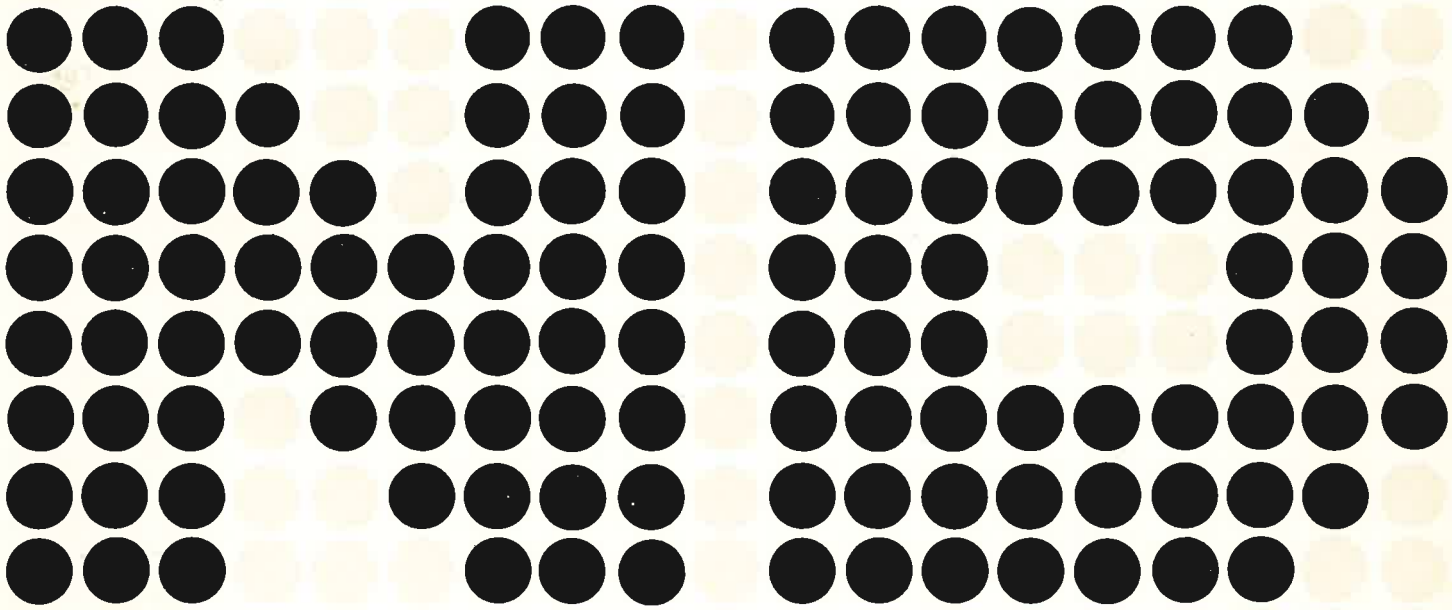
TEST PROGRAM DESCRIPTIONS

NORSK DATA A.S



TEST PROGRAM DESCRIPTIONS

REVISION RECORD

Revision	Notes
08/76	Original Printing
12/76	Second Edition
02/77	Third Edition - supersedes previous editions
05/77	Revision A. Revised page iii and added Chapter 14.
06/77	Revision B. Revised page iii and added Chapter 15.
09/78	Revision C. Added Chapter 16.
11/78	Revision D. Added Chapter 17.

Test Program Descriptions
Publication No. ND-62.009.03

NORSK DATA A/S
Lørenveien 57, Postboks 163 Økern,
Oslo 5, Norway

TABLE OF CONTENTS

\+ + +

Section:		
1	Test Reader and Punch TREPU	HAR 1269
2	How to Use PFAIL	HAR 1355
3	Test Paging	HAR 1358
4	Real Time Clock Stability Test	HAR 1397
5	Real Time Clock	HAR 1399
6	8K MOS Memory Test Program	HAR 1821
7	MOVER Memory Test Program	HAR 1863
8	TECOD Test Program	HAR 1451
9	DIMS User's Guide	HUT 1453
10	BIMS Users Guide	HUT 1872
11	MCOPY User Description	HUT 1649 - 1651
12	DRUMS User Description	HUT 1297
13	TCODR User Description	HAR 1299
14	CACHE Test User Description	HAR 2063
15	ERRCOR Test Program for Error Correction Logic and 21 Bits Memory Modules	HAR 2111
16	GREMS User Description	
17	FILE SYSTEM INVESTIGATOR Users Guide	

1 TEST READER AND PUNCH TREPU - HAR 1269

Patch if no punch test:

addr.	old	new
24/	135057	0

Read in the program. It starts automatically by printing:

TEST READER AND PUNCH.

Start tape reader interface test.

After this, the reader interface is tested without the reader. All the 256 possible characters are generated in the interface, read by the program, and checked. After the test, the program prints:

END TAPE READER INTERFACE TEST

START TAPE PUNCH INTERFACE TEST.

Then, the punch interface is tested without the punch. All the 256 possible characters are put into the buffer register and then read back again and checked. After the test, the program prints:

END TAPE PUNCH INTERFACE TEST

Type P (Punch), R (Read), S (Stop),
or anything else (octal delay).

The usual way of using the program is then to type P. The program will then punch a test tape. When the test tape is sufficiently long, type S and the program will stop. Put the test tape into the reader, type R and the program will read the test tape and check it. If anything else than P, R or S is typed, the program will print:

OCTAL DELAY.

The user must then type an octal number, followed by a non-octal character (CR or space). This number will be used in a MIN-LOOP between each character read from the tape or between each group of seven characters punched. An octal delay of 177777 will give the least possible delay, 0 (zero) will give the greatest delay. It is strongly advised to test both the reader and the punch with octal delays like the following:

100000, 140000, 160000, 170000, 174000, 176000, 177000, 177400,

and so on. This will put the greatest possible strain on these devices.

The test tape is punched in this way:

First all the characters from 0 to 0377 are punched. Then this sequence is repeated, but the characters are now shifted (rotated) 1 left. The characters punched will be 0, 2, 4, . . 0376, 1, 3, . . 0377. After this, the shift (rotate) is repeated, and so on.

The following error messages may be printed:

READ: XXXXXX SHOULD HAVE BEEN: YYYYYY
STOP . TO RESTART, TYPE ANY CHAR:

This means that XXXXXX was read when YYYYYY was expected. XXXXXX and YYYYYY are octal numbers. By inspecting the tape in the reader, the user may find out if the tape is punched incorrectly, or if the reader reads correctly. Remember that the character on the tape directly under the lamp is not yet read, if the reader works correctly. A common reader error may be that the reader reads the same character twice, or that it skips one or more characters. A common punch error is to punch two characters on top of each other or that one or more of the bits (channels) on the tape are failing.

After this error message, put the tape into the reader from the beginning and type any character.

TAPE PUNCH NOT READY

The punch is off, out of tape, etc.

TAPE READER NOT READY

Switch on the reader.

Tape reader status error: XXXXXX

This means that some of the unused bits of the status word (bit 1, bits 4-15) have become 1, or that bits 2 and 3 are equal.

Bit 0 is Interrupt enabled
Bit 2 is Reader active
Bit 3 is Ready for transfer

XXXXXX is the failing status word.

TAPE PUNCH STATUS ERROR: XXXXXX

This message is quite similar to that above.

READY-FOR-TRANSFER IS DEAD

This message may occur in both the reader and the punch interface test and means that READY-FOR-TRANSFER never comes on.

ERROR. FAILING AND EXPECTED CHARACTERS ARE:
XXXXXX YYYYYY

This message may occur in both the reader and the punch interface test. XXXXXX is the failing character and YYYYYY is the expected character.

2 HOW TO USE PFAIL (Power Fail Test Program) - HAR 1355

1. Read in the program. It starts in 020.

2. Lock the machine.

3. The program prints:

   POWER FAIL TEST PROGRAM

   version, date

   WHAT IS THE TERMINAL SPEED? etc., etc.

   This is answered by typing one of the digits 0-7.

   The program prints:

   INITIATE (I) OR RESTART (R)

4. Answer by typing I.

5. The program will now find the memory size, and store address in addresses (STX, X)
   If the machine has mini-paging and 64K memory, the program will print 3 error messages that should be ignored, and print a memory address (maximal.) that will be 177377.

6. The register save area will be filled with 070707.

7. The registers will be filled with a predetermined content.

8. When the machine executes a JMP * in level 0, it is ready for power fail. (The interrupt light on the panel is on).

9. SIMULATE POWER-FAIL BY REMOVING AND REINSERTING THE MAIN-PLUG.

10. After power fail (power off, power on), the program starts in 020 automatically.

11. Answer R.

12. The program will now check the memory. If a word does not contain its own address, an error message is printed. Otherwise: MEMORY OK.

13. The register save area will now be checked. If an error occurs, an error message will be printed. Otherwise: REGISTERS OK.

    If there is an error, the register save area might contain 070707. This means that the corresponding register is not saved.

    If the register save area contained 111111, it means that after the registers were saved, IOF and WAIT did not stop the program.

14. After the registers are checked, the program jumps to point 5.

3 TEST PAGING - HAR 1358

Test 1

Memory test of addresses 177400 - 177777, i.e., the page tables. First, in every word of the page table is stored its own address. The address is then read back and checked. After that, the page tables are filled with six different patterns, which are read back and checked. The six patterns are:

000000, 177777, 052525, 125252, 000377, 177400.

In all the following error messages, the expected result is printed first, and then the failing result.

Test 2

Page-not-in-core test. (WPM = RPM = FPM = 0).

The user's page table (page table 0) is filled with zeros, and then the program starts the user (all addresses from 0 to 177777 are checked). This should give "page-not-in-core".

Test 3

Instruction-fetch-not-permitted test. The user's page table (page table 0) is filled with 100004 (WPM = 1, page number 4).

After that, the user program SAA 123: MON 3; JMP *-1, is executed in all virtual addresses, with 125252 in the user's A register. This should give memory protect violation.

Test 4

Instruction-fetch-not-permitted test. The user's page table (page table 0) is filled with 040004 (RPM = 1, page number 4). After that, the user program SAA 0123: MON 4; JMP * - 1, is executed in all virtual addresses, with 125252 in the user's A register. This should give memory protect violation.

Test 5

Instruction-page-in-core test. The user's page table (page table 0) is filled with 020004 (FPM = 1, page number 4). After that, the user program SAA 0123: MON 5; JMP *-1 is executed in all virtual addresses, with 125252 in the user's A register.

SAA 0123 should be executed

Test 6

Data-read-not-permitted test. The user's page table is filled with 020004 (FPM = 1, page number 4). After that, the user program LDA *; MON 6; JMP *-1 is executed in all virtual addresses, with 125252 in the user's A register. This should give memory protect violation.

Test 7

Instruction-page-and-read-data-page-in-core test. The user's page table (page table 0) is filled with 060004 (WPM = 0, RPM = FPM = 1, page number 4). After that, the user program LDA *; MON 7; JMP *-1 is executed in all virtual addresses with 125252 in the user's A register. LDA * should be executed.

Test 8

Data-write-not-permitted test. The user's page is filled with 020004 (FPM = 1, page number 4). After that, the user program STA *; MON 010; JMP *-1 is executed in all virtual addresses, with 125252 in the user's A register. This should give memory protect violation.

Test 9

Instruction-page-and-write-data-page-in-core test. The user's page table is filled with 12004 (FPM = WPM = 1, page number 4). After that, the user program STA *; MON 011; JMP *-1 is executed in all virtual addresses, with 125252 in the user's A register. STA * should be executed.

Test 10

Data-read-and-write-not-permitted test. The user's page table is filled with 020004 (FPM = 1, page number 4). After that, the user program MIN *: MON 012; JMP *-1 is executed in all virtual addresses. This should give memory protect violation.

Test 11

Data-write-not-permitted test. The user's page table is filled with 060004 (RPM = FPM = 1, page number 4). After that, the user program MIN *: MON 013; JMP *-1 is executed in all virtual addresses. This should give memory protect violation.

Test 12

Instruction-page-and-read-data-page-and-write-data-page-in-core test. The user's page table is filled with 160004 (WPM = RPM = FPM = 1, page number 4). After that, the user program MIN *: MON 014; JMP *-1 is executed in all virtual addresses. MIN* should be executed.

4 REAL TIME CLOCK STABILITY TEXT - HAR 1397

Read in the program, it starts automatically and prints:

REAL TIME CLOCK STABILITY TEST
CLOCK DEVICE NO. (010, 014, ETC.):

Type the real time clock device number (usually 010) on the teletype, followed by space or carriage return.

The program uses all available memory for a table. The table is built in this way:

First zero is stored in a cell in memory. After that, the program checks if ready-for-transfer is one, and if not, it increases the memory cell with 1. This is done until ready-for-transfer comes on and then the program repeats itself, with the next memory cell, until the memory is full.

After the device number has been specified, the program starts to build the table. For a 16K machine, this will take approximately 100 seconds the first time, when the clock frequency is 100 microseconds. The second time, the frequency is 10 microseconds and the third it is 1 microsecond.

When available memory is full, the table is printed. It consists of two columns and a sum. The first column is the number of times a cell in memory has been MIN'ed between clock ready-for-transfer. This column should consist of consequtive numbers. The second column is the number of cells with this MIN count. The greatest numbers should be in the middle of the column. The sum is the sum of the second column and should be constant. If it varies, it may mean that ready-for-transfer is not turned off properly.

The building of the table may be interrupted by typing any character on the teletype (a feature for impatient users!).

There are two error messages from TSTAB:

CLOCK IS DEAD!

which means that READY-FOR-TRANSFER has not occurred in that amount of time in which a memory cell has been MIN'ed from zero to zero.

CLOCK STATUS ERROR: XXXXXX

which means that some of the unused bits in the status word (bit 1, bits 4-15) have become 1.

XXXXXX is the failing status word where

Bit 0 is interrupt enabled
Bit 2 is external hold pulse has occurred
Bit 3 is Ready for transfer

5 REAL TIME CLOCK - HAR 1399

Read in the program. It starts automatically and prints

TEST REAL TIME CLOCK
CLOCK DEVICE NO. (010, 014, ETC.).

Type the real time clock device number (usually 010) on the teletype followed by space or carriage return.

After this, the count test is started. This test sets the number of clock pulses between each interrupt to N, N=2, 4, 010, 020, ...,100000, and checks that the clock counts down to zero (or N) properly. The clock frequency in this test is 100 microseconds.

After this test, TREAL prints:

END COUNT TEST

After a little delay, the frequency test starts, TREAL prints:

ABCABC....ABC

60 characters are printed, one per second. Between each character, ready-for-transfer has occurred 100 times.

After A, the frequency is 100 microsec.
" B " " " 10 "
" C " " " 1 "

When 60 characters have been printed, the count test is restarted.

The following error messages may be printed by the program:

CLOCK .HANGUP

which means that ready-for-transfer never occurred in the frequency test.

CLOCK NEVER STARTED

which means that the clock never counted down to 1 before the count test.

READY FOR TRANSFER, BUT CLOCK WAS NEVER ZERO

which means that ready-for-transfer occurred before the clock had counted down to zero.

CLOCK COUNTED WRONG. XXXXXX INSTEAD OF YYYYYY

which means that the clock was expected to contain YYYYYY or YYYYYY-1, but contained XXXXXX.

READY-FOR-TRANSFER IS DEAD !

which means that in the middle of the count test, ready-for-transfer never occurred.

CLOCK STOPPED COUNTING

which means that the clock register did not change for a period of approximately 150 microseconds.

CLOCK STATUS ERROR: XXXXXX

which means that some of the unused bits in the status word (bit 1, bits 4-15) have become 1. XXXXXX is the failing status word, where

Bit 0 is interrupt enabled
Bit 2 is external hold pulse has occurred
Bit 3 is ready for transfer

6 8K MOS MEMORY TEST PROGRAM - HAR 1821

Load with MASTER-CLEAR, 400&
Restart with MASTER-CLEAR, RESTART

Tests in the program:

Test No 1 a) stores ZEROS in all memory cells, read and check.
b) stores ONES in all memory cells, read and check.

2. Stores 34 different patterns in the whole test area, read and check

3 Address in address test.

4 Stores the following pattern:

52525
125252
52525
125252
"
"
"

5 a) Stores ZEROS in all locations
b) Reads ZEROS in first location and replace it it with ONES.
c) Reads ZEROS in the next location and replace it with ONES.

At the end all locations are read and checked for all ONES.

6 a) Write a pattern in the test area.
b) Replace the pattern in the first location with the pattern inverted.
c) Read the inverted pattern
d) Read the previous location
e) Reset the inverted pattern

Repeat b - e both moves the inverted pattern one location for each time.

The test is done with 34 different patterns.

7 Address in address test followed by inverted address test.

10 Test refresh.
The test area is filled with ZEROS and ONES and read after approximately 5 seconds.

11 Identical with Section 4 in memory check program

12 Identical with Section 5 in memory check program

13 Identical with Section 6 in memory check program

14 Identical with Section 7 in memory check program

15 Identical with Section 11 in memory check program

16 Identical with Section 12 in memory check program

} HAR 1198

17 & 20 Walking Tests
a) Writes a pattern in the test area
b) Writes the pattern inverted in the first location
c) Checks that no other locations are changed
d) Reset inverted pattern

Repeats b-d, both moves inverted pattern one location for each time.

The test is run with 4 patterns.

0
-1
52525
125252

The test is run 20 times in the test area while test 17 is run in one module at the time.

Example of Printout Running 8K MOS MEMORY - TEST:

```
MEMORY MAP

MODULE  8K  16K  24K  32K  40K  48K  56K  64K

BANK 0   X    X    X    X

BANK 1

BANK 2

BANK 3

PAGING 8K MOS MEMORY EXERCISER FOR NORD-10

DID YOU KNOW THIS PROGRAM ?:N
FIRST THE PROGRAM RUNS TEST 1 & 2 IN THE PROGRAM
SCRATCH AREA.  IF ERRORS ARE DETECTED, A MESSAGE
WILL BE PRINTED AND THE MACHINE STOPS.
THEN LOWER BANK AND LOWER TEST ADDRESS HAS TO
BE SPECIFIED (OCTAL NUMBER) SIMILAR FOR UPPER BANK
AND UPPER TEST ADDRESS.
LOWER AND UPPER BANK MAY BE EQUAL.
TEST TO BE RUN MUST BE GIVEN AS A SEQUENCE OF
OCTAL NUMBERS.  (1-20) EACH NUMBER TERMINATED BY
CR.
THE SEQUENCE IS TERMINATED WHEN 0 IS GIVEN AS TEST
NUMBER.  THE FOLLOWING NUMBERS HAVE A SPECIAL
MEANING WHEN SPECIFYING THE TESTS.
77 (CR) MEANS ALL TESTS
66 (CR) MEANS ALL TESTS EXCEPT TEST 20
55 (CR) MEANS ALL TESTS EXCEPT TEST 17 & 20
TEST 17 & 20 ARE LONG TESTS.
A DESCRIPTION OF ALL TESTS SHOULD BE ATTACHED TO
THE PD-SHEET.

TO THE OTHER QUESTIONS YOU MAY ANSWER Y, WHICH
MEANS YES, OR ANY OTHER CHARACTER WHICH MEANS
NO.  WHEN ANSWERING NO TO "CONTINUOUS ERROR OUTPUT?",
ERRORS WILL BE SAVED IN A BUFFER AND DUMPED WHEN
THE TEST IS TERMINATED.  THE DUMP WILL CONTAIN THE
NUMBER OF ERRORS IN EACH BIT IN EACH 8K MODULE. (EVEN
AND ODD ADDRESSES).
TO STOP THE TESTS MANUALLY, ANY CHARACTER MAY BE
TYPED ON TTY.  (TYPE ONLY ONCE).  USUALLY THERE
WILL BE A DELAY BEFORE THE TESTS ARE STOPPED:
TEST RUNNING.   000001

TEST RUNNING.   000002
```

```
SELECT BANK (LOWER):  0
LOWER TEST ADDRESS ( >14000 IF BANK 0):  14001
SELECT BANK (UPPER):  0
UPPER TEST ADDRESS:  77777
SPECIFY TESTS TO BE RUN:
55
CONTINUOUS ERROR OUTPUT?:  Y
DO YOU WANT TESTS TO LOOP?:N
ENABLE PARITY ERROR DETECTION?:Y
TESTS RUNNING
```

7 MOVER MEMORY TEST PROGRAM - HAR 1863

The program will print a heading and a maximal address (the memory is supposed to be contiguous from 0 to the maximal address). Then the program will print: DO YOU KNOW THIS PROGRAM? If you answer N, you will get an explanation of the program on the teletype.

Then, the program will ask for 5 parameters: FLYTT, ADDRN, ADDRX, PROGN and a device number.

ADDRN and ADDRX are the lower and upper addresses of the memory area to be tested (for 32K they might be 0 and 077777). FLYTT is the number of words that the program moves in memory after each test. If FLYTT is positive, the program moves upwards; if negative, it moves downwards. If FLYTT is 0, the program will not move.

PROGN is the first location of the program (i.e., it defines where the program will be placed in the memory). PROGN must be in the memory area to be tested.

The device number defines error output. TTY: 0304, L-P:0430, F-P:0410, and so on.

If the program shall run in another memory bank, load it with MASTER CLEAR, NB400& where N is the bank number (1, 2, 3).

START ADDRESS = 400

For S-III put the tape into the reader.

Type PLACE T-R

For version HAR 1863B:

```
@LOOK-AT MEMORY
READY
0631/171001 171000
@

@MEMORY  0  2777
@DUMP "MOVER"  400  400
@MOVER
```

and continue as above.

8 TECOD TEST PROGRAM - HAR 1451

TECOD is a test program intended to test memory and disk.

It does this by performing six different tests:

1. Extra Blocks Test
2. Address Test
3. Disk Arm Test
4. Sector Counter Test
5. Disk Write Protect Test
6. Repeated Pattern Test

Extra Blocks Test

TECOD writes on the disk one sector, two sectors, three sectors, . . ., up to 24 sectors. After each write, the sector(s) is (are) read back and checked in memory. If an error occurs, an error message is printed.

NB: Version 1451B will produce some false address mismatches (status 040430). They may be removed with 120/135053 0.

Address Test

TECOD writes the disk full of addresses. Each sector is filled with its own address + word number. When the whole disk is written, the sectors are read back, one by one, and checked in memory. If an error occurs, an error message is printed.

Disk Arm Test

Every sector is read and checked in memory. If N is the maximal disk address of a cartridge (fixed or removable), the sectors are read in this sequence: 0,N,1, N-1, and so on. This will test that the disk arm moves properly. If an error occurs, an error message will be printed.

Sector Counter Test

This test checks that the sector counter counts properly, i.e., 0, 1, 2, . . ., 026, 027, 0, 1, and so on. At the end of the test, a table is printed out, demonstrating the stability of the frequency with which the counter counts. If an error occurs, an error message is printed.

Disk Write Protect Test

Seven of the eight disk protect switches are tested, one by one. For every switch tested, one disk write and one disk read transfer is performed per track. The user is guided through the test by messages on the teletype.

NB: The protect test will not work for Hawk disks. The test will be bypassed if nothing is answered after 90 seconds.

Repeated Pattern Test

A disk area, no greater than one cartridge (the size of this area is also dependent on the size of the memory area that is checked), is filled with a bit pattern, and then this bit pattern is read back to a specified memory area.

The writing is done by tracks, the reading by sectors.

Every time a disk sector is read, the memory address is incremented by one. Thus, the different sectors are read to different memory addresses.

After a sector has been read, the contents of the corresponding memory block is checked for correctness, and an error message is printed if necessary.

Also, after each read, a parity check and compare is performed.

Six different bits patterns are used. They are:

000000, 177777, 052525, 125252, 000377 and 177400.

When all bit patterns have been used, the program repeats the pattern test.

How to Use TECOD

Read in TECOD. If correctly read, TECOD prints:

TECOD - TEST CORE AND DISK.

PLEASE TURN DOWN THE DISK PROTECT SWITCHES BEFORE THE TEST STARTS.

LOWER CORE ADDRESS (NOT LESS THAN XXXXXX):

Answer by typing the start address of the area that you want to check, followed by space.

TECOD then prints:

UPPER CORE ADDRESS (NOT LESS THAN YYYYYY AND NOT GREATER THAN MAX. CORE ADDR):

Answer by typing the last address of the area that you want to check, followed by space. (One whole cartridge: Upper: =lower + 046076, if memory size permits it!)

TECOD then prints:

DISK SYSTEM I (1) or II (2):

Answer by typing 1 or 2.

TECOD then prints:

UNIT NO. (0 TO 3):

Answer by typing the unit number.

TECOD then prints:

REMOVABLE (R) OR FIXED (F) DISK ?

Answer by typing R or F.

TECOD will now do the EXTRA BLOCKS TEST, with the different patterns. Before each pattern, this message will occur:

EXTRA BLOCKS TESTED. PATTERN IS PPPPPP

If an error occurs, this error message will be printed:

XXXXXX NNNNNN WWWWWW

Where XXXXXX is the number of extra blocks (0 - 027), NNNNNN is the word number (0 - 06777) and WWWWWW is the failing word. It should have been equal to the pattern.

If WWWWWW is equal to NNNNNN it means that no word from the disk has been read into that memory word before the disk is read, the core block is filled with 0, 1, 2, . . . , 06777.

Then the address test is performed. Before the test, this message is printed:

DISK ADDRESS TEST IS STARTED

If an error occurs, this error message is printed:

DISKAD	WORDNO	WORD	EXP.
DDDDDD	NNNNNN	WWWWWW	CCCCCC

where DDDDDD is the disk address, NNNNNN is the word number (0 - 0177) and WWWWWW is the failing word. It should have been CCCCCC (=DDDDDD+NNNNNN).

If WWWWWW is equal to 077700 it means that no word has been read into that core word. Before the disk read, the block is filled with 077700, 077700, . . ., 077700.

Then the disk arm test is done. Before the test, TECOD prints:

DISK ARM TEST IS STARTED

After this, all sectors are read from the disk in such a way that the disk arm has to move violently. If an error occurs, this message is printed:

DISKAD	WORDNO	WORD	EXP.
DDDDDD	NNNNNN	WWWWWW	CCCCCC

This is the same error message as that one described under the address test.

After this, the sector counter test is done. Before the test, TECOD prints:

START SECTOR COUNTER TEST.

At the end of the test, a table is printed out. The left column contains the number of mins in a loop between sector counts and should consist of consequtive numbers. The right column contains the number of loops with the same min, count. The greatest numbers snould be in the middle of the column. A sum of all the numbers in the right column is printed. It should be:

Upper core address - Lower core address + 1

If the sector counter never changes, TECOD prints:

SECTOR COUNTER NEVER STARTED.

If the sector counter counts for a while and then stops, TECOD prints:

SECTOR COUNTER DIED.

After this error message, an incomplete table is printed.

If the sector counter counted incorrectly, TECOD prints:

SECTOR COUNTER COUNTED WRONG.
LAST VALUE (CORRECT) AND FAILING VALUE: CCCCCC FFFFFF

CCCCCC is the last correct value of the sector counter. FFFFFF is the incorrect (new) value. It should have been CCCCCC or CCCCCC + 1 (MODULU 030).

Then TECOD will print:

START DISK WRITE PROTECT TEST

If the fixed disk is tested, it will also print:

PROTECT SWITCH/MUST BE UP DURING THE REST OF
THE TEST!

After this, the disk protect switches will be tested. TECOD will ask for special settings of the protect switches by printing:

PLEASE PUT PROTECT SWITCH N UP (ALL OTHERS DOWN!)
TYPE ANY CHARACTER AFTERWARDS.

This message must be answered within 90 seconds (approximately), or the rest of the protect test will be skipped. N is in the range 0 to 6. Put up the requested switch and type a character on the teletype. TECOD will then write and read all tracks and check that the protect switches work properly.

The following error messages may occur:

MISSING DISK WRITE PROTECT ERROR!
DISK ADDRESS AND STATUS: AAAAAA SSSSSS

This means that a protect error was expected but it did not occur. Maybe the user forgot to put up the protect switch?

THIS DISK WRITE PROTECT ERROR SHOULD NOT HAVE OCCURRED!
DISK ADDRESS AND STATUS: AAAAAA SSSSSS

This means that an unexpected write protect error has occurred. Maybe the user put up a switch that was not requested?

WHAT ? DISK WRITE PROTECT ERROR ON READ ? NONSENSE!!
DISK ADDRESS AND STATUS: AAAAAA SSSSSS

This error message should not occur at all:

STATUS ERROR ON READ. TEST IS TERMINATED.
DISK ADDRESS AND STATUS: AAAAAA SSSSSS

This means that TECOD has tried to read several times with no luck. The test cannot continue.

PROTECTED TRACK WRITTEN UPON!
DISK ADDRESS: AAAAAA

This means that a disk area that TECOD expected to be protected is written upon. Maybe the user has not put up the requested switch?

When this test is finished, TECOD prints:

END DISK WRITE PROTECT TEST

PUT DOWN ALL PROTECT SWITCHES!
TYPE ANY CHARACTER AFTERWARDS!

After this, TECOD will put a pattern on the disk. When this is done, it prints:

PATTERN ZZZZZZ NOW ON DISK

Then, TECOD starts reading back the bit pattern. Every word is checked in memory after it has been read from the disk. If an error occurs, this error message is printed:

DISKAD	WORDNO	COREAD	WORD
DDDDDD	WWWWWW	CCCCCC	WWWWWW

DDDDDD is the disk address of the failing word, NNNNNN is the word number within the disk error (0 - 0177). CCCCCC is the memory address of the failing word. WWWWWW is the failing word. It should have been equal to the bit pattern.

If WWWWWW is equal to NNNNNN, it means that no word from the disk has been read into that core word. Before the disk read, the core block is filled with 0, 1, 2, . . ., 0177.

After each block has been read, the block is also parity checked and compare tested, by performing a disk parity check and a disk compare.

If an error occurs, a disk error message will be printed. For a parity check error, the disk status should be 041030; and for a compare error, the disk status should be 042030.

Error Messages from the Disk Routine

ERROR IN EXTRA BLOCKS: EEEEEE

EEEEEE is the number of extra blocks specified to the disk routine. It should be in the range 0 - 027. If this error message occurs, the program is probably destroyed, or the machine is very, very worn.

The main error message from the disk routine is this one:

DISK XXXXXX YYYYYY ERROR.
STATUS WORD: SSSSSS
DISK ADDR: DDDDDD
UNIT NO.: UUUUUU
MODUS WORD: MMMMMM
CAR, EXP, CAR, AND DIFF: UUUUUU EEEEEE DDDDDD

This message occurs whenever a disk transfer fails or if the disk routine is called with incorrect parameters.

XXXXXX can be READ, WRITE, PARITY, COMPARE and indicates which operation failed.

YYYYYY can be UNIT NOT READY, HANGUP, INCORRECT DISK ADDRESS STATUS, MODUS, CORE ADDRESS REGISTER.

UNIT NOT READY:	When activated, the disk did not become active, or not on cylinder.
HANGUP:	When activated, the disk stayed alive.
INCORRECT DISK ADDR:	Program probably destroyed.
STATUS ERROR:	Transfer failed. Look at status word description.
MODUS:	Program probably destroyed.
CORE ADDR. REGISTER:	Contents of core address register wrong after transfer. The incorrect and expected core address register is printed out, and their difference.

9 DIMS USER'S GUIDE - HUT 1453

Contents: Page:

Introduction	9-2
Simplified Parameters	9-3
Copy	9-3
Compare	9-5
Verify	9-5
Dump	9-5
Change	9-6
Parity Check	9-7
Set	9-8
Translate	9-8
Error Messages	9-9

Introduction

Read in "Disk Maintenance System" (hereafter called DIMS). if it is read correctly, the program will start automatically by printing:

DISK MAINTENANCE SYSTEM

version and date

If you don't know the answer to questions that DIMS may ask, Type X:

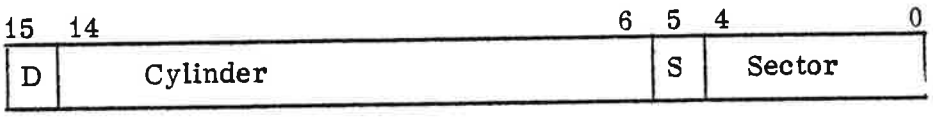
FUNCTION:

There are eight answers to this.

Type		DIMS answers by printing	
	CH		ANGE
	COM		PARE
	COP		Y
	D		UMP
	P		ARITY CHECK
	S		ET
	T		RANSLATE
	V		ERIFY

The eight functions CHANGE, COMPARE, COPY, DUMP, PARITY CHECK, SET, TRANSLATE, and VERIFY are described on the following pages.

CDC HAWK Disk Address Format:



D = disk
D = 0 i.e., removable cartridge
D = 1 i.e., fixed disk

S = surface
S = 0 i.e., upper surface
S = 1 i.e., lower surface

Simplified Parameters

If one of the four functions COMPARE, COPY, PARITY CHECK, VERIFY is selected, DIMS will print:

A WHOLE CARTR., S-III AND MACM-AREA, OR OTHER (W/S/O):

If the user types W as an answer, a whole cartridge will be compared/copied/parity-checked/verified. This is useful for backup purposes.

If the answer is S, only the disk area where S-III and MACM-AREA resides (starting at page 1, 0177 pages) is handled. This might be useful if SINTRAN is destroyed and if the user wants to reset SINTRAN without destroying files.

If O is answered, refer to descriptions of the four different functions.

If the answer was W or S, DIMS will ask for disk system, unit number and removable/fixed. The answer to disk system is 1 or 2, usually 1. The answer to unit is 0, 1, 2, or 3. The answer to removable/fixed is R or F.
REMEMBER: The answer is one single character. No carriage return must be given afterwards! If an incorrect character is given, DIMS will ask the question again.

Copy

Copy copies one disk area to another disk area.

DIMS will print

FROM:

As an answer to this, type,

S1/S2, or
U0/U1/U2/U3, or
An octal number in the range 0 to 162367, followed by space, or
CR (CARRIAGE RETURN)

S means disk system. If no disk system is specified, S1 is assumed. U means unit. If no unit is specified, U0 is assumed. Remember that the disk address must not have 1 both in bit 3 and bit 4, and bits 14 - 6 must not be greater than 623. This is the disk address of the first sector to be copied.

DIMS will then print

TO:

The answer to this is the same as the answer to from. This will be the disk address of the first sector of the new disk area.

DIMS will then print

AMOUNT:

As an answer, type

CR, or
an octal number in the range 1 to 045700, followed by space. This is the number of disk sectors to be copied.

DIMS will then print

BLOCK SIZE:

As an answer, type

CR, or
an octal number in the range 1 to 030.

CR means 030, which is a whole track. The disk transfers are done in blocks. A block may contain from 1 to 030 sectors.

DIMS will then check if the specified areas overlap. If they do, DIMS will print

DISK AREAS OVERLAP EACH OTHER.

If okay, print Y:

If you print Y, the copy will be executed. The execution starts by DIMS printing CR LF.

The overlap feature may be used to fill a whole disk with zeros (for instance). If "SET" is used to fill the first sector with zeros, then the disk area starting at disk address 0 may be copied to the disk area starting at disk address 1. The block size must be 1 (IMPORTANT!) and the amount must be 045677. "SET" can also fill a whole track with zeros. Copy is then executed with amount 045650. The block size is 030 and the disk addresses should be 0 and 030.

If a disk error should occur during the execution, an error message will be printed and the execution proceeds. Error messages from the disk routines are listed on page 9-9 .

When execution finishes, DIMS prints

END COPY

FUNCTION:

And you are back to the beginning again.

Compare

Compare compares the contents of two disk areas by reading the first area (disk read) and comparing it with the second area (disk compare).

Compare is used exactly like copy.

When execution is completed, DIMS prints:

END COMPARE

FUNCTION:

Verify

Verify compares the contents of two disk areas by reading them both from the disk and then comparing them word for word in core.

Verify is used exactly like copy.

If an error occurs, a heading will be printed and then one line per error. This line looks like this:

D1D1D1 U1U1U1 D2D2D2 U2U2U2 WWWWWW FFFFFF SSSSSS

where D1D1D1 and U1U1U1 are the disk address and unit number belonging to FFFFFF. D2D2D2 and U2U2U2 is the disk address and unit number belonging to SSSSSS. WWWWWW is the word number of FFFFFF and SSSSSS. It is in the range 0 to 05777, and indicates which word of the disk area that failed. FFFFFF and SSSSSS are the two words that should be equal. FFFFF is from the first disk area and SSSSSS from the second.

After the error message, verify proceeds.

When the execution finishes, DIMS prints:

END VERIFY

FUNCTION:

Dump

Dump dumps the contents of a disk area on the line printer or the teletype.

DIMS will print:

FROM:

The answer is the same as the answer described under copy.

DIMS will then print:

AMOUNT:

The answer is the same as the answer described under copy.

DIMS will then print:

DUMP ON TT (T) OR LPR (L):

The answer is T for teletype and L for line printer. The dump will loop like this:

```
000000 CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
000010 CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
SAME
000170 CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
```

where CCCCCC is any octal number and the first number on each line is the word number of the second word on the line. Identical lines will not be printed.

When the execution finishes, DIMS will print:

FUNCTION:

Change

Change inspects and changes single words on the disk.

DIMS will print:

DISK ADDRESS:____

The answer is the same as the answer described for From under Copy. This is the disk address of the sector you want to inspect/change.

DIMS reads the sector from the disk and prints:

```
SECTOR IS READ
LEGAL CHARACTERS ARE CR / 0 1 2 3 4 5 6 7
LEGAL ADDRESSES ARE 000 - 0177
```

After this, you may inspect and change the contents of the disk sector by using the teletype as you do under (NORD-10) operator's communications. The only difference is that you don't have to type all six digits of an octal number.

If an illegal character is typed (a character not from the set mentioned above), the character is ignored, and _ is printed.

Remember to use addresses in the range 0 to 0177!

When all changes are done, type @, and DIMS writes to the disk.

After that, DIMS prints:

SECTOR IS WRITTEN BACK

FUNCTION:

Parity Check

Parity check will read the contents of a disk area, without storing it in core, and check for parity.

DIMS will print:

FROM:

The answer to this is the same as the answer described in Copy.

DIMS will than print:

AMOUNT:

The answer is the same as the answer described in Copy.

DIMS will then print:

BLOCK SIZE:

The answer is the same as the answer described in Copy.

When the execution finishes, DIMS prints:

END PARITY CHECK

FUNCTION:

Set

Set will write 1-030 sector(s) on the disk. The contents of the sector(s) is specified by the user.

DIMS will print:

HOW MANY SECTORS (1-030):

Answer by typing an octal number in the range 1-030, followed by space.

DIMS will print:

SPECIFY THE FIRST WORD:

The answer to this is any octal number, followed by space. This will be the contents of the first word of the disk area.

DIMS will then print:

MODIFIER:

The answer is any octal number, followed by space. The modifier will be added to all subsequent words of the disk sector(s), in this way:

0/FIRST WORD
1/FIRST WORD + MODIFIER
2/SECOND WORD + MODIFIER (=FIRST WORD + 2*MODIFIER)
and so on.

DIMS will then print:

DISK ADDRESS:

The answer is the same as the answer to from described in Copy.

After the sector(s) has been written on the disk, DIMS prints:

FUNCTION:

Translate

Translate translates an octal number to a CDC disk address.

DIMS will print:

LOGICAL DISK ADDRESS:

The answer is any octal number in the range 0 to 045677, followed by space.

DIMS will then print:

LOGICAL DISK ADDRESS LLLLLL IS CDC DISK ADDRESS DDDDDD

Where LLLLLL is the word specified by the user, and DDDDDD is the disk address.

After this, DIMS prints:

FUNCTION.

Disk Error Messages

If a disk error occurs, this message will be printed:

DISK XXXXXX YYYYYY ERROR.
STATUS WORD: SSSSSS
DISK ADDR: DDDDDD
UNIT NO.: UUUUUU
MODUS WORD: MMMMMM

Where XXXXXX will be READ, COMPARE, WRITE OR PARITY CHECK; YYYYYY will be UNIT NOT READY, HANGUP, INCORRECT DISK ADDRESS, STATUS, MODUS or CORE ADDRESS REGISTER.

XXXXXX indicates which operation was performed. YYYYYY reflects the bits in the status word. SSSSSS is the status word and DDDDDD is the disk address of the sector that failed. UUUUUU is the disk unit number and MMMMMM contains parameters to the disk routine.

If the core address register fails, the error message will also contain this extra line:

CAR, EXP, CAR, AND DIFF: UUUUUU EEEEEE DDDDDD

Where UUUUUU is the incorrect core address register, EEEEEE is the expected core address register and DDDDDD is the octal difference between the core address register and its expected value after the disk transfer.

10 BIMS USER'S GUIDE - HUT 1871

Contents:

	Page:
Introduction	10-2
Copy	10-2
Compare	10-4
Verify	10-5
Dump	10-5
Change	10-6
Parity Check	10-7
Set	10-7
Translate	10-8
Format	10-8
Refresh	10-9
Clear-Device	10-10
Disk Error Messages	10-10

Introduction

Read in "Big Disk Maintenance System" (hereafter called BIMS) in the usual way (master clear, 400&). If it is read correctly, the program will start automatically by printing:

BIG DISK MAINTENANCE SYSTEM (BIMS)

version and date

IF YOU DON'T KNOW THE ANSWER TO QUESTIONS THAT BIMS MAY ASK, TYPE X.

IS THIS A 33 MBYTE DISK OR A 66 MBYTE DISK (3 OR 6)?

Answer this by typing 3 or 6.

BIMS then prints:

FUNCTION:

There are eleven answers to this:

Type	BIMS answers by typing	
CH		ANGE
CL		EAR-DEVICE
COM		PARE
COP		Y
D		UMP
F		FORMAT
P		ARITY-CHECK
R		EFRESH (REFORMAT)
S		ET
T		RANSLATE
V		ERIFY

The eleven functions CHANGE, CLEAR-DEVICE, COMPARE, COPY, DUMP, FORMAT, PARITY CHECK, REFRESH, SET, TRANSLATE and VERIFY are described on the following pages.

Copy

Copy copies the contents of one disk area to another disk area.

BIMS will print:

FROM:

It asks for the disk address of the first sector to be copied.

As an answer to this, type:

S1/S2 (S1 OR S2), AND/OR
U0/U1/U2/U3/U4/U5/U6/U7, AND/OR
E0/E1/(E1 only for the 66 mbyte disk),
and/or an octal number in the range 0-146517 for the 33 mbyte disk, any legal octal number for the 66 mbyte disk if E0, an octal number in the range 0-115517 if E1, followed by any non-digit.

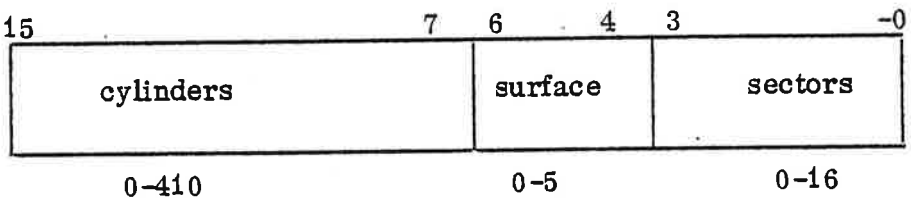
S means system. If no system is specified, S1 is assumed.
U means unit. If not unit is specified, U0 is assumed.
E means extended cylinder address. If no such is specified, E0 is assumed. If only carriage return is typed, 0 is assumed. If the disk address has 1 in bit 6, bits 5-4 must both be zero (101, 110, 111 in bits 6-4 are illegal).

BIMS will then print:

TO:

The answer to this is the same as the answer to From. This will be the disk address of the first sector of the new disk area.

SMD Address Format:



BIMS will then print:

AMOUNT:

As an answer, type:

CR, OR
An octal number in the range 1-100160 for the 33 mbyte disk. An octal number in the range 1-200460 for the 66 mbyte disk, followed by an non-digit. This is the number of disk sectors to be copied.

BIMS will then print:

BLOCK SIZE:

As an answer, type:

CR, OR
An octal number in the range 1 to 020.

CR means 020, which is a whole track. The disk transfers are done in blocks. A block may contain from 1 to 020 sectors.

BIMS will then check if the specified areas overlap. If they do, BIMS will print:

DISK AREAS OVERLAP EACH OTHER. IF OK, PRINT Y:

If you print Y, the copy will be executed. The execution starts by BIMS printing CR LF.

The overlap feature may be used to fill a whole disk with zeros (for instance). If "SET" is used to fill the first sector with zeros, then the disk area starting at disk address 0 may be copied to the disk area starting at disk address 1. The block size must be 1 (important!!), and the amount must be 100157 or 200457.

If a disk error should occur during the execution, an error message will be printed and the execution will proceed. Error messages from the disk routines are listed on page 10-10.

When execution finishes, BIMS prints:

END COPY

FUNCTION:

And you are back to the beginning again.

Compare

Compare compares the contents of two disk areas by reading the first area (disk read), and comparing it with the second area (disk compare).

Compare is used exactly like Copy.

When execution finishes, BIMS prints:

END COMPARE

FUNCTION:

Verify

Verify compares the contents of two disk areas by reading them both from the disk, and then comparing them word by word in memory.

Verify is used exactly like Copy.

If an error occurs, a heading will be printed, and then one line per error. The heading looks like this:

```
DSKAD1  EXTCY1  UNIT1  DSKAD2  EXTCY2  UNIT2
D1D1D1  E1E1E1  U1U1U1 D2D2D2  E2E2E2  U2U2U2

WORDNO  WORD1  WORD2
```

Where D1D1D1, E1E1E1 and U1U1U1 belong to FFFFFF, and D2D2D2, E2E2E2 and U2U2U2 belong to SSSSSS. The error lines contain three words:

```
WWWWWW  FFFFFF  SSSSSS
```

WWWWWW is the word number of FFFFFF and SSSSSS. It is in the range 0 to 017777, and indicates which word of the disk area failed. FFFFFF and SSSSSS are the two words that should have been equal. FFFFFF and SSSSSS are the two words that should have been second.

After the error message, Verify proceeds.

When the execution finishes, BIMS prints:

```
END VERIFY
```

```
FUNCTION:
```

Dump

Dump dumps the contents of a disk area on the line printer or the teletype.

BIMS will print:

```
FROM:
```

The answer is the same as the answer described under Copy.

BIMS will then print:

```
AMOUNT:
```

The answer is the same as the answer described under Copy.

BIMS will then print:

DUMP ON TT (T) OR LPR (L):

The answer is T for teletype and L for line printer. The DUMP will look like this:

```
000000 CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
000010 CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
       same
000770 CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
```

Where CCCCCC is any octal number, and the first number on each line is the word number of the second word on the line. Identical lines will not be printed.

When the execution finishes, BIMS will print:

FUNCTION:

Change

Change inspects and changes single words on the disk.

BIMS will print:

DISK ADDRESS:

The answer is the same as the answer described for FROM under Copy. This is the disk address of the sector you want to inspect/change.

BIMS reads the sector from the disk and prints:

```
SECTOR IS READ
LEGAL CHARACTERS ARE CR / 0 1 2 3 4 5 6 7 @
LEGAL ADDRESSES ARE 000-777
```

After this, you may inspect and change the contents of the disk sector by using the teletype as you do when N-10 is in stop mode (MOPC).

If an illegal character is typed (a character not from the set mentioned above), the character is ignored, and - is printed.

Remember to use addresses in the range 0 to 0777!

When all changes are done, type @, and BIMS writes the disk sector back on the disk.

After that, BIMS prints:

SECTOR IS WRITTEN BACK

FUNCTION:

Parity Check

Parity check will read the contents of a disk area, without storing it in core and check for parity.

BIMS will print:

FROM:

The answer to this is the same as the answer described under Copy.

BIMS will then print:

AMOUNT:

The answer is the same as the answer described under Copy.

BIMS will then print:

BLOCK SIZE:

The answer is the same as the answer described under Copy.

When the execution finishes, BIMS prints:

END PARITY CHECK

FUNCTION:

Set

Set will write one or more sectors on the disk. The contents are specified by the user.

BIMS will print:

HOW MANY DISK SECTORS (1-020):

The user must type an octal number in the range 1-020, followed by space.

BIMS will print:

SPECIFY THE FIRST WORD:

The answer to this is any octal number, followed by space. This will be the contents of the first word of the disk sector.

BIMS will then print:

MODIFIER:

The answer is any octal number, followed by space. The modifier will be added to all subsequent words of the disk sector, in this way:

0/FIRST WORD
1/FIRST WORD + MODIFIER
2/SECOND WORD + MODIFIER (=first word + 2 * modifier), and so on.

BIMS will then print:

DISK ADDRESS:

The answer is the same as the answer to FROM described under Copy.

After the sector has been written on the disk, BIMS prints:

FUNCTION:

Translate

Translate translates an octal number to a big disk address.

BIMS will then print:

THIS CORRESPONDS TO BIG DISK ADDRESS EN DDDDDD

Where EN is the extended cylinder address (E0 or E1) and DDDDDD is the block address.

After this, BIMS prints:

FUNCTION:

Format

Format will write the address parts of the tracks. Usually, this is not done, as it destroys the contents of the disk pack.

BIMS will first ask for the interface switch to be switched on. This is a red switch on one of the interface cards. When switched on, a small red lamp will light up. After this is done, type a character on the teletype.

BIMS will print:

DISK ADDRESS:

The answer is the same as the answer to "FROM" in Copy, but it must be the address of the beginning of a track (the octal address must end with 00/20/40/60).

BIMS will print:

NO. OF TRACKS:

Answer by typing an octal number in the range 1/04007, or in the range 1-010023 if 66 mbyte disk, or CR. CR (carriage return) means 04007/010023 tracks (a whole disk pack). After this, the specified track(s) will be formatted.

BIMS will then print:

END FORMATTING

REMEMBER THE FORMAT SWITCH!

FUNCTION:

The format switch should then be turned off again.

Refresh

This is a reformatting function. Tracks will be reformatted and information restored, if possible.

There are two modes of this function, refreshing of all specified tracks, or only those of the specified tracks that are bad. A bad track is a track with parity error or address mismatch. Address mismatch is an error in the address part of the track. Parity error may be an error in the address par, but it may also be a data error. In that case, the information on that track is impossible to correct (but the parity error will be removed).

BIMS will print:

REFRESH ALL TRACKS (A), OR ONLY BAD TRACKS (B):

Answer by typing A or B. If A, the user must turn on the interface format switch (see Format), and type A character on the teletype. If B, it is not necessary to do this before a bad track is found.

BIMS will print:

DISK ADDRESS:

The answer is the same as under Format.

If the mode is B (bad tracks only), BIMS will ask for the Format switch when (and only if) the first bad track is found.

Clear-Device

This function clears the device by doing a return-to-zero seek, followed by a device-clear.

BIMS will print:

DISK SYSTEM (1 or 2):

Answer by typing 1 or 2.

BIMS will print:

UNIT NO. (0-7):

Answer by typing one octal digit in the range 0-7.

The return-to-zero seek and the device-clear are then executed and BIMS will print:

FUNCTION:

Disk Error Messages

If a disk error occurs, this message may be printed:

```
STATUS ERROR.
BLOCK ADDRESS:  BBBBBB
EXT. CYL.  EEEEEE
UNIT NO.:  UUUUUU
STATUS:  SSSSSS
OPERATION WAS WRITE
```

If the core address register fails, the error message will contain some extra lines:

```
CORE ADDR REG ERROR.
EXP. CORE ADDR REG:  EEEEEE
FAIL. CORE ADDR REG:  FFFFFF
DIFFERENCE:  DDDDDD
```

11 MCOPY USER DESCRIPTION

HUT 1649 : TANDBERG, NORD-10
HUT 1650 : HP, NORD-10
HUT 1651 : NORD-1

MCOPY is a program to copy the entire contents of a disk or drum to or from magnetic tape. The program can be used with the following configurations:

Machine	Disk	Drum	Magnetic Tape
NORD-1	CDC 5Mb	All sizes	Hewlett Pack. Kennedy
NORD-10	CDC 5Mb CDC 33Mb CDC 66Mb	All sizes	Hewlet Pack Tandberg

This version of MCOPY cannot be used with NCR disks.

The program writes mag. tape blocks of either 3073 words for 5Mb disks, 8193 words for 33Mb and 66Mb disks, and 2049 words for drums (1 track of data + 1 word containing the disk/drum address), or blocks of 1024 words for all disk/drum types if the SINTRAN block size is desired. When the copying is done, the tape reverses to the beginning of the file and the program compares the contents of the tape with the disk. Any errors result in an error message (after several retries) and the number of errors and retries are listed at the end.

The program communicates extensively with the operator. Most of the messages and questions are self-explanatory, but a short description will be given here.

1. MAG. TAPE - DISK DRUM COPY
   VERSION X

2. STANDARD DEVICE NUMBERS?

   The following device numbers are standard:

NORD-1	Disk	= 144
	Drum	= 116
	Mag. Tape	= 134
NORD-10	Big disk	=1540
	Small disk	= 500
	Drum	= 540
	Mag. Tape	= 520

The answer is Y (yes) if all devices to be used have the standard device numbers. Otherwise, the answer N (no) must be given.

3. BIG DISK COPY?

Y if 33Mb or 66Mb disk copy

4. SYSTEM BACKUP (SMALL DISKS)?

(If not big disk copy.) If Y, the program will copy several 5Mb disks as follows: UNIT 0 - removable, UNIT 0 - fixed, UNIT 1 - removable, UNIT 1 - fixed, UNIT 2 - removable, etc. The total number copied is given by the next question.

5. NUMBER OF DISK CARTRIDGES:

(Only if system backup.) A maximum of 8 cartridges can be copied.

6. SMALL DISK COPY?

(If not big disk or system backup.) Y or N.

7. DRUM COPY?

(If none of the above.) Y. An N answer will go back to question 3.

8. HIGHEST DRUM ADDRESS:

(If drum.)

64K = 1777	512K = 17777
128K = 3777	1024K = 37777
256K = 7777	2048K = 77777

9. DRUM DEVICE NUMBER:

(If drum and not standard device numbers.)

10. DISK DEVICE NUMBER:

(If disk and not standard device numbers)

11. FIXED CARTRIDGE?

(If 5Mb disk and not system backup.)

12. REMOVABLE CARTRIDGE?

(If not fixed.)

13. DISK UNIT:

(If disk and not system backup.)

14. MAG. TAPE DEVICE NUMBER:

(If not standard.)

15. MAG. TAPE UNIT:

16. MAG. TAPE FILE NUMBER:

(If not big disk or system backup.) The first file is number 1 (not 0). If this is not the first copy in this run, a warning message will be given if the file number is not greater than the previous file number. A big disk copy and system backup always start with file 1. The tape is positioned at this point.

17. SINTRAN BLOCK SIZE (1K)?

Answer Y if a copy is desired that can be read on-line with the SINTRAN command COPY-DEVICE, or if a tape written with COPY-DEVICE is now to be read. A block size of 1024 will then be used. Note that this block size will cause the copying to go more slowlv. use more tape and that a check for correct disk address on the mag. tape record will not be made.

18. COPY TO MAG. TAPE?

Y or N.

19. COPY FROM MAG. TAPE?

(If previous answer was N.)

20. OK?

Y if everything all ready. This starts the copying process.

21. If the end of the tape is encountered during copying, appropriate messages are given to instruct the operator in changing tapes. The copy operation is completed entirely before the compare operation is started (all tapes must be mounted twice). If two tape units are available, they may be used interchangably, saving mounting time.

22. MORE TO BE COPIED?

(When copy and compare are done.) Y or N. If Y, the program starts up again from the beginning.

23. FINISHED

If errors are encountered during the copying or comparing, the following messages can be sent:

24. WRITE RING NOT PRESENT

25. MAG. TAPE ERROR MS MF ADR BLK

26. DISK/DRUM ERROR DS DF ADR BLK

27. COMPARE ERROR DS DF ADR BLK

The contents of the Mag. Tape block is not the same as the disk.

28. WRONG DISK/DRUM ADDRESS MS MF ADR BLK

The disk/drum address in the mag. tape block is wrong (out of step).

These messages contain the following fields:

MS = Mag. tape status
DS = Disk status
MF = Mag. tape function
DF = Disk function
ADR = Logical disk address (0 - 176 177)
BLK = Disk address field of mag. tape block

When the copying is done, the following messages will be sent:

29. NNN DISK/DRUM ERRORS

30. NNN MAG. TAPE ERRORS (includes wrong disk/drum address errors)

31. NNN COMPARE ERRORS

32. NNN MAG. TAPE READ RETRIES

33. NNN MAG. TAPE WRITE RETRIES

These last messages can be used to control the quality of the magnetic tapes used. More than 10 - 12 retries altogether indicate poor tape quality. An operation is attempted 12 times before an error message is given.

Function Codes

I MAG. TAPE

0	read one record
1	write one record
10	advance to end of file
11	reverse to end of file
12	write end of file
13	rewind
14	write skip
15	backspace one record
16	forwardspace one record
17	unload (Tandberg only)
20	read status

II DISK/DRUM

0	read
1	write
3	compare

Status Bits

Dev. / Bit	NORD 10					NORD 1			
	TB Mag.	HP Mag.	Big Disk	Small Disk	Drum	Disk	Ken. Mag.	HP Mag.	Drum
0	tape on-line	ready int. enabled				not ready	word count 0		comp. error
1	write enable	error int. enabled				time out	EOF		DMA error
2	load point	device active				DMA error	EOT		parity error
3	CRC error	device finished				protect violate	write protect		bit error
4	LRC error	inclusive OR of errors				address mismatch	data error	parity error	time out
5	control word error	write enable	illegal load	write protect violate		comp. error	parity error	write error	oper. dist.
6	bad data	LCR error	time out			parity error	overflow in read		protect violate
7	EOF		hardware error				DMA error		–
8	–	load point	address mismatch	bit error		modus bits	load point		–
9	EOT		parity error				unit select		–
10	word count not 0	parity error	compare error			–			–
11	DMA channel error					unit select	–	200 bpi	–
12	Overflow in read		abnor. compl.	transfer complete			–	556 bpi	–
13	tape busy	density select	not ready	transfer on			1600 bpi	800 bpi	–
14	form busy	tape ready	on cylinder				ready		–
15	–	–	external cylinder address	–	–	word count 0	transfer		–

12 DRUMS USER'S GUIDE - HUT 1297

Contents:

	Page:
Introduction	12-2
Copy	12-2
Compare	12-4
Verify	12-4
Dump	12-5
Change	12-6
Parity Check	12-7
Set	12-7
DRUM Error Messages	12-8

Introduction

Read in DRUMS. The program will start automatically by printing DRUM MAINTENANCE SYSTEM.

IF YOU DO NOT KNOW THE ANSWER TO THE QUESTIONS THAT DRUMS MAY ASK, TYPE X!

MAX DRUM ADDR (07777 FOR 256K):

Answer			
	07777	for	256K
	03777		128K
	01777		64K
	0777		32K

and so on. Remember to terminate with a non-octal character!

Drums then prints:

FUNCTION:

There are seven answers to this:

Type		drums answer by printing	
	CH		ANGE
	COM		PARE
	COP		Y
	D		UMP
	P		ARITY CHECK
	S		ET
	V		ERIFY

The seven functions CHANGE, COMPARE, COPY, DUMP, PARITY CHECK, SET and VERIFY are described in the following pages.

The meaning of the words sector and block will be as follows:

SECTOR	The smallest unit on the drum. It contains 64 (0100) words.
BLOCK	A collection of sectors, from 1 to 32 (1-040).

Copy

Copy copies one drum area to another drum area.

DRUMS will print:

FROM:

an an answer to this, type

U0, or
U1, or
U2, or
U3, or
an octal number in the range 0 to MMMMMM, followed by space, or
CR (Carriage Return).

MMMMMM is the highest legal drum address.

U means unit, if no unit is specified, U0 is assumed. If only carriage return is typed, U0 and 0 are assumed. This is the drum address of the first sector to be copied.

DRUMS will then print:

TO:

The answer to this is the same as the answer to from. This will be the drum address of the first sector of the new drum area.

DRUMS will then print:

AMOUNT:

as an answer, type:

CR, or
an octal number in the range 1 to AAAAAA followed by space (AAAAAA is the total number of sectors on the drum). This is the number of drum sectors to be copied.

DRUMS will then print:

BLOCK SIZE:

as an answer, type:

CR, or
an octal number in the range 1 to 040.

CR means 040, which is a whole track. The DRUM transfers are done in blocks. A block may contain from 1 to 040 sectors.

DRUMS will then check if the specified areas overlap. If they do, DRUMS will print:

DRUM AREAS OVERLAP EACH OTHER. IF OK, PRINT Y:

If you print Y, the copy will be executed. The execution starts by DRUMS printing:

CR LF.

The overlap feature may be used to fill a whole drum with zeros (for instance).

If "Set" is used to fill the first sector with zeros, then the drums area starting at drum address 0 may be copied to the drum area starting at drum address 1. The block size must be 1 (important!!), and the amount must be one less than the whole drum.

"Set" may also fill a whole track. The copying may then be done by tracks, which will be faster. The block size should then be 040, and the amount should be 040 (32) less than the whole DRUM.

If a DRUM error should occur during the execution, an error message will be printed and the execution will proceed.

Error messages from the DRUM routines are listed on page 12-8.

When execution finishes, DRUM prints:

END COPY

FUNCTION:

and you are back to the beginning again.

Compare

Compare compares the contents of two DRUM areas by reading the first area (DRUM read) and comparing it with the second area (DRUM compare).

Compare is used exactly like Copy.

When execution finishes, DRUMS prints:

END COMPARE

FUNCTION:

Verify

Verify compares the contents of two DRUM areas by reading them both from the DRUM, and then comparing them word by word in core.

Verify is used exactly like copy.

If an error occurs, a heading will be printed and then one line per error. This line looks like this:

D1D1D1 U1U1U1 D2D2D2 U2U2U2 WWWWWW FFFFFF SSSSSS

Where D1D1D1 and U1U1U1 are the DRUM address and unit numbers belonging to FFFFFF.

D2D2D2 and U2U2U2 are the DRUM address and unit number belonging to SSSSSS.

WWWWWW is the word number of FFFFFF and SSSSSS. It is in the range 0 to 03777 and indicates which word of the DRUM block that failed. FFFFFF and SSSSSS are the two words that should be equal. FFFFFF is from the first DRUM area and SSSSSS from the second.

After the error message, verify proceeds.

When the execution finishes, DRUM prints:

END VERIFY

FUNCTION:

Dump

Dump dumps the contents of a DRUM area on the line printer or the teletype.

DRUM will print:

FROM:

The answer is the same as the answer described under Copy.

DRUMS will then print:

AMOUNT:

The answer is the same as the answer described under Copy.

DRUM will then print:

DUMP ON TT (T) OR LPR (L):

The answer is T for teletype and L for line printer.

The dump will look like this:

```
000000 CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
000010 CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
       same
000170 CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
```

where CCCCCC is any octal number and the first number on each line is the word number of the second word on the line. Identical lines will not be printed.

When the execution finishes, DRUMS will print:

FUNCTION:

Change

Change inspects and changes single words on the DRUM.

DRUMS will print:

DRUM ADDRESS:

The answer is the same as the answer described under Copy. This is the DRUM address of the sector you want to inspect/ change.

DRUMS read the sector from the DRUM and prints:

SECTOR IS READ
LEGAL CHARACTER ARF CR / 0 1 2 3 4 5 6 7 @
LEGAL ADDRESSES ARE 000-077

After this, you may inspect and change the contents of the DRUM sector by using the teletype as you do under operators communication. The only difference is that you don't have to type all six digits of an octal number.

If an illegal character is typed (a character not from the set mentioned above) the character is ignored, and → is printed.

Remember to use addresses in the range 0 to 077!

When all changes are done, type @, and DRUM writes the drum sector back on the DRUM. After that, DRUM prints:

SECTOR IS WRITTEN BACK

FUNCTION:

Parity Check

Parity check will read the contents of a DRUM area without storing it in core and check for parity.

DRUMS will print:

FROM:

The answer to this is the same as the answer described under Copy.

DRUMS will then print:

AMOUNT:

The answer is the same as the answer described under Copy.

DRUMS will then print:

BLOCK SIZE:

The answer is the same as the answer described under Copy. When the execution finishes, DRUM prints:

END PARITY CHECK

FUNCTION:

Set

Set will write 1-040 sector(s) on the DRUM. The contents of this block are specified by the user.

DRUMS will print:

HOW MANY DRUM SECTORS (1-040):

Answer by typing an octal number in the range 1-040, followed by space.

DRUMS will print:

SPECIFY THE FIRST WORD:

The answer to this is any octal number, followed by space. This will be the contents of the first word of the first DRUM sector of the block.

Drums will then print:

MODIFIER:

The answer is any octal number, followed by space. The modifier will be added to all subsequent words of the drum sector(s), in this way:

0/FIRST WORD
1/FIRST WORD + MODIFIER
2/SECOND WORD + MODIFIER (=FIRST WORD + 2 * MODIFIER)
and so on.

DRUMS will then print:

DRUM ADDRESS:

The answer is the same as the answer to from described under Copy.

After the sector has been written on the DRUM. DRUMS print:

FUNCTION:

DRUM Error Messages

If a DRUM error occurs, this message will be printed:

```
DRUM XXXXXX YYYYYY ERROR,
STATUS WORD: SSSSSS
DRUM ADDR: DDDDDD
UNIT NO.: UUUUUU
MODUS WORD: MMMMMM
```

Where XXXXXX will be READ, COMPARE, WRITE or PARITY, and YYYYYY will be UNIT NOT READY, HANGUP, INCORRECT DRUM ADDRESS, STATUS, MODUS or CORE ADDRESS REGISTER. XXXXXX indicates which operation was performed. YYYYYY reflects the bits in the status word. SSSSSS is the status word and DDDDDD is the DRUM address of the first sector of the block that failed. UUUUUU is the DRUM UNIT unit number and MMMMMM is the modus word; it contains:

Bits	15-13	0
	12-11	OPERATION
	10-9	0
	8-7	UNIT NUMBER
	6-5	EXTRA CORE ADDRESS BITS
	4-0	NUMBER OF EXTRA BLOCKS

If the core address register fails, the error message will also contain this extra line:

CAR - EXP. CORE ADDR = AAAAAA

where AAAAAA is the octal difference between the core address register and its expected value after the DRUM transfer.

13 TCODR USER DESCRIPTION - HAR 1299

TCODR is a test program intended to test core and drum.

It does this by performing five different tests:

1. Extra Blocks Test
2. Address Test
3. Drum Write Protect Test
4. Sector Counter Test
5. Repeated Pattern Test

Extra Blocks Test

TCODR writes on the drum one block, two blocks, three blocks, . . ., up to 32 blocks, after each write, the block(s) is (are) read back and checked in core. If an error occurs, an error message is printed.

Address Test

TCODR writes the drum full of addresses, each block is filled with its own address + word number, when the whole drum is written upon, the blocks are read back, one by one, and checked in core. If an error occurs, an error message is printed.

Drum Write Protect Test

TCODR writes and reads all tracks on the drum and checks that the drum write protect feature works properly. TCODR also checks that the protected drum area is not written upon. By suitable messages on the teletype, the user is asked to participate in the test. If an error occurs, an error message is printed.

Sector Counter Test

TCODR reads the sector counter and checks that it counts properly (0, 1, 2, ..., 036, 037, 0, 1, and so on). At the end of the test a table is printed, demonstrating the stability of the counter. If an error occurs, an error message is printed.

Repeated Pattern Test

A drum area, not greater than one drum (the size of this area is also dependent on the size of the core area that is checked) is filled with a bit pattern, and then this bit pattern is read back to a specified core area. The writing is done by tracks, the reading by blocks.

Every time a drum block is read, the core address is incremented by one, thus, the different blocks are read to different core addresses.

After a block has been read, the contents of the corresponding core block is checked for correctness and an error message is printed if necessary.

Also, after the block has been read, it is parity checked and compared. If an error occurs, a drum error message is printed.

Six different bit patterns are used, they are:

000000, 177777, 052525, 125252, 000377 and 177400.

When all bit patterns have been used, the program repeats the pattern test.

How to Use TCODR

Read in TCODR, if correctly read, TCODR prints:

TCORD - - TEST CORE AND DRUM,

PLEASE SET DOWN ALL THE DRUM PROTECT SWITCHES!

MAX DRUM ADDR (07777 FOR 256K, ETC.):

Answer by typing the correct maximal drum address, followed by space, or CR (07777 FOR 256K, 03777 FOR 128K, ETC.):

TCODR then prints:

LOWER CORE ADDRESS (NOT LESS THAN XXXXXX):

Answer by typing the start address of the area that you want to check, followed by space.

TCODR then prints:

UPPER CORE ADDRESS (NOT LESS THAN YYYYYY, AND
NOT GREATER THAN MAX. CORE ADDR.,
ON WHOLE DRUM IS ZZZZZZ):

Answer by typing the last address of the area that you want to check, followed by space (one whole drum): UPPER:= LOWER+N+076, where N is the number of blocks on the drum).

TCODR then prints:

UNIT NO. (0 TO 3):

Answer by typing the unit number.

TCODR will now do the extra blocks test, with the different patterns, before each pattern, this message will occur:

EXTRA BLOCKS TESTED, PATTERN IS PPPPPP

If an error occurs, this error message will be printed:

XTRABL WORDNO WORD
XXXXXX NNNNNN WWWWWW

Where XXXXXX is the number of extra blocks (0-037), NNNNNN is the word number (0-04777), and WWWWWW is the failing word. It should have been equal to the pattern.

If WWWWWW is equal to NNNNNN it means that no word from the drum has been read into that core word. Before the drum read, the core block is filled with 0, 1, 2, ..., 04777.

Then the address test is performed. Before the test, this message is printed:

START DRUM ADDRESS TEST

If an error occurs, this error message is printed:

DRUMAD WORDNO WORD
DDDDDD NNNNNN WWWWWW

where DDDDDD is the drum address, NNNNNN is the word number (0-077) and WWWWWW is the failing word. It should have been DDDDDD+NNNNNN.

If WWWWWW is equal to 100000, it means that no word from the drum has been read into that core word. Before the drum read, the core block is filled with 100000, 100000, . . ., 100000.

After this, TCODR will start the drum write protect test. Before the test, this message is printed:

START DRUM WRITE PROTECT TEST

For every drum write protect switch tested, TCODR will also print:

PLEASE PUT PROTECT SWITCH N UP (AND ALL OTHERS DOWN!), TYPE ANY CHARACTER AFTERWARDS!

The answer to this is to flip up the requested switch, flip down any other (N is 0 to 7), and then type a character on the teletype.

TCODR will then write and read all tracks on the drum and check that the protect error occurs only at write transfers and only in the protected area, else an error message will be printed. TCODR also checks that the protected drum area is not written upon.

Protect switch 0 protects the first 16K with drum addresses 0 to 0377, switch 1 the next 16K with addresses 0400 to 0777, and so on. 128K is the maximal size of the protected area.

At the end of this test, TCODR prints:

END DRUM WRITE PROTECT TEST
PUT DOWN ALL PROTECT SWITCHES!
TYPE ANY CHARACTER AFTERWARDS.

The user must type a character on the teletype.

After this, the sector counter test is performed. TCODR prints:

START SECTOR COUNTER TEST

at the end of the test, a table is printed out, the left column contains the number of MIN's in a loop between sector counts and should consist of consequtive numbers. The right column contains the number of loops with the same MIN count. The greatest numbers should be in the middle of the column. A sum of all the numbers in the right column is printed, it should be:

UPPER CORE ADDR - LOWER CORE ADDR + 1

If the sector counter never changes, TCODR prints:

SECTOR COUNTER NEVER STARTED

If the sector counter counts for a while and then stops, TCODR prints:

SECTOR COUNTER DIED

After this error message, an incomplete table is printed.

If the sector counter counted wrong, TCODR prints:

SECTOR COUNTER COUNTED WRONG.
LAST VALUE (CORRECT) AND FAILING VALUE: CCCCCC FFFFFF

CCCCCC is the last correct value of the sector counter. FFFFFF is the incorrect (new) value. It should have been CCCCCC OR CCCCCC+1 (MODULO 0401).

After this, TCODR will put a pattern on the drum. When this is done, it prints:

PATTERN ZZZZZZ NOW ON DRUM

Then TCODR starts reading back the bit pattern. Every word is checked in core after it has been read from the drum. If an error occurs, this error message is printed:

DRUMAD WORDNO COREAD WORD
DDDDDD NNNNNN CCCCCC WWWWWW

DDDDDD is the drum address of the failing word. NNNNNN is the word number within the drum block (0-077). CCCCCC is the core address of the failing word. WWWWWW is the failing word. It should have been equal to the bit pattern.

If WWWWWW is equal to NNNNNN, it means that no word from the drum has been read into that core word, before the drum read, the core block is filled with 0, 1, 2, ..., 077.

After a block has been read from the drum it is parity checked and compare tested. This is done by performing a drum parity check and a drum compare. If an error occurs, a drum error message is printed.

14 CACHE TEST USER DESCRIPTION - HAR 2063

General

The program is divided into two parts:

- preliminary functional test
- a thorough test of functions in cache system, test 2 to 6 (see below)

If the functional test fails the user gets a question on the terminal to restart or continue. The user can specify if part 2 of the program (test 2 to 5) should run continuously or if it should stop after test 5.

During test 2 to 5 it is possible to limit error printout for each test to a number typed on the terminal and terminated with CR. If no limit is wanted the user types N.

After each test, the total number of errors detected for that test is printed.

The general principle for testing is to write different patterns to CACHE and memory for the same address locations. This is done by first writing to CACHE and memory, then inhibit CACHE and write to memory.

The program then reads and tests if output is from CACHE or memory. If output is different from what it is expected to be, there are several possible error reasons:

1. CACHE output but wrong content

   - one or more data bits are wrong
   - address to one or more data bits are wrong

2. Memory output instead of CACHE output, i.e., NIC - not in cache fails

   - page number bits fails
   - cache limits
   - used bits fails

3. CACHE output instead of memory output as 2 above.

The most common pattern to CACHE is address in address and in memory address inverted in address. This (address is displacement within one page) makes it relatively easy to see if output is from CACHE or memory.

Example:

Output 9B7 is stuck to one (CACHE data bit 4). An error printout can be: TEXT <addr.> <Read cont> <exp. cont.>

CACHE OUTPUT ERROR	040000	000020	000000
CACHE OUTPUT ERROR	040001	000021	000001
CACHE OUTPUT ERROR	040002	000022	000002

One possible error reason: CACHE data bit 4 stuck to one.

Cache Test Program

Initialization

1. Paging control system, interrupt system
2. Searching for memory limits

TEST 1

Functional test of cache

CACHE area page 10-12 (LIM 1) 20000 - 24000. Locations used WSPAC - 21300, TRAR1 - 21700.

- verification if CACHON is working
- test of CUP BIT using location 21700
- verification that data is coming from cache, using location 21300
- verification that data is coming from memory (used bit = 0)

If functional test fails type R to restart.

TEST 2

Test Source for Data Output, Cache or Memory, and Data Bits in Cache

Source for output:

This test tests if used bit is stuck to one or zero and if page number bits are stuck to one. CACHE area page 20 address 40000. Stores displacement into cache and complement to memory.

Error sources:

1. Used bit - stuck to one gives cache output, stuck to zero gives memory output
2. Page number bits - gives memory output
3. Data bits

Error printout:

text address, expected content, failing content.

Data bits in CACHE:

Test if stuck to one or zero.
Semi-random pattern. Error printout as above.

TEST 3

Test of Cache Inhibit Limits

- Sets upper limit to last page installed in memory, moves lower limit from page 0 to upper limit and tests that cache inhibit limits are working for each value.

- Sets lower limit to page 0 and moves upper limit from last page installed to page 0 and tests that cache inhibit limit is working for each value.

  The test is done by reading CUP bit to A register; TRA 12 inside or outside cache limits.

  Testing of each page is performed without using the same displacement within cache, test to page 0 is using location 0-1, page 1 location 2-3 and so on.

  Error printout: TRAC PNRB PNRC ADRC ULPNR

TRAC	Content of TRA instruction - CACHE status
PNRB	Dynamic limit
PNRC	Page of failing address
ADRC	Failing address
ULPNR	Upper limit page number

Error sources:

1. Used bit
2. Page number bits
3. Data bits, if data bits fail this test will cause the program to behave uncontrolled
4. Cache inhibit limits

TEST 4

Test of Page Number Bits in CACHE

- Test if bits stuck to one.

  CACHE area page 0 address 0-1777.
  Pattern in cache displacement (address in address).
  Pattern in memory complement of pattern in cache.
  If page number bits fail NIC (not in cache) will be true and data is read from memory.

- Test if bits stuck to zero.

  CACHE area page 377, bank 3 address 176000 - 177777.
  Pattern in cache displacement of pattern in cache.
  If installed memory is less than 377 pages, only cache will be working for these addresses. When page number bits fails response will be 0. Memory out of range interrupt is disabled.

- Checkerboard pattern.

  CACHE area page 125, bank 1 address 52000 - 53777.
  CACHE area page 252, bank 2 address 124000 - 125777.
  Pattern in cache even address 52525.
  Pattern in cache odd address 125252.
  Pattern in memory displacement (10 bits address in address)

- Test of address to page number bits.

  CACHE area page 0 and 377.
  Page 0 pattern in CACHE address in address.
  Page 377 address 1, 2, 4, 10, 20, 40, 100, 200, 400, 1000, 16 bits address in address.
  Memory: complement of address in address.

Error printout:

PAGE NO. BITS ERROR BANK NO. address failing content

Error reasons:

1. Used bit, will cause memory output
2. Page number bits, will cause memory output
3. Data bits, failing content.

TEST 5

Test of Used Bit

Test stuck to one or stuck to zero (see Test 3).

- Checkerboard pattern

  CACHE area page 2, i.e., address 4000 - 5777
  CACHE BACKGROUND PATTERN P1 = 125252
  CACHE PATTERN P2 = 52525
  MEMORY PATTERN P3 = 70707

  Tests first used bit in odd addresses and then in even addresses.

Error reasons:

1. Used bit failing one gives memory output P3 and failing zero gives cache output P1.
2. Page number bits will cause memory output
3. Data bits failing content.

Error printout:

ERROR: USED BIT = 0 address content
ERROR: USED BIT = 1 address content
BIT ERROR: IN CACHE address content

Test of Address to Used Bits

Sequence:

1. Initialization CACHE and MEMORY

   - 70000 to CACHE and MEMORY, enable CACHE
   - 7 to MEMORY, inhibit CACHE

2. Writes 07770 to address 1, 2, 4, 10, 20, 40, 100, 200 in each of the four chips and checks that no other locations are disturbed.

   Clear CACHE

LOOP: Enable CACHE
7770 to CACHE and MEMORY
Inhibit CACHE
7 to MEMORY
Read and test data content (whole CACHE)

Enable CACHE
7000 to CACHE
Inhibit CACHE
7 to MEMORY
Clear CACHE
Next address for write

Error Sources:

1. Used bit error

   - extra ones: output from cache instead of memory, i.e., output 70000 instead of 7.

   - missing ones: output from memory instead of cache.

2. Data bits error:

   - in CACHE output 07770 is modified
   - in CACHE output 70000 is modified
   - in Memory output 00007 is modified

3. Address error to used bit

   - responses as in 1 above.

Error Printout:

<Text> <address> <read content> <expected content>

Example:

ADDRESS BITS ERROR	000000	177777	000000
	000401	177376	000401
	000402	177375	000402

Error reason: 19D12 - floating, i.e., address pin (bit 8) to page number bit MR17 not connected. Address printout gives an indication that address bit 8 is failing but there is no information on which page-number-chip that is failing.

Test and Pattern Summary

TST2	Source output; cache or memory	CACHE pattern: 10 bits address (displacement) in address Memory pattern: complement of 10 bits address in address
TST21	Test CACHE data bits stuck	CACHE pattern: 0 or 177777 Memory pattern: 10 bits address in address
TST22	Test CACHE data bits "random pattern":	CACHE pattern: Random generator Memory pattern: 177400
TST3	CACHE Inhibit Limits:	Testing TRA CASTS (10) bit 1, when used bit is cleared for all limits.
TST4	Test of page number bits	
TST41	Test page number bits stuck	CACHE pattern: 10 bits address in address Memory pattern: complement of address in address
TST42	Test checkerboard pattern	CACHE pattern: 52525 (even addresses), 125255 (odd addresses) Memory pattern: 10 bits address in address
TST43	Test of address to page no. bits	CACHE pattern: 16 bits address in address (page 0 and page 377) Memory pattern: complement of address in address
TST5	Test of used bit	
TST51	Checkerboard pattern to used bits	CACHE background pattern: 125252 CACHE pattern: 52525 Memory pattern: 70707
TST52	Address to used bits	CACHE background pattern: 70000 CACHE pattern: 07770 Memory pattern: 00007

Error Printout Summary

Test 2	CACHE OUTPUT ERROR	⟨address⟩ ⟨failing content⟩ ⟨exp. content⟩
	MEMORY OUTPUT ERROR	⟨address⟩ ⟨failing content⟩ ⟨exp. content⟩
Test 21	CACHE OUTPUT ERROR	⟨address⟩ ⟨failing content⟩ ⟨exp. content⟩
Test 22	CACHE OUTPUT ERROR	⟨address⟩ ⟨failing content⟩ ⟨exp. content⟩

Test 3	LLE-OUT	i.e., Lower limit error - outside limits	⟨TRAC⟩ ⟨PNRB⟩ ⟨PNRC⟩ ⟨ADRC⟩ ⟨ULPNR⟩
	LLE-OUT U	i.e , Lower limit error - outside limits used bit fails	TRAC - CACHE status
	LLE-IN	i.e., lower limit error - inside limits	PNRB - Dynamic Limit
	ULE-OUT	i.e., Upper limit error - outside limits	PNRC - Page of failing address
	ULE-OUT U	i.e , Upper Limit error - outside limits used bit fails	ADRC - failing address
	ULE-IN	i.e., Upper Limit error - inside limits	ULPNR - upper limit page number

Test 4	PAGE NO. BITS ERROR BANK NO.:	⟨address⟩ ⟨failing content⟩
Test 41	" " "	"
Test 42	" " "	"
Test 43	ADDRESS BITS ERROR	⟨address⟩ ⟨failing content⟩ ⟨exp. content⟩
Test 51	ERROR: USED BIT = 0 ERROR: USED BIT = 1 BIT ERROR IN CACHE?	⟨address⟩ ⟨failing content⟩
Test 52	USED ERROR	⟨failing address⟩ ⟨failing content⟩ ⟨test address⟩ ⟨exp. content⟩

15 ERRCOR - HAR 2111

Test program for error correction logic, module 1125 and error correction bits on 21 bits memory module.

The program consists of 5 tests with subtests. Test 1 and 2 are used under program initialization and tests 3 to 5 are for testing error correction logic and memory modules.

TEST 1

Test 1 finds memory limits and test 2 finds which memory module has error correction bits. A memory map is printed. P means memory module with parity error detection. E means memory module with error correction bits and space means no memory module installed.

TEST 2

This test is performed by forcing an error to bit 15 and the interrupt system, level 14 parity error, should detect this error. PES (parity error status) bit 8 is set if the memory module tested has error correction bits. If this test fails (for example, no interrupt) the program has a scope modus. Test 2 is then repeated continuously without error messages and it is possible to scope the control logic for error detection and correction on module 1125 - Transceiver Data.

After these tests the program has an operator communication part where the operator can specify the following:

- stop after test 5 or repeated running of tests 3 - 5
- number of error printouts for each test or no limitation of error messages
- output device for error messages

During repeated running the message "start repeated running test 3 - 5" is printed after run number 1 - 10 - 100 - 1000 and then for each 1000 runs.

TEST 3

Test 3 is for testing error correction logic. This test is performed in 4 test modes. 20 locations are tested in each mode, and it is the first locations in the first available module with error correction bits which are used:

Mode 00: no error expected

Mode 11: forcing an error to bit 0 during write and expects parity error and bit 0 corrected during read

Mode 12: forcing an error to bit 15 during write and expects parity error and bit 15 corrected during read

Mode 13: forcing an error to bit 0 and bit 15 during write and expects parity error and none of bits 0 and 15 corrected during read because of multiple error.

Error printout, test 3:

PARITY ERROR: ⟨PES⟩ ⟨PEA⟩ ⟨Test mode⟩
DATA MISMATCH: ⟨ADDR⟩ ⟨Read Cont.⟩ ⟨Exp. Cont.⟩

Parity error is used when PES differs from expected value. Data mismatch is used when read content differs from expected content. Both printouts may be used for the same error.

If error correction network fails, test 3 may be used for scoping the error. The test routine is called for each test mode with the following parameters:

- test mode to be loaded to error correction control register
- write content, i.e., pattern written
- read content, i.e., expected pattern read
- PES, i.e., expected content of PES during interrupt

Patch for scoping error correction logic:

Address:	Statement:
3772	CALL FTST 3 (0, 70707, 70707, 0)
3777	CALL FTST 3 (11, 70706, 70707, 3500)
4004	CALL FTST 3 (12, 2707, 102707, 34500)
4011	CALL FTST 3 (13, 2706, 2706, 37500)
Mode 00:	3777/135054 → 124373 i.e., JPL I FTST3 → JMP * - 5
Mode 11:	4004/135050 → 124373
Mode 12:	4011/135044 → 124373
Mode 13:	4016/045040 → 124373

To avoid a call to error printout routine:

4165/135020 → 124002, i.e., JPL I ERR3 → JMP * + 2

TEST 4

Test 4 is for testing error correction bits in memory and has three subtests with different patterns to error correction bits.

Test 41 tests if bits struck to one or zero.

Test 42 uses a checkerboard pattern.

Test 43 uses a modified walking one and zero pattern.

TEST 5

This test, with subtests 50 and 51, is for testing whole memory, both modules with parity error detection and with error correction bits. Pattern used is a pseudo-random pattern. Test 50 is a direct read after write for each memory location. Test 51 is reading after writing to whole memory. I.e., if test 51 fails and not test 50, refresh errors or write disturbances are possible error reasons.

Error printout for test 4 and 5 is:

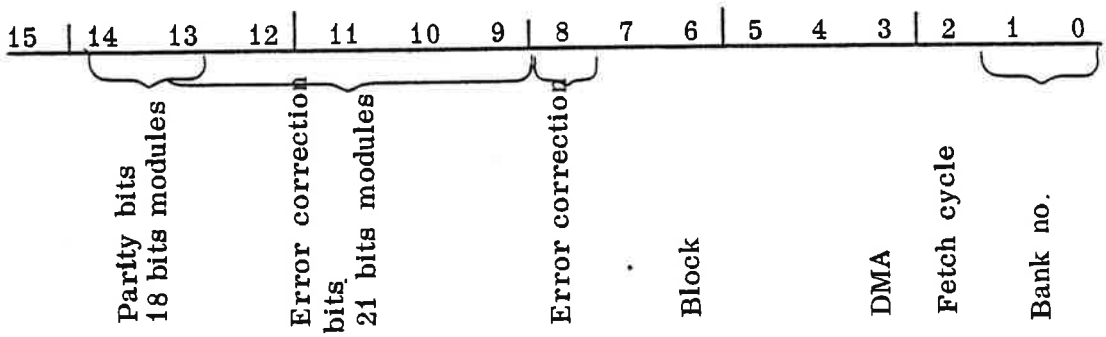
PARITY ERROR: ⟨Test no.⟩ ⟨PES⟩ ⟨PEA⟩ ⟨Read Cont.⟩ ⟨Exp. Cont.⟩ ⟨MULTIPLE ERROR or failing bit no.⟩

or

DATA ERROR: ⟨Test no⟩ ⟨ADDR⟩ ⟨Read Cont⟩ ⟨Exp. Cont⟩

Error detection is performed both with interrupt system, parity error on level 14, and with data comparing, read content versus expected content.

The different bits in PES have the following meaning:



If parity errors occur in bank 0 locations 1000 to 7000, the program prints out:

PROGRAM MEMORY ERROR: ⟨PES⟩ ⟨PEA⟩ ⟨Read Cont.⟩ ⟨MULTIPLE ERROR or failing bit no.⟩

If program memory error PME occurs during printout of other messages the program will stop after PME printout. The restart function will work if the program is stored in a module with error correction bits.

When searching for transient errors (repeated running) the program prints out run number when a failure occurs. An example of printout is shown below.

```
PARITY ERROR 51  034500  100363  034351  034351  I.E. BIT NO.15
PARITY ERROR 51  074500  101363  024753  024753  I.E. BIT NO.15
PARITY ERROR 51  074500  102363  015355  015355  I.E. BIT NO.15
      :
      :
00025  ERRORS
REPE = 03395
```

Number of errors for the test and run numbers are printed out in decimal, the rest is octal.

GREMS USER DESCRIPTION

CONTENTS

INTRODUCTION	PAGE 2
COPY	PAGE 3
COMPARE	PAGE 5
VERIFY	PAGE 6
DUMP	PAGE 7
CHANGE	PAGE 8
PARITY CHECK	PAGE 9
SET	PAGE 10
TRANSLATE	PAGE 11
FORMAT	PAGE 12
REFRESH	PAGE 13
CLEAR-DEVICE	PAGE 14
ERROR MESSAGES	PAGE 15

INTRODUCTION

READ IN "GREAT DISK MAINTENANCE SYSTEM" (HEREAFTER CALLED GREMS)
IN THE USUAL WAY (MASTER CLEAR, 1560&, *LOAD GREMS).
IF IT IS READ IN CORRECTLY, THE PROGRAM WILL START
AUTOMATICALLY BY PRINTING:

GREAT DISK MAINTENANCE SYSTEM (GREMS)

CURRENT PROG.NO. AND DATE

IF YOU DONT KNOW THE ANSWER TO QUESTIONS THAT GREMS MAY ASK,
TYPE X!

IS THIS A 38/76/288 MBYTE DISK (3/7/2) ?

ANSWER THIS BY TYPING 3 OR 7 OR 2.

GREMS THEN PRINTS

FUNCTION:

THERE ARE ELEVEN ANSWERS TO THIS:

TYPE CH	GREMS ANSWERS BY TYPING ANGE
CL	EAR-DEVICE
COM	PARE
COP	Y
D	UMP
F	ORMAT
P	ARITY-CHECK
R	EFRESH (REFORMAT)
S	ET
T	RANSLATE
V	ERIFY

THE ELEVEN FUNCTIONS CHANGE, CLEAR-DEVICE, COMPARE, COPY, DUMP,
FORMAT, PARITY CHECK, REFRESH, SET, TRANSLATE, AND VERIFY,
ARE DESCRIBED ON THE FOLLOWING PAGES.

COPY

COPY COPIES THE CONTENTS OF ONE DISK AREA TO ANOTHER DISK AREA.

GREMS WILL PRINT

FROM:

IT ASKS FOR THE DISK ADDRESS OF THE FIRST SECTOR TO BE COPIED.
AS AN ANSWER TO THIS, TYPE

CR (CARRIAGE RETURN), OR U0/U1/.../U7

CR IS THE SAME AS U0.
U0/U1/.../U7 MEANS THE FIRST SECTOR OF THE UNIT.

IF ANYTHING ELSE IS TYPED, GREMS WILL ASK FOR THE PARAMETERS:

DISK SYSTEM, WHICH IS 1 OR 2.
UNIT, WHICH IS 0-7.
CYLINDER, WHICH IS 0-0632 FOLLOWED BY CR FOR THE 38 MBYTE DISK,
OR 0-01466 " " " " " 76/288 MBYTE DISK.
SURFACE, WHICH IS 0-4 FOLLOWED BY CR FOR THE 38/76 MBYTE DISK,
OR 0-022 " " " " " 288 " "
SECTOR, WHICH IS 0-021 FOLLOWED BY CR.

GREMS WILL THEN PRINT

TO:

THE ANSWER TO THIS IS THE SAME AS THE ANSWER TO "FROM".
THIS WILL BE THE DISK ADDRESS OF THE FIRST SECTOR OF THE NEW
DISK AREA.

GREMS WILL THEN PRINT

AMOUNT:

AS AN ANSWER, TYPE

CR, OR
AN OCTAL NUMBER IN THE RANGE 1-110176 FOR THE 38 MBYTE DISK,
AN OCTAL NUMBER IN THE RANGE 1-220526 FOR THE 76 MBYTE DISK,
AN OCTAL NUMBER IN THE RANGE 1-1045572 FOR THE 288 MBYTE DISK,
FOLLOWED BY SPACE OR CR.
THIS IS THE NUMBER OF DISK SECTORS TO BE COPIED.
CR MEANS A WHOLE DISK PACK.

GREMS WILL THEN PRINT

BLOCK SIZE:

AS AN ANSWER, TYPE

CR, OR
AN OCTAL NUMBER IN THE RANGE 1 TO 022.

CR MEANS 022, WHICH IS A WHOLE TRACK.
THE DISK TRANSFERS ARE DONE IN BLOCKS.
A BLOCK MAY CONTAIN FROM 1 TO 022 SECTORS.

GREMS WILL THEN CHECK IF THE SPECIFIED AREAS OVERLAP.
IF THEY DO, GREMS WILL PRINT

DISK AREAS OVERLAP EACH OTHER. IF OK, PRINT Y:

IF YOU PRINT Y, THE COPY WILL BE EXECUTED.
THE EXECUTION STARTS BY GREMS PRINTING CR LF.

THE OVERLAP FEATURE MAY BE USED TO FILL A WHOLE DISK WITH
ZEROS (FOR INSTANCE).
IF "SET" IS USED TO FILL THE FIRST SECTOR WITH ZEROS,
THEN THE DISK AREA STARTING AT
CYLINDER 0, SURFACE 0, SECTOR 0,
MAY BE COPIED TO THE DISK AREA STARTING AT
CYLINDER 0, SURFACE 0, SECTOR 1.
THE BLOCK SIZE MUST BE 1 (IMPORTANT!!), AND THE AMOUNT
MUST BE 110175/220525/1045571.

IF A DISK ERROR SHOULD OCCUR DURING THE EXECUTION, AN ERROR
MESSAGE WILL BE PRINTED, AND THE EXECUTION PROCEEDS.
ERROR MESSAGES FROM THE DISK ROUTINES ARE LISTED ON PAGE 14.

WHEN EXECUTION FINISHES, GREMS PRINTS

END COPY

FUNCTION:

AND YOU ARE BACK TO THE BEGINNING AGAIN.

COMPARE

COMPARE COMPARES THE CONTENTS OF TWO DISK AREAS BY READING THE FIRST AREA (DISK READ), AND COMPARING IT WITH THE SECOND AREA (DISK COMPARE).

COMPARE IS USED EXACTLY LIKE COPY.

WHEN EXECUTION FINISHES, GREMS PRINTS:

END COMPARE

FUNCTION:

VERIFY

VERIFY COMPARES THE CONTENTS OF TWO DISK AREAS BY READING
THEM BOTH FROM THE DISK, AND THEN COMPARING
THEM WORD BY WORD IN THE MEMORY.

VERIFY IS USED EXACTLY LIKE COPY.

IF AN ERROR OCCURS, A HEADING WILL BE PRINTED, AND THEN
ONE LINE PER ERROR. THE HEADING LOOKS LIKE THIS:

```
UNIT1  CYLIN1 SURF.1 SECT.1 UNIT2  CYLIN2 SURF.2 SECT.2
U1U1U1 C1C1C1 F1F1F1 E1E1E1 U2U2U2 C2C2C2 F2F2F2 E2E2E2

WORDNO WORD1  WORD2
```

WHERE U1U1U1, C1C1C1, F1F1F1 AND E1E1E1 BELONGS TO FFFFF,
AND U2U2U2, C2C2C2, F2F2F2 AND E2E2E2 BELONGS TO SSSSSS.
THE ERROR LINES CONTAIN THREE WORDS:

```
WWWWWW FFFFFF SSSSSS
```

WWWWWW IS THE WORD NO. OF FFFFFF AND SSSSSS.
IT IS IN THE RANGE 0 TO 021777, AND INDICATES WHICH WORD
OF THE DISK AREA THAT FAILED.
FFFFFF AND SSSSSS ARE THE TWO WORDS THAT SHOULD HAVE BEEN EQUAL.
FFFFFF IS FROM THE FIRST DISK AREA, AND SSSSSS FROM THE SECOND.

AFTER THE ERROR MESSAGE, VERIFY PROCEEDS.

WHEN THE EXECUTION FINISHES, GREMS PRINTS:

```
END VERIFY
```

FUNCTION:

DUMP

DUMP DUMPS THE CONTENTS OF A DISK AREA ON THE LINE PRINTER OR THE TERMINAL.

GREMS WILL PRINT:

FROM:

THE ANSWER IS THE SAME AS THE ANSWER DESCRIBED UNDER COPY.

GREMS WILL THEN PRINT:

AMOUNT:

THE ANSWER IS THE SAME AS THE ANSWER DESCRIBED UNDER COPY.

GREMS WILL THEN PRINT:

DUMP ON TT (T) OR LPR (L):

THE ANSWER IS T FOR TERMINAL AND L FOR LINE PRINTER.
THE DUMP WILL HAVE A HEADING AND THEN A BLOCK LIKE THIS:

```
000000    CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
000010    CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
          SAME
000770    CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC CCCCCC
```

WHERE CCCCCC IS ANY OCTAL NUMBER, AND THE FIRST NUMBER ON EACH LINE IS THE WORD NUMBER OF THE SECOND WORD ON THE LINE. IDENTICAL LINES WILL NOT BE PRINTED.

WHEN THE EXECUTION FINISHES, GREMS WILL PRINT:

FUNCTION:

CHANGE

CHANGE INSPECTS AND CHANGES SINGLE WORDS ON THE DISK.

GREMS WILL PRINT:

DISK ADDRESS:

THE ANSWER IS THE SAME AS THE ANSWER DESCRIBED FOR "FROM" UNDER COPY.
THIS IS THE DISK ADDRESS OF THE SECTOR YOU WANT TO INSPECT/CHANGE.

GREMS READS THE SECTOR FROM THE DISK, AND PRINTS:

SECTOR IS READ
LEGAL CHARACTERS ARE CR / 0 1 2 3 4 5 6 7 @
LEGAL ADDRESSES ARE 000-777

AFTER THIS, YOU MAY INSPECT AND CHANGE THE CONTENTS OF THE DISK SECTOR BY USING THE TERMINAL AS YOU DO WHEN N-10 IS IN STOP MODE (MOPC).

IF AN ILLEGAL CHARACTER IS TYPED (A CHARACTER NOT FROM THE SET MENTIONED ABOVE), THE CHARACTER IS IGNORED, AND _ IS PRINTED.

REMEMBER TO USE ADDRESSES IN THE RANGE 0 TO 0777!

WHEN ALL CHANGES ARE DONE, TYPE @, AND GREMS WRITES THE DISK SECTOR BACK ON THE DISK.
AFTER THAT, GREMS PRINTS:

SECTOR IS WRITTEN BACK

FUNCTION:

PARITY CHECK

PARITY CHECK WILL READ THE CONTENTS OF A DISK AREA, WITHOUT STORING IT IN THE MEMORY, AND CHECK FOR PARITY.

GREMS WILL PRINT:

FROM:

THE ANSWER TO THIS IS THE SAME AS THE ANSWER DESCRIBED UNDER COPY.

GREMS WILL THEN PRINT :

AMOUNT:

THE ANSWER IS THE SAME AS THE ANSWER DESCRIBED UNDER COPY.

GREMS WILL THEN PRINT:

BLOCK SIZE:

THE ANSWER IS THE SAME AS THE ANSWER DESCRIBED UNDER COPY.

WHEN THE EXECUTION FINISHES, GREMS PRINTS:

END PARITY CHECK

FUNCTION:

SET

SET WILL WRITE ONE OR MORE SECTORS ON THE DISK.
THE CONTENTS IS SPECIFIED BY THE USER.

GREMS WILL PRINT:

HOW MANY DISK SECTORS (1-022):

THE USER MUST TYPE AN OCTAL NUMBER IN THE RANGE 1-022,
FOLLOWED BY SPACE OR CR.

GREMS WILL PRINT:

SPECIFY THE FIRST WORD:

THE ANSWER TO THIS IS ANY OCTAL NUMBER, FOLLOWED
BY SPACE OR CR.
THIS WILL BE THE CONTENTS OF THE FIRST WORD OF THE DISK
SECTOR.

GREMS WILL THEN PRINT:

MODIFIER:

THE ANSWER IS ANY OCTAL NUMBER, FOLLOWED BY SPACE OR CR.
THE MODIFIER WILL BE ADDED TO ALL SUBSEQUENT WORDS OF THE
DISK SECTOR. IN THIS WAY:
0/FIRST WORD
1/FIRST WORD+MODIFIER
2/SECOND WORD+MODIFIER (=FIRST WORD+2*MODIFIER)
AND SO ON.

GREMS WILL THEN PRINT:

DISK ADDRESS:

THE ANSWER IS THE SAME AS THE ANSWER TO "FROM" DESCRIBED
UNDER COPY.

AFTER THE SECTOR HAS BEEN WRITTEN ON THE DISK, GREMS PRINTS:

FUNCTION:

TRANSLATE

TRANSLATE TRANSLATES A LOGICAL DISK ADDRESS TO A PHYSICAL DISK ADDRESS, OR VICE VERSA.

GREMS FIRST PRINTS

LOGICAL OR PHYSICAL DISK ADDRESS (L/P):

IF THE ANSWER IS L, THEN GREMS WILL PRINT

LOGICAL ADDRESS:

THE ANSWER IS ANY OCTAL NUMBER IN THE RANGE 0-110175, OR 0-220525, OR 0-1045571, FOLLOWED BY SPACE OR CR.

GREMS WILL THEN PRINT:

DISK ADDRESS

CYL <CYL.NO.>
SUR <SURF.NO.>
SEC <SECT.NO.>

AFTER THIS, GREMS PRINTS:

FUNCTION:

IF THE ANSWER IS P, GREMS WILL PRINT:

DISK ADDRESS:

THE ANSWER TO THIS IS THE SAME AS THE ANSWER TO "FROM" UNDER COPY.

GREMS THEN PRINTS

LOGICAL ADDRESS <OCTAL NUMBER>

AFTER THIS, GREMS PRINTS

FUNCTION:

FORMAT

FORMAT WILL WRITE THE ADDRESS PARTS OF THE TRACKS.
USUALLY THIS IS NOT DONE, AS IT DESTROYS THE CONTENTS
OF THE DISK PACK.

GREMS WILL FIRST ASK FOR THE INTERFACE SWITCH TO BE SWITCHED ON.
THIS IS A RED SWITCH ON ONE OF THE INTERFACE CARDS.
WHEN SWITCHED ON, A SMALL RED LAMP WILL LIGHT UP.
AFTER THIS IS DONE, TYPE A CHARACTER ON THE TERMINAL.

GREMS WILL PRINT

DISK ADDRESS:

THE ANSWER IS THE SAME AS THE ANSWER TO "FROM" UNDER COPY,
BUT IT MUST BE THE ADDRESS OF THE BEGINNING OF A TRACK
(SECTOR NUMBER MUST BE ZERO).

GREMS WILL PRINT

NO. OF TRACKS:

ANSWER BY TYPING AN OCTAL NUMBER IN THE RANGE 1-04007,
OR 1-010023, OR 1-036425, OR CR.
CR (CARRIAGE RETURN) MEANS 04007/010023/036425 TRACKS (A WHOLE DISK PACK).
AFTER THIS, THE SPECIFIED TRACK(S) WILL BE FORMATTED.

GREMS WILL THEN PRINT

END FORMATTING

REMEMBER THE FORMAT SWITCH !

FUNCTION:

THE FORMAT SWITCH SHOULD THEN BE TURNED OFF AGAIN.

REFRESH

THIS IS A REFORMATTING FUNCTION.
TRACKS WILL BE REFORMATTED AND INFORMATION RESTORED, IF POSSIBLE.

THERE ARE TWO MODES OF THIS FUNCTION, REFRESHING OF ALL SPECIFIED TRACKS, OR ONLY THOSE OF THE SPECIFIED TRACKS THAT ARE BAD.
A BAD TRACK IS A TRACK WITH PARITY ERROR OR ADDRESS MISMATCH.
ADDRESS MISMATCH IS AN ERROR IN THE ADDRESS PART OF THE TRACK.
PARITY ERROR MAY BE AN ERROR IN THE ADDRESS PART, BUT IT MAY ALSO BE A DATA ERROR. IN THAT CASE, THE INFORMATION ON THAT TRACK IS IMPOSSIBLE TO CORRECT (BUT THE PARITY ERROR WILL BE REMOVED).

GREMS WILL PRINT

REFRESH ALL TRACKS (A), OR ONLY BAD TRACKS (B):

ANSWER BY TYPING A OR B.
IF A, THE USER MUST TURN ON THE INTERFACE FORMAT SWITCH (SEE FORMAT), AND TYPE A CHARACTER ON THE TERMINAL.
IF B, IT IS NOT NECESSARY TO DO THIS BEFORE A BAD TRACK IS FOUND.

GREMS WILL PRINT

DISK ADDRESS:

THE ANSWER IS THE SAME AS UNDER FORMAT.

GREMS WILL PRINT

NO. OF TRACKS:

THE ANSWER IS THE SAME AS UNDER FORMAT.

IF THE MODE IS B (BAD TRACKS ONLY), GREMS WILL ASK FOR THE FORMAT SWITCH WHEN (AND ONLY IF) THE FIRST BAD TRACK IS FOUND.

AFTER THE REFRESH IS DONE, GREMS WILL PRINT

END OF REFRESHER

REMEMBER THE FORMAT SWITCH !

FUNCTION:

CLEAR-DEVICE

THIS FUNCTION CLEARS THE DEVICE BY DOING A RETURN-TO-ZERO SEEK, FOLLOWED BY A DEVICE-CLEAR.

GREMS WILL PRINT

DISK SYSTEM (1 OR 2):

ANSWER BY TYPING 1 OR 2.

GREMS WILL PRINT

UNIT NO. (0-7):

ANSWER BY TYPING ONE OCTAL DIGIT IN THE RANGE 0-7.

THE RETURN-TO-ZERO SEEK AND THE DEVICE-CLEAR ARE THEN EXECUTED, AND GREMS WILL PRINT

FUNCTION:

DISK ERROR MESSAGES

IF A TRANSFER FAILS (STATUS ERROR), GREMS WILL TRY AGAIN.
HOWEVER, A WARNING MESSAGE IS PRINTED:

```
WARNING: MARGINAL RECOVERY.

SURF./SECT., CYL., UNIT, STATUS: <SURF./SECT.> <CYL.NO> <UNIT NO.> <STATUS>
```

IF THE RETRY ALSO FAILS, GREMS PRINTS:

```
STATUS ERROR.
SURFACE: <SURF.NO.>
SECTOR: <SECT.NO.>
CYLINDER: <CYL.NO.>
UNIT NO.: <UNIT NO.>
STATUS: <STATUS>
OPERATION WAS READ/WRITE/PARITY CHECK/COMPARE/FORMAT
```

IF THE CORE ADDRESS REGISTER FAILS, THE ERROR MESSAGE WILL
CONTAIN SOME EXTRA LINES:

```
CORE ADDR REG ERROR.
EXP.  CORE ADDR REG: EEEEEE
FAIL. CORE ADDR REG: FFFFFF
DIFFERENCE:          DDDDDD
```

FILE SYSTEM INVESTIGATOR

Users Guide

--ooOoo--

TABLE OF CONTENTS: PAGE:

17.1 Introduction 17-2

17.2 Page Types 17-3

17.3 Pointers 17-5

17.4 Available Commands 17-6

DB Dump Bit File 17-7
DD Dump Dir. Entry 17-7
DF Dump File 17-8
DO Dump Obj. Entry 17-9
DP Dump Page 17-9
DU Dump User Entry 17-9
HE Help 17-9
LF List Files 17-9
LU List Users 17-10
PF Pages in File 17-10
PL Page List 17-10
RE Restart 17-13
SO Set Output Device 17-13

--ooOoo--

17 FILE SYSTEM INVESTIGATOR - Users Guide

17.1 Introduction

The File System Investigator (FSI) has two uses:

1. To display the contents of a directory, in various ways.
2. To check a directory for errors.

The directory may be on drum, cartridge disk, floppy disk. 33/66 Mbyte disk. or 38/76/288 Mbyte disk.

FSI assumes that the directory is free of hardware errors, like format errors, parity errors etc.

The FSI is a stand-alone program. It cannot be run under SINTRAN. Furthermore, FSI cannot be used to correct a directory, as it has no write functions. FSI only reads and therefore it cannot harm the directory in any way.

All numbers input to and output from FSI are octal. The only exception is dates (date created, date opened). Dates are displayed as decimal numbers

Input is always taken from terminal 1. Output normally goes to terminal 1, but may be directed to other byte oriented units like other terminals, punch, line printer etc.

FSI is operated by two-letter commands (not followed by CR!). If the command requires more information, it will ask for it

To check a directory for errors, three commands are needed:

DD, LU, PL.

These commands will, in addition to the displaying of information, also do error checking.

17.2 Page Types

FSI assumes 11 different types of pages on a directory:

1. The Master Block

   This is always page 0. The directory entry is in words 01760-01777. See the File System Manual, Section 3.1.1.

2. Bit File Page

   Contains map of free and occupied pages. There may be up to 9 such pages.

3. The User File Index Block

   Contains from 1 to 8 pointers to pages in the user file. The first pointer is for user 0-31, the second for user 32-63, and so on

4 User File Page

   There may be up to 8 such pages. Each contains room for 32 user entries of 32 words. See the File System Manual, Section 3.1.3.

5. Object File Subindex Block

   This page will be created only when the number of users exceeds 64. Before this happens, this page is not needed. It will contain 1-4 pointers to object file index blocks.

6. Object File Index Block

   There may be up to 4 such pages. Each contains room for 512 pointers to object file pages. The first 8 pointers are for user 0, the next 8 are for user 1, and so on.

7. Object File Page

   One user may have 8 such pages, Each has room for 32 object entries of 32 words. See the File System Manual, Section 3.1.4.

Data File Subindex Block

If a user has an indexed file, and one of the pages in this file has a page number greater than 511. then this file must be subindexed, and a data file subindex block will be created. This page has room for 512 index block pointers.

Data File Index Block

Indexed files always has at least one such page. This page has room for 512 pointers to the pages of a data file.

Data File Page

A data page in a continuous or an indexed file.

Page

Any page on the device, with no meaning attached to it

17.3 Pointers

A pointer (or a file pointer) is a double word that contains a page number. It points to a page on the directory.

A pointer is correct only if it is

nonzero, and
positive, and
less than the directory size.

If the double word for a pointer is zero, it means that there is no pointer in that word.

An index block pointer is a pointer with bit 14 (016) in the first word set to 1.

A subindex pointer is a pointer with bit 15 (017) in the first word set to 1.

17.4 Available Commands

The available commands are:

DB	Dump the bit file
DD	Dump the directory entry
DF	Dump a file
DO	Dump an object entry
DP	Dump a page
DU	Dump a user entry
HE	Help. List the commands
LF	List file names for a user
LU	List user names
PF	List all page numbers for a given file
PL	List the numbers of all occupied pages on the directory, in ascending order, with corresponding user and file numbers.
RE	Restart the program
SO	Set output device number

DB Dump the Bit File

The bit file is a special continuous file, not belonging to any user. It contains information about free and occupied pages on the directory.

The length of the bit file depends on the size of the directory. For drum, cartridge disk, floppy disk and 33 Mbyte disk, the bit file occupies one page. For 38 Mbyte disk and 66 Mbyte disk, 2 pages. For 78 Mbyte disk, 3 pages, and for the 288 Mbyte disk, 9 pages.

When a page in the directory is free, the corresponding bit in the bit file is 0. When the page is occupied, the bit is 1.

To locate a bit in the bit file, for a given page number, PAGENO/020 yields the word number in the bit file, and the last 4 bits of the page number specifies the bit number in that word.

The bit file is described in the File System Manual, Section 3.1.2.

DD Dump the Directory Entry

The directory entry consists of 16 words. It is situated in the master block (page 0), words 01760-01777. It contains the directory name, pointers to the object file, the user file, and the bit file, and also the number of pages not reserved.

DD will display the information in the directory entry. In addition, some checking is also done.

FSI checks that:

- The bytes in the directory name contain no parity bit.
- The bytes are not zero.
- The three pointers point within the directory (corresponds to "outside device limits" in SINTRAN),
- Bits 017 and 016 in the first word of the object file pointer are unequal.
- Bit 016 of the first word of the user file pointer is 1.
- The number of pages not reserved is not greater than the the directory size.

FIS always fetches the directory entry from the device when the first command is executed. Therefore, if the first command is not DP, HE, RE, or SO, the directory entry will be checked at that time.

The directory entry is described in the File System Manual, Section 3 1.1.

If a directory entry dump is saved, this dump may be used to restore the directory entry if the master block is destroyed (by a disk failure, by doing @ CREATE-DIRECTORY instead of @ ENTER-DIRECTORY, etc.).

The @ CHANGE-DIRECTORY-ENTRY command will do this.

The change command in DIMS, BIMS, or GREMS, will also do this. For BIMS and GREMS, the directory entry may be found in word numbers 0760-0777. The disk address is 1 for BIMS, and CYL 0, SURF 0, SECT 1 for GREMS. For DIMS, the word numbers are 0160-0177, and the disk address is 7 or 100007 (R or F).

This will usually work because the pointers in the directory entry do not change, except when number of users pass from 63 to 64. The object file pointer will change when this happens. Also, note that number of pages not reserved usually will be wrong after this operation.

DF Dump a File

DF will dump the contents of a file. Each word of the file will be dumped as an octal number of 6 digits, or as two ASCII characters.

FSI will ask for the number of the user owning the file, and the file number. It will then ask if the dump should be octal or ASCII, and if the DUMP should contain all the pages of the file, or only up to the word pointed at by the max. byte pointer.

If the dump is wanted on the line printer, the punch, or another terminal, then the command SO should be executed first, before DF.

If the dump is octal, identical lines will be skipped

DO Dump an Object Entry

DO dumps the contents of an object entry. The names of the different fields of the object entry are also displayed.

DO will ask for the number of the user owning the file, and the file number.

The object entry is described in the File System Manual, Section 3.1.4.

DP Dump a Page

DP will dump the contents of a page. There need not be a directory on the device.

DP will ask for page number, word number of the first word to be dumped, and word number of the last word to be dumped. It will also ask if the dump is to be octal or ASCII.

In the octal dump, identical lines will be skipped.

DU Dump a User Entry

The contents of a user entry will be dumped. The names of the different fields of the user entry are also displayed.

DU will ask for the user number.

The user entry is described in the File System Manual, Section 3.1.3.

HE Help, List the Available Commands

HE displays a list of all the commands, together with a very short description of each command.

LF List File Names

LF displays a list of all file names for a user, together with the file numbers.

LF will ask for the user number.

LU List User Names

LU displays a list of all the user names, with user numbers.

The number of pages reserved by each user is accumulated and displayed at the end of the user list.

The user entries are checked as the list is displayed.

It is checked that:

- PAGES-RESERVED is within limits (not negative and not greater than maximum number of pages on the directory).
- PAGES-USED is within limits.
- The User index is as expected (0 for the first user, 1 for the next, and so on).

The user entry is described in the File System Manual. Section 3.1.3.

PF List All Page Numbers for a Given File

PF will list all the page numbers (page addresses) of a file, from the master block to the last data page.

PF will ask for the number of the user owning the file, and the file number.

The page numbers are then displayed with a text telling which type of page this is.

The page numbers are listed as one or two numbers per line. Two numbers mean all page numbers from the first to the second, both included.

At the end of the list, the number of pages in the file is displayed. This number is taken from the object entry.

PL Page List

PL will make a list of all the occupied pages on the directory, together with user number and file number for the page(s). The list will be in ascending order.

In addition to this, PL also does a lot of error checking

PL asks one question: The page list can be suppressed, or a part of it may be displayed. After this, PL searches through all the files on the directory, puts all the page numbers into a list, and sorts them. Then, the listing is displayed (or part of it), and the bit file is checked.

Below follows a list of some of the error messages from PL, with comments:

ERROR IN OBJECT FILE INDEX BLOCK.

One of the 512 pointers in the object file index block is wrong. It should have been 0, or a legal pointer.

ERROR IN DATAFILE SUBINDEX BLOCK.

One of the 512 pointers in the data file subindex block is wrong. It should have been 0, or a legal index block pointer.

ERROR IN DATAFILE INDEX BLOCK.

One of the 512 pointers in the datafile index block is wrong. It should have been 0, or a legal pointer.

SERIOUS ERROR IN THE BIT FILE, BIT IS 0, SHOULD BE 1.

This error should be corrected at once. @REGENERATE-DIRECTORY or @TEST-DIRECTORY will do this.
@CHANGE-BIT-FILE will also correct the error, and the information needed to execute this command.
BLOCK NO., WORD NO., BIT NO.,
is given in the FSI error message.

Remember that SINTRAN expects decimal block number!
If the page with the erroneous bit has been used by another file before the error is discovered, then the contents of that page may be destroyed (for instance if the spooling system used the page).

ERROR IN THE BIT FILE. BIT IS 1, SHOULD BE 0.

One of the free pages is marked as occupied, or this is a bad page detected when the directory was created. The last case is not an error. The first case is an error, but not serious, and may occur if abnormal stops of the system have occurred. The only consequence is that the directory is smaller than necessary.

@REGENERATE-DIRECTORY or @TEST-DIRECTORY will correct this error.

ERROR IN PAGES USED

If a file is indexed, FSI accumulates the number of pointers in the data file index block(s). This number is compared to PAGES-IN-FILE in the object entry, and if the two numbers are not equal, then the error message is printed. This error is not serious. It will be corrected by @ REGENERATE-DIRECTORY.

PAGE CONFLICT

This means that one or more pages are shared by two files. The reason may be a serious error in the bit file that causes a page to be allocated twice. The reason may also be that an index block is destroyed, and then this is not a real page conflict. But in both cases, it might be wise to use the command @ CHANGE-OBJECT-ENTRY, and change word 0 in the object entry to 0. This will cause the file to disappear without deleting any pages. After that, @ REGENERATE-DIRECTORY or @ TEST-DIRECTORY will correct the error.

If the page conflict is real. it would also be possible to remove the offending page from one of the files by using @ CHANGE-PAGE on the index block, and remove the pointer

Please note that in both cases the contents of both conflicting files may have been destroyed.

ERRORS OCCURRED IN OBJECT ENTRY

This may be one of several errors:

- Error in name string.
- Error in pointers to next or previous version. The user number in the left half of one of these two pointers points to another user.
- Error in object index of this entry.
- User number in the left half of this pointer, or file number in the right half, is not correct.

ERROR IN PAGES IN FILE

Negative, or greater than directory size.

ERROR IN FILE POINTER..

Bits 15 and 14 (017 and 016) in the left half of the file pointer are both 1, or
the file pointer is zero, but PAGES-IN file is not, or
the file pointer is subindex or index, and the page number is 0, or
the file pointer is not within limits.

CURRENT OPEN COUNT IS NOT ZERO

This may mean that the file has not been closed properly, maybe because SINTRAN stopped (fatal error, someone pressed the STOP button) while the file was still open.

Restart the Program

This command must be used when a new directory shall be investigated. It simulates the pressing of the buttons STOP, RESTART.

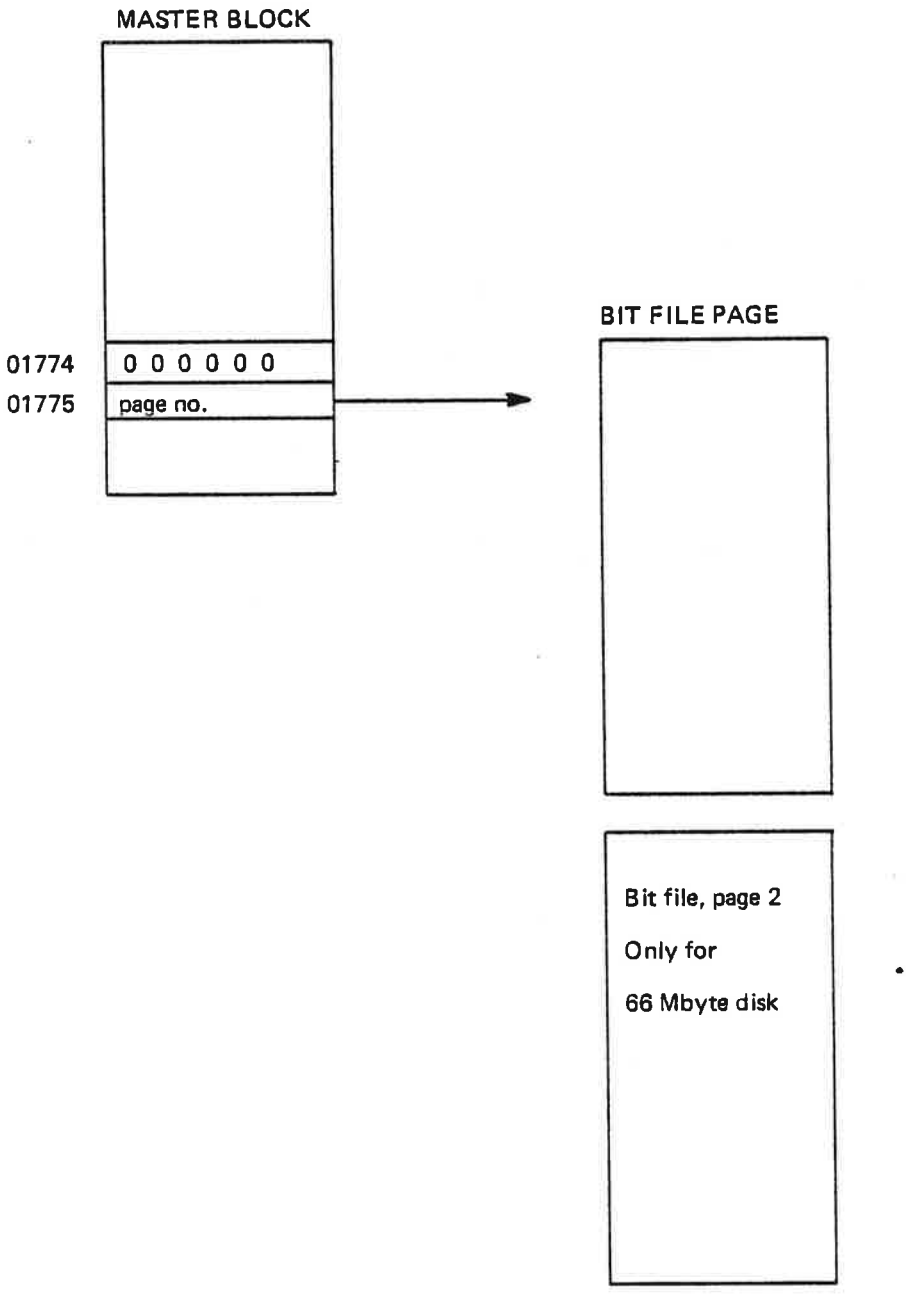
Set Output Device Number

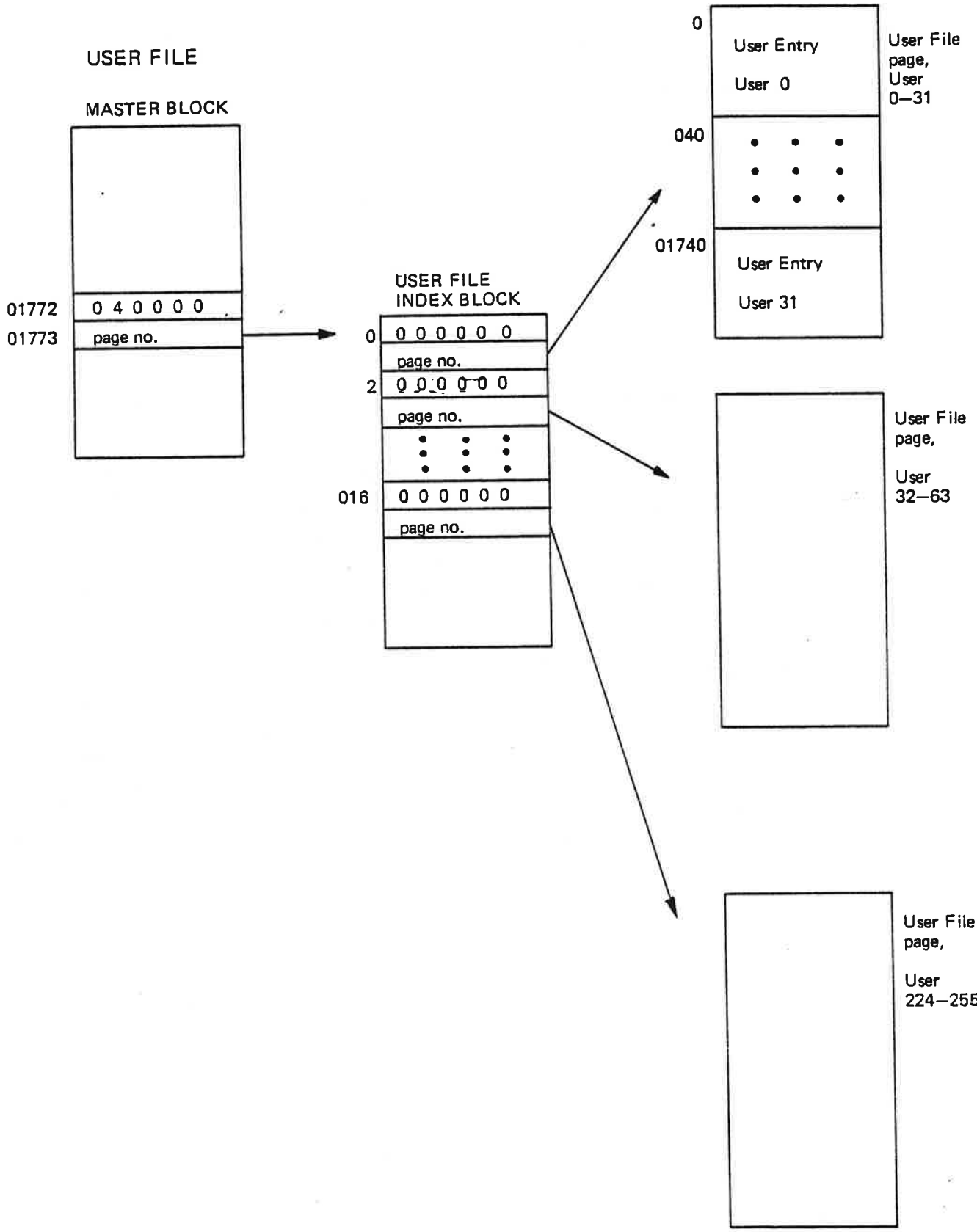
SO will change the output device number. It asks for this number. The answer may be:

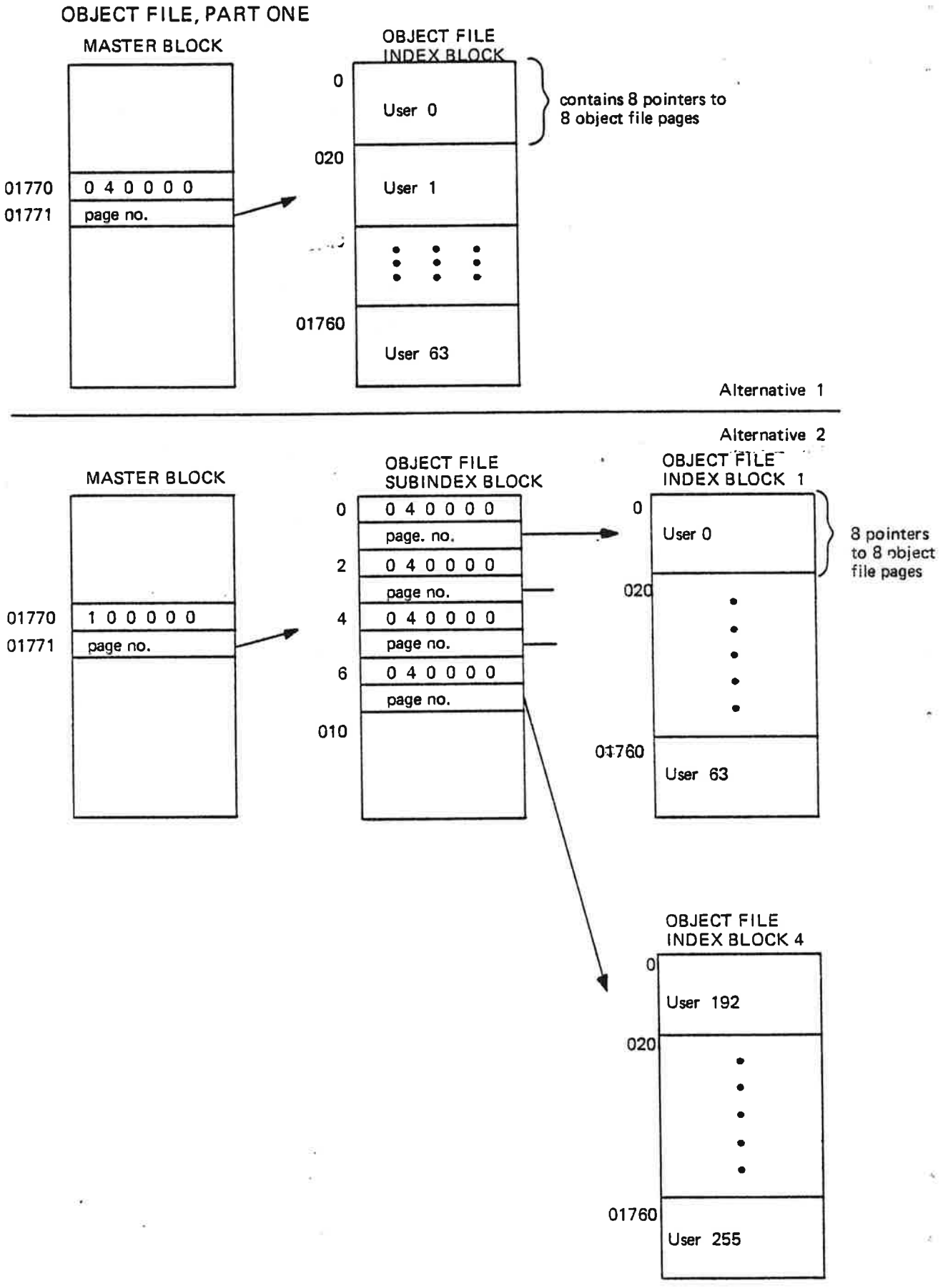
```
304 for terminal 1
314 for terminal 2
324 for terminal 3
.................
374 for terminal 8
410 for paper tape punch
430 for line printer
etc
```

After this, the output (dumps, lists) will go to the new output device.

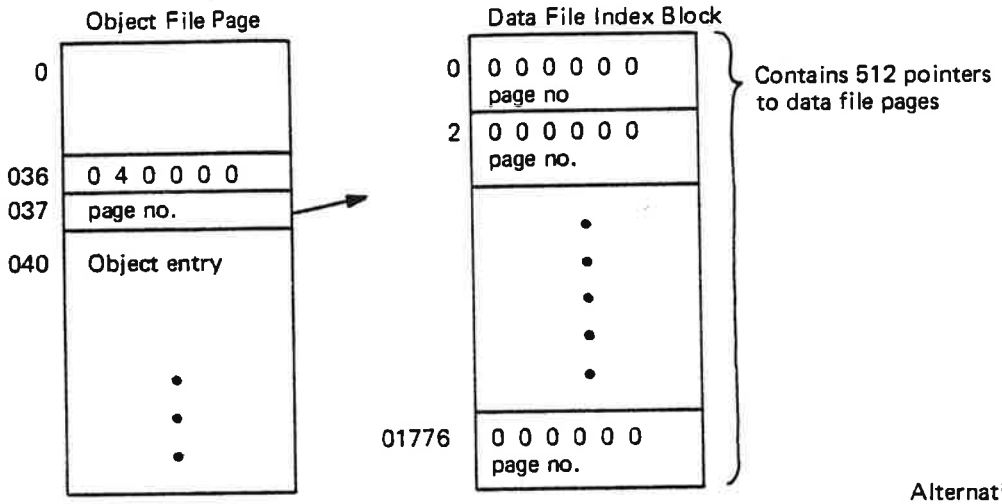
BIT FILE


MASTER BLOCK
01774
0 0 0 0 0 0
01775
page no.
BIT FILE PAGE
Bit file, page 2
Only for
66 Mbyte disk

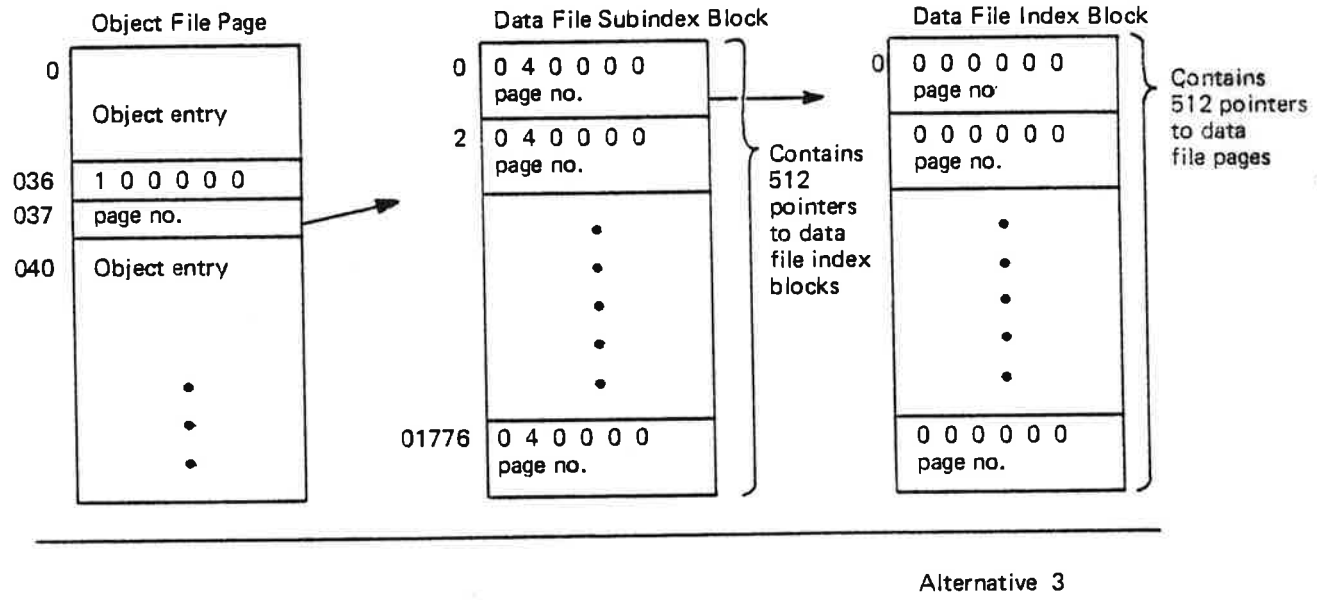

USER FILE
MASTER BLOCK
01772
0 4 0 0 0 0
01773
page no.
USER FILE
INDEX BLOCK
0
0 0 0 0 0 0
page no.
2
0 0 0 0 0 0
page no.
016
0 0 0 0 0 0
page no.
0
User Entry
User 0
040
01740
User Entry
User 31
User File
page,
User
0–31
User File
page,
User
32–63
User File
page,
User
224–255


OBJECT FILE, PART ONE
MASTER BLOCK
OBJECT FILE INDEX BLOCK
0
User 0
contains 8 pointers to 8 object file pages
020
User 1
01760
User 63
01770
0 4 0 0 0 0
01771
page no.
Alternative 1
Alternative 2
MASTER BLOCK
OBJECT FILE SUBINDEX BLOCK
OBJECT FILE INDEX BLOCK 1
0
0 4 0 0 0 0
page. no.
2
0 4 0 0 0 0
page no.
4
0 4 0 0 0 0
page no.
6
0 4 0 0 0 0
page no.
010
01770
1 0 0 0 0 0
01771
page no.
0
User 0
8 pointers to 8 object file pages
020
01760
User 63
OBJECT FILE INDEX BLOCK 4
0
User 192
020
01760
User 255

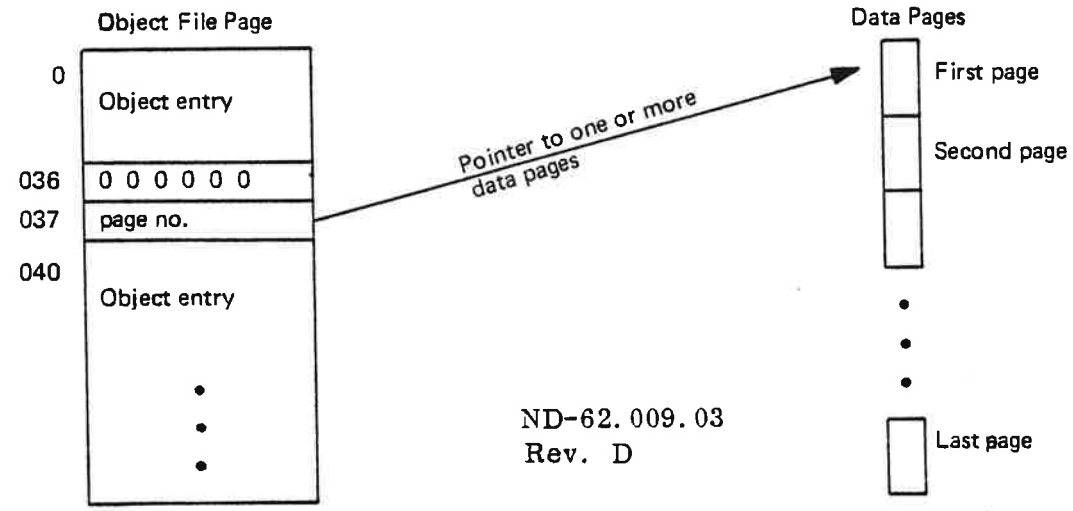
OBJECT FILE, PART TWO



Alternative 2



Alternative 3



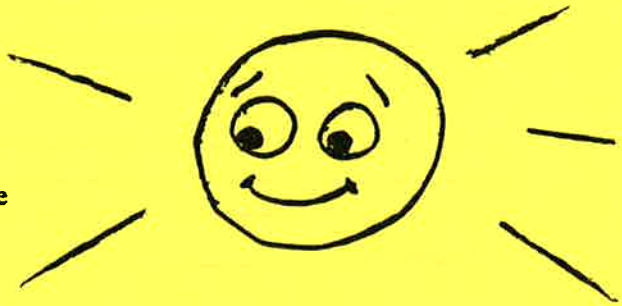
* * * * * * * * * * SEND US YOUR COMMENTS!!! * * * * * * * * * *

Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card - and an answer to your comments.

Please let us know if you

* find errors
* cannot understand information
* cannot find information
* find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!!



* * * * * * * * * HELP YOURSELF BY HELPING US!! * * * * * * * * *

Manual name: Test Program Descriptions Manual number: ND-62.009.03.

What problems do you have? (use extra pages if needed) _____

Do you have suggestions for improving this manual? _____

Your name: _____ Date: _____

Company: _____ Position: _____

Address: _____

What are you using this manual for? _____

Send to: Norsk Data A.S.
Documentation Department
P.O. Box 4, Lindeberg Gård
Oslo 10, Norway

Norsk Data's answer will be found on reverse side

Answer from Norsk Data _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Answered by _____ _____ _____ _____ Date _____

Norsk Data A.S.

Documentation Department

P.O. Box 4, Lindeberg Gård

Oslo 10, Norway

– we make bits for the future

NORSK DATA A.S BOX 4 LINDEBERG GÅRD OSLO 10 NORWAY PHONE: 30 90 30 TELEX: 18661