Test Program Description for ND 100/10S Volume 1

ND-30.005.01

NORSK DATA A.S

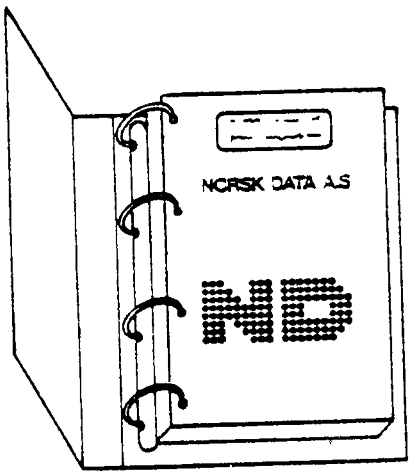
Test Program Description for ND 100/10S Volume 1

ND-30.005.01

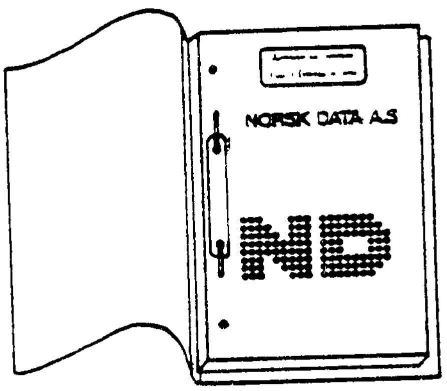
This manual is in loose leaf form for ease of updating. Old pages may be removed and new pages easily inserted if the manual is revised.

The loose leaf form also allows you to place the manual in a ring binder (A) for greater protection and convenience of use. Ring binders with 4 rings corresponding to the holes in the manual may be ordered in two widths, 30 mm and 40 mm. Use the order form below.

The manual may also be placed in a plastic cover (B). This cover is more suitable for manuals of less than 100 pages than for large manuals. Plastic covers may also be ordered below.



A Ring Binder



B Plastic Cover

Please send your order to the local ND office or (in Norway) to:

Documentation Department
Norsk Data A.S
P.O. Box 4, Lindeberg gård
Oslo 10

ORDER FORM

I would like to order

.... Ring Binders, 30 mm, at nkr 20,- per binder

.... Ring Binders, 40 mm, at nkr 25,- per binder

.... Plastic Covers at nkr 10,- per cover

Name ....
Company ....
Address ....
....
City ....

NOTICE

The information in this document is subject to change without notice. Norsk Data A.S assumes no responsibility for any errors that may appear in this document. Norsk Data A.S assumes no responsibility for the use or reliability of its software on equipment that is not furnished or supported by Norsk Data A.S.

The information described in this document is protected by copyright. It may not be photocopied, reproduced or translated without the prior consent of Norsk Data A.S.

Copyright © 1982 by Norsk Data A.S.

PRINTING RECORD

Printing	Notes
03/82	Version 01

Test Program Description for ND-100/10S, Volume 1
Publ. No. ND-30.005.01
March 1982

NORSK DATA A.S
P.O. Box 4, Lindeberg gård
Oslo 10, Norway

Manuals can be updated in two ways, new versions and revisions. New versions consist of a complete new manual which replaces the old manual. New versions incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the ND Bulletin and can be ordered as described below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and to give an evaluation of the manual. Both detailed and general comments are welcome.

These forms, together with all types of inquiry and requests for documentation should be sent to the local ND office or (in Norway) to:

Documentation Department
Norsk Data A.S
P.O. Box 4, Lindeberg gård
Oslo 10

Norsk Data A.S **M A N U A L**

TESTPROGRAM DESCRIPTION

VOLUME 1

THE PRODUCT

The manual describes the test programs used on the computers ND-100, NORD-10 S, NORD-10 and NORD-12. The programs are located on three flexible disks.

The programs most frequently used are contained on the first flexible disk. Programs not so often used are contained on floppy disks two and three.

THE MANUAL

The manual contains 6 chapters. The first chapter gives an introduction, describes the loading of test programs, the F-version of the test programs and the alternative console function. Chapter 1 also shows some examples and contains information on the TEST PROGRAM MONITOR HAR-2441E. The same chapter gives a description of the monitor commands and shows the differences in earlier versions of the TEST PROGRAM MONITOR.

Chapters 2, 3, 4, 5 and 6 give information on the test programs MEMORY, MOVER, LINE PRINTER, TERMINALS AND ASYNCHRONOUS DEVICES and the HDLC respectively.

THE READER

The manual is intended for all those in some way connected with maintaining the ND-100 and or the NORD-10 series computers.

PREREQUISITE KNOWLEDGE

Basic knowledge of the hardware in the ND-100 and or NORD-10 computer systems. This knowledge can be obtained by attending NORSK DATA courses and by studying NORSK DATA hardware manuals.

CHAPTER ONE

THE TEST PROGRAM DESCRIPTION

TABLE OF CONTENTS

1.THE TEST PROGRAM ORGANISATION

1.1.The Test Program Organisation

1.2.General

The test programs are contained on files with the same name as the program. The file may also be an abbreviation of the program name. Old program names, (older than spring 1981) also contain a program number. In order to obtain longer program names, program numbers have been omitted in later versions.

The programs can be obtained on paper tape or on diskettes. If paper tape is used, each program is on a seperate reel. If diskettes are used, the programs are located on three diskettes. Diskette one contains the test programs most frequently used, while diskette two and three contain test programs that are not used so frequently. Within each diskette the programs are arranged in an alphabetical order.

Example of labelling the three diskettes:

TEST-PROGRAM NO. 1 FOR ND-10, ND-12 AND ND-100 DIR. NAME: ND-10324B USER NAME: FLOPPY-USER	TEST-PROGRAM NO. 2 FOR ND-10, ND-12 AND ND-100 DIR. NAME: ND-10325A USER NAME: FLOPPY -USER	TEST-PROGRAM NO. 3 FOR ND-10, ND-12 AND ND-100 DIR.NAME: ND-10326A USER NAME: FLOPPY-USER

1.3.Reporting Errors in Testprograms

As soon as an error in a test-program is reported to **TECHNICAL SUPPORT**, a description of the error is included in the list called **"KNOWN BUT NOT CORRECTED ERRORS"**. The first time an error is reported in the list, it is marked with NEW==>. This is removed when the error has appeared once in the list. When a new program that corrects the error is released, the error disappears from the list.

Only the latest version of the test programs should be used.

1.4. Version Numbering of Programs and Directories

Both new and old programs include a version character as the last character of the name. The version character is increased each time a new program release is issued. This makes it easy to identify the program at hand.

DISC-TEMA-A:BPUN;1
Example of "new" filename without program number. Version number is A.

MOVER-1863C:BPUN;1
Example of "old" filename with program number. Version number is C.

The diskette directory name also has a version character. The version character is increased each time a new version of the diskette is released. A new version of at least one new program on the diskette causes a new release of the diskette.

A list of the latest versions of the programs and directories is found in the list called **"LATEST ISSUED TEST-PROGRAMS"**. The list also gives the initials of the persons responsible for the programs.

1.5. Distribution of the Lists

Both lists are regularly updated in the **Service Hand Book Vol.II.**

1.6. Program Descriptions

There is a **Program description** delivered with each diskette. It gives a short description of the programs, including start address, restart address and the program boundaries. The program boundaries define how much memory is required to load the program.

1.7. Where to obtain Programs

The programs may be obtained from your local ND office.

1.8.Loading of Test Programs

1.8.1.General

The examples given in the following chapters show the user input at a given point. Complete examples are given in section 1.11.

1.8.2.Stand Alone

The Microprograms in ND-10, ND-12 and ND-100 are capable of performing "binary load" from devices that conform with the programming specifications of a tape reader.

During loading, the floppy disc interface conforms with the programming specifications for the tape reader.

This facility is activated by, in OPCOM, typing the device number of the desired device (1560 for floppy and 400 for tape reader), followed by the sign & (no CR). The diskette must be inserted into the floppy drive connected to controller 1, unit 0. For further information on OPCOM see:

- ND-100 Reference Manual	ND-06.014
- ND-100 Functional Description	ND-06.015

1560&
Example of loading from diskette.

Since there are several programs on one diskette, we have to use the **FLOPPY MONITOR** to select the desired program. The program loaded and started by the micropogam is therefore the **FLOPPY MONITOR** (see section 1.9).

400&
Example of loading from tape reader.

Each program is on its own paper roll, therefore the program loaded and started by the microprogram is the program itself.

1.8.3.Loading under SINTRAN

Some test programs may be run under **SINTRAN,** see the**PROGRAM DESCRIPTION** for the desired program.

If the program is to be run under **SINTRAN** use the commands @PLACE-BINARY and @DUMP. See **PROGRAM DESRIPTION** for the desired program in order to find the start and restart addresses. The command @LOAD-BINARY may also be used. If unfamiliar with these commands see:

- SINTRAN III Reference Manual ND-60.128

Example using LOAD-BINARY:

```
@LOAD-BINARY (ND-10324B:FLOPPY-USER)DISC-TEMA-A
```

Example using PLACE-BINARY and DUMP:

```
@PLACE-BINARY (ND-10324B:FLOPPY-USER)DISC-TEMA-A

@DUMP "DISC-TEMA-A"
START ADDRESS: 0
RESTART ADDRESS: 20
```

1.9.FLOPPY MONITOR

This description is valid for the **F-version** of the **FLOPPY MONITOR** (FLOPPY MONITOR-2010F).

The **FLOPPY MONITOR** program has only 4 commands. They are:

LIST-FILE
LOAD-FILE
PLACE-FILE
HELP

The **FLOPPY MONITOR** is ready to receive a command when it has typed the character *.

All input is terminated by CR.

The commands and the filenames may be abbreviated following the standard SINTRAN rules for abbreviation. The parameter for the command may be given on the same line as the command.
The **FLOPPY MONITOR** translates lower case letters into upper case.

1.9.1.LIST-FILE <logical device>

This command lists all BPUN files found on the diskette. The list can be printed either on the console or on the line printer number 1 (device number 430), by answering 1 or 5 respectively to the question LOG DEV. Default logical device number is 1.

Example:

```
*LIST-FILE 1
FILE 0 : (ND-10324B:FLOPPY-USER)BIG-RAND-1876C:BPUN;1
FILE 1 : (ND-10324B:FLOPPY-USER)BIGFUNC-1824I:BPUN;1
FILE 2 : (ND-10324B:FLOPPY-USER)BIMS-1871C:BPUN;1
FILE 3 : (ND-10324B:FLOPPY-USER)CONFI-INV-1672M:BPUN;1
FILE 4 : (ND-10324B:FLOPPY-USER)DIMS-1453F:BPUN;1
FILE 5 : (ND-10324B:FLOPPY-USER)DISC-TEMA-A:BPUN;1
FILE 6 : (ND-10324B:FLOPPY-USER)EXTEN-ONE-1519B:BPUN;1
FILE 7 : (ND-10324B:FLOPPY-USER)FLOPP-FORM-1990A:BPUN;1
FILE 8 : (ND-10324B:FLOPPY-USER)FLOPPY-FU-1986F:BPUN;1
FILE 9 : (ND-10324B:FLOPPY-USER)FLOPPY-RAN-1988A:BPUN;1
FILE 10 : (ND-10324B:FLOPPY-USER)GREMS-2231G:BPUN;1
FILE 11 : (ND-10324B:FLOPPY-USER)HDLC-2-2370A:BPUN;1
FILE 12 : (ND-10324B:FLOPPY-USER)LP-TEST-1878B:BPUN;1
FILE 13 : (ND-10324B:FLOPPY-USER)MOVER-1863C:BPUN;1
FILE 14 : (ND-10324B:FLOPPY-USER)MPM-MAINT-2177B:BPUN;1
FILE 15 : (ND-10324B:FLOPPY-USER)MULTI-1820M:BPUN;1
FILE 16 : (ND-10324B:FLOPPY-USER)PFAIL-1355F:BPUN;1
FILE 17 : (ND-10324B:FLOPPY-USER)SUPER-RAND-2222C:BPUN;1
FILE 18 : (ND-10324B:FLOPPY-USER)T32KMOS-2178D:BPUN;1
FILE 19 : (ND-10324B:FLOPPY-USER)TANB-MAG-1559F:BPUN;1
FILE 20 : (ND-10324B:FLOPPY-USER)TTEST-1206A:BPUN;1
*
```

1.9.2.LOAD-FILE <file name>

Loads the specified file into memory and starts execution of the program.

Example:

```
*LOAD-FILE  DISC-TEMA-A
```

1.9.3.PLACE-FILE <file name>

The command is similar to **LOAD-FILE**, but execution of the program is not started. Instead the message **PLACED** is typed, and the machine goes into **STOP**. The first instruction after the **WAIT** is a jump (**JMP**) to the start of the "placed" program. Thus, when the required **OPCOM** activity is finished, the program may be started by using the **OPCOM** function.

The command is convenient when patching or a breakpoint is needed.

Example:

```
*PLACE-FILE  DISC-TEMA-A
PLACED
```

1.9.4.HELP

This command types the name and version number of the FLOPPY MONITOR, and a list of the 4 commmands.

Example:

```
*HELP
FLOPPY-MONITOR-2010F
LIST-FILE
LOAD-FILE
PLACE-FILE
HELP

*
```

1.9.5.Automatic Load Function

If the diskette contains only one :BPUN file and the X-register on level 15 (decimal) is different from 0, the FLOPPY MONITOR automatically loads this one.

1.9.6.FLOPPY MONITOR Error Reporting

The following program stops are defined (last instruction was WAIT):

WAIT 77 : not possible to read bootstrap from floppy (hardware error)

WAIT 0 : always when PLACE-FILE command is used (legal wait)

Any other error situation will give a self explanatory error message. If the error occurs after the loading of a program has started, an error message will be given and the FLOPPY MONITOR will be reloaded.

1.10.Alternative Console Function

The FLOPPY MONITOR and the programs that include the TESTPROGRAM MONITOR (see section 1.12), may be run on any terminal.

This is activated by the alternative console function. If a terminal different from device 300 (terminal 1) is to be used as console device, set the T-register on level 15 (decimal) to any legal terminal device number (310,320 etc.) prior to typing 1560&. The text

<IF HERE TYPE ANY CHAR<

will be printed on both device 300 and on the ´alternative´ console device. The first device that sends a character to the computer will be taken as console device. (see section 1.14.12)

This happens either when the FLOPPY MONITOR or when the TESTPROGRAM MONITOR is loaded.(See the example in section 1.11.3).

Example:

In OPCOM:

```
17R6/xxxxxx 340 <CR>
```

For further details on OPCOM:

- ND-100 Reference Manual	ND-06.014
- ND-100 Functional Description	ND-06.015

1.11. Examples of complete Loading of a Program

1.11.1. Example of Standard Load from Diskette

```
#1560&
*LOAD DISC-TEMA-A
DISC TEST AND MAINTENANCE SYSTEM (DISC-TEMA)

203134.A

ISSUED : 1. SEPTEMBER 1981

DISC NAME :
```

1.11.2. Example of Standard Load from Paper Tape

```
#400&
DISC TEST AND MAINTENANCE SYSTEM (DISC-TEMA)

203134.A

ISSUED : 1. SEPTEMBER 1981

DISC NAME :
```

1.11.3. Example of Load from Diskette with Alternative Console Function

```
#1560&
IF HERE TYPE ANY CHAR
*LOAD DISC-TEMA-A
IF HERE TYPE ANY CHAR
DISC TEST AND MAINTENANCE SYSTEM (DISC-TEMA)

203134.A

ISSUED : 1. SEPTEMBER 1981

DISC NAME :
```

1.11.4. Example of "PLACE" from Diskette

```
#1560&
*PLACE-FILE DISC-TEMA-A
#100/044405 4405
!
DISC TEST AND MAINTENANCE SYSTEM (DISC-TEMA)

203134.A

ISSUED : 1. SEPTEMBER 1981

DISC NAME :
```

1.12.TEST PROGRAM MONITOR HAR-2441E

This description concerns the E-version of the Test Program Monitor only (issued January 1982). Some facilities are not valid in earlier versions (see chapter 1.13). The PD-sheets will tell which version of the TEST PROGRAM MONITOR the different tests programs include.

1.12.1. General

The TEST PROGRAM MONITOR makes the different test programs behave in a uniform way. It is intended that the programs output as little text and questions as possible. If more information is required it may be asked for.

The monitor is based on a command structure i.e., the user can specify the desired action by writing a "command" to the program.

The command handling is done by the monitor, and when it is ready to accept a new command it types the sign >.
In order to return to the monitor, the tests may be terminated in different ways, but it is always possible to use **ESCAPE.**

The Monitor has a fixed set of commands common to all programs. There is also a set of programs or special commands that perform the intended tests.

1.12.2.Error Messages from the TEST PROGRAM MONITOR

The test program monitor enables the internal interrupt system on level 14. The following errors will be reported:

IIC:	4 =	Illegal instruction
IIC:	6 =	Privileged instruction (only valid for programs that use PAGING)
IIC:	7 =	IOX error
IIC:	8 =	Memory parity error
IIC:	9 =	Memory out of range
IIC:	10 =	Power fail

The error message also reports the address of the instruction that caused the error, the instruction itself and the level.

Example:

```
 INTERNAL INTERRUPT. IIC: 7 -IOX ERROR
AT ADDRESS: 013547 CAUSING INSTRUCTION : 164200 LEVEL : 0
```

1.12.3.Stand Alone Test Programs

All monitor commands can be executed stand alone.

1.12.4.Test Programs under SINTRAN

Programs that include the TEST PROGRAM MONITOR may be loaded under SINTRAN and some of the monitor commands may be executed. See description of each command (section 1.14). It then depends on the program whether the commands may be executed under SINTRAN or not. When trying a non-executable command, the message;

```
NOT EXECUTABLE UNDER SINTRAN
```

will be given. It is always possible to do an "EXPLAIN" command and get acquainted with the program.

1.12.5.Syntax of Commands and Parameters

Command handling is similar to that in SINTRAN. It is possible to abbreviate text strings separated by hyphens (-) up to the point of ambiguity. Commands and parameters on the same line may be separated with commas or spaces and terminated by CR. The commands always respond with the unabbreviated name.
The TEST PROGRAM MONITOR accepts lower case characters.

There are some editing possibilities:

CTRL A:	Deletes the last character written. Echoes with ^.
CTRL Q:	Deletes the whole line. Echoes with carriage return and line-feed.

1.12.5.1.Parameter Input

When a program expects input from the user, the following is done:
The program types the prompt and the user answers. If the parameter has been given on the same line as the command or an earlier parameter, the promt is not typed.

The program expects the answer in one of the following ways. (Listed in order of appearance.)

1) Number input

2) Menu selection

3) Yes/No selection

4) String input

The ways to ask for input, and to answer, are standardised as follows:

1) Number Input;
The prompt consists of a question, followed by a space, some information between brackets, a colon and a space. After the last space the program is waiting for user input.

Between brackets is the radix that the program expects to be used, and if the parameter value has a limited range, the minimum and maximum values are also given.
It is possible to overrule the radix suggested by the program by typing D, B or H after the number. This changes the radix of the input to decimal, octal or hexidecimal respectively.

If limits exist for the number, the input is checked against these limits.
The characters given in the input are checked to see if they are included in the radix in use. For use of details see 3.

Example of octal number input with limits:

```
CYLINDER NUMBER (0-1466 OCT.):
```

Example of decimal number input without limits:

```
GIVE LOGICAL-DEV-NO (DEC.):39
```

2) Menu Selection;
When a program enters a menu selection situation, the following applies: The prompt is typed followed by a space and colon. Help will give a list of the menu. The list has one menu entry on each line. The lines start with the menu entry number. If one of the menu entries is default, it will be marked with * (see the following example). To select from the menu, either the menu entry or its number is given.

If the given input is either nonexistent or ambiguous, an error message is typed and the input buffer is cleared.

Example of menu selection:

```
ADDRESS TYPE: HELP
1) *PHYSICAL-ADDRESS
2) LOGICAL-ADDRESS
3) PAGE-ADDRESS
4)  EXIT
ADDRESS-TYPE: PAGE-ADDRESS
```

3) Yes/No Selection;
Yes/No selection is a special case of menu selection with yes and no as the only menu entries. The question is typed followed by YES/NO. HELP does not work here. Numerical input is not legal here.

4) String Input;
String input is only used to read strings (i.e., when reading a file name as parameter).

1.12.5.2.Defaults

If parameters have default values, they are presented when answering with CR only. It is then possible to give another parameter or accept the default value by typing another CR.
If no default value exists, the question is repeated.

Example:

```
>PRINT-NOTE
PRINT-NOTE

NOTE NUMBER:(CR)ALL NOTES(CR)

1) THE IOX FOR THIS DISC SYSTEM GAVE IOX ERROR

2) IF YOU WANT TO DO THESE COMMANDS WITH THESE DISCS
 YOU MUST DO THE COMMAND DIRECTORY-MODE FIRST

3) THE SAME PATTERN WAS WRITTEN INTO BOTH MEMORY ADDRESS REGISTER
AND. . . .
. . . . .
```

1.12.5.3.Immediate Action Input

Immediate action input means that the keyboard of the console is scanned at a regular time period. Pressing an immediate action key, triggers the action associated with that key immediately. The TEST PROGRAM MONITOR has only two such keys: ESC and CTRL O. The various programs might at certain points specify other immediate action characters (see the description of the various programs).

1.12.6. Start-up Sequence of the TEST PROGRAM MONITOR

The following flow chart shows briefly what the Monitor does when a program is started or restarted.

Start address: 0

```
Clear internal registers.
Determine computer type.
Determine console (See chapter 4)
Type: PROGRAM TITLE.
Execute: Initial Command 1.
```

Restart address: 20

```
Allocate space for buffers.
Execute: Initial Command 2.
Clear interrupts on level 10, 11, 12 and 13.
Initialize clock, level 13 and 14.
Type:
THE COMMAND HELP GIVES YOU A LIST OF THE COMMANDS.
>
```

To the user, the difference between start and restart, is that the title is not typed after restart.

Note that the Initial Commands are different for every testprogram and may cause additional outputs to the ones mentioned above.

Example:

<u>Start:</u>

Title:

```
DISC TEST AND MAINTENANCE SYSTEM (DISC-TEMA)

203134.A

ISSUED : 1. SEPTEMBER 1981
```

Initial Command:

```
DISC NAME : D-7-1
DISC-75MB-1
DATA WAY TO DISC SYSTEM 1 TESTED
MEMORY ADDRESS REGISTER ON DISC SYSTEM 1 TESTED

THE COMMAND HELP GIVES YOU A LIST OF THE COMMANDS

>
```

<u>Restart:</u>

```
THE COMMAND HELP GIVES YOU A LIST OF THE COMMANDS

>
```

1.13.Earlier versions of the Test Program Monitor

This chapter describes the differences in earlier versions. The description of each program tells which (if any), version of the test program monitor it includes.

1.13.1.D-Version of TEST PROGRAM MONITOR

D-version (and earlier) of the TEST PROGRAM MONITOR does not permit typing parameters on the same line as the command. Each parameter has to be terminated with CR.
Except for SET-CONSOLE-DEVICE-NUMBER there are no default possibilities.

When typing a number it is not possible to overrule the radix by typing a character.

In addition to the present set of monitor commands, the D-version has the following commands:

1) CLEAR-PRINTER-BUFFER; This command clears the printer buffer. (See this section 4.)

2) DUMP-PRINTER-BUFFER <device number>; This command asks for a device number and dumps the content of the printer buffer to that device.

Example:

```
>DUMP-PRINT
DUMP-PRINTER-BUFFER

DEVICE NUMBER : 340

>
```

3) PROGRAM-PURPOSE; This command types the program purpose (as default of EXPLAIN-COMMANDS in newer versions of the monitor).

4) SET-PRINTER-DEVICE-NUMBER <device-number>; In addition to what is mentioned in section 1.13.1 the D-version has the possibility to set printer device to 0. Then the outputs meant for the printer are placed in the memory, and may later be dumped by the DUMP-PRINTER-BUFFER command.

5) STOP-SYSTEM; The D-version has both the commands STOP-SYSTEM (stand alone only) and EXIT (SINTRAN only).

1.13.2.C-Version of TEST PROGRAM MONITOR

Similar to the D-version, but cannot be loaded under SINTRAN. The command HELP does not clearly seperate between monitor commands and program commands, but program commands are always listed at the end.

1.13.3.Other Versions of TEST PROGRAM MONITOR

Some programs have a command handling similar to the TESTPROGRAM-MONITOR. See desription of each individual program

1.14.Description of each Monitor Command

1.14.1.DATCL

This command types the day, hour and minute according to the program clock.

Example:

```
>DATCL
DATCL
20,14:53
```

1.14.2.EXIT

If SINTRAN is running:

Exits from the test program back to SINTRAN.

If stand alone:

Turn off interrupt and paging before the computer is stopped (wait).

The first instruction after the wait is a jump back to the Test Program Monitor (>).

Example:

```
>EXIT
EXIT

@
```

1.14.3.EXPLAIN-COMMANDS <command>

This command will explain a given command more thoroughly. It asks for the command name.

Default: The purpose of the program is typed.

Example:

```
>EXPLAIN-COMMAND
EXPLAIN-COMMAND
COMMAND : PRINT-NOTE
PRINT-NOTE

PRINTS THE NOTES MENTIONED IN THE ERROR MESSAGES
ON THE DEVICE SELECTED BY THE SET-PRINTER-DEVICE-NUMBER
CR. PRINTS ALL NOTES
```

1.14.4.FLOPPY-LOAD

Loads from floppy system 1 unit 0 , same as 1560&.

This makes it possible to load the FLOPPY MONITOR (and another test program) when working on a terminal seperated from the computer room.

This command is not executable under SINTRAN.

Example:

```
>FLOPPY-LOAD
FLOPPY-LOAD

*
```

1.14.5.GO-TO-ADDRESS <address>

Starts execution at the address specified. A listing and good knowledge of the program is required to execute this command, except when used for address 0 (start) and 20 (restart).

Default: None.

Example:

```
>GO-TO-ADDRESS
GO-TO-ADDRESS

ADDRESS : 20

THE COMMAND HELP GIVES YOU A LIST OF THE COMMANDS
>
```

1.14.6.HELP <command>

Help asks for command and lists all commands that match the answer. Monitor commands and program commands are divided into two groups with headings.

Default: All available commands.

Example:

```
>HELP
HELP
COMMAND : SET

MONITOR-COMMANDS:
*****************
SET-CONSOLE-DEVICE-NUMBER
SET-PRINTER-DEVICE-NUMBER

PROGRAM-COMMANDS:
*****************
SET-DISC-CONTENT
SET-DISC-TYPE
SET-RETRY
```

1.14.7.LIST-SPECIAL-COMMANDS <command>

This command asks for command name and lists only the program commands that match the answer.

Default: All program commands are listed.

Example:

```
>LIST-SPECIAL-COMMANDS
LIST-SPECIAL-COMMANDS
COMMAND : SET

SET-DISC-CONTENT
SET-DISC-TYPE
SET-RETRY
```

1.14.8.MESSAGE <device number> <message>

This command is used to send a message to another terminal. The device number for the receiving terminal must be specified .

Defaults: None.
This command is not executable under SINTRAN.

Example:

```
>MESSAGE
MESSAGE

TO DEVICE NUMBER : 300

TYPE THE MESSAGE. CR TERMINATES
ENTER FLOPPY DISCETTE: TESTPROGRAM NO. 2 !

>
```

1.14.9.OPCOM

This command sets OPCOM on ND-100. If not ND-100, the text: "THIS OPTION IS ONLY AVAILABLE ON ND-100" is typed.
OPCOM communicates with the device number 300. It is advisable to use another terminal as console for the program when OPCOM is activated.
To leave OPCOM press ESC. on device number 300.
This command is not executable under SINTRAN.

For further details on OPCOM:

- ND-100 Reference Manual	ND-06.014
- ND-100 Functional Description	ND-06.015

Example:

```
>OPCOM
OPCOM
>
```

The yellow lamp(s) on the ND-100 are lit.

1.14.10.PRINT-NOTE <note number>

Some error messages refer to notes by giving a number and a closing bracket. This command asks for note number and prints that note on the device selected as printer device (see section 1.14.13).

Default: All notes are printed.

Example:

```
NO SUCH DEVICE 1)

>PRINT-NOTE
PRINT-NOTE

NOTE NUMBER : 1

1) THE IOX FOR THIS DISC SYSTEM GAVE IOX ERROR

>
```

1.14.11.PROGRAM-STATUS

This command types the status of different variables used in the program. The monitor types some variables (shown in the following example). The special programs may add variables used in program.

Example:

```
>PROGRAM-STATUS
PROGRAM-STATUS

-------------------------------------------------------------
*   DAY,    * CONSOLE* PRINTER*  STOP ON *  INTER-  *  PAGING *
* HOUR:MIN  * DEV.NO.* DEV.NO.* FULL PAGE*   RUPT   *         *
-------------------------------------------------------------
* 20,16:04  *   340   *   430   *    OFF   *    ON    *    OFF  *
-------------------------------------------------------------
>
```

1.14.12.SET-CONSOLE-DEVICE-NUMBER <device number>

Command to set the number of the terminal with which the program communicates. It is useful if a screen is wanted instead of a printing terminal. If the console device is not set, device number 300 is used. The alternative device number is written into the T-register on level 15, and will be used when loading other test programs.

Default: The message

THIS IS DEVICE NO:....,IF HERE PRESS ANY CHAR

is written to all possible terminal devices, and it is possible to reselect the console device.
This command is not executable under SINTRAN.

Example:

```
>SET-CONSOLE-DEVICE-NUMBER
SET-CONSOLE-DEVICE-NUMBER

DEVICE NO. : 340
```

Device number 340 will now type the > and become the console for the program.

1.14.13.SET-PRINTER-DEVICE-NUMBER <device number>

This command is used to set the device (terminal or printer) at which the program will print. If printer device is not set, the device selected as console is used. If stand alone, the command asks for device number. Under SINTRAN it asks for file name.

Default: Console device number.

Example:

Stand alone:
```
>SET-PRINTER-DEVICE-NUMBER
SET-PRINTER-DEVICE-NUMBER

DEVICE NO. : 430

>
```

Under SINTRAN:
```
>SET-PRINTER-DEVICE-NUMBER
SET-PRINTER-DEVICE-NUMBER

FILE NAME : L-P

>
```

1.14.14.START-CLOCK

This command activates the program clock. The clock is started automatically when the program is started. It is therefore not required to use this command unless the command STOP-CLOCK has been used.

This command is not executable under SINTRAN.

Example:

```
>START-CLOCK
START-CLOCK

>
```

1.14.15.STOP-CLOCK

Command to stop the program clock. The clock is started automatically when the program is started.

This command is not executable under SINTRAN.

Example:

```
>STOP-CLOCK
STOP-CLOCK

>
```

1.14.16.TERMINAL-MODE <yes/no>

Terminal mode gives the opportunity to set/reset display mode, i.e. stop on full page.

Default: The opposite of the present state.

Example:

```
>TERMINAL-MODE
TERMINAL-MODE
STOP ON FULL PAGE  (YES/NO) : YES

DISPLAY MODE : PRESS ANY CHARACTER TO GET NEXT PAGE.
```

1.14.17.UPDAT <day> <hour> <minute>

Asks for day, hour, minute and updates the program clock.

Defaults: Present values.

This command is not executable under SINTRAN.

Example:

```
>UPDAT
UPDAT
DAY (0-31) : 20
HOUR (0-24) : 14
MINUTE (0-59) : 53

>
```

CHAPTER TWO

MEMORY

TABLE OF CONTENTS

2.MEMORY

2.1.General

This test program will test memory modules on NORD-10 and ND-100 computer systems. It includes the TEST PROGRAM MONITOR and library.
The program permits to test systems with non correcting local memory, error correcting local memory and big multiport memory (BMPM) and it includes a simple "Service Program" which acts upon the multiport ERROR LOG in order to print the contents of the ERROR LOG and to scan the "Port Status".
One disc interface can be run in test mode, in parallel with the actual memory test.

A few things might be useful to know:

When running on a ND-100, extended addressing will always be set.

The first 64K of memory is mapped onto page table 0.

Memory is tested in blocks of 4K words (10.000 oct. words).

The 64K of memory currently being tested is mapped onto page table 1 and is accessed by enabling Alternative Page Table usage (setting bit 0 of the status word) and then turning the paging system on and off.

2.2.Initial Action

At startup, the program will do the following:

1. Allocate program buffers. They are placed directly after the program code and will contain information about the memory configuration.

2. Find out the computer type. (10, 10/S, 100).

3. Initialize Paging Control Registers and Page Index Table 0.

4. Find out the memory type and size and build the memory maps. When the computer has BMPM memory, the detection of the memory type (which is done by forcing parity errors) will cause errors to be reported to the ERROR LOG. These errors will be printed when using the BMPM Service Program command ERROR-LOG-SCAN.

5. Print the information found by steps 2 and 4.

6. Initialize first part of program variables:

    a. Indicate no tests skipped.
    b. Set lower and upper limits to include the total memory.
    c. Indicate that Single Test Mode is not selected.
    d. Indicate that Loop Mode is selected with Infinite loop.
    e. Indicate that Abort Mode is selected with abort after 8 error messages.
    f. Indicate that error message are not to be suppressed.

7. Clear the error lamps on ECCR and BMPM memory.

A restart of the program (at address 20) will have no influence on the setting of the different program variables i.e MODE-setting and memory-limits definitions will be unchanged.

2.3. Description of the Commands

2.3.1. RUN

This command will start the actual memory test sequence as defined by the **MODE** command. The following tests exist:

1) Read test on program part:

The memory area where the program is situated is read, in order to try to detect parity errors in that area. This test runs with paging off.

2) Addresses in addresses:

Memory is written with its own address as data and then read back. If the data read back is different from the memory address, an error message is given.

3) Write/read test (7 patterns):

Test patterns are written to and read back from memory. If the data read back is different from the data written, an error message is given.

The used patterns are:

0 52525 125252 44444 146314 31463 and 177777

4) Rapidly changing address bits:

The test computes a checksum by executing ADD ,X; ADD ,B.
X is incremented by 1 and B is decremented by 1 until the part of memory currently being tested is done. The test is repeated and the two results are compared. If they are different, an error message is given.

5) Parity error detection:

This test is only run if the tested memory has error correction. The memory is written to, while forcing parity errors, then read back. Each location read back should give an interrupt on level 14. If this is the case then all is ok. If no interrupt occurs or if the contents of the PES and/or PEA register is not as expected, an error message is given.

When running with multiport memory, this test will cause errors to be logged in the BMPM ERROR LOG. This must be taken into account when using the command **ERROR-LOG-SCAN** in the BMPM Service Program. It is therefore best to first run the Parity Error Detection test seperately, then clear the ERROR LOG, remove the Parity Error Detection test from the test sequence (see the command **SKIP-TESTS**) and then run the rest of the tests.

6) Walk test (34 patterns):

The patterns are:

0 1 2 4 10 20 40 100 200 400 1000 2000 4000 10000 20000 40000
100000 177777 177776 177775 177773 177767 177757 177737 177677
177577 177377 176777 175777 173777 167777 157777 137777 77777

Example of the **RUN** command:

```
>RUN
RUN

AREA TESTED: 0.5-0.15
READ TEST ON  PROGRAM  PART     === END OF TEST ===
ADDRESSES IN ADDRESSES          === END OF TEST ===
WRITE/READ TEST (7 PATTRENS)    === END OF TEST ===
RAPIDLY CHANGING ADDRESS BITS   === END OF TEST ===
PARITY ERROR DETECTION          === END OF TEST ===
WALK TEST (34 PATTERNS)         === END OF TEST ===

=== THE TESTS ARE NOW LOOPING ===
```

2.3.2.START-DMA-TRANSFERS

This command activates a driver on level 11 that does continuous DMA transfers in test mode on either device 500 or device 1540. The program will tell which DMA device is used. One can use this command to load the bus while running the actual memory test. The data transferred is not tested so the only error that can be caused by the DMA tranfers is MOR (Memory Out of Range).

IMPORTANT: switch power OFF on the discs before giving this command.

Example:

```
>START-DMA
START-DMA-TRANSFERS

DMA DEVICE : 500

== TRANSFERS STARTED ==

>
```

2.3.3.STOP-DMA-TRANSFERS

The command is used to stop the DMA transfers activated by the **START-DMA-TRANSFERS** command.

Example:

```
>STOP-DMA
STOP-DMA-TRANSFERS

== TRANSFERS STOPPED ==

>
```

2.3.4.PRINT-MEMORY-MAP

Used to print a map of the existing memory. The letters printed have the following meaning:

X: Memory without error correction

E: Memory with error correction

M: Multiport memory

Example:

```
>P-M-M
>PRINT-MEMORY-MAP

                 === M E M O R Y   M A P ===
       0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15  (4K UNIT)
      04 08 12 16 20 24 28 32 36 40 44 48 52 56 60 64  (X 1K)
BANK-----------------------------------------------
     0 E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E
     1 E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E
     2 E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E

      TOTAL MEMORY SIZE : 192K WORDS

>
```

2.3.5.DEFINE-TEST-AREA <lower bank> <lower 4k> <upper bank> <upper 4k>

This command allows definition of the memory area to be tested. It asks for 4 parameters. These parameters define lower and upper limits of the memory to be tested. The resolution of these limits is 4k (10.000 octal). For instance: If bank 2 and 3 both have 64k memory and one wants to test the upper 32k in bank 2 and the lower 32k in bank 3 then define the following values (decimal):

Lower bank:	2
Lower 4k unit:	8
Upper bank:	3
Upper 4k unit:	7

Example:

```
>D-T
DEFINE-TEST-AREA
LOWER    BANK   (0-2 DEC.): 1
LOWER 4K UNIT  (0-15 DEC.): 0
UPPER    BANK   (1-2 DEC.): 1
UPPER 4K UNIT  (0-15 DEC.): 7

>
```

2.3.6.DEFINE-DMA-AREA <lower bank> <upper bank>

Defining the lower and upper bank of memory to be used by the DMA transfers (see commands **START-** and **STOP-DMA-TRANSFERS**). The resolution is 1 bank (64k of memory).

Example:

```
>DEF-DMA
DEFINE-DMA-AREA
LOWER BANK (0-2 DEC.): 0
UPPER BANK (0-2 DEC.): 1

>
```

2.3.7.MODE <yes/no> Æ<value>Å <yes/no> Æ<value>Å <yes/no>

This command defines how the memory test should run and when to abort the test. The defaults for the different parameters are always equivalent to the current setting. The following possibilities exist:

1) Single Test Mode:	If selected, one test is run. The test to be run will be asked for when giving the command **RUN**.
2) Loop Mode :	Allows definition of how long the test(s) should run. If not selected, the test(s) will run once. If selected, a maximum loopcount telling how many times to run the test(s) may be selected. If 0 is given, the test will run infinite.
3) Abort Mode :	If not selected, tests will never be aborted. If selected, the number of error messages to be given before aborting the test, can be defined.
4) Suppress Errors :	If not selected, error messages will be printed. If selected, no error messages are printed except '*** ERROR ***' at certain intervals. May be used to speed up the program during an error condition.

Example:

```
>MODE
MODE
SINGLE TEST  (YES/NO): NO
LOOPMODE     (YES/NO): YES
HOW MANY TIMES (0=INFINITE) (DEC.): 10
ABORTMODE    (YES/NO): Y
AFTER HOW MANY ERRORS (1-65535 DEC.): 4
SUPP. ERRORS (YES/NO): N

>
```

2.4.REFRESH-TEST <no. of minutes delay>

This command allows one to test if the refresh on the memory modules functions correctly. Make sure the other tests run correctly before running this test. The default value for <delay> is 5 minutes.

Example:

```
>REF
REFRESH-TEST
DELAY BETWEEN WRITE AND READ IN MINUTES (DEC.): 1
 0,00:03 == START OF TEST ==
 0,00:04 === END OF TEST ===

>
```

2.5.SKIP-TESTS <yes/no> <yes/no> <yes/no> <yes/no> <yes/no>

With this command, one can define one or more tests that are to be skipped by the **RUN** command. The test(s) is(are) only skipped if the program is not in single test mode (see the **MODE** command). The defaults for the different parameters are always equivalent to the current setting.

Example:

```
>SKIP
SKIP-TESTS
ANSWER "YES" FOR EACH TEST YOU WANT TO SKIP

READ TEST ON  PROGRAM  PART     (YES/NO): Y
ADDRESSES IN ADDRESSES          (YES/NO): N
WRITE/READ TEST (7 PATTRENS)    (YES/NO): N
RAPIDLY CHANGING ADDRESS BITS   (YES/NO): N
PARITY ERROR DETECTION          (YES/NO): Y
WALK TEST (34 PATTERNS)         (YES/NO): N

>
```

2.6.BIG-MPM-SERVICE-PROGRAM

This command allows one to access the BMPM ERROR LOG. There are 5 commands possible. They are explained in the following sections.

2.6.1.CLEAR-ERROR-LOG

This command initializes the ERROR LOG module by reading it (the data read is not displayed). This is neccesary after power up.

Example:

```
>BIG
BIG-MPM-SERVICE-PROGRAM

SERVICE COMMAND: CLEAR
CLEAR-ERROR-LOG

== LOG CLEARED ==

SERVICE COMMAND: EXIT
EXIT

>
```

2.6.2.PORT-STATUS-SCAN

This shows the BMPM configuration concerning crates, banks, ports, lower-limit and upper-limit. Legal values are:

Crate: 0-7;   Bank: x or y;   Port: a, b, c or d;   Limits: 0-77

Example:

```
>BIG
BIG-MPM-SERVICE-PROGRAM

SERVICE COMMAND: PORT
PORT-STATUS-SCAN

SCAN REG. ERR. CRATE BANK PORT LOWER  UPPER (NOTE 3)
140000:    0     0    X    A   000000
100102     0     0    X    A          000002

** END OF SCAN **

SERVICE COMMAND: EXIT
EXIT

>
```

2.6.3.ERROR-LOG-SCAN

This command reads (and therefore resets) the ERROR LOG. It will tell either that no errors have been found or, if there were errors, in which crate, module and IC.

Please note that the errors, forced by the parity error detection test, will be reported to the ERROR LOG. Also, when building up the memory maps during program startup, parity errors are forced in order to detect the memory type, and these errors are also reported the first time we give the command error-log-scan.

Example:

```
>BIG
BIG-MPM-SERVICE-PROGRAM

SERVICE COMMAND: ERROR
ERROR-LOG-SCAN
CRATE MODULE IC-POSITION
  0     00         37
  0     00         37

SERVICE COMMAND: EXIT
EXIT

>
```

2.6.4.MODULE-TO-SLOT-CONVERSION

Gives the correspondence between "module" (as given by the ERROR LOG SCAN) and slot-position in the BMPM crate.

Example:

```
>BIG
BIG-MPM-SERVICE-PROGRAM

SERVICE COMMAND: MOD
MODULE-TO-SLOT-CONVERSION
    MODULE: 0  1  2  3  4  5  6  7  10  11  12  13
CRATE SLOT: 1  2  3  4  5  6  7  8  32  31  30  29

SERVICE COMMAND: EXIT
EXIT

>
```

2.6.5.EXIT

Leave the service-program

2.7.Error Messages

In general the error messages are printed in the form of tables because it is likely that a certain error is followed by many errors of the same type.
To speed up printout, this type of message will start by printing an error message header followed by lines of Error Data.
The header will be repeated at regular intervals.
The very first error message in an error sequence will be preceded by the text
'*** ERROR ***   TEST NR: <test no>'.

Error message printout can be suppressed by selecting the correct mode with the command **MODE**. If Suppress Error Message mode is selected then only the text '*** ERROR   TEST NR: <test no>' will be printed for every 65536 error messages (within one testroutine).
Each error message will cause a counter to be incremented and it is this counter that is used to find out if the test currently running should be aborted (in case Abort Mode is selected). Parity errors will also increment this counter.

2.7.1.Parity Errors

Parity errors might occur anywhere during the run of the testprogram. They are reported by a routine that runs on level 14.

Example:

```
>RUN

READ TEST ON PROGRAM PART     === END OF TEST ===
ADDRESSES IN ADDRESSES

*** MEMORY PARITY ERROR *** TIME: 0,00:05
***    ADDRESS (PEA): 000000  BANK  (DEC.): 1
***     STATUS (PES): 035001
*** PES-REG DECODED:
***    BIT 13 : FATAL ERROR. MULTIPLE ERROR OCCURRED
***    ERROR CORRECTION CODE (BITS 8-12 OF PES)(OCT.): 00
*** ( PREVIOUS LEVEL (DEC.): 0   ADDRESS: 021525 )
```

Decoding of the PES registers is done with respect to the computer type, since the format of the PES register is different for each computer type.
If the parity error occurs during an error message printout, the test currently running is aborted by doing a call to the the escape handling routine.

2.7.2.Errors during Test 1: Read test on program part

No error messages are printed by this test though parity errors may occur.

2.7.3.Errors during Test 2: Adresses in adresses

Example:

```
*** E R R O R *** IN ADDRESSES IN ADDRESSES
  TIME    BANK  ADDRESS  EXPECTED  FOUND (TEST 2)
 0,00:06    1   170060   170060    000001
 0,00:06    1   170061   170061    000002
 0,00:06    1   170062   170062    000003
 0,00:06    1   170063   170063    000004
*** ROUTINE ABORTED ***

>
```

2.7.4.Errors during Test 3: Write/read test patterns

The format of the error message given by this test is the same as the one in test 2.

Example:

```
*** E R R O R ***  IN WRITE/READ TEST (7 PATTERNS)
  TIME    BANK  ADDRESS  EXPECTED  FOUND (TEST 3)
 0,00:07   1    040200   125252    125242
 0,00:07   1    040202   125252    125242
 0,00:07   1    040206   125252    125242
 0,00:07   1    040210   125252    125242
*** ROUTINE ABORTED ***

>
```

2.7.5.Errors during Test 4: Rapidly changing address bits

Example:

```
*** E R R O R ***  IN RAPIDLY CHANGING ADDRESS BITS

                 LOWER    UPPERR
  TIME   BANK   ADDRESS  ADDRESS  CHECKSUM: FIRST  SECOND  (TEST 4) (NOTE 2)
 0,00:10   1    040000   047777             161140 161370
 0,00:10   1    050000   057777             160550 161010
 0,00:10   1    060000   067777             160560 161020
 0,00:10   1    070000   077777             160560 160570
*** ROUTINE ABORTED ***

>
```

2.7.6.Errors during Test 5: Parity error detection

This test has three different error message types. The first type is used when the forced parity does not cause interrupt on reading back memory.

Example:

```
*** E R R O R ***  IN PARITY  ERROR  DETECTION
DID NOT GET PARITY ERROR INTERRUPT (TEST 5)
  TIME    BANK  ADDRESS
 0,00,12    1   000000
 0,00,12    1   000001
 0,00,12    1   000002
 0,00,12    1   000003
*** ROUTINE ABORTED *

>
```

The second error message type is used when we have parity error interrupt but the data read back is not as expected.

Example:

```
*** E R R O R ***  IN PARITY  ERROR  DETECTION
   TIME   BANK  ADDRESS  EXPECTED  FOUND (TEST 5)
 0,00,15    0   050000   000000    000001
 0,00,15    0   050001   000000    000001
 0,00,15    0   050002   000000    000001
 0,00,15    0   050003   000000    000001
*** ROUTINE ABORTED ***

>
```

The third error message type is used when, after parity error interrupt, the contents of the PES and/or PEA register is not as expected. The format of this message is:

Example:

```
*** E R R O R ***  IN PARITY  ERROR  DETECTION
WRONG BANK/ADDRESS FOUND IN PES/PEA AFTER PAR.ERR.INTERRUPT (TEST 5)
  TIME  EXP. BANK/ADDRESS  FOUND BANK/ADDRESS  (PES-REG)
 0,00:24        1   050000           0     025500    037400
 0,00:24        1   050001           0     025500    037400
 0,00:24        1   050002           0     025500    037400
 0,00:24        1   050003           0     025500    037400
*** ROUTINE ABORTED ***

>
```

2.7.7.Errors during Test 6: Walk Test

Since the Walk Test is basically a read/write test, the error message format is the same as in tests 2 and 3.

Example:

```
*** E R R O R ***  IN WALK TEST (34 PATTERNS)
  TIME    BANK  ADDRESS  EXPECTED  FOUND (TEST 6)
 0,00:21   1    040201   000010    000000
 0,00:21   1    050200   000010    000000
 0,00:21   1    060200   000010    000000
 0,00:21   1    070200   000010    000000
*** ROUTINE ABORTED ***

>
```

CHAPTER THREE

MOVER

TABLE OF CONTENTS

3.MOVER

3.1.General

This is a memory testprogram that has the capability of moving itself to a new place in memory, each time a test cycle is completed. This has two advantages above other memory test programs:

1. The program "seems" not to occupy any memory, thereby allowing to test the memory where the program was previously located.

2. Program instructions are fetched from a different place in memory after each move.

However, on the ND-100 only a maximum of 64K of memory in bank 0 can be tested whereas on the NORD-10 it is possible to select the BANK the program will run in, thereby allowing to test 256K of memory (by running the program 4 times).

The following test patterns are used:

000000, 177777, 125252, 052525, 016343, 161434, 000776

3.2.Stand alone/SINTRAN

The program can run as a stand-alone program or under SINTRAN. However one should realise that when running under SINTRAN, this program is not so much a memory test as a "system" test, testing parts of SINTRAN, disc transfers etc. Also the address in error messages is a virtual-memory address and has no direct correspondence with a physical-memory address.

3.3.Parameters

After startup, the program will immediately start by asking if you know this program, a question that should be answered by "Y" or "N" without <CR>. Answering "Y" will cause an explanatory text to be typed. Then the following parameters are asked for (any "non octal" character terminates a parameter):

FLYTT: This is the number of memory locations that the program will move itself after it has completed one test cycle. If FLYTT is positive the program will move upwards, if negative it will move downwards and if zero, the program will stay at one place.
Legal values for FLYTT are: 0 and (+ or -) 1, 2, 4, 10, 20, 40, 100, 200 and 400 (octal).

ADDRN: Lower address of the memory area to be tested. (0, 20000 etc.)

ADDRX: Upper address of the memory area to be tested. (017777, 77777, 177777 etc.)

PROGN: This is the address where the program should initially be placed in memory before starting the tests. PROGN can be either 0 or >=400 but must be within the tested memory area.

ADDRN, ADDRX+1 and PROGN must all be divisible by FLYTT exept when FLYTT is 0.

Finally the program will ask where to print error messages. Under SINTRAN a file name is expected as "error message device", when running stand alone a hardware device number is expected (304, 314, 410, 430 etc.)

3.4. Example of Execution under SINTRAN

User input is underlined.

```
@MOVER


MOVING MEMORY TEST PROGRAM

HAR-1863C    APRIL 29, 1977

MAXIMAL MEMORY ADDRESS: 177777

DO YOU KNOW THIS PROGRAM (Y OR N) ? Y
FLYTT: 400
ADDRN: 0
ADDRX: 177777
PROGN (ADDRS. FROM 1 TO 002735 ARE ILLEGAL): 0
FILE NAME FOR ERROR MESSAGES: TERMINAL

ERROR MESSAGES WILL APPEAR AS 4 NUMBERS:

PROGN   FAILING ADDRESS   EXPECTED CONTENTS   WRONG CONTENTS

043000 127636 052525 050525
102400 125777 052525 161434
102400 125776 125252 016434
102400 125775 177777 052525

      and so on.....
```

CHAPTER FOUR

LINE PRINTER TEST PROGRAM

TABLE OF CONTENTS

4.LINE PRINTER TEST PROGRAM

4.1.GENERAL

This program has various tests for line-printers. Most of the tests are visual; The program sends different patterns or control characters that have to be checked by the user. All commands use the device number/file name that is specified when entering the program. To change this device number/file name the monitor command **SET-PRINTER-DEVICE-NUMBER** must be used.

NOTE: The time-out used when writing one character is 20 sec. This is because some slow printers have big buffers, and might use that time to get ready.

All test are stopped by pressing ESCAPE on the console.

4.2.INITIAL ACTION

After loading, the program asks for the printer device number. The number is checked to see if it is a likely number for printers and that it exists. If not it will ask again. When running under SINTRAN it asks for the file name.

4.3.DESCRIPTION OF THE COMMANDS

4.3.1.COLUMN-TEST <character> <column from (to)>

This command prints the same character in one or several specified columns. The default character is "E", but any character can be specified. The command asks for column (FROM,<to>) and when two numbers are given, seperated with comma, the program will print the specified character between these two columns. The first number has to be smaller than the last. If only one number is given, the character will only be printed in that column. The default answer will be all columns.

Example:

```
>COLUMN
COLUMN-TEST
CHARACTER: CR E CR
COLUMN (FROM,<TO>) (1-132 DEC.): 40,50
```

4.3.2.FORMAT-TEST <channel>

On some printers it is possible to make the line-printer skip to certain lines on the page by giving different control characters. These positions are refered to as channels. In addition it is possible to do Form-Feed (FF), Carriage-Return (CR) and Line-Feed (LF).
This command tests the form- and channel-feed on printers with such option. First a form-feed is done and the sentence "*****THIS IS TOP OF FORM" is written on the first line of the page. Then it skips to the specified channel and writes "******THIS IS CHANNEL NN". The command will then ask for a new channel until EXIT is answered.

Example:

```
>FORMAT
FORMAT-TEST
CHANNEL: 5
CHANNEL-5           % Printer does a Form-feed then skips to Channel 5.
CHANNEl: EXIT
EXIT
>
```

4.3.3.INTERFACE-TEST <interface type>

As interface type one of these can be chosen:

1) CDC-LP-INTERF.-1047
2) TERMINAL-BUFFER-1095
3) PARALLEL-BYTE-1109
4) PARALLEL-1130-1140
5) DUAL-ASYNC-MODEM-1147
6) FLOPPY-4TERM-3010
7) EIGHT-TERMINAL-3013

The program tests for illegal characters on interfaces that should detect those, by sending out all illegal characters and checking that the error bit is set on the interface.
Then it runs through a dataway test by writing all legal characters to the interface in testmode, and reads them back again. The command also checks the ident code for the device number.
This command is not executable under SINTRAN

Example:

```
>INTERFACE
INTERFACE-TEST
INTERFACE: CDC
CDC-LP-INTERF.-1047

ILLEGAL CHARACTER TESTED.
INTERFACE TESTED WITH ALL CHAR.
TEST ARE LOOPING.
```

4.3.4.MODE <line lenght> <small letters> <skip pattern> <loop mode>

This command sets/resets different parameters used by the other commands. The command **PROGRAM-STATUS** will present the state of these parameters.

- LINE LENGTH (1-136 DEC.):
  Sets the maximum line lenght for the printer. Default is 132.

- SMALL LETTERS (YES/NO):
  Sets/resets if the patterns in **RUN-TEST-PATTERN** should include small letters (96 characters). The default value is NO.

- SKIP PATTERN (1-5 DEC.):
  It is possible to skip some (or all) patterns used by the command **RUN-TEST-PATTERN.** Answer with the numbers of the patterns on one line seperated by comma or space, terminate with CR.
  Default answer is NONE SKIPPED, e.g. Reset.

- LOOPMODE (YES/NO):
  Initially the program is in loopmode and runs infinite. When answering YES it will ask
  HOW MANY TIMES (DEC.):
  and the input decides the number of loops the test should run.
  Default is INFINITE.

4.3.5.RUN-TEST-PATTERNS

This command runs 5 different test-patterns on the Line-printer. Each pattern starts with a form-feed and then the pattern number.

TEST-PATTERN 1:	All characters are printed, then shifted one position to the left untill all characters are printed in all positions.
TEST-PATTERN 2:	On the first line the maximum number of characters will be printed, on the second line only the first character. The pattern will then alternate between decreasing the maximum line lenght and increasing the minimum line lenght by one for each line printed.
TEST-PATTERN 3:	All characters are printed, filling up one line each.
TEST-PATTERN 4:	The letter "M" is printed in every other column (1,3,5,...), seperated by spaces.
TEST-PATTERN 5:	The letter "M" is printed in every other column (2,4,6...), seperated by spaces.

Example:

```
>RUN
RUN-TEST-PATTERNS


>
```

4.3.6.STRESS-BUSY-TEST

Missing characters on line-printers could be because the printer sends the busy signal too late, and cannot accept characters already sent from the CPU. This command tries to make the printer go busy by printing only one number from 0-9 on each line, and then doing a Form-feed.

Example:

```
>STRESS
STRESS-BUSY-TEST
```

Correct output on the printer should then be like this:

```
STRESS-BUSY-TEST
0
1
2
3
4
5
6
7
8
9
```

4.3.7.PROGRAM-STATUS

This is a Monitor command but in addition to variables presented by the Monitor, it will also present variables set by the **MODE** command.

Example:

```
>PROGRAM-ST
PROGRAM-STATUS

-------------------------------------------------------------
*   DAY,    * CONSOLE* PRINTER*  STOP ON *  INTER- *  PAGING *
* HOUR:MIN  * DEV.NO.* DEV.NO.* FULL PAGE*   RUPT  *         *
-------------------------------------------------------------
*  7,12:40  *    1   *    1   *    OFF   *  SINTRAN RUNNING  *
-------------------------------------------------------------


-------------------------------------------------------------
* LINE LENGHT *    CHARACTER SET   *       PATTERNS USED      *
-------------------------------------------------------------
*     132     * 64 - CAPITALS ONLY *  1 *  2 *  3 *  4 *  5 *
-------------------------------------------------------------

>
```

CHAPTER FIVE

TEST PROGRAM FOR TERMINALS AND ASYNCHRONOUS DEVICES

T A B L E   O F   C O N T E N T S

5.TEST PROGRAM FOR TERMINALS AND ASYNCHRONOUS DEVICES

5.1.GENERAL

This program tests asynchronous interfaces and does some tests to check terminal devices both on input and output. In all tests the program will continuously check the status word, parity, interrupt and ident codes.

In addition there are commands to set- and find-speed (both on interface and terminal), and one command (**TRANSLATE**) that gives cross-reference between different types of terminal numbers (device, logical, ident-code).
One should remember that after hardware changes (switches, cables etc.), it is important to do a Master Clear and Restart (20!) of the program. After changing of cards etc. the program should be reloaded.

5.2.INITIAL ACTION

When the program is loaded different tables will be allocated and set to zero. No parameters are asked for before the command processor is entered. At restart only the error table and error descriptors are reset.
The defined device table will still be valid.

5.3.THE DEVICE TABLE

Tests can be run on one device or on several devices simultanously. In order to test several devices at the time, a device table has to be defined. (see command **DEFINE-DEVICE-TABLE**, 5.4.1.).

The device table is used by the following commands:
KEYBOARD-TEST
When a device table has been defined, these commands will start by asking:

USE DEVICE TABLE (YES/NO): (default answer is YES).

If YES, the commands will run through the test on all devices in the table. The command **SET-SPEED** will ask for one speed only and set the same speed on all devices.
If NO, the commands will ask for a device number instead. Default is console device number.

It is possible to define a device table with only one device in it.

5.4.DESCRIPTION OF THE COMMANDS

5.4.1.DEFINE-DEVICE-TABLE <yes/no for each present device>

This command finds out which devices are present and asks if tests should be run on a device. If YES, the device number is entered into a table which might be used in tests. Default answer is "NO". The command does no testing on the interfaces. The device table is presented by the command **PROGRAM-STATUS** (see 5.4.13.).
When executed under SINTRAN this command will write
TERMINAL RESERVED FOR TEST-PROGRAM
on each terminal in the table. When the program is left (EXIT), the text
TERMINAL IS RELEASED
will be written on the terminals.

Example:
```
>DEFINE-DEVICE-TABLE
DEFINE-DEVICE-TABLE

TEST FOLLOWING DEVICE NUMBERS ?
  300  (YES/NO): CR NO CR
  340  (YES/NO): Y
  350  (YES/NO): Y
  360  (YES/NO): Y
  370  (YES/NO): Y
>
```

5.4.2.DISPLAY-TEST <device no.>

This command runs 5 different test-patterns on the screen:

Pattern 1: All characters in all positions.

Pattern 2: Alternating Max. and Min. lines (decreasing/increasing).

Pattern 3: All characters. One line each.

Pattern 4: Alternating "M" and spaces.

Pattern 5: Alternating "U" and "*".
Patterns 4 and 5 hold for about 5 sec.

Example:

```
>DISPL
DISPLAY-TEST
TERMINAL DEV. NO. (OCT.): CR 300 CR

ABCDEFGHIJKLMNOPQRSTUVWXY.......
```

5.4.3.FILL-SCREEN <device no.> <character>

This command sends one character continously. Any character (also control characters) can be specified. "H" is default character.

Example:

```
>FILL
FILL-SCREEN
TERMINAL DEV. NO. (OCT.): 340
CHARACTER: A AAAAAAAAAAAAAAAA..............
```

5.4.4.FIND-SPEED <device no.>

This command finds the speed on the interface of the specified device. It is done by writing characters in testmode for 5 seconds while counting how many characters are written, thus calculating the speed. This command is not executable under SINTRAN.

Example:

```
>FIND
FIND-SPEED
TERMINAL DEV. NO (OCT.): 340

SPEED IS 9600 BPS.

>
```

5.4.5.INTERFACE-TEST <interface type> <all devices (yes/no)> <device no.>

The command tests the following interfaces:

1) ASYNC-MODEM-1046
2) TERM-BUFFER-1095
3) FOUR-ASYNC-1122
4) DUAL-ASYNC-1147
5) FOUR-TERMINAL-3010
6) EIGHT-TERMINAL-3013

On interfaces with more than one device the program asks if all devices on the card should be tested (default answer is YES). If NO is answered it asks for device number and tests that. Otherwise it asks for the FIRST device number on the card, and checks that the other device numbers are present.

For an 8-terminal card the first device number of the second group must also be specified.

Then the program asks if tests are to be run at all speeds on the interface. This will take approximately 1,5 minutes/device number.

The test writes the following patterns to the interface in test-mode:

(000,177,152,125,000,125,152,177,000)

and checks for correct status-word, ident-code and that data is correctly transmitted. When testing at all speeds, the speed is checked before running the pattern on each speed.

This command is not executable under SINTRAN.

Example:
```
>INTERFACE
INTERFACE-TEST FOUR-TERM
INTERFACE TYPE:
FOUR-TERMINAL-3010

TEST ALL DEVICES ON CARD (YES/NO): CR YES CR

FIRST DEVICE NUMBER (OCT.): 340

TEST WITH ALL SPEEDS (YES/NO): YES

TESTING DEVICE NO.:
(TAKES 1.5 MIN PR DEV.NO.)     340  350  360  370
TESTS ARE LOOPING
```

5.4.6.KEYBOARD-TEST <device no.> <hard copy terminal (y/n)>

After displaying all characters, the test asks for input from the keyboard. All characters are typed as they appear then checked to see if they are correct (with even parity). If a wrong character is typed it is not necessary to start all over.

Note that special characters will vary from terminal to terminal. But the character on the screen should of course correspond to the character on the keyboard.
First the program displayes capital letters and asks for them to be typed, then it displayes the same string, but asks you to type them as Control characters. Next it displayes a string of special characters (NOTE: This string starts with a space !). Finally the test asks for lower case letters, if YES, they should also be typed.
The test may be stopped by pressing the ESC key.

If the device table has been defined, the program prints IF HERE TYPE ANY CHAR on each device and runs the test on the keyboard that is first touched. Then it waits for the next keyboard to be typed at. The console device will list the devices that are tested.
When run under SINTRAN it is only possible to test the "console-device" keyboard, i.e., the terminal on which the program is run.

Example:

```
>KEYBOARD
KEYBOARD-TEST

DO YOU HAVE HARDCOPY TERMINAL (Y/N) ?NNO

TYPE THE FOLLOWING STRING, UPPER-CASE CHARACTERS.

ABCDEFGHIJKLMNOPQRSTUVWXYZÆØÅ
ABCDE...........
```

5.4.7.LIST-TERMINAL-DEVICES

This command finds all terminal devices present in the machine and presents them in a table. The ones that have probably no terminal connected are masked with a hyphen (-). To verify if there is a terminal connected or not the program checks for framing error (bit 5 in status-word). This might occur even if a terminal is connected, and the best way to reset it is by doing a Master Clear and then a Restart (20 !) on the machine.

Under Sintran all available (not reserved by others) logical devices are presented.

Example:

```
>LIST-TERMINAL
LIST-TERMINAL-DEVICES

-------------------------------------------------------------
*                  TERMINAL DEVICES PRESENT                 *
-------------------------------------------------------------
*   300  *   340  *   350  *   360- *   370- *   1300  *
-------------------------------------------------------------
*  1310  *  1320- *  1330- *        *        *         *
-------------------------------------------------------------
- MEANS PROBABLY NO TERMINAL CONNECTED.
>
```

5.4.8.LOOP-TEST <device no.>

This command writes the following patterns to the interface:

(000,177,152,125,000,125,152,177,000)

and then expects to read back the same as an input to the interface.

This means that a dummy plug that connects output and input on the interface, or a modem in local loop-test has to be used.

Example:

```
>LOOP
LOOP-TEST

TERMINAL DEVICE NO.:  340
```

5.4.9.MODE <loopmode> <abortmode> <suppress errors>

Sets different modes used by other commands:

1. Loopmode	- Tests will loop given times. - Initially in loopmode (infinite).
2. Abortmode	- Tests are aborted after a given number of errors. - Initially not aborted.
3. Suppress errors	- Tests are run without giving error-messages. - Initially not suppressed.

Even if errors are supressed, the number of errors in the last test will be recorded and presented by **PROGRAM-STATUS**.

Default will always be the previous setting.
The command **PROGRAM-STATUS** (see 5.4.13.)will present the setting of the modes.

Example:

```
>MODE
MODE
LOOPMODE            (YES/NO): Y
HOW MANY TIMES        (DEC.): CR INFINITE CR
ABORTMODE           (YES/NO): Y
AFTER HOW MANY ERRORS (DEC.): CR 10 CR
SUPRESS ERRORS      (YES/NO): NO

>
```

5.4.10.SET-SPEED <device no.> <speed>

This command sets (by software) the speed on the interface of the specified device, and then does a "Find-speed" to check it. This takes 5 seconds.

If the interface has more than one device number, the speed should be set by software on all devices to avoid problems on the interface.

Default answer on speed is 9600 BPS.
This command is not executable under SINTRAN.

Example:

```
>SET-SPEED
SET-SPEED
TERMINAL DEV. NO. (OCT.):  340

SPEED: 4800
4800

TAKES 5 SEC. TO CHECK SPEED.
SPEED SET TO 4800 BPS.
>
```

5.4.11.TRANSLATE <input type> <number>

This command gives a cross-reference between terminal no., device no., ident-code and logical no. Any of them can be input. Default is a list of all numbers. This command gives output to the printer device.

Example:

```
>TRANS
TRANSLATE

INPUT TYPE: LOGIC
LOGICAL-DEV-NO

GIVE LOGICAL-DEV-NO (DEC.): 39

---------------------------------------------------------------------
!   TERM.NO.   !  DEVICE   !  IDENT-   ! LOGICAL DEV.NO.!BACK-   !
!  OCT.  DEC.  !    NO.    !   CODE    !   OCT.     DEC. !GROUND  !
---------------------------------------------------------------------
!    10    8   !    370    !     47    !    47       39  ! BAK08  !
---------------------------------------------------------------------

INPUT TYPE: EXIT
EXIT

>
```

5.4.12.VERIFY-TERMINAL-SPEED <device no.>

This command finds the speed of the terminal sets the interface to the same speed.

To find the speed one has to press ESC a few times on the specified terminal. The speed is then written out on both the specified terminal and the console device.
It is not possible to find the speed of the console device.

Some speeds (7200, 3600, 900) can be measured, but it is not possible to set the speed on the interfaces. The message will only appear on the console device.

Example:
```
>VERI
VERIFY-TERMINAL-SPEED

TERMINAL DEV. NO. (OCT.): 340

PRESS ESC. ON TERMINAL 340 TO FIND TERMINAL SPEED.
SPEED ON THE TERMINAL IS 4800 BPS.
>
```

5.4.13.PROGRAM-STATUS

This is a monitor command but in addition to variables presented by the monitor it will also present variables set by different program commands **DEFINE-DEVICE-TABLE** (5.4.1.) and **MODE** (5.4.9.). Note that the terminal device numbers are the ones defined in 5.4.1. and not all devices present in the machine.

Example:
```
>PROGRAM-ST
PROGRAM-STATUS

-------------------------------------------------------------
*   DAY,     * CONSOLE* PRINTER*  STOP ON *  INTER- *  PAGING *
* HOUR:MIN  * DEV.NO.* DEV.NO.* FULL PAGE*   RUPT   *         *
-------------------------------------------------------------
* 26,10:24  *   300   *   300   *    OFF    *    ON    *   OFF   *
-------------------------------------------------------------
*                    DEFINED DEVICES TABLE                   *
-------------------------------------------------------------
*    340   *    350   *    360   *    370   *          *         *
-------------------------------------------------------------
LOOPMODE : INFINITE
ABORTMODE: AFTER 10 ERROR(S)
ERRORS IN LAST TEST: 248

>
```

5.5.ERROR MESSAGES

Error messages always appear with this heading:

```
*** ERROR ***  DEVICE NO.:  XXXX     TIME:   dd,hh:mm
```

An error will only be reported once. If there is a change in status or other errors occur it will cause a new error message. After an error message the last read status will be printed and decoded. Some of the error messages may have a number added to it "n)". The number refers to a note (see **PRINT-NOTE** section 1.14.10) that gives further information.

Example:

```
*** ERROR ***  DEVICE NO.:  350      TIME:    10,15:30
WRONG IDENT CODE. FOUND: 0. EXPECTED: 44           3)    LEVEL:   12
STATUS: 004041
   DEVICE NOT READY.
   FRAMING ERROR.                                  4)
   CARRIER MISSING.
```

CHAPTER SIX

HDLC TEST PROGRAM

TABLE OF CONTENTS

6. HDLC TESTPROGRAM

6.1.GENERAL

This program is meant to be a verification and debugging program for (DMA) HDLC interfaces. The program is a stand alone version and contains three seperate tests and a number of commands to specify parameters when one wishes to change default values.

All commands are given in "SINTRAN III" format, i.e., a string (abbreviated) specifying the command, followed by parameters (if any). If a parameter is not specified at once, a prompt line is written. All parameters have default values, used if CR is given. The prompt sign is *.

Editing facilities are CNTR-A to delete the last character, CNTR-Q to delete the entire line.

6.2.INITIAL ACTION

When the program starts it checks how much memory there is (up to 256k), if there is a disc connected and puts its ident-code in a table.
Initially the device number is set to 1640.

6.3.NORMAL PROCEDURE TO TEST HDLC

6.3.1.VERIFICATION OF DEVICE NUMBER AND IDENT CODE.

Run NORD-CONFIGURATION-INVESTIGATOR.

6.3.2.VERIFICATION OF A HDLC INSTALLATION.

The tests listed below should in most cases be sufficient to verify the hardware on a HDLC/MEGALINK interface.

```
 1) *CHANGE-DEVICE         % Specify dev.no to be tested and mode.
 2) *BASIC-IO-TEST         % Tests the IOX`s (1181 card)
 3) *BASIC-DMA-TEST        % Tests the DMA channel and "HDLC CPU"
 4) *FIND-SPEED            % Prints the speed
 5) *LOOP-TEST             % Run data in specified mode
```

If two HDLC/MEGALINK is in the same machine and link cable connecting them, then this test should be added:

```
 - *SPECIFY-LOOP-CONFIGURATION   % Specify the configuration
                                       to be used in

 - *EXTENDED-LOOP-TEST
```

All tests given above should first be run in maint.mode.

6.3.3.HDLC-2 TEST-AREA ILLUSTRATION

```
             --------------------------------
DMA        ! HDLC DMA CONTROL/SHADOW        !
<=======>!  *BASIC-DMA-TEST                 !<--! 1181/1151 connection
         !                    *LOOP-TEST    !   !
         !                                  !   !
         !....................              !   !
IOX/IDENT! HDLC DATA          *FIND-SPEED   !<--!
<=======>!  *BASIC-IO-TEST                  !              !-! !-!
         !                       line       !<------------->! ! ! !=
         !                       circuitry!                 !-! !-!
             --------------------------------
                                 !                              !
<--- All tests in maint.mode -->!                               !
                                                                !
<--- All tests not in maint.mode and "dummy plug" ------------>!

<--- All tests not in maint.mode without dummy plug --------------->
     Data and clocks have to be returned from either
         - a modem in local loop test
           or
         - a link(modem or computer link) connected
           to a remote CPU w/HDLC echoing the signals.
```

6.3.4.HDLC-2 TEST-SETUPS.

```
 ---------------
!               !
! HDLC DMA      !
!               !
!               !<------------> No external equipment connected
 ---------------

*LOOP-TEST in maint.mode                  % Press any char.for status
```

```
 ---------------
!               !
! HDLC DMA      !
!               !
!               !<------------> Dummy plug or modem in local loop test.
 ---------------

*LOOP-TEST not in maint.mode              % Press any char.for status
```

6.3.5.TWO HDLC/MEGALINK IN THE SAME MACHINE.

```
 ---------------                   Plug panel.
!               !                 /
! HDLC DMA      !                /
! ex.dev no.1640! intern cable !!
!               !<------------>!!---
 ---------------               !!   !
                                    !   Computerlink cabel or
                                    !   synch.modem connection
 ---------------                    !
!               !                   !
! HDLC DMA      !                   !
! ex.dev-no.1660!              !!   !
!               !<------------>!!---
 ---------------               !!
```

```
*SPECIFY-LOOP-CONFIGRATION          % This is a typical setup
GENERATOR OUT:1640                  % for running the
GENERATOR IN:1640                   % EXT-LOO-TEST where the two
MAINT(Y/N):N                        % interfaces has dev.numbers
ECHO IN:1660                        % 1640 and 1660
ECHO OUT:1660                       %
*EXTENDED-LOOP-TEST                 % Press any char.for status
```

6.4.DESCRIPTIONS OF THE COMMANDS

6.4.1.BASIC-DMA-TEST

This test checks the DMA-channel on the interface of the computer memory. During the first part of the test the "INIT" function is used with all possible variations of the parameter values (PCR, SAR, CHLR, D1, D2, MAXBL). The result is read back with the "DUMP" function and tested for errors.

During the second part of the test the interface memory (register 40-377) is loaded from the computer memory ("LOAD") and read back again ("DUMP"). The "LOAD"-address starts at the lower end of physical memory and is moved 340 octal words upwards for each transfer. The "DUMP" address starts at the upper end of the memory and is moved downwards. When all memory has been tested, the test is repeated with a different data pattern.

The data patterns used are:

1) All zero.
2) "Address in address" (16 bit).
3) 125252
4) 077176
5) 077577
6) 177777

If any difference is found between the data written ("LOAD") to the interface, and the data read back ("DUMP"), the data is read back once more to a fixed buffer in lower memory. Error printout takes the following form:

```
xx DMA ERROR FROM yyy/ddd >< to zzz/ppp (qqq)

  xx  = Word index in block (0 <= xx < 377).
  yyy = Address from which word was transfered TO interface.
  ddd = Contents of address yyy.
  zzz = Address to which data was transfered FROM interface.
  ppp = Contents of address zzz.
  qqq = Contents of word no. xx in fixed buffer after extra "DUMP".
```

Example:

```
*BASIC-DMA-TEST

                    (Takes approximately 1 minute,
                    depending on memory size.)
*
```

6.4.2.BASIC-IO-TEST

This is an elementary test to check for IOX-errors. Checks READY-FOR-SENDING and DATA-SET-READY. The test also checks ident-codes and interrupt levels of the interface.

Example:

```
*BASIC-IO-TEST
                     (Takes 2 sec.)
*
```

6.4.3.CHANGE-DEVICE

This command selects the interface on which most of the tests are run. Initially the default value will be 1640, later the default will be the same as the one last given.

This command also asks if maintenance mode is wanted or not. In maintenance mode the output of the interface is looped internally back to the input. Default is Y (maint. mode).

Example:

```
*CHANGE-DEVICE
DEV.NO: 1660
MAINT.MODE (Y/N): Y

*
```

6.4.4.CHANGE-TEST-MODE

This command controls the test sequence and error printout mode. The example below should be self explanatory.

Some of the questions will be omitted, depending on the answer given to a previous question. For example, if the answer to "SUPPRESS ERROR PRINT" was "NO", the question "RING BELL ON ERROR" is not asked.

Example:

```
*CHANGE-TEST-MODE
LOOP IN TEST (Y/N) : Y
REPEAT COUNT:                          (0=forever)
SUPPRESS ERROR PRINTOUT (Y/N):
SHORT ERROR PRINT(Y/N):
RING BEL ON ERROR (Y/N) :

*
```

6.4.5.DATA-PATTERNS

Frames may have different data contents, as specified by this command. Pattern specifications may be repeated, but no more than 7 can be specified. Give parameters on the same line.

Example:

```
*DATA-PATTERN
PATTERN CODE:
ZERO-BYTES: 0
377  BYTES: 1
252  BYTES: 2
INCREASING: 3
DECREASING: 4
177  BYTES: 5
176  BYTES: 6
SELECT (END WITH CR) : 0,1,0,2,4,5,3

*
```

6.4.6.DISC-OFF

Turns off running the disc interface in testmode.

Example:

```
*DISC-OFF

*
```

6.4.7.DISC-TEST-ON

During the **LOOP-TEST** and **EXTENDED-LOOP-TEST** it is possible also to run the disc interface in testmode to load the bus.

Example:

```
*DISC-TEST-ON
TURN ALL DISC UNITS OFF LINE

*
```

6.4.8.EXIT

This command just stops the program.

Example:

```
*EXIT

#
```

6.4.9.EXTENDED-LOOP-TEST

This command does asymmetric loop testing depending on how the command **SPECIFY-LOOP-CONFIGURATION** is set up. It is possible to specify one device to generate frames and one device to receive and check them (might also be same device). In addition it is possible to specify device(s) to echo the frames.

This test will continue infinite and it is possible at any time to have a status report by pressing any key on the keyboard. Status report is the same as described under **LOOP-TEST**.

Example:
```
*EXTENDED-LOOP-TEST
```

6.4.10.FIND-SPEED

Finds the rate of the transmit clock. This test will transmit bytes for one second and compute the clock rate from the number of bytes transmitted. (If speed is 307,20 kbits/s or higher the program will say 307,20 kbits/s).

If the interface is operated in maintenance mode (internal looping), the data rate is equal to the internal clock rate. If the interface is not in maintenance mode, the data rate is equal to the modem clock rate, or to the internal clock rate if a CPU-CPU cable is connected to the interface.
If the speed is 307,20 kbits/s or higher, the program will still say

Example: `*FIND-SPEED`

```
CLOCK RATE : 76.81 KBITS/S

*
```

6.4.11.FRAME-SIZE

The size of the frames to be transmitted during the **LOOP-TEST** and the **EXTENDED-LOOP-TEST** may be changed by this command. Frame no. 1 will be generated with the first frame size specified, frame number 2 with the second, and so on. If you only want one fixed frame size, specify only one.

It is possible to specify up to 27 different frame sizes from 2 to 1024 bytes.

Example;
```
*FRAME-SIZE
FRAME SIZE (IN BYTES) OR CR: 2,13,200,400,1,3,5CR
SIZE OR CR: CR

*
```

6.4.12.HDLC-NOISE

This command makes it possible to run another HDLC (than the one being tested) in maintenance mode, in order to add more load to the bus while doing other tests.

Example:

```
*HDLC-NOISE
HDLC-DEVNO: 1660

*
```

6.4.13.HELP

List all commands in the program.

Example:

```
*HELP
COMMANDS?CR
BASIC-IO-TEST
BASIC-DMA-TEST
LOOP-TEST
EXTENDED-LOOP-TEST
CHANGE-TEST-MODE
RUN
FIND-SPEED
FRAME-SIZE
DATA-PATTERNS
HDLC-NOISE
CHANGE-DEVICE
SELECT-PROTOCOL
SET-BLOCK-PARAMETERS
SET-MEMORY-LIMITS
SPECIFY-LOOP-CONFIGURATION
HELP
DISC-TEST-ON
DISC-OFF
EXIT

*
```

6.4.14.LOOP-TEST

During the loop test, frames of different size and data-patterns are transmitted. Received frames (if any) are checked for corruptions. Looping depends on the mode given in the last **CHANGE-DEVICE** command. If maintenance mode is not used, looping must be provided by:

- cable looping
- modem looping
- external echo device (HDLC-interface in another machine).

All frames transmitted have the following format:

FLAG, ADDRS-BYTE, SEQUENCE-BYTE, DATA BYTES, FCS1, FCS2, FLAG

ADDRS-BYTE	= Constant value, initially zero, changed by **SET-PROTOCOL** command
SEQUENCE-BYTE	= Frame sequence counter from 0-255.
DATA	= N data bytes (0 <= N <= 1022).
FCS1 FCS2	= Frame check sequence (CRC).

The **LOOP-TEST** will continue infinitly. Every time a key is depressed on the terminal, a status report will be written. Afterwards the test continues.

Example of status report:

```
                 FRAMES O/U-RUN CRC-ERR DATA-ERR ABORT  ASTAT LSTAT
GENERATOR OUT:    1890     0       0       0      0     7404  5404
GENERATOR IN :    1890     0       0       0      0     3750  3750
```

The differents columns signify:

FRAMES	= Number of frames transmitted (generator out) or received (generator in).
O/U-RUN	= Number of over/under-runs detected.
CRC-ERR	= Number of CRC-errors detected.
DATA-ERR	= Number of data errors detected, i.e. CRC of received was correct, but the data was still corrupted.
ABORT	= Number of aborts detected.
ASTAT	= Accumulated status bits so far (receiver/transmitter transfer status register).
LSTAT	= Last status read from Receiver/transmitter transfer status.

Example:

```
*LOOP-TEST
```

6.4.15.RUN

This command runs either the **BASIC-IO-TEST**, the **BASIC-DMA-TEST** or both.

Example:

```
*RUN
I/O-DMA OR BOTH (0,1,CR): CR

BASIC-IO-TEST

BASIC-DMA-TEST

BASIC-IO-TEST

BASIC-DMA-TEST  etc..
```

6.4.16.SELECT-PROTOCOL

This command can modify the control register of the MPCC on the HDLC interface. See the manual HDLC - High Level Data Link Control Interface, ND-12.018.01 section II.2.1.1 for further information.

Specify a number corresponding to the bits you wish to set in each parameter.

Example:

```
*SELECT-PROTOCOL
PCR-REG: 20
SYNC/ADDR: 100
CHAR.LENGHT REG: 347

*
```

6.4.17.SET-BLOCK-PARAMETERS

This command specifies the DMA block size (1-2000 oct.) and displacements (0 - block size) used by the **LOOP-TEST** and **EXTENDED-LOOP-TEST** commands.

Example:

```
*SET-BLOCK-PARAMETERS
DMA BLOCK-SIZE: 1000
DISP1: 100
DISP2: 100

*
```

6.4.18.SET-MEMORY-LIMITS

Initially the program will use all of the memory (except the part occupied by the program itself) for the DMA transfers. If for some reason you wish to restrict the use to only a part of the memory, use this command.
The parameters are given as page numbers. Default values will be the initial values.

Example:

```
*SET-MEMORY-LIMITS
SPECIFY (DEC.) PAG NO. BETWEEN 8 AND 159  (This  is  highest  page  in
machine)
LOWEST PAGE: 8
HIGHEST PAGE: 127

*
```

6.4.19.SPECIFY-LOOP-CONFIGURATION

This command sets up how to run the **EXTENDED-LOOP-TEST.** Logically two functions are used:

- A generator transmits frames (generator out) and checks received frames for corruptions (generator in).

- Echo (if used) will transmit (echo out) frames exactly as received by "echo in" (if no CRC-errors or overrun).

Some examples are given on the following page.

Examples:

```
*SPECIFY-LOOP-CONFIGURATION
GENERATOR OUTPUT DEVNO: 1640
GENERATOR INPUT DEVNO (0=NONE): 1660
MAINT.MODE(Y/N): Y

                                (1640.Tx =====> 1660.RX)

- - - -

GENERATOR OUTPUT DEVNO: 1640
GENERATOR INPUT DEVNO (0=NONE): 1640
MAINT.MODE(Y/N): N
ECHO INPUT DEVNO (0=NONE): 1660
ECHO OUTPUT DEVNO : 1660

                                (1640.Tx =====> 1660.Rx
                                 1640.Rx <===== 1660.Tx)

- - - -

GENERATOR OUTPUT DEVNO: 1640
GENERATOR INPUT DEVNO (0=NONE): 1700
MAINT.MODE(Y/N): N
ECHO INPUT DEVNO (0=NONE): 1660
ECHO OUTPUT DEVNO : 1660

                   (1640.Tx =====> 1660.Rx/Tx =====> 1700.Rx)

- - - -

GENERATOR OUTPUT DEVNO: 1640
GENERATOR INPUT DEVNO (0=NONE): 1660
MAINT.MODE(Y/N): N
ECHO INPUT DEVNO (0=NONE): 1640
ECHO OUTPUT DEVNO : 1660

                   (1640.Tx =====> 1640.Rx-1660.Tx =====> 1660.Rx)
```

CHAPTER SEVEN

DISC-TEMA

7.DISC-TEMA

7.1.Introduction

This program is a general tool for use on discs. It has a set of commands which enable you to format, dump contents, change single words or check parity on discs. It can also copy, compare and verify the contents of two discs.

At the start of the program some tests are run. During exectution of the commands the disc status is read and errors, if any, are reported. This is a useful test in itself.

The program is written for 30MB, 38MB, 60MB, 75MB, 90MB, 288MB, 2-75MB and 3-75MB discs.

This description covers the B versjon of DISC-TEMA. The program includes the E version of the testprogram monitor.

The program has been written to make it easy to run with the modes that are assumed to be the most common ones.

Almost all of the commands are used in the same way. The errors are also reported in the same way. The description of the various commands cover only what is special to the commands.

7.2."TO" and "FROM" discs

The program refers to a "TO" and a "FROM" disc. The disc types of the "TO" and "FROM" discs must therefore be set before executing the commands. This might be done either in the initial action (see 7.7.1) or by means of the command **SET-DISC-TYPE.**
The types specified are printed by the command **PROGRAM-STATUS.**
To see which disc is used for each command, look at the description for the individual commands.

7.3.Parameter input in the commands

All commands start by typing the DISC TYPE of the disc being used.
All commands ask for a starting point, where the function associated with the command starts.
The next command asks for the AMOUNT; indicating how many sectors the function is to perform upon.
In most cases the job is so big that the hardware cannot handle it in one transfer, therefore the program asks for the BLOCK SIZE in which it will divide the total amount of sectors.

7.3.1.Starting point

The starting point is the physical address of the first sector where the function will start. It is given as CYLINDER, SURFACE and SECTOR. If the command involves two discs, the program asks for two seperate parameters Default answer on all parameters in the physical address is 0. Min values are also 0. Max values are the max for the selected DISC TYPE.

Example of specifying the starting point :

```
CYLINDER NUMBER (0-1466 OCT.): 11
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 3
SECTOR   NUMBER  (0-21 OCT.):11
```

7.3.2.AMOUNT

AMOUNT is the number of sectors for the function to perform upon.
From the given starting point, the program calculates the remaining number of sectors on the disc.
If the command involves two discs, the remainder is calculated for both discs.

The smallest remainder is set to be, both default and maximum AMOUNT. Minimum AMOUNT is 1. In case of FORMAT, the AMOUNT is given as tracks not sectors. The question specifies whether sectors or tracks are wanted.

Example of specifying amount

```
AMOUNT (NO. OF SECTORS) (1-216755 OCT.): 4573
```

7.3.3.BLOCK SIZE

If the AMOUNT specified is greater than that be handled by the disc controller in one transfer, the program will divide the job into several transfers.

The size of each such transfer is called : BLOCK SIZE. The BLOCK SIZE is the number of sectors acted upon in each transfer.

Maximum BLOCK SIZE is the maximum transfer that the disc controller can handle. If specified AMOUNT is smaller than this, Maximum BLOCK SIZE is equal to the specified AMOUNT.

Minimum BLOCK SIZE is 1.

Default BLOCK SIZE is equal to the max BLOCK SIZE.

If AMOUNT is set to 1, BLOCK SIZE is not asked for.

It should be noted that the ECC disc only uses the address part of the first sector in each transfer. There might therefore be an un-detected address error on the disc surface because the bad address part was not used. SINTRAN almost always uses a BLOCK SIZE of 2 (1 page). It should also be remembered that when using a smaller BLOCK SIZE, the function will take a longer time.

Example of specifying BLOCK SIZE.

```
BLOCK SIZE (1-132 OCT.): 11
```

7.3.4.Example of parameter answering

```
>PARITY-CHECK
PARITY-CHECK
DISC-75MB-1
CYLINDER NUMBER (0-1466 OCT.): 11
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 3
SECTOR   NUMBER  (0-21 OCT.): 11
AMOUNT (NO. OF SECTORS) (1-216755 OCT.): 4573
BLOCK SIZE (1-132 OCT.): 11
```

7.4.Definition

All commands start by doing a "cleaning up sequence". This sequence is the only thing done by the command **CLEAR-DEVICE** (see 7.10.3.). The program then continues from the given start point, block size by block size. AMOUNT is decremented by BLOCK SIZE after each transfer. The transfer counter is incremented by 1 for each transfer. The technique used to find the address for the next transfer, is to take the start point and add the BLOCK SIZE to form the new address. When the AMOUNT is zero the job is finished.

Before each transfer the PID bit 11 is cleared. After the transfer, the program checks to see if the disc interface has set the bit again. The ident given by the disc interface is also checked.

7.5.Directory mode

In most cases one wants to start the operation at address 0, take the whole disc as the AMOUNT and the maximum BLOCK SIZE. (Typical back up situation). In order to simplify this situation, the program may be in two different modes - **DIRECTORY** or **PHYSICAL.** The program is initially in **DIRECTORY** mode after loading from the floppy. This means that the commands **COPY** and **VERIFY** will only ask for enough parameters to know the directory in question. The function is then performed with logical address 0 as start address, the whole directory as AMOUNT and BLOCK SIZE set to maximum. If it is required to specify other areas, AMOUNT, or BLOCK SIZE, the command **DIRECTORY-MODE** with the answer NO must now be used.
The commands **CHANGE** and **DUMP-DISC-CONTENT** are not affected by the **DIRECTORY-MODE.**

7.6.Overlap test

If the command involvs two disc, it is tested if the two disc areas overlapp each other. If they do, the program asks wheter this is OK or not.
Two disc areas overlapp each other if the following is true:

1) They are both on the same disc system.

2) They are both on the same disc unit.

3) The lowest logical address added to the amount is greater than the higest loggical address.

Default answer is YES.

```
DISC AREAS OVERLAP EACH OTHER. OK ? (YES/NO): YES
```

7.7.Initial action

If DISC-TEMA is stared in address 0, the whole initial action described belove is executed. If DISC-TEMA is started in address 20, no action is done except for what is done by the TESTPROGRAM-MONITOR itself (see 1.12).

7.7.1.Initial question for DISC TYPE

The program starts by asking for a DISC TYPE. Most installations have only one DISC TYPE, therefore the program asks only once for DISC TYPE. The type specified is then taken both as the "TO" and the "FROM" disc. When different DISC TYPES are to be used, only one of the types will be specified here. In order to set up the desired DISC TYPE, the command **SET-DISC-TYPE** must be used to set up the relevant discs.

An IOX for the disc is issued in order to see that it does not give an IOX ERROR. If the IOX gives IOX ERROR the message NO SUCH DEVICE 1) is typed an the question is repeated. Specifying the DISC TYPE is done in a table selection manner (see 1.12).

Standard SINTRAN names are used.

Default DISC TYPE is DISC-75MB-1.

7.7.2.Initial test

When the DISC TYPE is specified, the memory buffers are initialised, in order to write the contents with correct parity in the buffers. Then the dataway to the specified disc controller is tested. This is done by writing a counting pattern into the block address register 1. If the contents read back from the block address register 1 is wrong, the same pattern is written into the memory address register. Depending on whether this also gives an error, the conclusion is that the error generally concers the dataway to the disc interface or, that the error is special for the block address register. In both cases error messages are printed, specifying whether it is a dataway error or block address register error. The error messages include notes that explain the case.

Then the whole memory address register (16 bit for ND-10, 24 bit for ND-100) is tested with a counting pattern. If the pattern read back is not the same as that written, an error message is printed. When this test has been done the command processor is entered, and the desired command may be executed.

7.7.3.Example of initial action

```
DISC TEST AND MAINTENANCE SYSTEM (DISC-TEMA)

203134.B

ISSUED : 15. JANUARY 1982

DISC TEST AND MAINTENANCE SYSTEM (DISC-TEMA)

DISC NAME: DISC-75MB-1
DISC-75MB-1
MEMORY BUFFERS INITIALIZED
DATA WAY TO DISC SYSTEM 1 TESTED
MEMORY ADDRESS REGISTER ON DISC SYSTEM 1 TESTED

THE COMMAND HELP GIVES YOU A LIST OF THE COMMANDS

>
```

7.8.Information typed by PROGRAM-STATUS

The command **PROGRAM-STATUS** tells you whether the program is in **DIRECTORY** or **PHYSICAL** mode. It also types the disc specified as **"TO"** and **"FROM"**. The number of transfers is also printed.

```
>PROGRAM-STATUS
PROGRAM-STATUS

-------------------------------------------------------------
*   DAY,    * CONSOLE* PRINTER*  STOP ON *  INTER- *  PAGING *
* HOUR:MIN  * DEV.NO.* DEV.NO.* FULL PAGE*   RUPT  *         *
-------------------------------------------------------------
* 28,11:10  *      1  *     1  *    OFF   *  SINTRAN RUNNING *
-------------------------------------------------------------
* "FROM" DISC   * "TO" DISC     * MODE       * TRANSFERS      *
-------------------------------------------------------------
* DISC-75MB-1   * DISC-75MB-1   * DIRECTORY *              0 *
-------------------------------------------------------------
>
```

7.9.Sintran/stand-alone

When the program is run under SINTRAN, none of the commands using the disc are executable. The only executable commands are TRANSLATE and the legal monitor commands.

7.10.Available commands

The following commands exist in the program : CHANGE, CLEAR-DEVICE, COMPARE, COPY, DUMP-DISC-CONTENT, FORMAT, PARITY-CHECK, SET-DISC-CONTENT, SEEK, TRANSLATE, VERIFY, ALIGN, SET-DISC-TYPE, DIRECTORY-MODE, CLEAR-TRANSFER-COUNTER, SET-RETRY

7.10.1.ALIGN <Unit> <Cylinder> <Head>

This command positions the disc heads at the align cylinder on a CE Pack. (Not applicable to 150MB MMD) There is an immediate reaction on C and H. If H is pressed the question for head will come up again. If C is pressed the question for cylinder will come up again.

The "FROM" disc is used.
This command is not affected by directory mode.
This command is not executable under SINTRAN.

```
>ALIGN
ALIGN

DISC-75MB-1
UNIT NUMBER (0-3 OCT.): 0
CYLINDER (0-1466 OCT.): 365
HEAD (0-4 OCT.): 0
HEAD (0-4 OCT.): 1                % H is pressed
HEAD (0-4 OCT.): 2                % H is pressed
HEAD (0-4 OCT.): 3                % C is pressed
CYLINDER (0-1466 OCT.): 4         % H is pressed
HEAD (0-4 OCT.): 0                % ESC is pressed
>
```

7.10.2.CHANGE <Unit> (<Sub Unit>) <Starting point>

A command to inspect and change single words on the disc.
AMOUNT is always one sector.
Changes are done in a MOPC way.
If the contents has been changed it is written back upon detection of the sign "@". The "FROM" disc is used.
This command is not affected by directory mode.
This command is not executable under SINTRAN.

```
>CHANGE
CHANGE

DISC-75MB-1
UNIT NUMBER (0-3 OCT.): 0
CYLINDER NUMBER (0-1466 OCT.): 1000
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 2
SECTOR   NUMBER  (0-21 OCT.): 11

READY
30/123507 123407
000031 * 123510 123410
000032 * 123511 123411
000033 * 123512 123412
000033 * 123512 40/177777 0
000041 * 177776 @
SECTOR WRITTEN BACK
>
```

7.10.3.CLEAR-DEVICE <Unit>

Clears the device by doing a return-to-zero seek . Followed by a DEVICE-CLEAR .

The "FROM" disc is used.
This command is not affected by directory mode.
This command is not executable under SINTRAN.

```
>CLEAR-DEVICE
CLEAR-DEVICE

DISC-75MB-1
UNIT NUMBER (0-3 OCT.): 0
>
```

7.10.4.CLEAR-TRANSFER-COUNTER

This command sets the transfer counter to zero.

The transfer counter is typed by the command **PROGRAM-STATUS.**

This command is not affected by directory mode.
This command is not executable under SINTRAN.

```
>CLEAR-TRANSFER-COUNTER
CLEAR-TRANSFER-COUNTER

>
```

7.10.5.COMPARE <Unit> (<Sub Unit>) (<Starting point>) <Unit> (<Sub Unit>) (<starting point> <Amount> <Block size>)

Compares the contents of two disc areas by reading the first area, (disc read) and comparing it with the second area (disc compare). NB! Disc read uses error correction, but disc compare does not!

This command is not executable under SINTRAN.
This command is affected by directory mode.

See also VERIFY.

```
>COMPARE
COMPARE

FROM DISC-75MB-1
UNIT NUMBER (0-3 OCT.): 0
CYLINDER NUMBER (0-1466 OCT.): 0
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 0
SECTOR   NUMBER  (0-21 OCT.): 0

TO DISC-75MB-1
UNIT NUMBER (0-3 OCT.): 0
CYLINDER NUMBER (0-1466 OCT.): 0
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 0
SECTOR   NUMBER  (0-21 OCT.): 0
AMOUNT (NO. OF SECTORS) (1-220526 OCT.): 220526

DISC AREAS OVERLAP EACH OTHER. OK ? (YES/NO): YES
BLOCK SIZE (1-132 OCT.): 132
>
```

7.10.6.COPY <Unit> (<Sub Unit>) (<Starting point>) <Unit> (<Sub Unit>) (<Starting point> <Amount> <Block Size>)

Copies the contents of one disc area to another disc area. By doing a read from the "FROM" disc and a write to the "TO" disc.

This command is not executable under SINTRAN.
This command is affected by directory mode.

```
>COPY
COPY

FROM DISC-75MB-1
UNIT NUMBER (0-3 OCT.):0


TO DISC-75MB-1
UNIT NUMBER (0-3 OCT.):1

>
```

7.10.7.DIRECTORY-MODE <Yes/No>

Sets or resets DIRECTORY MODE e.g., when the commands COMPARE, COPY, FORMAT, PARITY and VERIFY are executed, the program will only ask for parameters concerning definition of the directory (not detailed disc addresses). (See 7.5).

Default is to change.

```
>DIRECTORY-MODE
DIRECTORY-MODE

DIRECTORY MODE ON (YES/NO): NO

>
```

7.10.8.DUMP-DISC-CONTENT <Unit> (<Sub Unit>) <Starting point> <Amount> <Block size>

Dumps the contents of a disc area on the device specified by the command : **SET-PRINTER-DEVICE-NUMBER.**

The "FROM" disc is used.
This command is not affected by directory mode.
This command is not executable under SINTRAN.

```
>DUMP-DISC-CONTENT
DUMP-DISC-CONTENT

DISC-75MB-1
UNIT NUMBER (0-3 OCT.): 0
CYLINDER NUMBER (0-1466 OCT.): 300
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 2
SECTOR   NUMBER  (0-21 OCT.): 7
AMOUNT (NO. OF SECTORS) (1-22 OCT.): 1
BLOCK SIZE (1-3 OCT.): 1

DISC DUMP OF
DISC SYSTEM : 1 UNIT : 0
CYLIN :  300 SURFACE :  2 SECTOR : 7

  0 123456 123457 123460 123461 123462 123463 123464 123465
 10 123466 123467 123500 123501 123502 123503 123504 123505
 20 123506 123507 123510 123511 123512 123513 123514 123515
 30 123516 123517 000000 000000 000000 000000 000000 000000
 40 000000 000000 000000 000000 000000 000000 000000 000000

SAME

750 000000 000000 000000 000000 442200 125100 177777 044403
760 063204 014560 025400 014230 422340 177400 166427 024533
770 025604 032132 021620 032126 764731 154100 623207 054127
>
```

7.10.9.FORMAT <Unit> (<Sub Unit>) (<Starting point> <Amount>)

FORMAT will write the address part of the sectors. Usually, this is not done with a system disc as it destroys the contents of the disc pack.
AMOUNT means the number of tracks to format with this command. Block size is always one track.

The "FROM" disc is used.
This command is affected by directory mode.
This command is not executable under SINTRAN.

```
>FORMAT
FORMAT

DISC-75MB-1
UNIT NUMBER (0-3 OCT.): 0
CYLINDER NUMBER (0-1466 OCT.): 300
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 2
AMOUNT (NO. OF TRACKS) (1-6121 OCT.): 7
>
```

7.10.10.PARITY-CHECK <Unit> (<Sub Unit>) (<Starting point>) <Amount> <Block Size>)

Reads the contents of a disk area without storing it in the memory, and checks for parity.

The "FROM" disc is used.
This command is affected by directory mode.
This command is not executable under SINTRAN.

```
>PARITY-CHECK
PARITY-CHECK
DISC-75MB-1
CYLINDER NUMBER (0-1466 OCT.): 11
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 3
SECTOR   NUMBER  (0-21 OCT.): 12
AMOUNT (NO. OF SECTORS) (1-216754 OCT.): 4573
BLOCK SIZE (1-132 OCT.): 11
>
```

7.10.11.SEEK <Unit> (<Sub Unit>) (<Starting point>) (<Sub Unit>) (<Starting point>)

The command starts consecutive seeks between the two disc addresses specified. To stop the seeking press <ESC>. The "FROM" disc is used. The command is not affected by directory mode and is not executable under SINTRAN.

```
>SEEK
SEEK

FROM DISC-75MB-1
UNIT NUMBER (0-3 OCT.): 0
CYLINDER NUMBER (0-1466 OCT.): 0
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 0
SECTOR   NUMBER  (0-21 OCT.): 0

TO DISC-75MB-1
CYLINDER NUMBER (0-1466 OCT.): 1466
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 2
SECTOR   NUMBER  (0-21 OCT.): 11           % ESC is pressed
>
```

7.10.12.SET-DISC-CONTENT <Unit> (<Sub Unit>) <Starting point> <Amount> <First Word> <Modifyer>

SET will write one or more sectors on the disc. The contents are specified by the user.
The pattern specified as **FIRST WORD** is written into the first word on the first sector specified. In the second word of the sector is written the first word pluss the **MODIFYER.** In the third word is written the second word pluss the **MODIFYER,** and so on.
The contents of word n may be calculated by the following equation.:

(n)=n*MODIFYER+FIRST WORD

Where is the word number calculated from the beginning of the first sector, specified. The "FROM" disc is used.
This command is not affected by directory mode and not executable under SINTRAN.

```
>SET-DISC-CONTENT
SET-DISC-CONTENT

DISC-75MB-1
UNIT NUMBER (0-3 OCT.): 0
CYLINDER NUMBER (0-1466 OCT.): 700
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 0
SECTOR   NUMBER  (0-21 OCT.): 4
AMOUNT (NO. OF SECTORS) (1-101722 OCT.): 10
BLOCK SIZE (1-132 OCT.): 11
FIRST WORD (0-177777 OCT.): 0
MODIFIER (0-177777 OCT.): 1
>
```

7.10.13.SET-DISC-TYPE <Disc Type> <Disc Type>

Sets the disc type for the "FROM" and the "TO" discs. Because they are both set to the same at program start, this is only useful if the two differ.

Default is not to change type.

```
>SET-DISC-TYPE
SET-DISC-TYPE

"FROM" DISC DISC NAME: DISC-30MB-1
DISC-30MB-1


"TO" DISC DISC NAME: DISC-2-75MB-1
DISC-2-75MB-1

>
```

7.10.14.SET-RETRY <Read Retry> <Write Retry>

Sets number of retries on read and write independently. Default values are

```
Read retry  : 28
Write retry : 4
>SET-RETRY
SET-RETRY

NUMBER OF RETRIES ON READ (0-64 DEC.): 24

NUMBER OF RETRIES ON WRITE (0-64 DEC.): 4

>
```

7.10.15.TRANSLATE (<Disc-Type>) <Address Type> (<Sub Unit>) <Address>

Translates between logical, physical or page disc address. The answer is always a table containing all address types in both octal and decimal. If two different disc types are specified the program asks for DISC TYPE.
Example:

```
>TRANSLATE
TRANSLATE
ADDRESS TYPE: PHYSICAL-ADDRESS
PHYSICAL-ADDRESS

DISC-75MB-1
CYLINDER NUMBER (0-1466 OCT.): 11
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 3
SECTOR   NUMBER  (0-21 OCT.): 11
-------------------------------------------------
*      * CYLI * SUR  * SEC * LOGICAL * PAGE NO.*
*      * NDER * FACE * TOR * ADDRESS *         *
-------------------------------------------------
* OCT. *   11 *    3 *  11 *    1551 *     664 *
-------------------------------------------------
* DEC. *    9 *    3 *   9 *     873 *     436 *
-------------------------------------------------
ADDRESS TYPE: PAGE-ADDRESS
PAGE-ADDRESS

DISC-75MB-1
PAGE NUMBER (0-110252 OCT.): 11
-------------------------------------------------
*      * CYLI * SUR  * SEC * LOGICAL * PAGE NO.*
*      * NDER * FACE * TOR * ADDRESS *         *
-------------------------------------------------
* OCT. *    0 *    1 *   0 *      22 *      11 *
-------------------------------------------------
* DEC. *    0 *    1 *   0 *      18 *       9 *
-------------------------------------------------
ADDRESS TYPE: LOGICAL-ADDRESS
LOGICAL-ADDRESS

DISC-75MB-1
LOGICAL ADDRESS (0-220525 OCT.): 112
-------------------------------------------------
*      * CYLI * SUR  * SEC * LOGICAL * PAGE NO.*
*      * NDER * FACE * TOR * ADDRESS *         *
-------------------------------------------------
* OCT. *    0 *    4 *   2 *     112 *      45 *
-------------------------------------------------
* DEC. *    0 *    4 *   2 *      74 *      37 *
-------------------------------------------------
ADDRESS TYPE: EXIT
EXIT
>
```

7.10.16.VERIFY <Unit> (<Sub Unit>) (<Starting point>) <Unit> (<Sub Unit>) (<Amount> <Block Size>)

VERIFY compares the contents of two disc areas by reading them both from the discs, and then comparing them word by word in the memory. This command is affected by directory mode.
This command is not executable under SINTRAN.

```
>VERIFY
VERIFY

FROM DISC-75MB-1
UNIT NUMBER (0-3 OCT.): 0
CYLINDER NUMBER (0-1466 OCT.): 14
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 1
SECTOR   NUMBER  (0-21 OCT.): 21


TO DISC-75MB-1
UNIT NUMBER (0-3 OCT.): 2
CYLINDER NUMBER (0-1466 OCT.): 14
SURFACE  NUMBER ( DATA HEAD ) (0-4 OCT.): 1
SECTOR   NUMBER  (0-21 OCT.): 21
AMOUNT (NO. OF SECTORS) (1-216373 OCT.): 1
>
```

7.11.Error messages

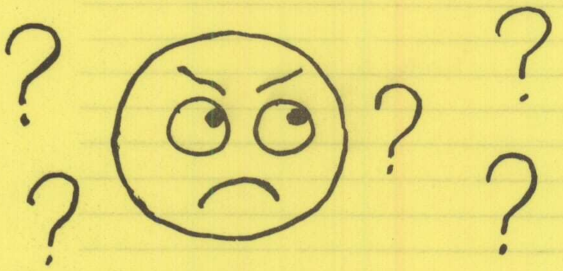
The program has a common error routine. The error message consists of the text ERROR !!! followed by the time that the error occurred, printed as DD,HH:MM. Next is the DISC SYSTEM, UNIT and SUB UNIT (if the disc has SUB UNITs). Then the PHYSICAL ADDRESS and the DEC. PAGE ADDRESS where the error occurred is printed.
After that the HARDWARE STATUS that describes the error is printed as an octal number. The meaning of the bits in the status word is then typed out in clear text.
Then the ADDITIONAL STATUS is printed out, both as an octal number and in clear text. Either the HARDWARE STATUS or the ADDITIONAL STATUS might be left out if it does not include any error information.
Lastly, the operation that caused the error, is printed.

Example of error printout:

```
ERROR !!! 0,00:05
DISC SYSTEM : 1 UNIT : 0
CYLINDER:    11 SURFACE:   3 SECTOR: 11
DEC. PAGE NO.:436

HARDWARE STATUS 020030
 DISC UNIT NOT READY
 NOT ON CYLINDER
OPERATION WAS : READ PARITY
```

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

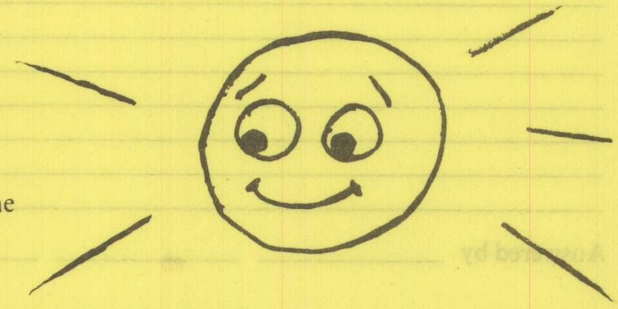


Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card - and an answer to your comments.

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 Description for ND 100/10S Volume 1

Manual number: ND-30.005.01

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

I I
I I
I I

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