At this point, the diskette containing the program may be removed from the disk drive as it is not needed for the actual test. The message "6840 Timer Test Diagnostic 1% Version 1.0" is returned to the screen to identify the test. Several prompts appear on the screen asking for infomation needed to run the test. The first is "Enter Processor Speed in MHz [1 or 2]". Only if "2" is entered for the processor speed, will the next prompt "3509 Stretched Cycle Memories [y or n]?" appear on the screen. The test will begin execution for the 2 MHz processor after the second prompt has been answered.

After the prompts have been correctly answered, the message "--Checking Consistency of Timers" appears on the screen. When this test has been completed, the message "---> Timer Consistency Verified Good" appears on the screen if the test had no errors. If the test found an error, the message "--Inconsistency Found in Reported Timer Values" should appear on the screen and a string of numbers will be returned indicating an error in timer values (see documentation for CFTEST diagnostic). TIMETEST allows for only 1% error in timing measurements. The next message "Testing Timers --" appears on the screen . A "+" is returned to the screen for every good pass and an error message is returned if there exists an error (see ERROR MESSAGES). The test will continue until "Control C" is typed. When "Control C" is typed, the message "System Stopped" will appear on the screen and the system will have to be booted again.

ERROR MESSAGES

NOTE: The 3509 stretched cycle memory at 2 MHz may return the error message "ERROR FOUND IN TESTING INTERRUPT TIMER". If this should happen, CFTEST needs to be run to verify the error -- if there actually was an error.

ERROR FOUND IN TESTING INTERRUPT TIMER -- Timer 1 of the 6840 is not functioning properly or the input reference circuitry is malfunctioning.

ERROR IN TESTING TIMER 2 — A problem exists with timer 2 of the 6840.

ERROR IN TESTING TIMER 3 — A problem exists with timer 3 of the 6840.

CFTEST
(UniFlex Version)

The CFTEST diagnostic may be used by the UniFlex system user or by qualified technical personnel as an aid in testing or repairing the 6840 interrupt timer. The test is used to judge the consistency of the interrupt timer circuitry on an MP-ID circuit board.

PROCEDURE

NEEDED: One disk containing the program CFTEST.

NOTE: CFTEST can only be run on systems that have an MP-ID board. The system MUST BE in single user mode.

In order to run CFTEST, type "cftest" and return. The message "6840 Timer Consistency Diagnostic" should appear on the screen, followed by a series of numbers. All the numbers returned should be the same. If the numbers differ at all, then there exists an inconsistency in the interrupt timer. The test may be exited by typing "Control C". Upon typing "Control C", the message "System Stopped" will appear on the screen and the system will have to be re-booted.

The table below describes the different timer speeds and the series of numbers displayed to the screen.

NUMBER DISPLAYED TO THE SCREEN	PROCESSOR SPEED	MEMORY TYPE	POWER LINE FREQUENCY
01	2 MHz	non-stretched	50 Hz
02	2 MHz	stretched	50 Hz
03	2 MHz	non-stretched	60 Hz
04	2 MHz	stretched	60 Hz
05	1 MHz	non-stretched	50 Hz
06	1 MHz	non-stretched	60 Hz

Using CFTEST with processor speeds other than 1 MHz or 2 MHz can cause unpredictable results.

SYSTEM SUPPORT PROGRAMS

for UNIFLEX T.M.

DRWTST

Perform general read/write test.

SYNTAX

```
     drwtst <pathname> [+options]
```

DESCRIPTION

The DRWTST program performs a generalized read/write test on the file or device specified. During each pass, blocks are constructed with a known pattern and written to disk. The file system is updated to force UniFlex to actually write the blocks to disk, then each block in turn is read back in and the pattern verified. The pattern consists of a random 16-bit number, left rotated through 256 words in each block of the file or device. If an error occurs, the current block number is reported to standard output.

Since this program uses read/write drivers with retry capability, a properly working device should never show any errors. Three types of errors are detectable: First, write errors. These errors occur when drwtst is (for some reason) unable to write a block to the disk. Second are read errors, which indicate that the block being read shows an error. Third, and by far the most serious, are data errors. These errors indicate that the block was read with no error indication, but that the data it contains is wrong.

If the pathname specified is a block device, the read/write test begins at block 2 to preserve the super block and its density information. The file structure (if any) on the specified device is destroyed. If the specified name is a directory, or if no name is specified, a file called TEST.XXXXX (Where XXXXX=task number) is created and used for the test. This file is removed at the end of the test. By using this option, a block device can be mounted and tested, without destroying the file structure it contains. If a file name is specified, that file is used for the read/write test. The test will default to the current size of the file.

The options that can be specified are:

+b=nn	Specifies number of DISK BLOCKS (min 25)
+p=nn	Specifies number of PASSES (min 1)

Make a File System on a Device

SYNTAX

```
mkfs <device> <proto> [+<options>]
```

DESCRIPTION

The mkfs program creates an empty file system by writing on the specified device. The device must be a block device, and if it has removable media, that media must be in place and initialized. The mkfs program writes the UniFlex super block, FDN blocks, and free space chain on the specified device, then an empty root directory is constructed. If a prototype file is specified, it is used to initialize the contents of the file system. Note that the mkfs program will happily write over an existing file system, (destroying it) (totally!) so be certain that none of the data already on the device is significant. Options applicable to the mkfs program are:

+b=<boot name>	- specifies name of boot block
+f=<fdn space>	- specifies the number of FDN's
+k=<swap space>	- specifies size of swap space
+n=<blocks>	- specifies size of file system
+u=<uniflex>	- specifies name of UniFlex

<u>For floppy disks only</u>:

+d - specifies double density (FD only)
+s - specifies double sided (FD only)

The prototype file contains tokens separated by spaces or tab characters, and is organized into lines separated by newline characters. Each line of the prototype file specifies a file or directory to be present in the newly created file system. The first token in each line is the file name to be processed. If the file name consists of the single character "$", then this line is considered to be the end of a directory. The next token is a single character and designates what type of file the name is to be. The next token is two octal digits and specify the permissions for the output file. Following the permissions is the name or user number of the owner of the file. File type tokens are:

c - character device name
b - block device name
d - directory name (starts recursion)
f - ordinary file (data)
s - ordinary file, but set "s+" permission bit

If the file type token specifies a character or block device, the next two tokens should be decimal numbers specifying the major and minor device numbers, respectively. If the file type token specifies an ordinary file, the next token (if any) specifies the path name of the file to be copied to the output file system. If the file type token specifies a directory, this directory is created on the output file system, and the following lines (up to a line containing a single "$") specify files to be contained in this directory. It is permitted to recursively define output directories. The following prototype file will serve as an example:

```
uniflex  f 70 0 /uniflex
etc        d 75 0
    log        d 75 0
        password f 64 0 /etc/newpass
        $
    init       s 75 root /etc/init
    login      s 75 root /etc/login
    ttylist    f 64 root /etc/newttys
    $
bin        d 75 bin
    shell        f 75 /bin/shell
    $
act        d 75 bin
    jobacct   f 60 root
    history   f 60 root
    $
tmp        d 77 bin
    $
dev        d 77 root
    tty00      c 66 root 0 0
    tty01      c 66 root 0 1
    fd0        b 64 root 0 0
    $
```

The file owner token may either be a decimal number or a character string of up to eight characters. If a number is specified, this number is used as the owner id of the specified file. If a character string is specified, this is used as the name of the owner. The password file contained on the <u>output</u> file system is searched to determine the owner id number. The two permission digits specify read, write, and execute permission for the file's owner, and for others, respectively. For example, permissions of "77" specify read, write, and exec permission for everybody, while permissions of "60" specify read and write permission for only the file's owner. Remember that for directories, "execute permission" means "permission to search a pathname".

If a file system produced by mkfs is to be bootable, a boot block must be specified. The name of the boot block is specified through the use of the "+b=<boot name>" option. If the specified boot block file cannot be located in the current directory, then the mkfs program searches the directory "/etc/boots". Once the file is located, it is read and verified (as much as possible), then written as block zero on the specified device. This block contains the initial program load (IPL) for UniFlex and is read into memory by the ROM boot routine. Remember that these IPL programs are different for various configurations of the UniFlex system. In particular, the IPL programs supplied with Version 1 and Version 2 UniFlex are not compatable. The following table details the boot block files normally contained in the directory "/etc/boots" and the configuration in which they should be used:

Boot Name	Processor	Disk Unit
cds	MP-09	CDS-1, CDS-2, CDS-3, CDS-4
dmf2	MP-09	DMF2, QUME or REMEX drives
dmf2C	MP-09	DMF2, CALCOMP or SHUGART
dmf3	MP-09	DMF3, QUME drives
win1	MP-09	DMF3 5.25" Winchester Unit
cds-a	MPU-1	CDS-1, CDS-2, CDS-3, CDS-4
dmf2-a	MPU-1	DMF2, QUME or REMEX drives
dmf3-a	MPU-1	DMF3, QUME drives
win1-a	MPU-1	DMF3 5.25" Winchester Unit

By default, the mkfs program computes the number of FDN's to be generated as approximately ten percent of the number of blocks on the file system. This number is generally correct, i.e., the file system runs out of FDN's and free space at approximately the same time. For certain applications, it may be desirable to explicitly specify the number of FDN's to be created. The mkfs option "+f=<fdn's>" allows this specification. Note that this number is the actual number of FDN's, not the number of fdn blocks.

If the generated file system is to contain swap space, the "+k=<swap space>" option should be used to reserve space for swapping operations. Note that if the +b option has been specified, the swap space will default to 16K since a minimum amount of swap space is required for booting. The size of the swap space is specified in kilobytes, and should be a multiple of four. A good rule of thumb is to assign 96K of swap space for each attached terminal, with a recommended minimum of 200K. If insufficient swap space is allocated, the message "Swap Space!" is issued and the UniFlex system stops. Remember that assigning more swap space than is actually used in a system does not improve performance and reduces the size of the available data space. Swap space is placed at the end of the media, past the area that contains the active file system.

The mkfs program automatically determines the size of the device (number of blocks) unless this size is explicitly specified. By using the "+n=<blocks>" option, a smaller file system may be constructed. The maximum size for a UniFlex file system is 8,388,607 blocks. Of course, the file system must _fit_ on the designated device media. For floppy disks, density and side information must be specified through use of the "d" and "s" options. The mkfs program will default to single sided, single density. If the density and side flags are incorrectly specified, the floppy disk media will be incorrectly initialized. Remember that the mkfs program does not actually format the media.

On a bootable file system, the boot block contains the name of the operating system to be booted. The IPL program searches the root directory of the file system for the operating system file. The name of this file is "uniflex" by default, but may be changed in certain instances. For example, the current TSC distribution provides a UniFlex with CDS root, swap, and pipe devices, and this UniFlex is called "uniflex1". In this case, the UniFlex name would be set by the option "+u=uniflex1". Then when making a bootable CDS, the "uniflex1" file should be copied to the CDS unit and installed.

Write a boot block onto a device.

SYNTAX

```
     putboot <device> <boot file>
```

DESCRIPTION

The putboot program reads the boot file and verifies (as well as it can) that it represents a valid boot program. The boot block is then written to block zero of the specified device. Putboot searches for the boot file at the specified path name, and if not found, then searches the directory "/etc/boots". The boot blocks normally supplied in this directory are detailed below:

Boot Name	Processor	Disk Unit
cds	MP-09	CDS-1, CDS-2, CDS-3, CDS-4
dmf2	MP-09	DMF2, QUME or REMEX drives
dmf2C	MP-09	DMF2, CALCOMP or SHUGART
dmf3	MP-09	DMF3, QUME drives
win1	MP-09	DMF3 5.25" Winchester Unit
cds-a	MPU-1	CDS-1, CDS-2, CDS-3, CDS-4
dmf2-a	MPU-1	DMF2, QUME or REMEX drives
dmf3-a	MPU-1	DMF3, QUME drives
win1-a	MPU-1	DMF3 5.25" Winchester Unit

```
chd /etc
putboot   boots/cds  /dev/hd0
```

READ

Read a file or device and report errors.

SYNTAX

```
     read <pathname> [ <pathname> . . . ]
```

DESCRIPTION

The read utility checks for errors in the files or devices specified. Errors are reported by path name and block number. For ordinary files, the byte count of the file is given at end of file. For devices, the block count of the device is given.

For example, suppose a user (Jerry, in this case) suspects that there are errors in a file in which he keeps back-order log sheets (bool-sheet). He can use the read program to check the file:

```
        ++ read /usr/jerry/bool-sheet
```

The above example will read the file "/usr/jerry/bool-sheet" and report the total number of bytes and any information about read errors found in the file. Similarly, if a programmer suspects errors or contamination on a floppy disk, he can use the read program to read an entire floppy disk:

```
        ++ read /dev/fd2
```

The above example will read the device "/dev/fd2" and report the number of blocks and any read errors found while reading the device. Note that an error will occur when "reading off the end of the disk", i.e. READ ERROR at block 2464. The correct sizes for various devices is detailed below:

DMF2/3 SS/SD — 616 Blocks	CDS-0 M-10 — 16,400 Blocks
DMF2/3 SS/DD — 1232 Blocks	CDS-1 M-20 — 32,800 Blocks
DMF2/3 DS/SD — 1232 Blocks	CDS-2 M-40 — 65,600 Blocks
DMF2/3 DS/DD — 2464 Blocks	CDS-3 M-80 — 141,120 Blocks
	CDS-4 M-160 — 281,232 Blocks
DMF3-504 — 7276 Blocks	
DMF3-509 — 14,552 Blocks	
DMF3-513 — 21,828 Blocks	
DMF3-518 — 29,104 Blocks	

Perform Memory Diagnostics

SYNTAX

    testmem

DESCRIPTION

The TESTMEM program performs several different memory diagnostics over the main memory of a UniFlex system. The UniFlex system must have a 6116 RAM installed on the MP-09 CPU board (see application note #139A), and a CT-82 or 8200 series terminal. This test must be run by the system manager in single user mode. The test is designed to run unattended, and in normal circumstances, should be left running overnight if memory problems are suspected.

When the testmem program is run, it updates any active file systems and then takes control of the computer from UniFlex. Since the test destroys all of the contents of main memory, the UniFlex operating system cannot continue to run at the same time. Once running, the diagnostic maintains a physical memory segment map on the right side of the screen, and an error summary on the left. Status messages detail the progress of each test, and what type of diagnostic procedure is currently in use. Cumulative error statistics are shown, along with an indication of which physical pages have shown errors in the past. The format of the testmem screen is shown in Figure 1.

```
                -- Pathological Case Memory Test 2.1:4 --

                                              ----- Physical Memory Map -----
   CPU: 2 MHz
Memory: 256 K       AVL Downcount          0  * * * * * * * * . . . . . . . .
  Pass:     3                              1  * * * * X X X X . . . . . . . .
Errors:     1       Writing Memory         2  * * * * * * * * . . . . . . . .
                                           3  * * * * * * * * . . . . . . . .
----- Failed Memory Locations ------       4  * * * * * * * * . . . . . . . .
Pass Errs Addr Bad Bits  Last Err Cl       5  * * * * * * * * . . . . . . . .
  1.  12  03F7 ----11-- ----0----  Y       6  * * * * * * * * . . . . . . . .
                                           7  * * * * * * * * . . . . . . . .
                                           8  . . . . . . . . . . . . . . . .
                                           9  . . . . . . . . . . . . . . . .
                                           A  . . . . . . . . . . . . . . . .
                                           B  . . . . . . . . . . . . . . . .
                                           C  . . . . . . . . . . . . . . . .
                                           D  . . . . . . . . . . . . . . . .
                                           E  . . . . . . . . . . . . . . . .
                                           F  . . . . . . . . . . . . . . . .
```

Figure 1.

As the memory test runs, the cursor character is continuously displayed on the physical memory map, underneath the page that is currently being tested. The line labeled "Memory:" on the testmem screen shows the amount of memory that the test thinks is in the system. If there are permanent errors in the memory, this value may be less than the actual amount of memory installed. In this case, the physical memory map will reveal the address of the missing memory. The pass counter is incremented each time one of the memory tests completes. In order to run all eight tests, an absolute minimum of eight passes must be run. The error counter reflects the number of passes that had errors. The actual failure counts are shown in the last error summary.

The last error summary is located in the lower left hand corner of the screen, and scrolls down with each reported error. For this error, the pass number is reported in the column labeled "Pass". In addition, the current memory page is marked with an "X" character in the physical memory map. The total number of errors located this pass is reported in the column labeled "Errors". In the column "Addr" is the address of the last error that was detected in a particular memory page. This address is a full 20-bit physical address. The "Bad Bits" column shows which of the eight data bits were found to be in error, and is cumulative for all bit errors in a memory page. Bits found bad are marked with a "1" character. In the column labeled "Last Err", the pattern of the last error is shown. Bit positions containing a hyphen were correct, those containing a "0" or a "1" were wrong, and the character shown reflects what was actually read from the memory. Note that the "Last Err" field occurred at the address reported in the "Addr" column. Finally, the column labeled "Clr" reports whether the memory test was able to clear the error by writing back the apropriate data.

Memory Circuit Architecture

MOS memory circuits may be classified as either static or dynamic. Static memories typically use two or three transistors per memory bit, and these transistors form a stable circuit element (essentially an RS flip-flop) that will hold a particular value indefinitely. When a static memory is read, the data in a particular cell is simply gated to an output line. Notice that reading a static memory does not involve changing the data held in any particular cell.

Dynamic memories normally use one transistor per memory cell, and hold actual bit value as a charge in a capacitor. The transistor is used to dump the capacitor's charge into a sense line, and to store a new charge value when the bit cell is written. Dynamic memories are three to five times as dense as static memories and require less power to operate. However, dynamic memories require that each bit cell be periodically rewritten to retain data.

When a dynamic memory is read, the bit cell capacitor charge is dumped to a row line leading to a sense amplifier. The output from this amplifier (which reflects that particular bit's value) is gated to an output line, and is also used to write the data back into the capacitor. This rewriting is termed refresh. In most dynamic memories, an entire row of data is refreshed by reading (or writing) any single bit in that row. It is possible (and usual) to refresh a memory without having any data presented on its output lines. Typical memory units require that each memory row (of 128 or 256 bits) be refreshed every two to four milliseconds. The length of time that a bit cell will hold its value without refresh depends upon memory circuit temperature, and is significantly longer at low temperatures. Suspected refresh problems may be aggrivated by heating the memory board.

Classification of Failures

Board-level memory systems display eleven distinct types of failures. The testmem program has been designed to check for all eleven cases. Some of these cases are specific to dynamic memories, while others apply to both static and dynamic memories. Most of the memory boards currently in use are dynamic memories. The eleven types of memory failures are detailed below:

1. Intersitial Interference Errors. These errors occur when a memory cell changes value when one of the immediately adjacent memory cells is written. These errors are typically pattern sensitive and may manifest themselves only after extended testing of a memory part.

2. Charge Leakage Problems (Short Refresh). Since data held in dynamic memories consists of very small charges contained in very small capacitors, the data will eventually be lost as the charge leaks off. Manufacturers test memory IC's to be sure that this leakage is sufficiently small that the memory data is held at least through one refresh time, plus a safety margin. As the memory part ages the leakage current tends to increase somewhat. A very small percentage of parts (less than .004 percent) will leak to such an extent that normal refresh is unable to retain memory data in all cases.

3. Column and Row Driver Failures. Memory circuits are written by applying a voltage pulse to the memory array through column and row drivers. Failures in these drivers causes unwritable memory cells. These errors manifest themselves as non-clearable errors at random addresses within a memory segment.

4. Sense Amplifier Failures. When data is read in a dynamic memory, a very small capacitor charge is dumped onto a column line and is subsequently amplified by a column sense

amplifier. The signals are typically a few millivolts and must be amplified for use. Sense amplifier failures manifest themselves as sporadic bit read failures at a particular pattern of addresses.

5. Mobile ION Problems. Because of the extremely small geometry of the capacitor used to hold dynamic memory data, a single positively charged ION (Typically sodium, Na+) can incapacitate a bit cell. This failure shows up after a few hundred or thousand hours of in-service time. This error is permanent and is the typical cause of failure of a dynamic memory part.

6. Ionizing Radiation Failures (Alpha Particles). With the advent of 64K and 256K dynamic memories, bit cell geometry has become so small that energetic alpha particles present in normal background radiation (from the decay of potassium-40) can generate sufficient secondary electrons to corrupt the charge in one or more memory capacitors. Most materials have very high cross sections for alpha particles of this energy, thus they do not penetrate very far into most materials. Semiconductor manufacturers are coating their silicon chips with various plastics (which do not contain potassium) to separate them from packaging materials (which do contain potassium). In this way, the chip is "insulated" from alpha particles.

7. Refresh Failure (Normal State). Dynamic memory boards keep their memories refreshed by a combination of timers and counters. The purpose of this circuitry is to assure that each row of every memory IC is read within the refresh time which is typically two to four milliseconds. Failure of this circuitry will not necessarily cause the memory board to stop functioning altogether.

8. Refresh Failure (Forced Refresh). Under normal operating conditions, sufficient non-VMA (Valid Memory Address) bus cycles are provided by the processor or DMA to permit memory refresh to occur on a non-interference basis. However, if an insufficient number of "dead" cycles are presented, memory boards will take steps to insure that the memory remains refreshed. SWTPC M-64 and M-256 boards can halt the processor, if necessary, to force the generation of dead cycles. Since this circumstance never occurs under typical operating conditions, this circuitry may fail and the board will appear to operate normally.

9. Address Buffer Failures. System bus addresses are buffered and latched before presentation to the memory circuits. If these address buffers fail, the memory board will develop address convergence errors. That is, one physical memory cell may respond at more than one logical memory address. This problem manifests itself as the failure of "blocks" of

memory locations within each memory segment. Block error counts will normally be close to a power of two, for example, a consistent block count of 118 to 132 suggests a failure of the A5 address input line.

10. Data Buffer Failures. System bus data is passed through data buffers while being written or read from the memory circuits themselves. Failure of data buffers is reflected as "missing" memory, i.e., memory that should respond to a physical bus address and simply does not appear to exist. This is shown by the physical memory map displayed by the testmem program.

11. Miscellaneous. There are several other board level problems that may be revealed by the testmem program. Among these are socketing problems on the memory IC's, power regulator failures, shorts, connector problems, and the like. If a system shows what seems to be memory related failures and no problems are reported by the testmem program, it is strongly suggested that the system be subjected to a vibration test while the testmem program is running. This is conveniently done by using a vibrator-massager to shake each memory board in turn.

Memory Diagnostics

Eight types of memory diagnostics are run by the testmem program. Each one is designed to detect specific types of memory failures; thus the many different tests complement each other to provide an extremely reliable memory function test. The eight different tests are detailed below:

Test #1 - AVL UPCOUNT

This is an Address Variable (AVL) test that uses ascending memory addresses to generate a hash value for each memory cell. The hash value is regenerated from the addresses during the checking phase and compared against the contents of memory. This diagnostic is most effective at locating errors of types 1, 3, 9, and 10.

Test #2 - AVL DOWNCOUNT

This is an Address Variable (AVL) test that uses decending memory addresses to generate a hash value for each memory cell. The hash value is regenerated from the addresses during the checking phase and compared against the contents of memory. This diagnostic is most effective at locating errors of types 1, 3, 9, and 10.

Test #3 - AVL WITH REFRESH

This is an Address Variable (AVL) test that uses ascending memory addresses and a memory row refresh counter to generate a hash value for each memory cell. After the memory has been written, a delay interval is added (in which there are no memory accesses) to test dynamic memory refresh circuitry. The hash value is regenerated from the addresses and refresh counter during the checking phase and compared against the contents of memory. This diagnostic is most effective at locating errors of types 2, 5, and 7.

Test #4 - PRN UPCOUNT

This is a Pseudo-Random Number (PRN) test that writes memory in ascending address order. The Pseudo-Random Number Generator is a linear congruential generator that has its results inverted every other test to provide for both incrementing and decrementing pseudo-random numbers. This diagnostic is most effective at locating errors of types 3, 4, and 10.

Test #5 - PRN DOWNCOUNT

This is a Pseudo-Random Number (PRN) test that writes memory in decending address order. The Pseudo-Random Number Generator is a linear congruential generator that has its results inverted every other test to provide for both incrementing and decrementing pseudo-random numbers. This diagnostic is most effective at locating errors of types 3, 4, and 10.

Test #6 - ROTATING BIT

This test alternately rotates a single bit (either one or zero) through the entire memory structure and validates the results at each step. Because each cell must be verified nine times per verification pass, this test is rather slow. This diagnostic is most effective at locating errors of types 1, 4, and 11.

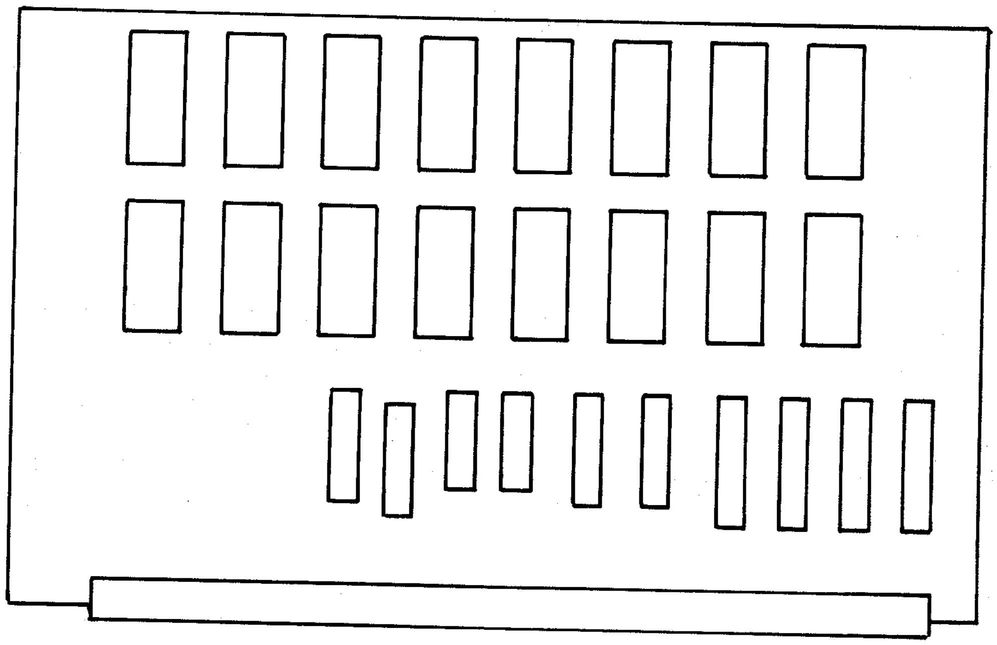
Test #7 - DATA RETENTION

This test alternately writes zeros or ones in all memory, then enters a delay cycle in which there are no memory accesses. The memory is then verified to insure that data has been retained. This diagnostic is most effective at locating errors of types 5, 6, and 7.

Test #8 - DEAD CYCLE HALT

This test writes a known pattern into each page of memory in turn. It then uses a low-dead-cycle loop to force the M-64 or M-256 boards into a condition called refresh starve. The boards will then HALT the processor in order to perform a burst refresh. During the Refresh Halt portion of the test, these boards should turn their HALT LED on, indicating the refresh starve condition. Data is then verified in each memory page. This test is designed to specifically detect class 8 errors.

Southwest Technical S-32 Memory Board



The S-32 Memory board uses CMOS 2K x 8 Static RAMS like Hitachi 6116's. Since each memory chip contains all eight bits of a data word, the bits-in-error field cannot help locate the faulty part. The RAM's on the board are arranged in pairs, to make 4K pages. Note that static RAM's cannot have refresh problems. The RAM chips are in ascending address order, from the top right hand corner proceeding left and down. See the documentation on the S-32 board.

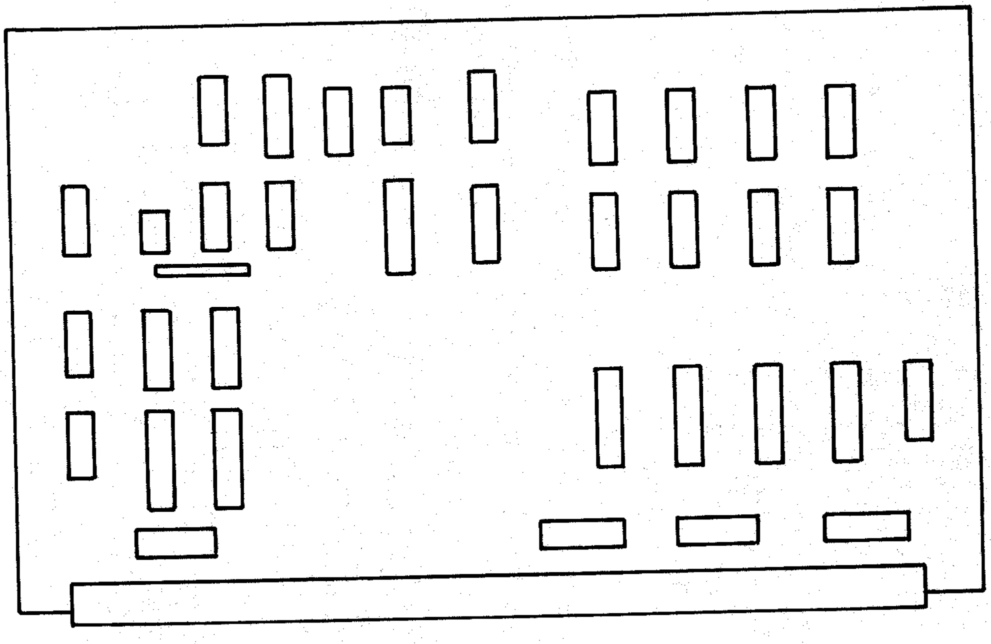
This is the memory map for a SWTPC S-32 memory board configured at position zero, with both upper and lower 16K enabled. Remember that ROM's will not appear in the physical memory map.

Jumper	Address Range
0	00000-07FFF
1	10000-17FFF
	- - -
6	60000-67FFF
7	70000-77FFF

Max Memory Configuration: 256K

```
  ----- Physical Memory Map -----
0 * * * * * * * * . . . . . . . .
1 . . . . . . . . . . . . . . . .
2 . . . . . . . . . . . . . . . .
3 . . . . . . . . . . . . . . . .
4 . . . . . . . . . . . . . . . .
5 . . . . . . . . . . . . . . . .
6 . . . . . . . . . . . . . . . .
7 . . . . . . . . . . . . . . . .
8 . . . . . . . . . . . . . . . .
9 . . . . . . . . . . . . . . . .
A . . . . . . . . . . . . . . . .
B . . . . . . . . . . . . . . . .
C . . . . . . . . . . . . . . . .
D . . . . . . . . . . . . . . . .
E . . . . . . . . . . . . . . . .
F . . . . . . . . . . . . . . . .
```

Southwest Technical M-64 Memory Board



The M-64 Memory board uses NMOS 64K x 1 dynamic RAMs like Hitachi 4864-3's or TMS 4164's. The maximum access time on these RAMs is 200 nSec., with a maximum cycle time of 375 nSec. Refresh is implemented with an 8-bit row counter and a refresh cycle is conducted on every bus cycle that does not access the memory board. A fail-safe timer is used to halt the processor if necessary to guarentee refresh. For bit definitions, see the documentation on the M-64 memory board.

This is the memory map for a SWTPC M-64 memory board configured at slot zero, position zero.

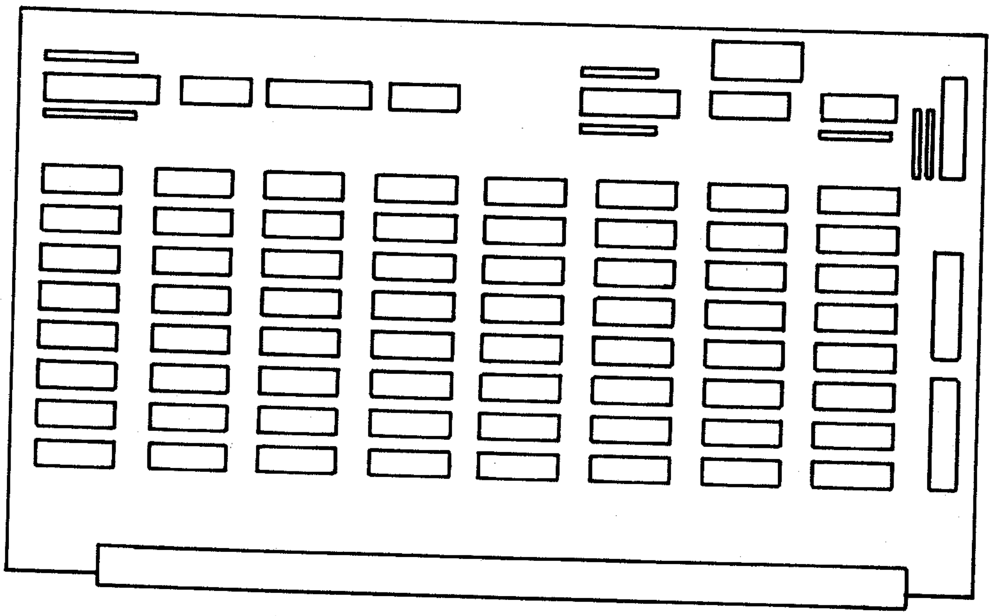
Switches 0 and 1		Switches 2 and 3	
00	0xxxx-3xxxx	00	x0000-x3FFF
01	4xxxx-7xxxx	01	x4000-x7FFF
10	8xxxx-Bxxxx	10	x8000-xBFFF
11	Cxxxx-Fxxxx	11	xC000-xFFFF

Each board responds in four groups of 16K bytes, for a total of 64K bytes of memory.

Max Memory Configuration: 512K

```
  ----- Physical Memory Map -----
0 * * * * . . . . . . . . . . . .
1 * * * * . . . . . . . . . . . .
2 * * * * . . . . . . . . . . . .
3 * * * * . . . . . . . . . . . .
4 . . . . . . . . . . . . . . . .
5 . . . . . . . . . . . . . . . .
6 . . . . . . . . . . . . . . . .
7 . . . . . . . . . . . . . . . .
8 . . . . . . . . . . . . . . . .
9 . . . . . . . . . . . . . . . .
A . . . . . . . . . . . . . . . .
B . . . . . . . . . . . . . . . .
C . . . . . . . . . . . . . . . .
D . . . . . . . . . . . . . . . .
E . . . . . . . . . . . . . . . .
F . . . . . . . . . . . . . . . .
```

Motorola SMS-3509 Memory System



The SMS3509 memory system uses NMOS 16K x 1 dynamic RAMs like Motorola MCM4116A25's. The maximum access time on these parts is 250 nSec, with maximum cycle time of 425 nSec. Refresh is implemented using a 7-bit PMFUR with access delay. For this reason, every memory access will be a stretched cycle. The memory array is set up so that bit zero is at the bottom of the board with bit seven at the top, and row zero is at the right with row seven at the left.

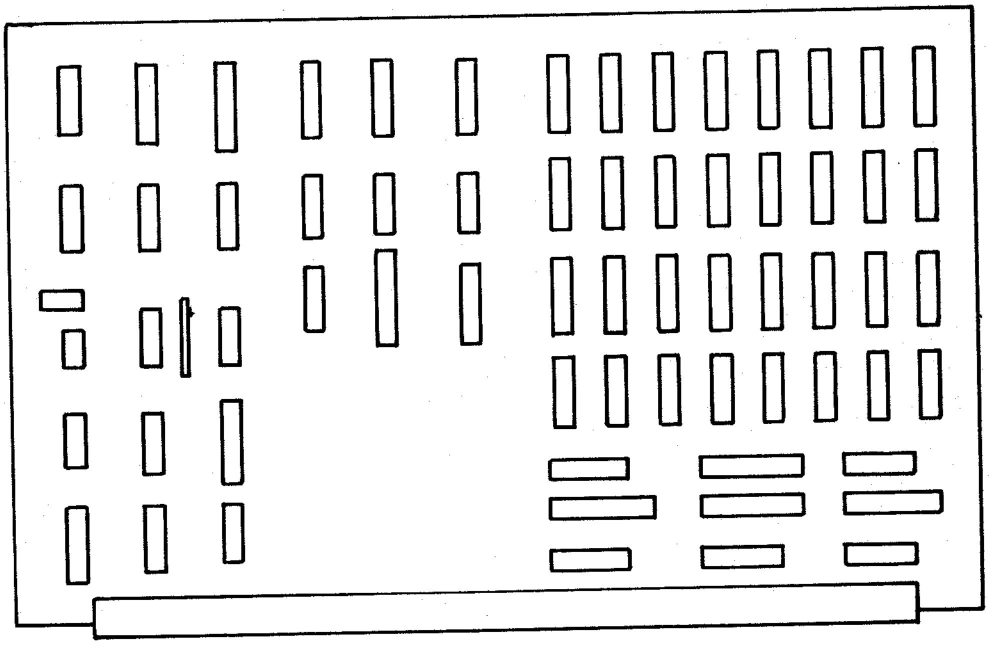
This is the memory map for an SMS-3509 memory array with 128K of memory. Additional array boards of 128K each may be added up to a total of three.

Array 1	Array 2	Array 3
00000-03FFF	04000-07FFF	08000-0BFFF
10000-13FFF	14000-17FFF	18000-1BFFF
- - -	- - -	- - -
60000-63FFF	64000-67FFF	68000-6BFFF
70000-73FFF	74000-77FFF	78000-7BFFF

Max Memory Configuration: 384K

```
  ----- Physical Memory Map -----
0 * * * * . . . . . . . . . . . .
1 * * * * . . . . . . . . . . . .
2 * * * * . . . . . . . . . . . .
3 * * * * . . . . . . . . . . . .
4 * * * * . . . . . . . . . . . .
5 * * * * . . . . . . . . . . . .
6 * * * * . . . . . . . . . . . .
7 * * * * . . . . . . . . . . . .
8 . . . . . . . . . . . . . . . .
9 . . . . . . . . . . . . . . . .
A . . . . . . . . . . . . . . . .
B . . . . . . . . . . . . . . . .
C . . . . . . . . . . . . . . . .
D . . . . . . . . . . . . . . . .
E . . . . . . . . . . . . . . . .
F . . . . . . . . . . . . . . . .
```

Southwest Technical M-256 Memory Board



The M-256 memory board uses NMOS 64K x 1 RAMs like MCM6665 or TMS4164's. The parts must have a maximum access time of 200 Nsec and a maximum cycle time of 375 Nsec. Refresh is implemented using an 8-bit row counter and a 1.5 mSec power reduction timer. A refresh cycle is conducted on each bus cycle that is not a board access until the entire array is refreshed. Refresh is then suspended until the 1.5 mSec timer starts a new refresh group. See the M-256 documentation.

This is the memory map for a SWTPC M-256 memory board jumpered into slot zero with all banks on.

Slot	Addresses	Bank	Addresses
0	x0000-x3FFF	0	0xxxx-3xxxx
1	x4000-x7FFF	1	4xxxx-7xxxx
2	x8000-xBFFF	2	8xxxx-Bxxxx
3	xC000-xFFFF	3	Cxxxx-Fxxxx

From one to four of the 64K banks may be enabled at any one time. The banks are switched selected.

Max Memory Configuration: 960K

```
   ----- Physical Memory Map -----
0  * * * * . . . . . . . . . . . .
1  * * * * . . . . . . . . . . . .
2  * * * * . . . . . . . . . . . .
3  * * * * . . . . . . . . . . . .
4  * * * * . . . . . . . . . . . .
5  * * * * . . . . . . . . . . . .
6  * * * * . . . . . . . . . . . .
7  * * * * . . . . . . . . . . . .
8  * * * * . . . . . . . . . . . .
9  * * * * . . . . . . . . . . . .
A  * * * * . . . . . . . . . . . .
B  * * * * . . . . . . . . . . . .
C  * * * * . . . . . . . . . . . .
D  * * * * . . . . . . . . . . . .
E  * * * * . . . . . . . . . . . .
F  * * * * . . . . . . . . . . . .
```

FDBUG

FDBUG
(UniFlex Version)

Disclaimer

*UniFlex is a trademark of Technical Systems Consultants

FDBUG - FLOPPY DISK DIAGNOSTIC PROGRAM

Table of Contents

FDBUG

FDBUG is a diagnostic tool designed to assist qualified technical personnel in finding and rectifying malfunctions that may occur in DMF3 floppy disk units.

INSTALLATION PROCEDURE

REQUIRED: A diskette containing a bootable UniFlex system.
An FDBUG release diskette.
A DMF2 or DMF3 / 8 inch floppy disk combo.

To install the FDBUG diagnostic program, boot UniFlex, then mount the disk containing the program "FDBUG", and copy the FDBUG program from the release disk. It is suggested that the FDBUG program be placed in a directory called "/etc/diagnostics". The FDBUG program should have owner SYSTEM, with no execute permission for others. Due to the potentially destructive nature of this test, it is advised that this program not be placed in the execution directories "/bin" and "/usr/bin". An example of installing the FDBUG program is:

```
++ /etc/mount /dev/fdl /usr2                 (mount FDBUG disk on fdl)
++ crdir /etc/diagnostics                    (make diagnostic directory)
++ copy /usr2/fdbug /etc/diagnostics         (move onto UniFlex disk)
++ owner system /etc/diagnostics/fdbug       (assign SYSTEM as owner)
++ perms o-rwx /etc/diagnostics/fdbug        (remove exec permission)
++ /etc/unmount /dev/fdl                     (unmount the FDBUG disk)
```

At this point, the FDBUG program may be run by typing "/etc/diagnostics/fdbug" and return. The FDBUG program will update all file systems, then begin execution. You must be using a SWTPC CT-82 or 8200 series terminal.

NOTE: The diagnostic program "FDBUG" moves in and takes over the entire UniFlex operating system. Since the UniFlex routines are destroyed in the process, the only way to terminate the FDBUG program is to RESET the computer. This effectively takes the system down in so far as normal usage is concerned. This implies that the "FDBUG" program must not be run on a system in multi-user mode.

FDBUG DISPLAY FORMAT

All of the FDBUG functions operate in frame mode, i.e., the terminal screen does not scroll. Some functions generate more than one page of data, and provide mechanisms for changing from one page to another. When the FDBUG program is running, the screen will always show a header line and status information as follows:

```
         "-- FDBUG FLOPPY DISK DEBUG PROGRAM - VERSION X.X --"

COMMAND:|_|           BLOCK:  00000 HD/CYL/SEC:  00 0000 00 STATUS:  00 00
         |                                        |    |   |          |  |
         |                                        |    |   |          |  |
         +-- CURSOR         HEAD/DENSITY/DRIVE -----+    |   |          |  |
                                CYLINDER ---------------+   |          |  |
                                     SECTOR ---------------+          |  |
                                                                      |  |
                                          TYPE I STATUS --------------+  |
                                                                         |
                                          TYPE II STATUS ----------------+
```

"COMMAND" is followed by the cursor, indicating that the program is waiting for the next command string to be entered. Commands consist of strings of one to twenty-one characters, and are terminated by the return key.

"BLOCK" is the current block number. The current block is set by several commands, and may be incremented or decremented by using the "+" or "-" commands. Commands that operate upon disk media typically use the current block number for their manipulations.

"HD/CYL/SEC" "HD" indicates the actual disk head, density and drive number. "HD" can be interpreted by the following chart:

```
                        BIT NUMBERS
                  7  6  5  4  3  2  1  0
                  |  |  |  |  |  |  |  |
   unfdefined ---+  |  |  |  |  |  |  +--- drive 0
   density ---------+--+  |  |  |  +------ drive 1
   head 1 ----------------+  |  +--------- drive 2
                             +------------ drive 3
```

"STATUS" shows the status of the floppy disk controller. Two sets of hexadecimal numbers are shown. The first number is the 8-bit status returned after a TYPE I command has been executed. The second number is the 8-bit status returned after a TYPE II command has been executed. A detailed explanation of these status bytes follows.

FLOPPY DISK CONTROLLER STATUS

The TYPE I status is an 8-bit value which shows the ending status after a TYPE I command has been executed. In order to determine which bits are set, the hexadecimal number must be converted to binary. For example, if the status was "50", then by converting the hexadecimal number to binary, the number would be "01010000". The binary number "01010000" has bits 6 and 4 set which may be interpreted by looking at the chart below. (WRITE PROTECTED and SEEK ERROR)

TYPE I STATUS BITS

```
                                    _______________________________
                                   | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
                                     |   |   |   |   |   |   |   |
BIT NUMBER                           |   |   |   |   |   |   |   |
                                     |   |   |   |   |   |   |   |
7 -- NOT READY ----------------------+   |   |   |   |   |   |   |
6 -- WRITE PROTECTED --------------------+   |   |   |   |   |   |
5 -- HEAD ENGAGED ---------------------------+   |   |   |   |   |
4 -- SEEK ERROR ---------------------------------+   |   |   |   |
3 -- CRC ERROR --------------------------------------+   |   |   |
2 -- TRACK 0 --------------------------------------------+   |   |
1 -- INDEX --------------------------------------------------+   |
0 -- BUSY -------------------------------------------------------+
```

SUMMARY OF TYPE I STATUS BITS:

NOT READY -- This bit, when set indicates the drive is not ready. No diskette is installed, the door is not closed, or a non-existent drive is specified.

WRITE PROTECTED -- Writing cannot be completed because the diskette is write protected.

HEAD ENGAGED -- When set, the bit indicates the head is loaded and engaged.

SEEK ERROR -- When set, the desired track was not verified. This bit is set to 0 when updated.

CRC (Cyclic Redundancy Check) ERROR -- The CRC "checksum" read at the end of an ID or data field did not agree with the data read off the disk. One or more bits of the data read will be invalid.

TRACK 0 -- When set, indicates Read/Write head is positioned to Track 0.

INDEX -- When set, indicates index mark detected from drive.

BUSY -- When set, a command is in progress.

TYPE II STATUS BITS

```
                            _______________________________________
                           | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
                             |   |   |   |   |   |   |   |
    BIT NUMBER               |   |   |   |   |   |   |   |
                             |   |   |   |   |   |   |   |
    7 -- NOT READY ----------+   |   |   |   |   |   |   |
    6 -- RECORD TYPE / WRITE PROT --+   |   |   |   |   |   |
    5 -- DATA COMPARE ------------------+   |   |   |   |   |
    4 -- RECORD NOT FOUND ------------------+   |   |   |   |
    3 -- CRC ERROR -----------------------------+   |   |   |
    2 -- LOST DATA ---------------------------------+   |   |
    1 -- DRQ -------------------------------------------+   |
    0 -- BUSY --------------------------------------------------+
```

SUMMARY OF TYPE II STATUS BITS:

NOT READY -- Indicates the drive is not ready. TYPE II commands will not execute unless the drive is ready.

RECORD TYPE / WRITE PROT -- Indicates the record-type code from data field address mark or a write protect.

DATA COMPARE -- Data read was not the same as the data written or data previously written was not the same as the data read. (Used in diagnostic mode only)

NOTE: This bit is software generated by the diagnostic program and is not a part of the actual status bytes returned by the disk controller hardware.

RECORD NOT FOUND -- Indicates that the desired track, sector or side were not found.

CRC ERROR -- The CRC "checksum" read at the end of an ID or data field did not agree with the data read off the disk. One or more bits of the data read will be invalid.

LOST DATA -- Indicates the computer did not respond to DRQ line within the allocated time.

DRQ -- (Data Request bit) When set, indicates that the sector buffer is ready to accept data or contains data to be read out by the host.

BUSY -- A command was issued to the controller and no completion status has been presented. No other bits or registers are valid when this bit is set.

DETAILED DESCRIPTION OF COMMANDS

The following commands may be performed in any order and/or may be listed one after the other with or without being separated by blanks.

COMMAND "@":

The "@" (ampersand) command fills the buffer with a pre-defined pattern. Type an "@" command followed by a "<" (right arrow) command and the screen should reveal the buffer memory consisting of a recognisable pattern. The "@" command followed by a 16 bit hexadecimal number will fill the buffer with the specified 16 bit hexadecimal number. For example, by typing "@FFFF" the buffer will be filled with a series of "FF's".

COMMAND "C":

The "C" command actually clears the buffer of all existing memory and replaces the buffer memory with all zeroes. Type a "C" command followed by a "<" (right arrow) command and the screen should reveal the buffer memory of all zeroes.

COMMAND "H":

The "H" command is the "FDBUG" help command. Type "H" and return. The screen should reveal a short summary of all existing commands. Typing non-existent commands should make the bell sound.

COMMAND "L":

The "L" command lists the status definitions.

COMMAND "M":

The "M" command enables the user to actually modify the buffer memory. Typing "M" should reveal the current buffer memory and the cursor will be positioned at the top left corner. In order to move the cursor, type the appropriate arrow on the terminal keyboard.

COMMAND "R":

The "R" command reads a sector of the current working block.

COMMAND "Sn":

The "S" command followed by a number, performs a non-verified seek to the desired block.

COMMAND "U":

The "U" command reads and displays the current controller status, head, cylinder and sector registers. It does not update the current block position. Note that the controller status is displayed in both the I and II positions.

COMMAND "W":

The "W" command writes a 512 byte block to a sector on the disk.

COMMAND "Z":

The "Z" command restores the disk to cylinder 0. A restore command must be issued after a drive select command to synchronize the disk drive and the controller.

COMMAND "[":

The right bracket is an indication of the start of a repeating command routine.

COMMAND "]":

The left bracket indicates the end of a repeating command routine.

COMMAND ".":

The period command updates the status display and displays the disk status after a read, write or seek command.

COMMAND "?":

The question mark command tests the status for certain errors. A repeated command routine is aborted upon finding an error if this command is used within the command routine.

COMMAND "<":

The left arrow command displays the upper part of the buffer.

COMMAND ">":

The right arrow command displays the lower part of the buffer.

COMMAND "+n":

The plus sign followed by a number increments the current block number by "n".

COMMAND "-n":

The negative sign followed by a number decrements the current block number by "n".

COMMAND ",":

The "," (comma) command provides a delay.

COMMAND "\":

The backslash command clears the lower part of the screen.

COMMAND "Bn":

The "B" command followed by a number sets the block number to "n".

COMMAND "Nn":

The "N" command followed by a hexadecimal number "n" sets the number of heads on the disk. The default value is n = 2.

COMMAND "%n":

The "%" (percent) command followed by "n" sets the density. (n = 00 for double density and n = FF for single density) The default value is n = 00.

COMMAND "V":

The "V" command sequences all drives down (deselects all drives).

COMMAND "D":

The "D" command initializes the special diagnostic mode. The "D" command must be executed before the ";" command may be issued. The upon executing the "D" command the lower part of the screen should look like this:

```
Pass: ---- Errors: ---- Type: -- Fatalities: ---- Fatality Type: --
```

TYPE I STATUS BITS

10000000	80	Not Ready
01000000	40	Write Protected
00100000	20	Head Engaged
00010000	10	Error
00001000	08	CRC Error
00000100	04	Track 0
00000010	02	Index
00000001	01	Busy

TYPE II STATUS BITS

10000000	80	Not Ready
01000000	40	Type/Wt Prot
00100000	20	Data Compare
00010000	10	Not Found
00001000	08	CRC Error
00000100	04	Lost Data
00000010	02	DRQ
00000001	01	Busy

"Pass" is the number of passes made by the ";" command in hexadecimal.

"Errors" is the total number of errors encountered while the command routine was executing.

"Type" is the type of error encountered. The error type can be interpreted by converting the hexadecimal number to binary and looking at the "TYPE II" status bit chart provided above.

"Fatalities" is the total number of fatalities recorded while using the ";" command.

"Fatality Type" is the type of fatality found. A fatality is considered to be a "20", a Data Compare error.

COMMAND ";":

The semicolon command may be executed only after the "D" command is issued and a read has been performed. The ";" command increments the pass counter and looks at the status bytes to see if any errors exist. If an error exists, "Pass" is incremented, "Errors" is incremented, the "Type" of error is displayed. If the "STATUS" is good, "PASS" is incremented and a data compare is performed. (the data is compared to the standard pattern generated by the "@" command) If the data compare is good, the routine ends. If the data compare is bad, "Errors" is incremented, the "Type" of error is displayed, "Fatalities" is incremented and the "Fatality Type" is displayed.

COMMAND "^n":

The up arrow command followed by a number "n" sequences the specified drive number "n" up. If an undefined drive is selected or the drive will not come ready, FDBUG will lock up and the system will have to be reset.

COMMAND "$n":

The "$" (dollar sign) command followed by a four or eight bit hexadecimal number "n" sets the stepping rate of the drive. The default factor is n = 00. The following chart gives the hexadecimal number "n" and its corresponding stepping rate.

n	Rate
00	3mS
01	6mS
02	10mS
03	15mS

SUMMARY OF COMMANDS

" " --	Separate commands
@ --	Fill buffer with data
C --	Zero out buffer
H --	The help command returns all commands followed by a short summary of each command.
L --	Lists status definitions
M --	Modify buffer memory
R --	Read a sector
Sn --	Seek
U --	Read controller status
W --	Write a sector
Z --	Restore the disk
[--	(open bracket) Begin repeating group
] --	(close bracket) End repeating group
. --	(period) Display disk status
? --	(question mark) Test status for errors
, --	(comma) Provide a delay
< --	(left arrow) Display upper half
> --	(right arrow) Display lower half
+n --	Increment block value
-n --	Decrement block value
D --	Set Disktest Format
; --	Display Disktest Status
\ --	Clear Screen
Bn --	Set Block Number
$n --	Set Step Rate
Nn --	Set Number of Heads
%n --	Sets density
V --	Sequence drives down

EXAMPLE DISKTEST SEQUENCE

The following is a command sequence that may be used to test the operation for a double sided, double density disk unit.

1. Initiate the FDBUG diagnostic
2. Install a formatted scratch diskette in drive 0
3. Type ^0ZD to select drive 0, restore the drive and set special diagnostic mode.
4. Enter the following command string.

```
                                        [ZS999@WSFZCS999R.;]
                                        |||   ||| |||   ||||
                                        |||   ||| |||   ||||
Begin repeat ---------------------------+||   ||| |||   ||||
Restore ---------------------------------+|   ||| |||   ||||
Seek to block 999 ------------------------+   ||| |||   ||||
Fill buffer with pattern ---------------------+|| |||   ||||
Write buffer to disk --------------------------+| |||   ||||
Seek to block 0F -------------------------------+ |||   ||||
Restore ------------------------------------------+||   ||||
Clear buffer --------------------------------------+|   ||||
Seek to block 999 ----------------------------------+   ||||
Read block into buffer ---------------------------------+|||
Display controller status--------------------------------+||
Compare buff / update hist display -----------------------+|
Repeat entire sequence ------------------------------------+
```

DISK SURFACE VERIFICATION

This sequence can be used to verify that the entire disk surface is useable. This sequence uses the special 'disktest' mode and will report the total number of blocks tested in the 'Pass' field. Note that it is normal for this sequence to start showing errors at the point where the disk is not formatted-this implies that the test must be manually monitored to see where the failures begin.

```
Select / restore drive: ^0ZS0
Set 'disktest' mode:    D
Perform verification:   [@WCR.;+]
```

SIMPLE DISK SURFACE READ INTEGRITY TEST

This short routine to verify that all disk blocks are readable. The test will terminate when a hard error is reached (normally the end of the disk).

```
Select / restore drive: ^0ZS0
Do read test:           [R.?+]
```

RMSBUG

Diagnostic: RMSBUG

RMSBUG is a diagnostic tool designed to assist qualified technical personnel in finding and rectifying malfunctions that may occur in DMF3 Winchester Disk units.

IMPORTANT NOTE

Improper use of the diagnostic program "RMSBUG" will result in loss of information contained on the disk units. Southwest Technical Products Corporation specifically disclaims any responsibility or liability for any such damages incurred or generated by the "RMSBUG" diagnostic program. This program is not sold nor intended for distribution to persons unfamiliar with the disk units or the operation of diagnostic tools. RMSBUG remains the sole property of Southwest Technical Products and may not be reproduced or distributed without prior written permission.

RMSBUG
(UniFlex Version)

Disclaimer

RMSBUG - MICRO WINCHESTER DIAGNOSTIC PROGRAM

Table of Contents

RMSBUG

RMSBUG is a diagnostic tool designed to assist qualified technical personnel in finding and rectifying malfunctions that may occur in micro-Winchester disk units.

INSTALLATION PROCEDURE

REQUIRED: A diskette containing a bootable UniFlex system.
A RMSBUG release diskette.
A DMF3 / WD1000 / RMS disk drive combo.

To install the RMSBUG diagnostic program, boot UniFlex, then mount the disk containing the program "RMSBUG", and copy the RMSBUG program from the release disk. It is suggested that the RMSBUG program be placed in a directory called "/etc/diagnostics". The RMSBUG program should have owner SYSTEM, with no execute permission for others. Due to the potentially destructive nature of this test, it is advised that this program not be placed in the execution directories "/bin" and "/usr/bin". An example of installing the RMSBUG program is:

```
++ /etc/mount /dev/fdl /usr2                 (mount RMSBUG disk on fdl)
++ crdir /etc/diagnostics                    (make diagnostic directory)
++ copy /usr2/rmsbug /etc/diagnostics        (move onto UniFlex disk)
++ owner system /etc/diagnostics/rmsbug      (assign SYSTEM as owner)
++ perms o-rwx /etc/diagnostics/rmsbug       (remove exec permission)
++ /etc/unmount /dev/fdl                     (unmount the RMSBUG disk)
```

At this point, the RMSBUG program may be run by typing "/etc/diagnostics/rmsbug" and return. The RMSBUG program will update all file systems, then begin execution. You must be using a SWTPC CT-82 or 8200 series terminal.

NOTE: The diagnostic program "RMSBUG" moves in and takes over the entire UniFlex operating system. Since the UniFlex routines are destroyed in the process, the only way to terminate the RMSBUG program is to RESET the computer. This effectively takes the system down in so far as normal usage is concerned. This implies that the "RMSBUG" program <u>must</u> <u>not</u> be run on a system in multi-user mode.

RMSBUG DISPLAY FORMAT

All of the RMSBUG functions operate in frame mode, i.e., the terminal screen does not scroll. Some functions generate more than one page of data, and provide mechanisms for changing from one page to another. When the RMSBUG program is running, the screen will always show a header line and status information as follows:

```
              "-- RMSBUG WINCHESTER DEBUG PROGRAM - VERSION X.X --"

COMMAND:|_|            BLOCK:  00000 HD/CYL/SEC:  00 0000 00 STATUS:  00 00
         |                                         |   |   |           |  |
         |                                         |   |   |           |  |
         +-- CURSOR                   HEAD --------+   |   |           |  |
                                      CYLINDER --------+   |           |  |
                                      SECTOR --------------+           |  |
                                                                       |  |
                                     WD1000 STATUS --------------------+  |
                                                                          |
                                     WD1000 ERROR BITS -------------------+
```

"COMMAND" is followed by the cursor, indicating that the program is waiting for the next command string to be entered. Commands consist of strings of one to twenty-one characters, and are terminated by the return key.

"BLOCK" is the current block number. The current block is set by several commands, and may be incremented or decremented by using the "+" or "-" commands. Commands that operate upon disk media typically use the current block number for their manipulations.

"HD/CYL/SEC" "HD" indicates the actual disk head and drive number. The head selected can be interpreted by the following chart:

Bit 2	Bit 1	Bit 0	Head Number Selected
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

The drive selected can be interpreted by the following chart:

Bit 4	Bit 3	Drive Number
0	0	1
0	1	2
1	0	3
1	1	4

"CYL" and "SEC" are the cylinder and sector number that the last disk operation was performed on.

"STATUS" shows the status of the WD1000 controller. Two sets of hexadecimal numbers are shown. The first number is the 8-bit WD1000 CONTROLLER status. The second number is 8 ERROR BITS of the controller. This byte is valid only if the ERROR BIT in the status byte is set. A detailed explanation of these status bytes follows.

WD1000 CONTROLLER STATUS

The WD1000 CONTROLLER status is an 8-bit value returned to the computer from the WD1000 controller board. In order to determine which bits are set, the hexadecimal number must be converted to binary. For example, if the WD1000 CONTROLLER status was "50", then by converting the hexadecimal number to binary, the number would be "01010000". The binary number "01010000" has bits 6 and 4 set which may be interpreted by looking at the chart below. (DRIVE READY and SEEK COMPLETE)

WD1000 CONTROLLER STATUS BITS

```
                                  ________________________________
                                 |_7_|_6_|_5_|_4_|_3_|_X_|_1_|_0_|
                                   |   |   |   |   |   |   |   |
   BIT NUMBER                      |   |   |   |   |   |   |   |
                                   |   |   |   |   |   |   |   |
   7 -- BUSY ----------------------+   |   |   |   |   |   |   |
   6 -- READY -------------------------+   |   |   |   |   |   |
   5 -- WRITE FAULT -----------------------+   |   |   |   |   |
   4 -- SEEK COMPLETE -------------------------+   |   |   |   |
   3 -- DRQ ---------------------------------------+   |   |   |
   2 -- -----------------------------------------------+   |   |
   1 -- SEEK OUT OF RANGE ---------------------------------+   |
   0 -- ERROR BIT ---------------------------------------------+
```

SUMMARY OF WD1000 CONTROLLER STATUS BITS:

BUSY -- A command was issued to the WD1000 controller and no completion status has been presented. No other bits or registers are valid when this bit is set.

READY -- The disk unit has been sequenced up and has presented READY status to the controller. The SELECT light on the front panel should be on. The WD1000 will not execute any commands unless the ready bit is set.

WRITE FAULT -- Indicates a WRITE FAULT condition exists in the drive. The WD1000 will not execute any command if this bit is set.

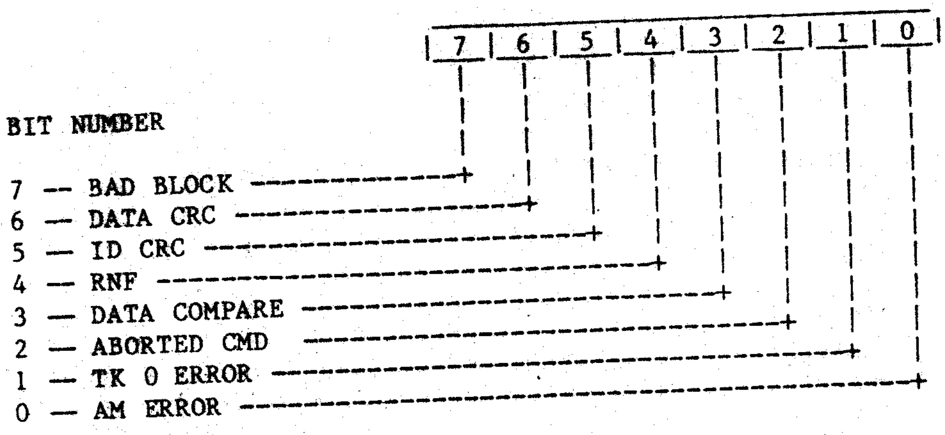
SEEK COMPLETE -- Indicates the true state of the SEEK COMPLETE line on the selected drive.

DRQ -- (Data Request bit) When set, it indicates that the sector buffer is ready to accept data or contains data to be read out by the host. The data request bit is reset when the sector buffer has been fully read from or written to.

SEEK OUT OF RANGE -- A seek was executed to a non-existing block.

ERROR BIT -- The error bit, when set, indicates that one or more bits are set in the error register. The error bit is reset on receipt of a new command.

ERROR BITS



SUMMARY OF ERROR BITS:

BAD BLOCK -- Indicates that a BAD BLOCK mark has been detected in the specified ID field. If the command issued was a write sector command, no writing will be performed. If generated from a read sector command, the data field will not be read. Note that bad blocks will not be detected if the flaw is in the ID field.

DATA CRC -- Indicates that a CRC error was encountered in a data field during a Read Sector Command.

ID CRC — Indicates that a CRC error was encountered in an ID field.

RNF -- When set, this bit indicates that an ID field containing a specified cylinder, head, sector number or sector size was not found.

DATA COMPARE -- Data read was not the same as the data written or data previously written was not the same as the data read. (Used in diagnostic mode only)

ABORTED CMD — Indicates that a valid command has been received that cannot be executed based on status information from the drive. For example, if a write sector command has been issued while the WRITE FAULT line is set, the ABORTED CMD. bit will be set. Interrogation of the Status and/or Error Registers by the host must be performed to determine the cause of failure.

TK 0 ERROR — Will be set during a Restore command if, after issuing 1023 stepping pulses, the TRACK 0000 line was not asserted by the drive.

AM ERROR -- Will be set during a Read Sector command if, after successfully identifying the ID field, the Data Address mark was not detected within 16 bytes of the ID field.

DETAILED DESCRIPTION OF COMMANDS

The following commands may be performed in any order and/or may be listed one after the other with or without being separated by blanks.

COMMAND "@":

The "@" (ampersand) command fills the buffer with a pre-defined pattern. Type an "@" command followed by a "<" (right arrow) command and the screen should reveal the buffer memory consisting of a recognisable pattern. The "@" command followed by a 16 bit hexadecimal number will fill the buffer with the specified 16 bit hexadecimal number. For example, by typing "@FFFF" the buffer will be filled with a series of "FF's".

COMMAND "C":

The "C" command actually clears the buffer of all existing memory and replaces the buffer memory with all zeroes. Type a "C" command followed by a "<" (right arrow) command and the screen should reveal the buffer memory of all zeroes.

COMMAND "Fn":

The "F" command formats a track on the disk. The format command will only be executed when the sector register is equal to one. N specifies the sector interleave (1-9) to be used on the disk. If 0 or no value is entered, a preset default sector interleave is used.

COMMAND "H":

The "H" command is the "RMSBUG" help command. Type "H" and return. The screen should reveal a short summary of all existing commands. Typing non-existent commands should make the bell sound.

COMMAND "L":

The "L" command lists the status definitions.

COMMAND "M":

The "M" command enables the user to actually modify the buffer memory. Typing "M" should reveal the current buffer memory and the cursor will be positioned at the top left corner. In order to move the cursor, type the appropriate arrow on the terminal keyboard.

COMMAND "R":

The "R" command reads a sector of the current working block. An implied seek must be performed before this command will execute.

COMMAND "Sn":

The "S" command followed by a number, performs a non-verified seek. Note that SEEK COMPLETE is not sampled after a seek command, so repetitive seek commands such as "[S0 S1000]" may give unpredictable results.

COMMAND "U":

The "U" command reads and displays the current WD1000 status, error, head, cylinder and sector registers. It does not update the current block position.

COMMAND "W":

The "W" command writes a 512 byte block to a sector on the disk.

COMMAND "Z":

The "Z" command restores the disk to cylinder 0.

COMMAND "[":

The right bracket is an indication of the start of a repeating command routine.

COMMAND "]":

The left bracket indicates the end of a repeating command routine.

COMMAND ".":

The period command updates the status display and displays the disk status after a read, write or format command.

COMMAND "?":

The question mark command tests the WD1000 status for certain errors. A repeated command routine is aborted upon finding an error if this command is used within the command routine.

COMMAND "<":

The left arrow command displays the upper part of the buffer.

COMMAND ">":

The right arrow command displays the lower part of the buffer.

COMMAND "+n":

The plus sign followed by a number increments the current block number by "n".

COMMAND "-n":

The negative sign followed by a number decrements the current block number by "n".

COMMAND ",":

The "," (comma) command provides a delay.

COMMAND "\":

The backslash command clears the lower part of the screen.

COMMAND "Bn":

The "B" command followed by a number sets the block number to "n".

COMMAND "D":

The "D" command initializes the special diagnostic mode. The "D" command must be executed before the ";" command may be issued. The upon executing the "D" command the lower part of the screen should look like this:

```
Pass: ---- Errors: ---- Type: -- Fatalities: ---- Fatality Type: --
```

Status Bits

10000000	80	Busy
01000000	50	Ready
00100000	20	Write Fault
00010000	10	Seek Complete
00001000	08	DRQ
00000100	04	
00000010	02	Seek Out of Range
00000001	01	Error Bit

Error Bits

10000000	80	Bad Block
01000000	50	Data CRC
00100000	20	ID CRC
00010000	10	RNF
00001000	08	Data Compare
00000100	04	Aborted Cmd.
00000010	02	Tk 0 Error
00000001	01	AM Error

"Pass" is the number of passes made by the ";" command in hexadecimal.

"Errors" is the total number of errors encountered while the command routine was executing.

"Type" is the type of error encountered. The error type can be interpreted by converting the hexadecimal number to binary and looking at the "Error Bits" chart provided above.

"Fatalities" is the total number of fatalities recorded while using the ";" command.

"Fatality Type" is the type of fatality found. A fatality is considered to be a "08", a Data Compare error.

COMMAND ";":

The semicolon command may be executed only after the "D" command is issued and a read has been performed. The ";" command increments the pass counter and looks at "STATUS" to see if any errors exist. If an error exists, "Pass" is incremented, "Errors" is incremented, the "Type" of error is displayed. If the "STATUS" is good, "Pass" is incremented and a data compare is performed. (the data is compared to the standard pattern generated by the "@" command) If the data compare is good, the routine ends. If the data compare is bad, "Errors" is incremented, the "Type" of error is displayed, "Fatalities" is incremented and the "Fatality Type" is displayed.

COMMAND "^n":

The up arrow command followed by a number "n" sequences the specified drive number "n" up. If an undefined drive is selected or the drive will not come ready, RMSBUG will lock up and the system will have to be reset.

COMMAND "X":

The "X" command sends a hardware reset to the WD1000 controller. It clears the cylinder, sector addresses and selects drive 0.

COMMAND "$n":

The "$" (dollar sign) command followed by an eight bit hexadecimal number "n" sets the stepping rate of the heads. The default factor is n = OF. The following chart gives the hexadecimal number "n" and its corresponding stepping rate.

n	Rate	n	Rate
00	10uS	08	4.0mS
01	0.5mS	08	4.5mS
02	1.0mS	0A	5.0mS
03	1.5mS	0B	5.5mS
04	2.0mS	0C	6.0mS
05	2.5mS	0D	6.5mS
06	3.0mS	0E	7.0mS
07	3.5mS	0F	7.5mS

COMMAND "Nn":

The "N" command followed by a hexadecimal number "n" sets the number of heads on the disk. The default value is n = 8.

COMMAND Tn":

The "T" command followed by a hexadecimal number "n" will attempt to clear a soft or hard read error on the current block. "n" specifies the desired sector interleave factor for the automatic track reformatting function of T.

SUMMARY OF COMMANDS

" "--	Separate commands
@ --	Fill buffer with data
C --	Zero out buffer
Fn --	Formats a track with the given interleave
H --	The help command returns all commands followed by a short summary of each command.
L --	Lists status definitions
M --	Modify buffer memory
R --	Read a sector
Sn --	Seek
U --	Read WD1000 status
W --	Write a sector
Z --	Restore the disk
[--	(open bracket) Begin repeating group
] --	(close bracket) End repeating group
. --	(period) Display disk status
? --	(question mark) Test status for errors
, --	(comma) Provide a delay
< --	(left arrow) Display upper half
> --	(right arrow) Display lower half
+n --	Increment block value
-n --	Decrement block value
D --	Set Disktest Format
; --	Display Disktest Status
\ --	Clear Screen
X --	Reset WD1000 Controller
Bn --	Set Block Number
$n --	Set Step Rate
Nn --	Set Number of Heads
Tn --	Clear read error on current block

SAMPLE DISKTEST PATTERN FOR RMS 518 DISK DRIVE

The following sequence can be used to test the integrity of data transfers between the computer and the disk. This test writes a specific test pattern at block 5000 (B5000@W), then reads the data in block 0 to move the heads around (B0R), clears the data buffer to 0 (C), reads the data back from block 5000 (B5000R), displays the disk status (.), and checks the data for errors and updates the display (;). The block number '5000' can be changed for different size disks.

```
Select / restore drive: ^0ZS0
Do the disktest:           [B5000@WB0RCB5000R.;]
                           ||    ||| |||    ||||
Open repeat routine -------+|    ||| |||    ||||
Set block to 5000 ----------+    ||| |||    ||||
Set buffer to test pattern ------+|| |||    ||||
Write buffer to disk -------------+| |||    ||||
Set block to 0 --------------------+ |||    ||||
Read block 0 ------------------------+||    ||||
Clear buffer -------------------------+|    ||||
Set block to 5000 ---------------------+    ||||
Read block 5000 ----------------------------+|||
Display disk status --------------------------+||
Display disktest status -----------------------+|
Close repeat routine ---------------------------+
```

DISK FORMAT SEQUENCE

The following sequence can be used to format a disk. This format will automaticaly fill all sector data with zeros.

Select / restore drive:	^0ZS0	
Format (manual stop):	[F.+11]	This format will not stop when the end of the disk is reached. You must manually stop it when the desired number of tracks has been formatted.
Format (auto stop):	[FR?.+11]	This format will read the first record of the newly formatted track and will exit when a format has failed.

DISK SURFACE VERIFICATION

This sequence can be used to verify that the entire disk surface is useable. This sequence uses the special 'disktest' mode and will report the total number of blocks tested in the 'Pass' field. Note that it is normal for this sequence to start showing errors at the point where the disk is not formatted-this implies that the test must be manually monitored to see where the failures begin.

```
Select / restore drive: ^0ZS0
Set 'disktest' mode:     D
Perform verification:    [@WCR.;+]
```

SIMPLE DISK SURFACE READ INTEGRITY TEST

This short routine to verify that all disk blocks are readable. The test will terminate when a hard error is reached (normally the end of the disk).

```
Select / restore drive: ^0ZS0
Do read test:            [R.?+]
```

CLEARING SOFT AND HARD READ ERRORS ON WINCHESTER DRIVES

It is sometimes desirable to clear a soft or hard read error on a drive while maintaining as much data integrity as possible. The following procedure can be used to correct read errors discovered by the previous disk surface read integrity test.

First try REPEATEDLY reading the block in question with the 'R' command. If you can get at least one good read, the internal buffer will be filled with the correct data for the block and may be able to be re-written with the 'W' command. This procedure will work if the ID field of the block is intact but the data and/or data CRC part is flaky. If this is succesful, try re-reading the block several times to be sure that the error is really cleared.

If you are unable to get a good read on the block by using the 'R' command, then the 'T' command must be used to reformat and clear the error. The T command functions by reading ALL of the blocks on the track with the bad block, reformats the track with the given sector interleave, and replaces the data. Note that the block that was originally bad will now read correctly, but may be filled with zeroes rather than the original data. Extreme care should be used when clearing errors with this function since at least one block of data may be lost.

EXAMPLE TO CLEAR A READ ERROR ON BLOCK 1000:

Select and restore drive:	^0ZS0
Set bad block number:	B1000 or S1000
Try repeated reads:	R R R etc. until you give up
If one read succesful:	W then R to verify that error cleared
If unsuccessful:	Tn to clear and reformat (n=interleave)