ATPS

MISC52 performs various operations on an IOU52 controller/disk

```
.   SUBJECT DEVICE : disks supported by IOU52s.

        ROM-version: -

    CONTROLLER     : IOU-52
        Board-ver. : -
        ROM-version: B

    SYSTEM REQUIRED
        CPU-type   : See directory listing
        Min. memory: 32K bytes
```

GENERAL OUTLINE

This program allows several operations to be performed on one or more disks supported by the IOU52. These operations include

- format with optional verify
- read/display alternate-sector map
- read user-sectors, checking only status
- force alternate user-sector(s)

PROGRAM-LOGIC

Initialization

1. Load MISC52 using standard program-loading procedure. After the heading is displayed, the program checks whether the processor-type is later than Q29. If so, the program will use fast-read instruction for reading sequential user-sectors.

The program then instructs:

"ENTER DEVICE-ADDRESSES (0H):"

Enter a 2-digit device-address xy for each disk-drive to be accessed by the program. "x" is always 0, and "y" is the IOU52 address ($1 -- $F). A device-address list terminated by an "R" will allow repeating of the selected operation. Up to 4 device-addresses can be entered. Pressing TAB, RETURN, or TRANSMIT without entering any address will cause the program to return to the program loader.

For each device-address entered, the following items are checked:

Device-address-entry format is correct and within bounds. If not, the program reports

<u>ILLEGAL VALUE FOR DEVICE-ADDRESS.</u>

Then it re-requests the entry of device-addresses.

Device-controller status is not $FF, meaning that the controller is active. If not, the program reports

<u>DEVICE xy NOT ACTIVE.</u>

Then it skips to check the next device-address.

RS2 value returned from the controller is $FD, meaning it is a disk-controller. If so, the program performs a seek and read of sector 0, and if it detects inoperable-status, it reports

<u>DEVICE xy NOT OPERABLE.</u>

Then it skips to check the next device-address.

If status $DX is returned after the read, meaning the disk is not formatted for the IOU-52 controller, the program reports

<u>DEVICE xy NOT FORMATTED FOR THE IOU 52.</u>

Error-statuses other than $8x or $Dx will be reported as follows:

<u>DEVICE xy REPORTS ERROR-STATUS xx.</u>

The controller-ROM version-# is checked in the upper 5 bits of status 0 after control $A0 is issued. And the first byte of the ID string returned after control $7F contains $52. If none of the above is true, the program reports

<u>DEVICE xy NOT AN IOU 52.</u>

Then it skips to check the next device-address.

If any device-address was skipped in the above checking-procedure, the program will report

```
THE DEVICE(S) REPORTED ABOVE WILL NOT BE ACTIVATED.
```

For all devices that are activated, the program obtains and displays the controller- and drive-profile information on the screen. The operator should check each profile carefully before proceeding. The display for a 20-megabyte Marksman on controller $0D is shown as follows:

```
CONTROLLER                0D
ROM-LEVEL                  B
DRIVE-TYPE          MARKSMAN
CAPACITY               20 MB
CONFIGURATION             02
# OF CYLINDERS           213
# OF HEADS                 4
# OF SECTORS / TRACK      28
```

2. A menu is then displayed for selection of the operation to be performed.

```
PROGRAM MENU:
1.  FORMAT DISK WITH OPTIONAL VERIFICATION
2.  READ/DISPLAY ALTERNATE-SECTOR MAP
3.  READ USER-SECTORS - STATUS-CHECK ONLY
4.  FORCE ALTERNATE USER-SECTOR
5.  OEM AUTOMATIC-FORMAT MODE
ENTER SELECTION:
```

Enter the # corresponding to the desired operation. If TAB, TRANSMIT, or RETURN is pressed without any entry, the program will request device-addresses again. If the number entered is not on the menu, the program reports:

"INVALID OPTION. PLEASE TRY AGAIN."

and then asks for the operation-# again.

3. Except for force-alternation, all operations have the option of output-messages on a printer. First, the program instructs:

"ENTER OUTPUT-DEVICE ADDRESS (H):" .

A 1-digit hex number should be entered. If TAB, TRANSMIT, or RETURN is pressed with no entry, the program assumes device 0 as the output-device. The output-device is checked for operability, and the following message is displayed when inoperable-condition is detected.

"OUTPUT-DEVICE INOPERABLE."

If the device is operable, the program asks:

"TEST-MESSAGE ON OUTPUT-DEVICE?"

Press F3 if no message is desired; press F2 otherwise. When F2 is pressed, the ":" prompt appears. Enter a message of up to 62 characters. If a non-zero device-# is specified as the output-device, the following line will be printed, followed by the test-message, if any.

"MISC52 830328 Q30 (test-message)"

Execution

I. FORMAT DISK WITH OPTIONAL VERIFY

A. First, the program instructs:

"ENTER FORMAT-DATA ($6DB9/hhhh):"

Enter a 4-digit hex data-pattern, which will be written to all sectors on the disk. $6DB9 will be the default data-pattern when TAB/RETURN/TRANSMIT is pressed without any entry. Then the program asks:

"DISABLE VERIFICATION?"

Press F2 to skip steps C and D below. Press F3 otherwise. The program then displays:

"STANDARD INTERLACE FOR Q29 IS 5; Q30 IS 4; Q64 IS 3."
"NON-STANDARD INTERLACE DESIRED?"

Press F3 for standard format; and press F2 for unusual applications.

B. The format-process is performed by the firmware when the following events are executed by the program.

- device-control $75 is issued.
- 2-byte interlace-parameter is sent.
- a 768-byte data-buffer containing $FFs is sent.

As the program starts formatting, it declares

"FORMATTING IN PROGRESS.....".

At the completion of formatting, it declares

"FORMATTING SUCCESSFUL".

If error-status is detected during format, the program displays

"XX STATUS DETECTED DURING FORMAT."

C. To verify the format, the program first writes to all user-sectors using the format-pattern specified in step A, including the decimal sector-numbers. No read-after-write check is performed for these writes to save time. On the top of the screen,

"WRITING SECTOR xxxxxx"

is displayed, and the sector-# is updated every 100th sector. At completion,
"WRITE COMPLETED."

is displayed. If error-status is detected during write,

"SECTOR xxxxxx ERROR-STATUS xx IN WRITE-PHASE."

is displayed.

D. The last step is to verify the disk is written correctly. The verification takes the form of a write, except the data passed to the IOU52 are not written onto the disk. Instead, the data are used for comparison. When an error-status is returned for a bad sector, the program puts the bad-sector # into a table. During the verification, the program displays

"VERIFYING SECTOR xxxxxx."

and at completion,

"VERIFICATION COMPLETED."

is displayed. Finally, the program uses the table that stores bad-sector #'s to carry out force-alternation of all the bad sectors found by the verification-process. When more than 150 sectors are found to be bad, the table is full. The program will display bad-sector information on the screen as follows:

```
"MORE THAN 150 BAD SECTORS HAVE BEEN DETECTED."
"SECTOR xxxxxx"
•
•
•
"THE ABOVE SECTORS NEED TO BE FORCE-ALTERNATED."
```

* In an effort to make the format/verify operation less time-consuming, steps C and D use short writes of 16 bytes of data to each sector.

II. READ ALTERNATE-SECTOR MAP

In this operation, the program issues device-control $72 to the IOU52 to start the process of passing alternate-sector-map entries from the IOU52 to the processor. The program reads one entry at a time and stores all 16 bytes of each entry into a table until end of map is detected. During the reading of the map-entries, if an error-status occurs, the program reports

"ERROR READING ALTERNATE-SECTOR MAP."

Each map-entry will be processed by the program to be displayed on the output-device. A typical alternate-sector map for a 20MB Marksman is shown below:

```
"ALTERNATE-SECTOR MAP FOR DISK 0D (INTERLACE:5)"

"      BAD PHYSICAL              BAD             "
"  CYLINDER HEAD SECTOR     LOGICAL ALTERNATE    "

"     004     02    00      000504 F 023688      "
"     064     03    01      007185 F 023733      "
"     157     00    17      017677 F 023689      "

"  END OF ALTERNATE-SECTOR MAP.                  "
```

If no entry is found in the map, the program reports:

"ALTERNATE-SECTOR MAP EMPTY."

For disks that were previously formatted for the IOU42 and subsequently have been formatted for the IOU52, an option is available to display both the IOU52 and IOU42 sector-#'s for each entry in the map.

III. READ USER-SECTORS - CHECK STATUS ONLY

This operation involves reading a selected range of user-sectors. Only status is checked after reading each sector. Data cannot be checked for practical reason. An option is available to disable the read-retry operation so that any problem in reading any sector will be made more visible. A 6-digit starting-sector # can be entered when program instructs:

"ENTER STARTING-SECTOR # (000000/dddddd):"

The default sector-# is 0. As the program checks the sectors,

"VERIFYING SECTOR xxxxxx"

is displayed on the top of the screen, and the sector-# is updated every 100th sector. The verification continues until the last sector is checked, or when either F2 or F3 is turned on. At completion, the program displays:

"VERIFICATION COMPLETED."

Any error-status received during verification is reported as follows:

"xx STATUS READING SECTOR xxxxxx."

IV. FORCE ALTERNATE SECTOR(S)

This operation allows sector(s) be force-alternated through manual specification. Sector-information is accepted in several formats. All formats have 3 parameters; the first 2 being a 3-digit cylinder-# and a 2-digit head-#, the 3rd parameter being different for all formats. They are each described as follows:

A. 5-digit BYTE-#
Used for Marksman, Memorex, and Fujitsu Eagle.

B. 2-digit sector preceded by "F"
Used for Fujitsu 168MB and 84MB.

C. 2-digit sector-# preceded by "P"
Used for formatted IOU52 disks, any type.

D. 2-digit sector-# preceded by "L"
Used for formatted IOU42 disks, any type.

Note: when converting an IOU42 disk to an IOU52 disk, the format described in "D" above should be used. In that case, the information from the Qantel defect-label on the IOU42 disk can be used directly.

V. OEM AUTOMATIC-FORMAT MODE

This operation is designed to cover all the proper operations for formatting a disk and execute them in a logical manner. The following steps are executed sequentially, and sector-information for force-alternation is entered before the first step is executed.

A. FORMAT DISK.
B. WRITE/VERIFY FORMATTED DISK.
C. FORCE-ALTERNATE SECTOR.
D. DISPLAY ALTERNATE-SECTOR MAP.

SAFT tests the 3740 STAND ALONE FLOPPY DISK DRIVE.

```
    SUBJECT DEVICE : 3740 STAND ALONE FLOPPY DISK DRIVE

        ROM-version: -

    CONTROLLER     : IOU-40
        Board-ver. : -
        ROM-version: -

    SYSTEM REQUIRED
        CPU-type   : See directory listing
        Min. memory: 16K bytes
```

This program tests the 3740 Stand-alone Floppy Disk drive. One or more Drives may be configured for this test, each with its own IOU-40.

Install the Floppy Drive, via a standard QSP cable, to channel A of the IOU-40, (top connector). Turn on power, at the rear of the unit, and install the diskette, making sure that the lever is fully closed.

Please note that the 3740 format is: single Density, single sided, soft sector, 128 bytes/sector and 28 sectors/track. A Format operation is a part of the automatic test, the program will therefore accept any un-protected 8 inch diskette.

Load the program using the standard ATP loading procedure, with the name SAFT.

The program will ask for the IOU-40 device addresses. Each device will be checked to verify that it is an IOU-40 and is operational.

Next the question "SINGLE TEST ?" will be asked. Answer using Flag 2 (yes) or Flag 3 (no). If the selection is yes, the following list will be displayed:

```
                    TEST BLOCK NAMES ARE:

        WRT   WRITE - TRACK/SEC/DATA           (MANUAL)
        RED   READ  - TRACK/SEC/DATA           (MANUAL)
        FMT   FORMAT                           ( AUTO )
        PH1   WRITE TEST - ALL, WORST CASE     ( AUTO )
        PH2   READ TEST  - ALL, WORST CASE     ( AUTO )
```

Enter the desired test name and the program will continue. If the single test option is not selected the program will execute the automatic tests in order, (begining and ending with Format).

Next enter the iteration count. This is the number of times the test will be executed for each controller. Specify whether to halt after an error by answering "yes" or "no" (f2 or f3) to the question "HALT AFTER ERROR".

The program will now ask "OUTPUT DEVICE:". Enter a valid printer address to record test names, iteration counts and error messages as they occur. The terminal will also display this information.

Finally a "TEST MESSAGE" may be recorded on the printer at the beginning of the test (as well as the control line of a VT3). Answer "yes", as before, to allow entry of this message.

A description of each test follows, first the MANUAL tests (executed only by specifing "SINGLE TEST"), and then the AUTOMATIC section will be described.

A. MANUAL SECTION

1. WRITE
   a. A single sector write is performed at the specified track and sector. The operator enters a single byte to be repeated 128 times.

2. READ
   a. A single sector read is performed at the specified track and sector. The sector is display on the terminal and the output device (if selected).

B. AUTOMATIC SECTION

1. FORMAT
   a. The 3740 Stand Alone firmware is commanded to format the diskette. This is done in approximately 3 minutes and includes a read pass following the write

2. PHASE 1 - WRITE TEST
   a. Writes the entire diskette, tracks 0 to 76 (hex 4C), and sectors 1 to 26 (hex 1A). The data written is the worst case data pattern, $B6DB6D, repeated.
   b. A two byte "header", consisting of the track and sector numbers (in hex) is written at the beginning of each sector.

3. PHASE 2 - READ TEST
   a. Data written in Phase 1 are read back and verified.

During execution of Phase 1 and 2, the upper line of the terminal will indicate the following, (from left to right): track (hex), sector (hex), test name, iteration (entered and actual) and an "h", if the halt on error option has been selected.

At the end of the test sequence the iteration count is incremented and displayed along with the number of the device just tested. When the count entered at the start of the test is reached, and the last device has been tested, the test is complete.

Errors are reported either by a simple message, in the case of a data miscompare or timeout of some kind, or by the display of the error code from the IOU-40 or 3740 firmware, with the appropriate message.

STAPE tests the IOU-49 STREAMING TAPE DRIVE CONTROLLER.

SUBJECT DEVICE :	IOU-49
ROM-version:	-
CONTROLLER :	-
Board-ver. :	-
ROM-version:	-
SYSTEM REQUIRED	
CPU-type :	See directory listing
Min. memory:	32K bytes

This program tests the IOU-49 streaming tape drive controller. One or more Cipher tape drives may be configured for this test, each with its own IOU-49.

Load the program using the standard ATP loading procedure, with the name STAPE.

The program will ask for the devices to be tested. Each device will be checked to verify that it is an IOU-49 and is operational. The IOU-49 rom date and version will be displayed for each device entered.

Next the question "SINGLE TEST ?" will be asked. Answer using Flag 2 (yes) or Flag 3 (no). If the selection is yes, the following list will be displayed:

TEST BLOCK NAMES ARE:

UNL	WRITE PROTECT AND UNLOAD TEST	(MANUAL)
END	END OF TAPE TEST	(MANUAL)
PRE	PRELIMINARY FUNCTIONAL TEST	(AUTO)
PH1	WRITE ENTIRE TAPE TEST	(AUTO)
PH2	READ ENTIRE TAPE TEST	(AUTO)
PH3	STREAMING TEST	(AUTO)

Enter the desired test name and the program will continue. If the single test option is not selected the program will execute all the automatic tests in order. Next enter the iteration count. This is the number of times the test will be executed for each controller. Specify whether to halt after an error by answering "yes" or "no" (f2 or f3) to the question "HALT AFTER ERROR".

The program will now ask "OUTPUT DEVICE:". Enter a valid printer address to record test names, iteration counts and error messages as they occur. The terminal will also display this information.

Finally a "TEST MESSAGE" may be recorded on the printer at the beginning of the test (as well as the control line of a VT3). Answer "yes", as before, to allow entry of this message.

A description of all the tests follows, first the MANUAL tests (executed only by specifing "$INGLE TEST"), and then the AUTOMATIC section will be described.

A. MANUAL SECTION

1. WRITE PROTECT AND UNLOAD TEST
   a. A write is attempted with the tape write protected, this causes an automatic unload.
   b. The tape is manually reloaded with a write ring.
   c. An Unload command is executed.
   d. The tape is manually reloaded.

2. END OF TAPE TEST
   a. Maximum length blocks are written to the end of tape mark. The total block count is displayed. This also serves as a test of overall tape quality.

B. AUTOMATIC SECTION

1. PRELIMINARY TEST
   a. Write 10 maximun length blocks of channel-test data in IPL mode. Channel test data consists of the following 26 byte pattern:

```
$0102040810204080C0E0F0F8FC
 FEFDFBF7EFDFBF7F3F1F0F0703
```

   b. Rewind, read and verify each block in IPL mode.
   c. Rewind and do a Read Check of each block.
   d. Backspace and Erase the 10th block.
   e. Backspace and rewrite the 10th block and 2 end of file marks.
   f. Rewind and search for 2 file marks.
   g. Rewind, read and verify each block in IPL mode.
   h. Rewind and write 10 maximum length blocks of worst-case data in Buffer mode (with interrupts enabled). Worst-case data is the pattern $B6DB6D.
   i. Rewind, read and verify each block in Buffer mode (with interrupts enabled).
   j. Rewind, Stream write 10 maximum length blocks of worst-case data.
   k. Do a reverse backup check.
   l. Do a forward backup check.
   m. Repeat the above sequence at 50ips and then at 100ips, (the first sequence is at 25ips).

2. PHASE 1 - IPL MODE WRITE
    a. Write incremental block lengths. Each block 1 byte greater than the preceding. The data pattern is increased by 1 each block, from all $00's to all FF's.
    b. Write 26 blocks of increasing length, each block more than the last by the amount of the next value in the 26 byte channel test pattern (previous page). The last block written is the maximum length of 16k ($4000). The data pattern is the channel test pattern repeated.
    c. Write decremental block lengths to the end of tape mark. Data written is the worst-case pattern, $B6DB6D, repeated.

3. PHASE 2 - IPL MODE READ
    a. Data written in Phase 1 are read back and verified.

3. PHASE 3 - STREAM MODE WRITE
    a. Write 512 maximum length blocks of worst-case data.
    b. Do a reverse backup check.

At the end of the test sequence the iteration count is incremented and displayed along with the number of the device just tested. When the count entered at the start of the test is reached, and the last device has been tested, the test is complete.

Errors are reported either by a simple message, in the case of a data miscompare or timeout of some kind, or by the display of the controller's Read Id bytes, with their interpretation, in the case of a status error. Examples are shown on the next page.

Examples:

```
ERROR #01
INCORRECT READ DATA

ERROR #02  TST:PRE  ITR:01  BLK:0001  CTL:20  SPD:025
 MODE:IPL  ST0:84   EXT 15:80 16:00 17:00 18:00 19:00
TAPE INOPERABLE
```

In the first example, above, no status error has occurred, but the data read back from the tape did not agree with what was written. The second example is an error reported by the controller. The fields displayed are as follows:

1. Error number.
2. Test Block in progress.
3. Test iteration number.
4. Last block written or read.
5. Last control command sent to controller.
6. The current speed in inches per second.
7. The mode of operation (Ipl, Buffer or Streaming).
8. Status 0.
9. External status bytes.
10. A message (or messages) derived from the external status bytes.

All errors are numbered and a device is aborted should the limit of 9 errors be reached in any one test sequence. After an error, the test in which it occured is restarted from the beginning.

The first line of the terminal's screen indicates program information as follows:

1. The block number of the current read or write.
2. The address of the device being tested.
3. The name of the test block being run.
4. The number of the test iteration with the count entered at the beginning of the test - seperated by a slash.
5. An "h" in the right corner, if the halt on error option was selected.

Example:

```
BLK=1234  ADR=4  TST=PRE  ITR=02/04h
```

CPU64 tests peripheral-independent CPU macro-instructions.

```
SUBJECT DEVICE : CPU Q64

    ROM-version: -3

CONTROLLER     : -
    Board-ver. : -
    ROM-version: -

SYSTEM REQUIRED
    CPU-type   : See directory listing
    Min. memory: 32K bytes
```

CPU64 tests all Q64 macro instructions except reads, writes, CTL's, and RIO. The program tests an instruction for the iteration count before testing the next instruction. Unless the operator enters an iteration count and/or turns on FLAG 3, the program assumes the default values, which are: execute $200 iterations and list test block names. Halting-after-error and repeating-the-test are operator-selectable.

INITIALIAZATION

1. Using the name CPU64, load the program.

2. The program will identify itself and report the serial number of the CPU board.

3. The program will request "ITERATIONS (HHHHHH):".

    a. To either return to the loader, or enable the default values, press RETURN. The program will ask "RETURN TO LOADER?".

        (1). To return to the loader, press FLAG 2

        (2). To enable the default values, press FLAG 3 and proceed to step 4.

    b. To override either or both of the default values, enter either a 1- to 6-hexadecimal-digit number, or turn on FLAG 3 before pressing RETURN.

4. The program will ask "HALT AFTER ERROR?". Hit either

   a. F2 to halt the CPU after any error, or

   b. F3 to not halt, but instead to report the error and skip to the next instruction.

5. The program will ask "REPEAT TEST?". Hit either

   a. F2 to run the test repeatedly, or

   b. F3 to run only one pass of the test.

EXECUTION and ERROR-REPORTING

Consult the operating-instructions for CPU2930 or CPU2.

ERROR CODES

If the program reports that the CPU executed MID erroneously (error 00), the report means that the execution did not put $000005 into memory-locations $D, E, and F.

If the program reports an error while executing ML, ANI, ORI, XRI, TBI, CP1, or CP2, use the explanations in these operating instructions.

To get an explanation of any other error reported by CPU64: search the operating-instructions for CPU2930 or CPU2 to find the error-code. When the code is found, the explanation associated with it is the best available explanation.

ML : MOVE LEFT

Initialize a test area from $3000 to $40FF with $00's.

Load (LD) $FF into the accumulator.
Move (ML) $FF into address $D.
00 Addresses $C through $F do not contain $30FF30FF.

Load (LD) address $00 with $00, remainder of accumulator contains $FF.
Move left address $00 for a count of 15.
01 Address $F does not contain $00.

The data in the error code explanation describe the origin, count, and fill character, then the origin, count, and destination of the move left instruction, and the type of test. A compare left to verify the move left branched on not equal. Fill-character is ASCII.

	ORGN	CT	F	ORGN	CT	DEST	TYPE
02	300B	A	1	3000	A	300B	SHORT MOVE, NO BASEFAULT
03	30C0	A0	2	3020	A0	30C0	MEDIUM MOVE, NO BASEFAULT
04	32C0	B0	3	3200	B0	32C0	LONG MOVE, NO BASEFAULT
05	380B	9	4	37FF	9	380B	SHORT MOVE, SOURCE BASEFAULT
06	3820	2F	5	37DB	2F	3820	MEDIUM MOVE, SOURCE BASEFAULT
07	3860	AF	6	375B	AF	3860	LONG MOVE, SOURCE BASEFAULT
08	37FF	8	7	3808	8	37FF	SHORT MOVE, DEST. BASEFAULT
09	37FF	9E	8	38FE	9E	37FF	MEDIUM MOVE, DEST. BASEFAULT
10	37FF	AE	9	3910	AE	37FF	LONG MOVE, DEST. BASEFAULT
11	37FF	7	H	3FFF	7	37FF	SHORT MOVE, BOTH BASEFAULT
12	37FF	9D	J	37F0	9D	37FF	MEDIUM MOVE, BOTH BASEFAULT
13	37FF	AD	K	3F60	AD	37FF	LONG MOVE, BOTH BASEFAULT
14	3810	20	M	3815	6	3820	SHORT RIPPLE-MOVE
15	3010	40	N	3030	30	3040	MEDIUM RIPPLE-MOVE
16	3100	100	P	3150	B0	3170	LONG RIPPLE-MOVE

Set Bank D before executing the next test, then restore Bank C

17	-	-	-	0000	4000	4000	SUPER LONG MOVE

Part of the program is moved for a count of $2800 to a destination, beyond the program, that varies with each iteration of the test-block by an offset of $00 to $0F.
18 A compare left failed to branch on equal.

ANI : AND IMMEDIATE

	Hex 00 ANI with hex FF in byte 15.
00	The program then did a BNZ.
01	The result, in byte 15, is not hex 00.
	Hex FF ANI with another hex FF in byte 15.
02	The program then did a BEQ.
03	The contents of byte 15 is not hex FF.
	Hex AA ANI with hex 55.
04	The program then failed to BEQ.

ORI : OR IMMEDIATE

	Hex FF ORI with hex 00.
00	The program then did a BEQ.
01	The result is not hex FF.
	Hex AA ORI with hex 55.
02	The program then did a BEQ.
03	The result is not hex FF.
	Hex 00 ORI with hex FF in byte 15.
04	The program then did a BEQ.
05	The result is not hex FF.

XRI : XOR IMMEDIATE

	Hex 00 XRI with hex 00 in byte 15.
00	The program then did a BNZ.
01	The result is not hex 00.
	Hex FF XRI with hex 00 in byte 15.
02	The program then did a BEQ.
03	The result is not hex FF.
	Hex FF XRI with hex FF in byte 15.
04	The program then did a BNZ.
05	The result is not hex 00.

TBI : TEST BIT IMMEDIATE

Hex FF was used to test hex 00.
00 The program then did a BNZ.

Hex 66 was used to test hex 99.
01 The program then did a BNZ.

Hex 01 was used to test hex 99.
02 The program then did a BEQ.

Hex 80 was used to test hex 99.
03 The program then did a BEQ.

Hex 18 was used to test hex 99.
04 The program then failed to BNZ.

CP1 : COMPARE IMMEDIATE, 1 byte

The test values reside in the field: $AA00807FAA.

Hex 00 was compared to another hex 00.
00 The program did a BMI.
01 The program failed to BEQ.

Hex 80 was compared to hex 7F.
02 The program did a BEQ.
03 The program failed to BMI.

Hex 7F was compared to hex 80.
04 The program did a BEQ.
05 The program failed to BP.

Hex 7F was compared to hex 7F.
06 The program did a BMI.
07 The program failed to BEQ.

CP2 : COMPARE IMMEDIATE, 2 bytes

Hex 00 was compared to hex 0000.
00 The program then failed to BEQ.

Hex 80 was compared to hex 007F.
01 The program then failed to BMI.
02 After BMI, the program did either a BEQ, or failed to BNZ.

```
.   Hex FF was compared to hex 1000.
03  The program then did a BMI.
04  The program then did a BEQ.
05  The program then failed to BNZ.
```

DISC52 tests Fujitsu, Marksman, and Memorex disk-drives.

```
SUBJECT DEVICE : Fujitsu, Marksman, & Memorex disk-drives

    ROM-version: -

CONTROLLER     : IOU-52
    Board-ver. : -
    ROM-version: 5

SYSTEM REQUIRED
    CPU-type   : See directory listing
    Min. memory: 32K bytes
```

Some modest assumptions were made when the test was designed. Service-representatives and factory-personnel should check the following list of assumptions before drawing the conclusion that the test-program or the disk-drive is faulty:

1. The disk-pack has been initialized properly and completely.

2. Power is on, and the disk is selected.
3. MASP/ZSP and AMI switches on the tester-panel are OFF.

4. The operator has read and understood the operating-instructions.

5. The processor, the TPU, and the device for reporting errors work correctly.

The program begins with an initialization-procedure. This is followed by 6 test-blocks, which may be discontinued if the operator initializes an interpreter-routine called FRODO, described on page 012.

The test-initialization is executed once at the beginning of the test. It allows the operator to enter certain parameters needed by the program and enables or disables certain routines within the test in accordance with options selected by the operator.

The program refers to each test-block by the first three letters of the name of the block. Each block is composed of sections, each of which is designated by a letter, and some sections are composed of sub-sections, each of which is designated by a decimal digit. For example, the first section of the "PATTERN" test-block is composed of five sub-sections, one for each pattern-group used in that section. The first sub-section is identified as "PAT A1". The significance of this identification is made recognizable only by a familiarity either with the operation of this test, or with "DISC52 TEST LOGIC, a document included with these operating-instructions.

Test-blocks are ordered in a definite sequence, designed to perform in the beginning some basic, simple, fast operations yielding questionable information, and then to work toward complex, extensive, slow operations yielding accurate information in great detail.

The degree of certainty that the disk is operating correctly increases with the number of test-blocks that have successfully run to completion, rather than with the length of time (the number of iterations) during which any one block is run. Thus, eight minutes of testing with an iteration-count of 01 is more useful than a half hour of testing with an iteration-count of FF. In the former case the test will have completed the first five blocks, giving almost complete assurance that the disk is operating correctly. In the latter case however, the test will still be executing the initial test-block, giving some indication that the test can probably write and read hex zeros to sector 00000, and hex FF's to sector 00111.

With each successive test-block the extent of the error-analysis performed by the program increases. As the extent of the error-analysis increases, the assurance that the error-report is accurate * also increases. This implies that error-reports in the first two or three blocks may be inaccurate. This is precisely the situation, and it is a deliberate result of ordering and of designing test-blocks to yield the maximum assurance of disk-operability in the shortest time. The programmer is aware that obscure error-code explanations, or error-codes suggesting a false cause of failure, are very frustrating to technician's trying to repair the disk. Any reasonable suggestion of a method of avoiding this problem will be gratefully accepted.

* "accurate" here means correct and complete in its description of a failure of a test-segment designed to detect that and only that failure.

INITIALIZATION

1. Using the test-name "DISC52" load the program.

2. If the CPU is later than Q29, the program will report "FAST READ/WRITE WILL BE USED FOR TESTING.". On any CPU the program will then request "DISK-ADDRESSES (0H):". Enter the two-digit address of each disk-drive to be tested. Up to four addresses may be entered. After the last device-address, an "r" may be entered. Entering "r" causes the last five test-blocks (ADDRESS test through EXTENSIVE test) to repeat endlessly. The program will scan the list of addresses to test whether all devices can be selected.

   A. If the program fails to detect a device at any one of the addresses entered, it will report "NO DISK-DRIVE PRESENT AT ADDRESS xx" and will return to requesting "DISK-ADDRESSES (0H):".

   B. If not all disk-drives are correctly configured IOU-52s, the program will display "WRONG DISK-TYPE AT DEV xx" and will halt. After ST-SP has been pressed, the program will return to requesting "DISK-ADDRESSES (0H):"

3. For each disk, the program will report
   the IOU-52 address,
   the ROM-version of the IOU-52,
   the disk-drive type,
   the capacity of the disk,
   the setting of the configuration-switches on the IOU-52,
   the number of cylinders,
   the number of heads, and
   the number of sectors per track.

   The program will then instruct "PLEASE EXAMINE EACH DEVICE-PROFILE CAREFULLY. HIT EITHER F2 OR F3 TO CONTINUE.". Either

   A. if you need to power-down the system to change any profile, power-down, or

   B. else hit either flag.

4. The program will request "OUTPUT-DEVICE ADDRESS:". Either

    A. enter the address of the device that is to report errors and test-status, or

    B. if this device is device 0, either

        1. enter "0", or
        2. hit RETURN, entering nothing.

5. The program will ask "SELECT OPTIONS?".

    A. If the operator answers "NO", the test will be performed in its normal mode of operation, which means:

        1. After DISC52 has been initialized (proven by the display of "CONTROLLER x HAS LEVEL x ROMS DATED yymmdd."), FRODO will be accessible either

            A. at any time, by branching to $1000, or

            B. during a Halt-after-Error, by pressing Flags 2 and 3 and then ST-SP;

        2. the alternate-sector map will not be displayed, but the current number of alternate sectors will be displayed;

        3. the program will halt after most errors;
        4. errors will be reported on device 0;

        5. all blocks in the automatic part will be executed the same number of times (to be specified below in step 5); and

        6. track-checking will be performed on most seeks.

    B. If, instead, the operator answers "YES", the program will issue the 5 requests listed below.

        1. "FRODO ONLY?". A "YES" answer will initiate FRODO, abandoning execution of the test-blocks of DISC52.

        2. "PRINT ALTERNATE SECTORS?".

            A. A "YES" answer will cause the program to display the alternate-sector map when executing the Initial test-block.

            B. A "NO" answer will cause the program to display only the current number of alternate sectors.

5. B. 3. "HALT AFTER ERROR?". A "NO" answer allows the program to continue testing a disk even though that disk has failed one of the tests. Also, if the disk being tested becomes INOP, a "NO" answer allows the program to skip to the next disk that was specified by the operator.

4. The program will ask "ENABLE TICK-MODE TEST?". A "NO" answer directs that the program skip the tick-mode test in the Extensive test-block. A "YES" answer directs that the program execute the tick-mode test only when at least 4 iterations of the Extensive test is specified (in step 5B5A below). For more information about the tick-mode test, see "Tick-mode-test Logic" on page 011.

5. "ENTER ITERATION-COUNTS (HH)?".

A. A "YES" response allows the operator to enter a separate 1- or 2-hex-digit iteration-count for every test-block, in order to get the option of establishing different emphases upon different test-blocks. When circumstances indicate the desirability of running the test for a long period of time, the operator may specify a "01" iteration-count for the INITIAL and the ADDRESS tests, "03" for the HARMONIC test, "20" for the RANDOM test, and "0A" for the EXTENSIVE test. There is little advantage in executing the ADDRESS test more than once, or in executing the INITIAL test more than three times. An iteration-count of at least 04 should be specified for the extensive test to assure that every sector on the disk is tested. Entering nothing (just pressing RETURN) = entering "00".

B. A "NO" response is a choice to execute every test-block of the automatic part the same number of times. This number must be specified in step 6 below.

6. If the operator has not enabled in step 5B5 above the option permitting the specification of separate iteration-counts, the program will request "ITERATION-COUNT (HH):". Type a one- or two-hex-digit number to execute the test-blocks that many times.

The program will now execute test-blocks as directed through the iteration-counts. All testing of one disk is completed before the testing of another disk is begun.

While the test is running, the operator can monitor and control the test with flags 2 and 3:

F2 reports progress;
F3 reports progress and terminates the current block;
F2 followed by F2 prints an error-summary; and
F2 followed by F3 aborts the test (no error-summary).

When the program reports "TEST COMPLETED.", it will sound the alarm of device 0 and will instruct "HIT EITHER FLAG TO RESUME PROGRAM.". Then when the operator hits either flag, the program will re-start.

THE SIX TEST-BLOCKS

The program will perform these blocks "automatically" (without a need for operator-intervention). The execution-time for one iteration of each test-block run on a Q30 is shown below.

EXECUTION-TIME (in minutes) FOR 1 ITERATION OF EACH TEST-BLOCK

	DISK-DRIVE (identified by capacity expressed in megabytes)									
		Fujitsu			Marksman			Memorex		
	67	84	134	168	10	20	40	25	50	75
INI	.	0.04	.	.	.	0.05	.	.	.	.
ADD	.	1.06	.	.	.	0.36	.	.	.	.
PAT	.	1.02	.	.	.	1.18	.	.	.	.
HAR	.	3.16	.	.	.	1.50	.	.	.	.
RAN	.	1.04	.	.	.	1.43	.	.	.	.
EXT	.	9.67	.	.	.	2.60	.	.	.	.
	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
tot	.	15.99	.	.	.	7.12	.	.	.	.

The execution-time for any block except EXTENSIVE is the product of the execution-time for one iteration of the block and the number of iterations executed. In contrast, the execution-time for the EXTENSIVE block varies according to the following table.

iterations executed	multiplier of the execution-time for 1 iteration of EXTENSIVE
2	3
3	6
4	10
5	11
n	2n, wherein "n" is any integer > 5

DISC52 TEST-LOGIC

Error-output and progress-reports are keyed to a code composed of two or three fields. The first field is the first three letters of the name of a test-block. The second field, separated from the first by a blank, is a letter designating a section of the test-block. The third field is a decimal digit designating a subsection (if one exists) of the section.

Initial Test (INI)

A. Read the alternate-sector map. Then, depending upon option 5B2, display either the alternate-sector map, or the number of alternate sectors.

B. Cursory Data-transfer Test
A seek precedes each write and each read.

1. Write "00"s to a sector (S0) on the FE cylinder, head 0.

2. Write "FF"s to a sector (S1) on the FE cylinder, head 1.

3. Read S0, checking for "00"s.
4. Read S1, checking for "FF"s.

C. Check Termination-interrupt Feature.

Address Test (ADD)

A. Check whether all sectors on a track can be addressed.
B. Check whether all heads on a cylinder can be addressed.
C. Check whether all cylinders can be addressed.

Pattern-test (PAT)

A. Buffer-turn-around Test. A seek-instruction is issued to sector 00000 at the beginning of this test, resulting in reads of the controller's buffers, not of the disk-pack. No status-checking or retry is attempted. Each write in Section A is immediately followed by a read-and-check-data operation. Data-patterns are as follows:

1. The 256 hex characters are written three times in descending order (FF through 00). Single write and read.

2. The last 762 positions are filled with the same ASCII character. 256 write/reads. The first fill-character is FF, and the last is 00. When the fill-character is FF, a three-second time-delay is inserted between the write and the read to determine whether the buffer can retain data over an extended period of time.

3. Pairs of characters fill the last 762 positions. Each pair is composed of FF's complements. The first pair written is 00FF, the next pair is 01FE, the next is 02FD, etc. There are 128 write/read attempts.

4. The last 762 positions are filled with 761 $00's and one $FF. The $FF assumes different positions within the field. On the first write/read the $FF occupies the 7th position, on the second write/read it occupies the 14th position, on the third it occupies the 21st position, etc. There are 108 write/read attempts.

5. The last 762 positions are filled with 761 $FF's and one $00. The $00 assumes different positions within the field. On the first write/read the $00 occupies the 7th position, on the second write/read it occupies the 14th position, on the third it occupies the 21st position, etc. There are 108 write/read attempts.

B. Write to Disc. A seek-instruction precedes every write. The read-after-write check is inhibited to save time. Consecutive sectors starting from 00000 are written, and on each write the seek-parameter is increased by one (with an add-decimal instruction). The data-patterns are identical in content and in sequence to those used in the buffer-turn-around test (Section A).

C. Read from disk. The data written in section B are read and checked. If a status-error occurs during the read, the data-check is skipped.

Harmonics Test (HAR)

Pairs of seek-and-write or seek-and-read instructions are issued to the disk. The seek-addresses change so that the head travels back and forth across the disk for each pair of seeks in a constantly varying manner.

The pattern of seeks is generated as follows: the first pair of seeks is to track 000 and the last track. In each consecutive pair of seeks the track-number of the first seek is increased by one, and the track-number of the second seek is decreased by one. These changes continue until the first seek is to the last track, and the second seek is to track 000.

A. Write-Phase. Writing is done with track-check and read-after-write check. Seeks to the first address in the pair are followed by writes to sector 1, and seeks to the second address are followed by writes to sector 2. The data written are the seek-parameter block, the pass-number, and hex zeros.

B. Read-Phase. Reading is done in the same order used in the write-phase. The first seven bytes are checked after each read.

ERRORS

Errors can occur on either the write- or the read-phase. Errors are counted, and the sector-addresses at which they occurred are saved in a table, which is printed after the phase is completed. If a status-error has occurred, the status is printed below the sector-address at which the error occurred. On the read-phase, if the status is correct, but the data read are incorrect, the letters "DE" (for Data-Error) are printed below the sector-address where the error occurred. The program makes no attempt to analyze or to correct any error. Probably only invalid seeks and data-errors are significant. HOWEVER, DATA-ERRORS OCCURRING DURING THE READ-PHASE ON SECTORS THAT HAD INVALID-SEEK STATUS DURING THE WRITE-PHASE SHOULD BE IGNORED.

Random Write/Read Test (RAN)

A. Write-Phase. Four hundred writes to random sector-addresses with pseudo-random data are performed with read-after-write checking.

B. Read-Phase. The sectors that were written in the write-phase are read, and the data are checked. Any temporary read-error is reported.
Additional iterations of the test do not repeat the same sequence of sector-access of previous iterations.

Extensive Write/Read Test (EXT)

The purpose of this test is to detect errors. It serves no other purpose. It is difficult to use as a diagnostic tool. Its operation is difficult to understand. Consider thyself forewarned.

A. Writing is done in the following manner. Each pass writes one sector on each track, starting at track 000. On each consecutive write the sector-number increases by 111 with no carries. Twenty passes make a scan, and ten scans are required to write the entire disk. The data written are composed of the parameter-block, a five-digit write-number, and a two-byte fill-character in the last 756 positions. The fill-characters used are 6DB3, 36DB, B6D9, and FFFF. The read-check-after-write is inhibited to save time.

B. Reading is done with the same seek-pattern used to write the disk. A single read-scan follows every write-scan. The program will report any write- or read-error.

The number of write/read-scan pairs depends upon the iteration, as shown below.

iteration-count	number of write/read-scan pairs
1	1
2	3
3	6
4	10
5	11
n	2n, wherein "n" is any integer > 5.

C. After ten scans every sector on the disk will have been written. Then the program performs a uniqueness-test, in which every sector on the disk is read, and the first twelve characters are checked. Reading is done by sequentially incrementing the seek-parameter. After the uniqueness-test the procedure is restarted from the beginning.

Tick-mode Test

To understand the tick-mode test, remember these 6 points.

1. The tick-mode test is executed only when at least 4 iterations of the Extensive test have been completed.

2. While the tick-mode test is running, do not end the test (and do not IPL) because ending the test may leave a sector ticked, and this condition may confuse a customer.

3. Tick-normal Mode. When the program writes to a sector, it will set the tick-indicator of the sector.

4. Tick-reset Mode. (Even) when the program writes to a sector, it will not set the tick-indicator of the sector.

5. Tick-scan Mode. The controller will scan a specific sector-range for sectors which have been ticked, and will return to the processor the number of each ticked sector located.

6. When the tick-mode test is enabled, the program runs in tick-reset mode for all test-blocks except for the Extensive.

<u>Tick-mode-test Logic.</u> Before the program starts the Extensive test, it enables tick-normal mode so that setting the tick-indicator of a sector will accompany writing to the sector. After EXT A and B are done, all sectors should have been ticked. When the program has completed the uniqueness-check in EXT C, it activates tick-scan mode, wherein the controller(s) should be expected to return to the CPU the number of every sector on every disk. Then the program enables tick-reset mode and writes ($DB) to every sector, turning-off the tick-indicator for every sector. Then the program re-activates tick-scan mode. This time, the controller is expected not to return any sector-numbers. If it returns any, the error will be reported.

FRODO : FAST REALIZATION OF DISC-OPERATIONS

OPERATING-INSTRUCTIONS

. In using FRODO the operator may select and perform certain diagnostic routines. After FRODO has begun operation, DISC52 cannot be resumed from the point at which FRODO began; instead, DISC52 can then be started only at its beginning. The capability of operating the disk currently under test by DISC52, through the entry of a series of codes representing I/O, test-, and housekeeping-functions, has been implemented through the use of an interpreter-routine (in the last program-block of DISC52) called FRODO. Elements of this system include an interpreter-program, a set of instruction codes, a read-buffer, a write-buffer, and two seek-parameters. Contents of the write-buffer and of the seek-parameters may be modified by the operator during program-execution.

INITIALIZATION

FRODO may be initialized by any one of six methods. These methods are outlined below and are then summarized in the table that follows the outline. One method of initialization is group 1 of the outline, another is group 2, and the others are the four possible variations in group 3.

```
1.  During the initialization of DISC52
    A.  "SELECT OPTIONS", and
    B.  select "FRODO ONLY".

2.  A.  During the initialization of DISC52
        1.  "SELECT OPTIONS",
        2.  do not select "FRODO ONLY", and
        3.  do not select "Halt-after-Error",
    B.  complete the initialization of DISC52, and
    C.  branch to $1000.

3.  A.  During the initialization of DISC52, either
        1.  do not "SELECT OPTIONS", or
        2.  A.  "SELECT OPTIONS",
            B.  do not select "FRODO ONLY", and
            C.  select "Halt-after-Error",
    B.  complete the initialization of DISC52, and
    C.  either
        1.  branch to $1000, or
        2.  during a Halt-after-Error press
            A.  Flags 2 and 3 and
            B.  ST-SP.
```

In the following table the six methods of initializing FRODO are keyed to groups in the outline above.

method	group
1	1
2	2
3	3A1 + 3B + 3C1
4	3A1 + 3B + 3C2
5	3A2 + 3B + 3C1
6	3A2 + 3B + 3C2

When FRODO is initialized, the last 762 positions of the write-buffer are filled with hexadecimal 00's, and the seek-parameters are set to 000000 and 021111.

USER-PROGRAM ENTRY

The user-program is composed of a series of two- or three-character codes. Each code will cause a specific function to be performed. Some codes require the entry of one, two, or three hexadecimal or decimal digits immediately following the code. Codes are entered from device zero's keyboard and may be separated from other codes by spaces and/or carriage-returns, but no separation is required. The program will request "INSTRUCTIONS:". The operator may now enter all the codes needed to perform the desired task. If a typographical error is made before the line is terminated, the operator may backspace to the error and retype the line correctly from that point. After the last line of the user's program has been entered, the program-entry phase must be terminated by pressing Flag 2 and/or Flag 3 and then TERM or Carriage-Return. These flags select one of the following three execution-modes for the user's program.

1. Flag 2 causes the program to be executed at machine-speed and to be repeated until stopped by the user.

2. Flag 3 causes the user's program to halt after each instruction is executed, and to be repeated until stopped by the user.

3. Flags 2 and 3 cause the user's program to be executed once at machine-speed, and then FRODO will return to requesting "INSTRUCTIONS:".

USER-PROGRAM EXECUTION

After program-entry has been terminated, the interpreter will start executing each coded instruction in turn. The first step in the execution of each instruction is the recognition of the code. If the interpreter cannot recognize a code, it will display "ILLEGAL ENTRY", and will then return to requesting "INSTRUCTIONS:". If the interpreter recognizes the code, it will proceed to check device zero's flags. If any of these Flags are on, the interpreter will procede according to the instructions given by the Flags as described above.

FRODO CODES

After recognizing the code and checking for device zero's Flags, the interpreter will perform the operation specified by the code. Following is an explanation of these operations, in ASCII order of the codes.

CODE OPERATION

ACD Add constant to data. When overflow occurs, skip to next "EL".

ACH Add constant to head. The head-number of seek-parameter #1 is increased by one. If the head-number overflows, the cylinder-number is increased by one. When the highest seek-parameter allowed is exceeded, seek-parameter #1 is reset to zero, and the interpreter automatically skips to the code following the next EL instruction of the user's program.

ACS Add constant to sector. The sector-number of seek-parameter #1 is increased by one. When the sector overflows from 9 to 0, the head-number is increased by one. If the head-number overflows, the cylinder-number is increased by one. When the highest seek-parameter allowed is exceeded, seek-parameter #1 is reset to zero, and the interpreter automatically skips to the code following the next EL instruction of the user's program.

ACT Add constant to track. The cylinder-number of seek-parameter #1 is increased by one. When the highest seek-parameter allowed is exceeded, seek-parameter #1 is reset to zero, and the interpreter automatically skips to the code following the next EL instruction of the user's program.

BL, EL Begin loop, end loop. A BL instruction must immediately preceed, and an EL instruction must immediately follow, any portion of the user's program which is to be repeated. The instructions between the BL and the EL instructions will be repeated until ended by one of the following three events.

1. The user presses Flag 2 or 3 on device zero. If this occurs, the program will continue to execute any remaining instructions in the loop, but will fall through the EL instruction and will then continue with the remainder of the program following the loop.

2. A CSxx instruction followed by an SOF instruction does not detect the expected status, or a CD instruction followed by an SOF instruction detects a data-error. In either case, the loop is immediately terminated.

3. An ACS, an ACH, or an ACT instruction increments seek-parameter #1 past the highest valid sector-number. In this case the loop is immediately terminated.

CD Check data. The contents of the read-buffer are compared to the contents of the write-buffer. If an error occurs, and if the SER instruction has not been executed, the error will be displayed.

CH	Compare header.
CRC	Print CRC of sector just read.

CSxx Check for status xx. For 250 milliseconds this routine tries to obtain the status-byte specified by the user (represented by xx). If the status fails to become the required configuration within 250 milliseconds, the last status-byte obtained is output to device zero. If the SER instruction has been executed, errors are not reported.

CTxx	Send device-control xx to the IOU-52.
DAS	Disable alternate-sectoring.
EAS	Enable alternate-sectoring.

ED Enter data. Hex data are read into the write-buffer from device zero during user-program execution. If one or two bytes are entered (two or four hex digits), then those characters will be propagated through the entire write-buffer. Otherwise, up to one typewriter-line of data may be placed into the first 64 positions of the write-buffer following the seek-parameter. On CRT's only one character may be entered. This will fill the write-buffer.

EL See BL, EL.

ES1 Enter seek-parameter #1. The user may enter a 6-digit decimal number during the execution of the user's program. This number becomes seek-parameter #1 and is used when the SK1 instruction is executed.

ES2 Enter seek-parameter #2. Same as ES1 except that the number becomes seek-parameter #2 and is used for the SK2 instruction.

IRC Inhibit read-check. An inhibit-read-after-write-check instruction (CTL 02) is issued to the disk, causing the disk-controller to skip the read-after-write check (and possibly also the rewriting of sectors on which a write-error has occurred) on the next write-instruction.

ITC Inhibit track-check. An inhibit-track-check instruction (CTL 01) is issued to the disk, causing the disk-controller to skip the automatic track-check that would have been performed on the next seek-instruction.

MRW Move read-buffer to write-buffer.

PCS Print changing status. Every 425 micro-seconds a status-in instruction is issued to the disk. The status obtained is kept in a stack. If the status changes within 250 milliseconds, the new status is placed in the stack. This procedure continues until the status has remained constant for 250 milliseconds, at which time all the different status-bytes obtained are displayed.

PE Fetch the status-buffer from the IOU-52, and display it.
PHD Print the header of the sector just read.

PRB Print read-buffer. Same as print write-buffer except that the read-buffer is printed.

PS Print status. Status is printed after the current operation is completed (after END has become set, or after BUSY has become reset).

PWB Print write-buffer. The hex contents of the write-buffer are output on device zero. Flag 2 terminates the current line, and Flag 3 terminates the operation.

RB Read the device-buffer.

RD Read disk. Data are read from the disk into the read-buffer. Before the read is performed, the first sixteen bytes of the read-buffer are cleared to hex zeros. The read-routine does not include a seek, a status-check, or data-checks.

RIO Reset I/O. A reset-I/O instruction is sent to the disk. On the IOU-52, reset I/O sets invalid-seek status, resets Flag 1, and resets INOP if the INOP was caused by the FPT flip-flop being set. --Fpt is set by writing to a file-protected disk.-- Invalid-seek status sets END status. When END becomes set, service-request status will be reset. When invalid-seek is true, and service-request is false, as should occur after RIO, an RES2-N signal is generated in the controller. RES2-N resets the RD, the WRT1, the WRT2, the WRT3, the WRT4, the ERR, and the MRK flip-flops.

RM	Micro read.
RS	Read status 1.
RTN	Return to main program.

SER Suppress error-report. The display of all error-reports is suppressed after this code is encountered. Instead, signal-lights one and two will blink every time an error occurs. Error-reporting is automatically re-enabled during program-entry.

SK1 Seek 1 A seek using seek-parameter #1 is performed. Seek-parameter #1 is placed into the first six positions of the write-buffer. When FRODO is first started, seek-parameter #1 is set to 00000.

SK2 Seek 2 Same as SK1 except that seek-parameter #2 is used. When FRODO is first started, seek parameter #2 is set to 21111.

SOF Skip on failure. This instruction follows a CD (check data) or a CSxx (check for status xx) instruction. If a failure occurs in the previous instruction, the interpreter will skip the remaining instructions in a loop and will leave the loop, or will restart the program. Thus, if the CD instruction finds a data-error, or if the CSxx instruction fails to detect the expected status (xx), the SOF instruction will skip to the instruction of the user's program following the next EL instruction. If no loops occur in the program, the program will be restarted from the beginning.

SR	Set read.
SWR	Set write.
TER	Terminate.
TR	Fetch the trace-buffer from the IOU-52, and display it.

TSP Type seek-parameters. Seek-parameters 1 and 2 are output to device 0.

WM Micro-write.

WRT Write to disk. Data in the write-buffer are written to the disk. The write-routine does not include a seek or a status-check.

THE USE OF WRITE- AND OF READ-BUFFERS

The write- and the read-buffers used by FRODO occupy the same memory-locations as the write- and the read-buffers used by DISC52. All writes and all reads use these buffers. When a seek is performed, the seek-parameter used to access a sector is always placed in the first six positions of the write-buffer. Thus, the sector-address is always the first datum written to every sector.

DISC52 ERROR-CODES

Subroutine Errors

d2a A short write has occurred. The number of characters transferred from memory to the disk-controller is less than the count specified in the Write-instruction. The actual hex number of characters transferred is written above the error-message line.

d2b The write-routine has detected invalid-seek status after completion of a seek-instruction. Two consecutive invalid seeks must occur before this error occurs the first time.

d2c The write-routine has detected a status-condition from which it cannot recover. This error-code is output under any of the following conditions:

1. An attempt to recover after an invalid seek has produced neither good status nor invalid-seek status.

2. Unrecoverable error-status was detected after the write. This could indicate a very badly damaged disk.

3. Marked-sector status was detected after the write, but the read-after-write check was inhibited at the time. This probably indicates a failure of the controller.

4. One of nine re-writes after marked-sector produced neither good nor marked-sector status.

d2d After a read- or a write-operation, an error-status was detected. The types of error and their corresponding error-status are as follows.

ERROR-STATUS	TYPE OF ERROR
$24	irrecoverable
$44	marked sector
$64	invalid seek
$84	drive not ready or drive write-protected
$B4	alternated sector not found in map
$D4	drive not formatted
$F4	serious drive-problem

d2j A short read has occurred. The number of characters read by the processor is less than the count specified in the read-instruction. The actual number of (hex) characters transferred is printed above the error-message line.

d2k The read-routine has detected invalid-seek status after completion of a seek-instruction. Two consecutive invalid seeks must occur before this error occurs the first time.

d2l Read-error status was detected after a sector was read -- a sector that had previously been written without the automatic read-after-write check. A rewrite was immediately initiated, but the write-routine has reported that the rewrite was unsuccessful for some reason. Errors that occurred during the rewrite were printed, but the error-routine was prevented by the rewrite-condition from halting at that time.

d2m The read-routine has detected a status-condition from which it cannot recover. This error-code is output under any of the following conditions:

1. An attempt to recover from invalid-seek has produced neither good status nor invalid-seek status.

2. Error-status was detected on a sector which had been previously written with the automatic read-after-write check enabled.

3. Marked-sector status was detected on a sector which had been previously written while the automatic read-after-write check was disabled.

d2r The data-check subroutine has detected an error during the Pattern-block, the Random Write/Read block, or the Extensive Write/Read block. The four lines following the error-code line contain a summary of the error. This error is caused by a difference between the data read from a sector on the disk and the data last written to that sector. The program does not halt after this type of error.

Initial-test Errors

D4L An attempt to write all hex zeros to FE cylinder, head 0, has resulted in some kind of error-status.

d4m An attempt to write all hex FFs to FE cylinder, head 1, has resulted in some kind of error-status.

d4n An attempt to read all hex zeros from FE cylinder, head 0, has resulted in some kind of error-status.

D4Q An attempt to read all hex FFs from the FE cylinder, head 1, has resulted in some kind of error-status.

D4X The controller has failed to cause an interrupt after interrupt-mode was enabled, and a Seek-instruction was issued and allowed 500 milliseconds to complete.

d4y A read did not reset the pending interrupt, or the disable-interrupt (write-control) instruction did not reset the pending interrupt, or the disable-interrupt instruction did not work.

d4z Although interrupt-mode had been enabled, a write-instruction did not cause an interrupt upon its completion.

Address-test Errors

d5a An attempt to write, and then to read, all sectors on head 0, track 000 has failed. All writes and all reads to the track on which the test was run completed without any error-status being reported.

Only the first six bytes of the block were checked on the read-phase. The error can be caused either by a failure to read or to write data correctly, or by a failure of the disk to seek to the unique locations specified by the seek-instruction. It is the latter of the two that the test is intended to check, but it should be kept in mind that data-errors will invalidate this particular test.

d5b An attempt to write, and then to read, all heads (sector 0) on track 000 has failed. All writes and all reads to the track on which the test was run were completed without any error-status being reported. See the second paragraph of d5a above for additional explanation.

d5c An attempt to write, and then to read, all tracks (head 0), using sector 0 has failed. All writes and all reads to the sector on which the test was run were completed without any error-status being reported. See the second paragraph of d5a above for additional explanation.

d5d An attempt to do section A, B, or C of the address-test has failed after the second "automatic" restart was unsuccessful. Three status-errors were encountered while writing or reading the section. Pressing ST-SP causes restart of the current section from the beginning. Pressing Flag 2 and ST-SP causes this iteration of the address-test to be aborted.

Pattern-test Error

d6a A seek to sector 00000 has failed to give correct status 1/2 second after the seek-instruction was issued.

Harmonic-test Error

d7a The Harmonic Test has failed. Pressing ST-SP allows repeating the test from the beginning. The two kinds of errors possible on the read-phase of the test are status-errors and data-errors. Data-errors are identified by a "DE" printed below the address of the sector upon which the error occurred. A data-error occurs when the status for the read is correct, but the first seven bytes of the block are incorrect. Status-errors are reported by printing the status that occurred (when the block was read or written) under the sector-address where the error occurred. The program will report only the first sixteen errors to occur.

Extensive-test Errors

d9a The uniqueness-check (performed after every tenth scan of the extensive-test) has failed. The hex representation of the first six characters of the sector on which the failure occurred is printed on the line above the error-message. Data-errors can cause this failure.

d9b The uniqueness-check (performed after every tenth scan of the extensive-test) has failed in an obscure fashion. This failure can be caused by a data-error.

DISC52 ERROR-HANDLING

After DISC52 reads a sector, if it then detects unrecoverable-status ($24), it will perform in sequence the steps described below unless some condition (as described) forces a different sequence (specified).

1. Re-read sector, and check status. If no error, increment soft-error count, and return.

2. Increment hard-error count. If sector was written with verify, skip to step 5.

3. Re-write sector with verify, and report "SECTOR RE-WRITTEN WITH VERIFY.".

4. Search alternate-sector map for sector. If found, report "SECTOR HAS BEEN ASSIGNED AN ALTERNATE.", and return.

5. Force alternate sector.

6. Search alternate-sector map for sector. If found, report "SECTOR HAS BEEN ASSIGNED AN ALTERNATE.". If not found, report "FAIL TO ALTERNATE SECTOR.".

IOU-52 CONFIGURATION-SWITCH SETTINGS FOR EACH DISK-DRIVE

	Fujitsu				Marksman			Memorex		
capacity in M-bytes	67	84	134	168	10	20	40	25	50	75
switch-settings ($)	91	92	93	94	01	02	03	81	82	83

THE CONFIGURATION OF EACH DISK-DRIVE

drive ***	3 contiguous groups of sectors			s/t **	hd **	cyl **	alt **
	user's	FE's	reserved*				
Fj 67	0--078719	--078815	--079007	24	4	823	168
Fj 84	0--098399	--098519	--098759	24	5	823	216
Fj134	0--157439	--157631	--158015	24	8	823	360
Fj168	0--196799	--197039	--197519	24	10	823	456
Mk 10	0--011759	--011815	--011927	28	2	213	84
Mk 20	0--023519	--023631	--023855	28	4	213	196
Mk 40	0--047039	--047263	--047711	28	8	213	420
Mx 25	0--030535	--030623	--030799	22	4	350	154
Mx 50	0--061071	--061247	--061599	22	8	350	330
Mx 75	0--091607	--091871	--092399	22	12	350	506

*** Fj = Fujitsu Mk = Marksman Mx = Memorex

** s/t = number of sectors per track
hd = number of heads
cyl = number of cylinders
alt = number of alternate sectors

* The Engineering department's "IOU52 External Specification" describes the reserved sectors in more detail.

MEM64 tests system 64's main memory.

```
SUBJECT DEVICE : MEM64

    ROM-version: -

CONTROLLER     : -
    Board-ver. : -
    ROM-version: -

SYSTEM REQUIRED
    CPU-type   : See directory listing
    Min. memory: 32K bytes
```

This program is operable on system 64 series, with MOTHER board and 2 DAUGHTER boards. The amount of memory on any board will be an integral multiple of 512K.

The size of each memory-chip (RAM) that may be tested is 64K. The program assumes that the layout of every chip is 4 square matrix of 64 rows by 256 columns, and that the physical arrangement of the rows and the columns inside the chip is sequentially ordered.

Testing parity-memory features in the MEM64 board is limited. MEM64 will test all memory present (except the lowest 16K), without the operator having to move memory-boards or to reload the test.

INITIALIZATION

When a repetitious operation of the test is required, as in Incoming-inspection, the operator may initialize the test by entering a single character. The operator may, however, select any combination of test-options, iteration-counts, and test-areas desired. One option enables the program to "report chip-failure only once", so that only the first failure of a chip will be reported, any subsequent failure of the same chip remaining unreported.

1. The program will report the actual amount of memory present. The actual memory present must divisible by 512K, otherwise error message will report.

```
CAUTION: INCOMPLETE 512K MODULE DETECTED.
         PLEASE CHECK BEFORE PROCEEDING.
         F2 TO CONTINUE; F3 TO LOADER.
```

2. The program will report the memory-address space that may be tested. The 2 hexadecimal addresses separated by a hyphen are the first and the last addresses of the memory present on the MEM64 MOTHER boar operator is responsible for ascertaining the completeness and the accuracy of this report.

3. The program can operate in any one of six modes:

N - NORMAL	I - INCOMING
O - OVEN	F - FINAL

X - Operator select options, with no summary of test-block errors

' - Operator-select-options, with a summary of test-block errors.

Each of the modes N, I, O, and F differs from the other five modes in the options enabled or disabled. Mode N differs further in the number of iterations executed for each test-block. In mode X or ', the operator may enable or disable any of the options and may also determine the iteration-count for each test-block. The modes relate to the options as shown in the 1st table following, and to the iteration-counts as shown in the 2nd table following.

OPTIONS ENABLED (e), DISABLED ( ), OR EITHER (x)

o p t i o n	N	I	O	F	X	'
Halt after error	e			e	x	x
Report chip-failure once		e	e	e	x	x
Repeat test		e	e	e	x	x
Enable parity-fault	e	e	e	e	x	x
Select a ddress range	x					x

ITERATION-COUNTS may vary from 0 through $FFFF.)

t e s t - b l o c k	m o d e		
	N	I, O, and F	X and '
FILCHK (fill and check)	1	2	x
SADLIN (shorted address-line)	1	2	x
MARCH	1	4	x
ADCOMP (address-complement)	1	4	x
COLSAM (column-sense amplifier)	1	4	x
KLUDGE	1	1	x
DELAY	1	5	x
ROWPAT (row-pattern)	1	1	x
MULFIL (multiple fill)	1	2	x

The program will ask "ENTER MODE (N, I, O, F OR X):". Answer with any one of the seven alternatives described below (A through G).

A. To return to the program loader, hit RETURN.

B. To select mode N either "N" or "n". In this mode the program assumes options as shown in previous tables and enabling selection of address range.

1. Press F2 for 'YES'. The program will request the 'START ADDR:' and 'END ADDR:'. Minimum address range is 16K. The program will use the boundaries specified throughout the test.

2. Press F3 for 'NO'. Testing address range will be the actual memory present.

C. To select mode I enter either "I" or "i". In this mode the program assumes options as shown in previous tables and starts testing.

D. To select mode O enter either "O" or "o". In this mode the program assumes options as shown in previous tables and start testing.

E. To select mode F enter either "F" or "f". In this mode the program assumes options as shown in previous tables and starts testing.

F. To select mode X, enabling the selection of any combination of test-options and iteration-counts, but without any summary of test-block errors, enter "X" or "x".

1. For each of the options the program will ask whether to enable the option, by displaying the option-name followed by a "?". Hit either F2 for "YES" or F3 for "NO".

2. The program will ask "ENTER ITERATION-COUNTS?". This question means "Do you wish to enter a separate iteration-count for each test-block, disabling the program from assigning to every test-block an iteration-count of 1?". Hit either

A. F3 for "NO", establishing for every test-block an iteration-count of 1, or

B. F2 for "YES". For each test-block the program will request the iteration-count for the block, by displaying the name of the block. Either

1. hit RETURN to skip the test-block, or

2. enter a 1- to 4-digit hexadecimal number to execute the block that number of times, except that the count for the DELAY block selects the number of seconds of delay, up to 63 ($3F).

G. To select mode ' (single-quote), enabling the selection of any combination of test-options and iteration-counts (the same as in mode X described above), and providing a summary of test-block errors, enter "'", a single quote. Finish initializing this mode as instructed for mode X in steps 2E1 through 2E2 above.

INITIAL TESTING

Before the program begins executing the test-block(s) selected, it tests whether writing to any DAUGHTER board affects any other DAUGHTER board. If the writing affects another board, the program will display an error-message in which the test-block name is "BOUND" (boundary-test).

EXECUTION

Test-blocks are executed in the following order, the same order in which their names appear when iteration-counts are being entered.

FILCHK
SADLIN
MARCH
ADCOMP
COLSAM
KLUDGE
DELAY
ROWPAT
MULFIL

A block is executed as many times as is necessary to satisfy the block's iteration-count before the next block is started.

After all iterations of all blocks are completed, the program will state "TEST COMPLETED.", will alert the operator to the completion by beeping the alarm of device 0, and will instruct "TO RESTART HIT ANY FLAG.". To silence the alarm and to restart, hit any flag of device 0.

FLAG-FUNCTION DURING EXECUTION

If Flag 2 is pressed while a test-block is being executed, the test will display a progress-report consisting of the current block's name and iteration-number. If Flag 3 is pressed, the progress-report will be output, followed by the word "TERMINATED", and the program will immediately terminate execution of the current block and will begin executing the next one. Note that there may be up to several seconds delay before the program responds to a flag that has been turned on. If Flags 2 and 3 are both turned on within one second of each other, a progress-report and an error-summary will be output. If Flag 2 is pressed, and after two second, F3 is pressed, a progress-report and 'TEST ABORTED' will be output.

ERRORS

Up to 100,000 errors are allowed before the program outputs the message "ERROR-OVERFLOW; TEST ABORTED." and halts. If an error occurs, the following information will be output:

1. The current test-block's name and iteration-number. (This is not output on subsequent errors on the same block unless the iteration number changes, or the name and the iteration-number are about to roll off the CRT screen.)

2. The address at which the error occurred.

3. The actual contents of that address shown as eight binary bits.

4. The correct data that should have been at the address, shown as eight binary bits.

5. The address of the pattern-location, output only for certain test-blocks.

```
    Example:     ADDR        CONTENTS     CORRECT      PADDR

                 0040FF      11111111     00000000     4000
```

If the "HALT AFTER ERROR" option is not selected, the program will continue executing the current test-block after the error-message is output. Because of this, it is strongly recommended that the "REPORT CHIP-FAILURE ONCE ONLY" option be used when not halting after error, especially if no-one is present while the test is running.

If the "HALT AFTER ERROR" option is selected, following the error-halt, the operator may turn on Flag 2 and/or Flag 3 to achieve the following results:

Both Flags off to continue with the current test-block.

Flag 2 on to continue the current block, but to skip to the next test-area.

Flag 3 on to skip to the next test-block.

Flags 2 and 3 on to output an error-summary and to halt.

After the operator presses ST-SP, Flags 2 and 3 will perform the functions described above.

Transient errors occur when a long scan or a long compare-instruction finds an error, but the data is correct a short time later when the error-routine tries to report the error. The address-range in which the error occurred is reported.

MEM64 TEST-BLOCK LOGIC

As an aid in understanding this document, the following explanation of some of the terms used is included. "Low to high" means starting from the lowest memory-address and progressing to the highest memory-address in the area being tested. "high to low" has the opposite meaning. "Chip-size areas" are 64K. A "checkerboard-pattern" is a pattern of alternating 00's and FF's moved into a chip in such a way that each 00 will be surrounded by two, three, or four FF's, and each FF will be surrounded by two, three, or four 00's. This is accomplished by moving the column-length string "00FF00FF00FF..." to all even-numbered columns, and the column-length string "FF00FF00FF00..." to all odd-numbered columns. Example:

```
00 FF 00 FF
FF 00 FF 00
00 FF 00 FF
FF 00 FF 00
```

"All memory" refers to that portion of memory that is to be tested. It is all memory that lies above the first 16K area.

"Complementary data" means FF used in place of 00, and 00 in place of FF.

THE TEST-BLOCKS

FILCHK

Fill and Check memory eight times per iteration.

1 and 2. Fill and check all memory, low to high, for 00 and then for FF.

3 and 4. Fill and check all memory, high to low, for 00 and then for FF.

5 and 6. Fill and check each 16K area, low to high, for 00 and then for FF.

7 and 8. Fill and check each 16K area, high to low, for 00 and then for FF.

SADLIN

Shorted-address-line test

Part 1 tests all D-boards. Part 2 tests the MOTHER board. All filling and all checking are performed low to high.

In Part 1 the D-board is filled with 00's. The first address on the D-Board is selected as the first pattern-address. An FF is moved to the pattern-address, and all other locations are checked to see if they contain 00's. This procedure is continued with different pattern-addresses, which are generated by ORing a single one-bit onto the first board-address. For example, pattern-adresses for the test area containing addresses 4000--7FFF would be 4000, 4001, 4002, 4004, 4008, 4010, 4020, 4040, 4080, 4100, 4200, 4400, 4800, and 5000 etc.

Part 2 performs a similar operation on the MEM64 board, the only difference being the larger address-range.

MARCH

Marching ones and zeros.

All memory is filled low to high with 00's. Proceeding from low to high, one byte at a time is checked for 00 and then rewritten with an FF. When all memory has been checked and rewritten, it is checked low to high for all FF's. (If any errors occur, all memory is rewritten with FF's.) Then, proceeding high to low, one byte at a time is checked for FF and then rewritten with a 00. When all memory has been checked and rewritten, it is checked high to low for all 00's.

The procedure described in the proceeding paragraph is repeated, but complementary data are used.

ADCOMP

Address-complementing test

This procedure is perfomed on test-area size areas, proceeding from low to high memory.

A checkerboard-pattern is moved to the test-area. Then the data are checked and rewritten as follows: at test-area-address zero, check data, and write data-complement; at test-area-address maximum, check data, and write data-comple ment at test-area-address zero, check data-complement. Then this process is repeated for test-area-address one and maximum-minus-one. This entire process - -read and write low address, read and write high address, read low address, increment low address, decrement high address -- continues until the low and the high addresses meet in the middle of the chip. If no errors occurred, then the complementary checkerboard-pattern is checked high to low.

After this procedure has been performed on all memory, it is repeated starting with the complementary checkerboard-pattern.

COLSAM

Column-sense-amplifier test

This test is performed one column at a time (64 bytes) and proceeds from low to high memory. First a series of 00's followed by a single FF is written and checked. Then a series of FF's followed by a single 00 is written and checked. Then the next column is tested. This continues for all memory.

KLUDGE

This test is performed two bytes at a time on all memory, proceeding high to low. It is similar to Block 7 in MEM.

The pattern data is initially set to FFFF. This data is moved to the highest two bytes in memory. Then the data is decremented by 1 and moved to the next two bytes. This continues until all memory is filled. Then all memory is checked, high to low, two bytes at a time.

Next the initial-pattern's value is decremented by 4K (1000 hex), and the fill-and-check is repeated. This continues until memory has been filled and checked sixteen times.

DELAY

All memory is filled low to high with a checkerboard-pattern. Then a delay occurs during which none of the chips being tested is accessed. The delay-time in seconds is equal to the iteration count or 63, whichever is less. After the delay all memory is checked high to low.

This procedure is repeated with the complementary checkerboard-pattern.

ROWPAT

Pingpong between columns within a row.

All memory is filled with 00's. Then, each row in a 16K test-area is tested, one at a time. The pattern-address is set to the memory-address which is the first column in the row. The pattern-location is checked for 00. The FF is put into the pattern-location. Now the following "ping pong" routine is executed: the pattern-location is checked for FF, a memory-cell in the same row is checked for 00, the pattern-location is checked for FF, another memory-cell in the same row is checked for 00. This is repeated until every non-pattern location in the row is checked for 00. Then a 00 is put back into the pattern-location, and the next cell in the row becomes the new pattern-location. The new pattern-location is checked for 00, and receives an FF. The ping-pong routine is executed for this pattern-address. This entire procedure is repeated until each cell in the row has been the pattern-address once. Then the program does the next row in the chip.

When this is completed for all memory, it is repeated using complementary data.

MULFIL

Multiple-Fill Test (similar to Block 6 on MEM)

The test is executed low to high on 16K test-areas of memory.

The 16K test-areas is filled with 00's. The first address of the test-area becomes the pattern-address. An FF is moved to the pattern-address for a period of 2 milliseconds. Then (on the first iteration) the data in the pattern-address's row and column are checked to make sure they are unchanged. (On subsequent iterations the entire test-area is checked to make sure it is unchanged.) The 00 at the pattern-address is restored, the pattern-address is incremented, the 2-millisecond fill is performed, and the row-and-column (or the test-area) check is performed. This procedure is repeated until the pattern-address reaches the end of the test-area.

Then the next test-area is checked in the same way. When all memory has been checked, the complementary data-pattern is used, and the test-block is executed again.