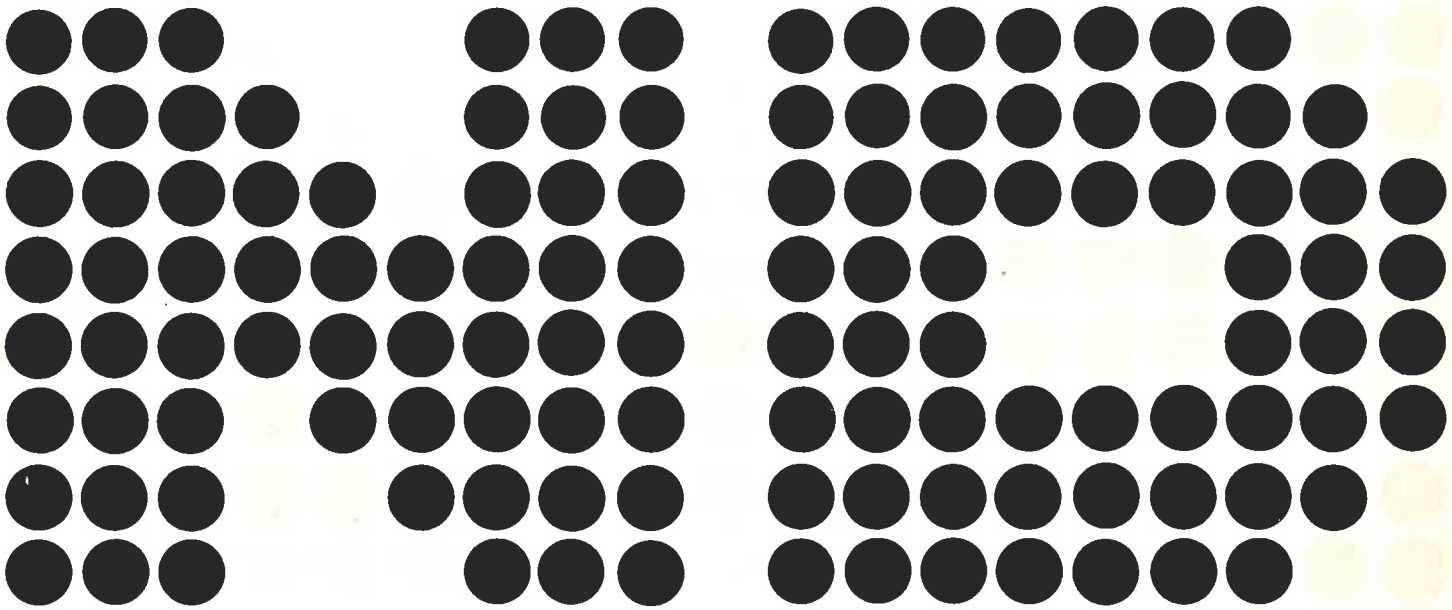
SINTRAN III
TIME-SHARING/
BATCH GUIDE

NORSK DATA A.S



SINTRAN III
TIME-SHARING/
BATCH GUIDE

Part 1
of
SINTRAN III
User's Guide

NOTICE

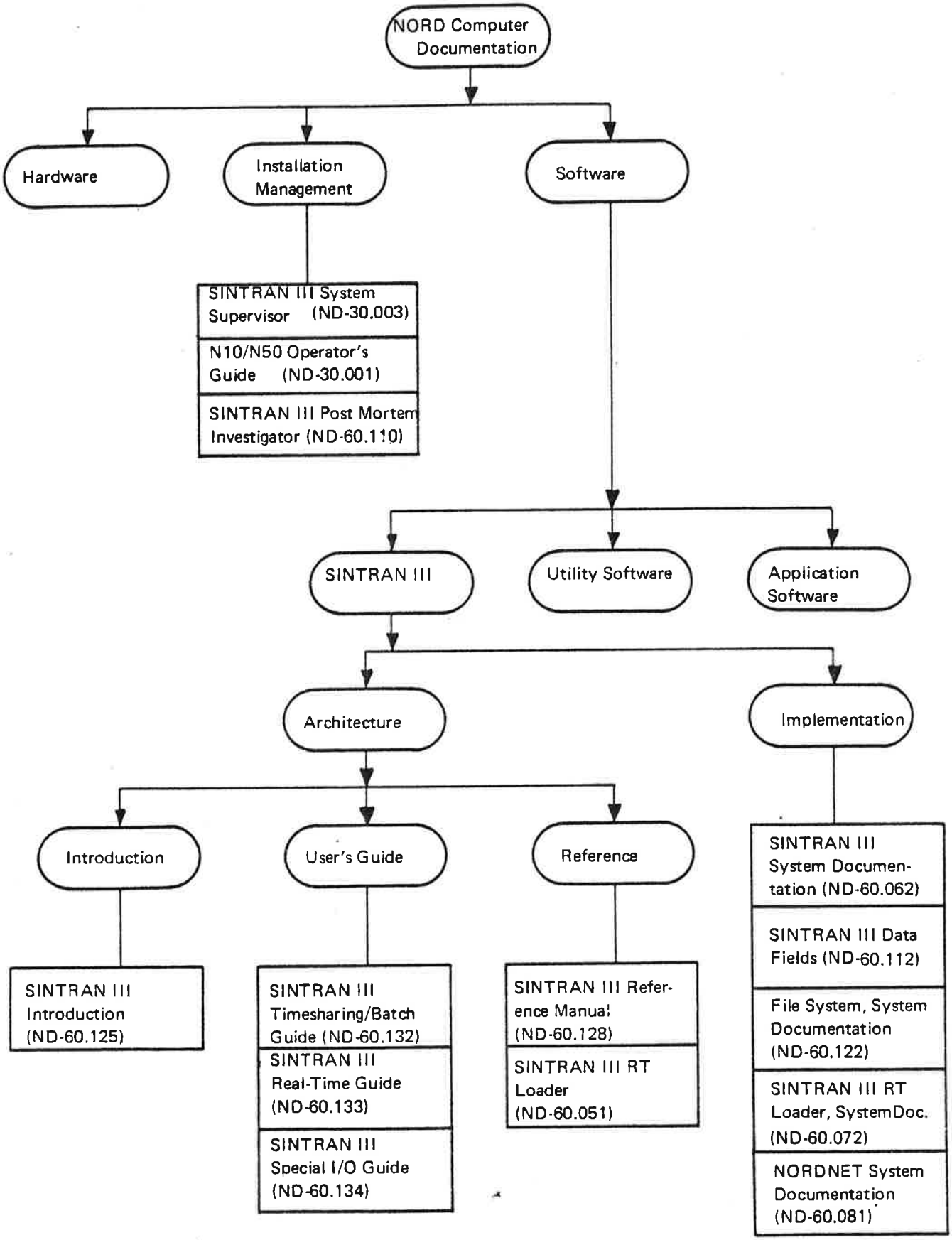
The information in this document is subject to change without notice. Norsk Data A.S assumes no responsibility for any errors that may appear in this document. Norsk Data A.S assumes no responsibility for the use or reliability of its software on equipment that is not furnished or supported by Norsk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

PRINTING RECORD

Printing	Notes
06/80	First Edition

SINTRAN III Time-sharing/Batch Guide
ND-60.132.01

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



SINTRAN III DOCUMENTATION OVERVIEW

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

PREFACE

THE READER

This manual is intended for the user who needs a thorough demonstration of the time-sharing/batch facilities in SINTRAN III.

PREREQUISITE KNOWLEDGE

The reader should be familiar with the contents of the manual

SINTRAN III INTRODUCTION (ND-60.125)

THE MANUAL

This manual demonstrates command and monitor calls available for the time-sharing/batch user. The functions are ordered according to functional category as opposed to the SINTRAL III REFERENCE MANUAL which is fully documented in alphabetical order.

Related manuals guiding the user's are:

SINTRAN III REAL TIME GUIDE (ND-60.133), and
SINTRAN III SPECIAL I/O GUIDE (ND-60.134)

The SPECIAL I/O GUIDE documents commands and monitor calls for special I/O and NCRDNET even if they are accessible by the time-sharing/batch user. The introduction of this manual gives further details.

Other SINTRAN III manuals are:

SINTRAN III REFERENCE MANUAL (ND-60.128),
SINTRAN III SYSTEM SUPERVISOR (ND-60.103), and
SINTRAN III RT LOADER (ND-60.051)

This manual (partially) obsoletes the following manual:

SINTRAN III User's Guide (ND-60.050)

THE PRODUCT

This manual documents the 1980 version of SINTRAN III/VS.

Contents

1. Introduction

1.1. General

The NORD computer system is a medium scale, general purpose computer system which, because of its modular design, can be configured for a wide variety of applications.

SINTRAN III is a multiprogramming, multi-lingual, real-time operating system that supervises the processing of user programs submitted to a NORD computer system. SINTRAN III relieves the user from much of the mundane work connected with developing and running of user programs.

The modular design allows a minimal software configuration of only 64 Kbytes, the memory resident SINTRAN III/RT. The virtual storage version, SINTRAN III/VS may be expanded with up to 32 Mbytes of main memory, large disks, connections to other computers, etc. Several NORD computers may therefore form a computer network.

This manual is primarily oriented towards the SINTRAN III/VS system. The SINTRAN III/RT system is documented in "SINTRAN III - CORE USER'S GUIDE (ND-60.082)".

Each terminal and batch activity is run as a separate process. These processes are called background programs and are run on a time sharing basis in the NORD computer. The other type of processes, foreground programs, are created by real time programmers. They normally run with a higher priority in the computer. In general, a real time program may be either a background or foreground program.

1.2. User Categories

While this manual is only concerned with time-sharing users, other types of users recognised in SINTRAN III are the real-time user, developing foreground programs, and system supervisors, managing the computer installation. Another type of user is the "end user", performing aplication oriented tasks such as data entry or data inquiry through application programs. This user will not be discussed in the SINTRAN III documentation.

1.2.1. Time-Sharing and Batch Users

This user group will normally access the system through a terminal or by submitting jobs for batch processing from disk files or card decks containing the necessary commands and data.

Time-sharing and batch users are known to the system by names which must be presented at the beginning of the terminal session

or as the first command in a batch job. The user name may be protected by a password choosen by the user to prevent unauthorized access to the system.

The user name must be established by those responsible for the system. Time-sharing and batch users access the system in its "background" processing mode, normally executing programs with a lower priority than time critical real-time programs executing in "foreground" mode. The time-sharing and batch processes are normally given processing time on an equal basis in a round-robin fashion.

Trying to access real-time user and system supervisor commands is not permitted and will produce an error message.

1.2.2. Real-Time Users

The operating system recognizes a special user name RT as a privileged user capable of controlling the real-time processing programs. This user name should be protected by a password to prevent unauthorized access.

This user is permitted to use commands for developing and running real-time programs. The manual "SINTRAN III REAL-TIME GUIDE" (ND-60.133) gives the details about this activity.

The user RT may also use the time-sharing and batch facilities for program development and testing purposes. Trying to access system supervisor commands is not permitted and will produce an error message.

1.2.3. System Supervisor

The operating system recognizes a special user name SYSTEM as a privileged user having the access to all commands in SINTRAN III. The special SYSTEM commands facilitate creating and deleting users, creating and deleting directories, together with other commands for system management. The manual "SINTRAN III SYSTEM SUPERVISOR" (ND-60.103) gives the details about system management.

This user name should be protected by a password.

1.3. Commands and Monitor Calls

A command may be entered from a terminal, a mode input file, or a batch input file. The command is checked for validity and invokes the appropriate function. Commands are clasified according to user categories.

Time-Sharing/Batch Commands.

- calling compilers, assemblers, editors, other subsystems and user programs

- creating, deleting and maintaining files

- reserving and releasing files, etc.

The commands for SINTRAN III communication and special I/O are not included in this manual but may be found in the SINTRAN III SPECIAL I/O GUIDE (ND-60.134). The commands are,

@DEVICE-FUNCTION

@IOSET

@LIST-DEVICE-FUNCTION

@LIST-REMOTE-QUEUE

@LOCAL

@NORD-50

@REMOTE

Real-Time Commands

- starting and stopping RT programs

- operating new RT programs using the RT loader

- reserving and releasing peripheral devices on behalf of RT programs, etc.

System Supervisor Commands

- starting and stopping the system

- creating, entering and releasing file directories on disk

- creating and deleting users

- allocating resources, etc.

A monitor call is a special instruction in machine code. It is used to call service functions within the operating system. Monitor calls as machine code can only be performed in MAC, NORD PL, etc. In high level code such as FORTRAN, etc. a corresponding set of subroutine calls is used. In this manual, the examples show only the FORTRAN call, if it is available. The Reference Manual documents the complete definition including calls in both MAC and FORTRAN. Some monitor calls are restricted to real-time users only. They are

documented in a separate manual, SINTRAN III REAL TIME GUIDE (ND-60.133). Also, this manual does not document calls for SINTRAN III communication and special I/O. They are,

IOSET (MON 141)

MAGTP (MON 144)

ACM (MON 145)

GRAPHIC (MON 155)

TRACB (MON 156)

WRQI (MON 163)

XMSG (MON 200)

HDLC (MON 201)

Refer to the SINTRAN III SPECIAL I/O GUIDE for these calls.

1.4. Hardware Environment

The minium hardware configuration required to run SINTRAN III/VS is:

- NORD-10/S or NORD-100 with memory management system
- main memory of 128 Kbytes
- disk unit with controller
- floppy disk unit or paper tape reader
- console terminal

The range of standard peripherals include alphanumeric, graphic, and color display systems, hard-copy terminals, printers, plotters, paper tape readers and punches, card readers and punches, magnetic tape stations, floppy disks, hard disks, A/D and D/A conversion, telex, modem and CAMAC interfaces. Processor options includes local and multiport memory up to 32 Mbytes, Direct Memory Access and cache memory. Norsk Data marketing information gives further details about the hardware environment.

1.5. SINTRAN III Subsystems

Some subsystems are integrated into the operating system. They are the file system, the RT loader and the spooling system. The other subsystems are separated from but run under the operating system. They are compilers text editors, program loaders, data base systems, etc. Norsk Data marketing information gives further details about these subsystems.

2. The Terminal Session

2.1. The NORD Terminals

A user's terminal is used for the interactive communication with the system. All terminals connected to a system can function as user terminals. However, one terminal is always selected as the error device. This is the terminal where the system error messages will appear.

Each NORD computer system contains a console terminal as shown in fig. 2-2. It is first used by the System Supervisor to get SINTRAN III on the air. After this is done, the console terminal will normally function as the error device. However, the System Supervisor can select any terminal to be the error device. So, a console terminal is always the same physical terminal while the error device is a function that can be assigned to any device.

Appendix A of SINTRAN III REFERENCE MANUAL (ND-60.128) shows the standard terminals delivered with NORD systems. If your terminal is not documented here, refer to the manufacturer's "User' Manual" for definition of the keyboard layout, or contact your local ND representative.

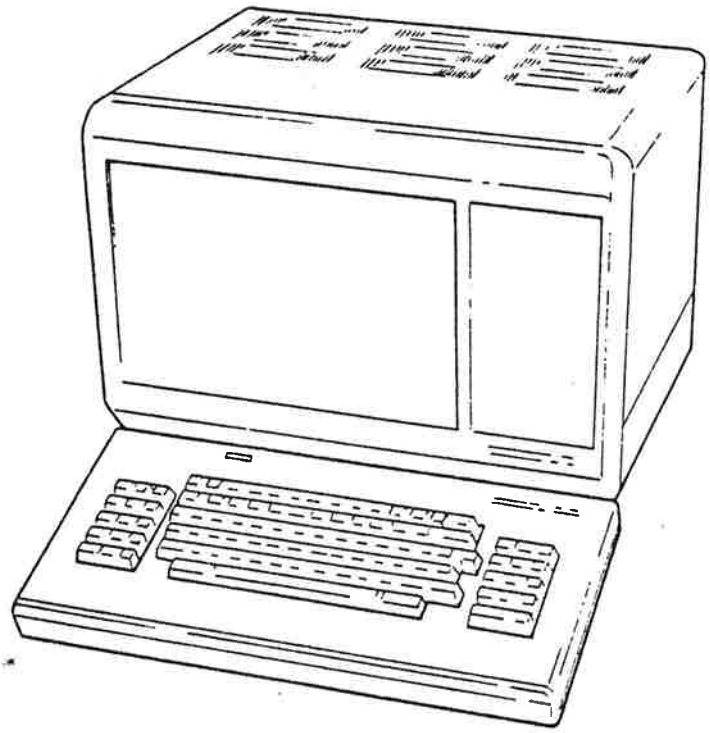


Fig. 2-1 A User CRT Terminal

2.2. Logging in

Having read the manual SINTRAN III INTRODUCTION (ND-60.125) you should be familiar with logging in. In this section we will give a more general explanation of this procedure.

The System Supervisor is responsible for getting SIII on the air and for configuring the terminals (i.e. cabling, selecting speed of transmission, e.t.c.). The rest of the procedure is carried out by the user (see fig. 2-3).

After turning on the terminal and hitting ESCAPE you enter your user name and (if used) password and project number. There may be some delay before you get the @, but once you do SIII is ready to accept your commands. The delay depends on the processing load on the computer.

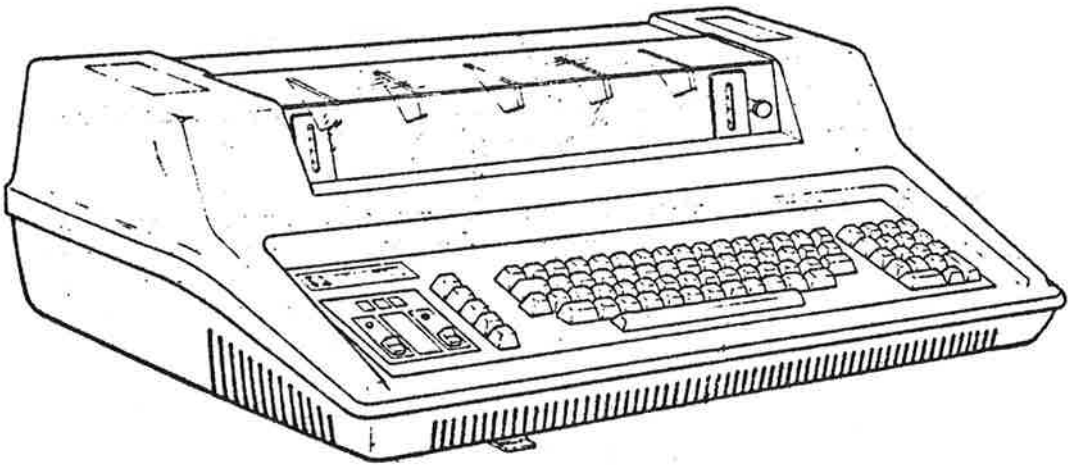
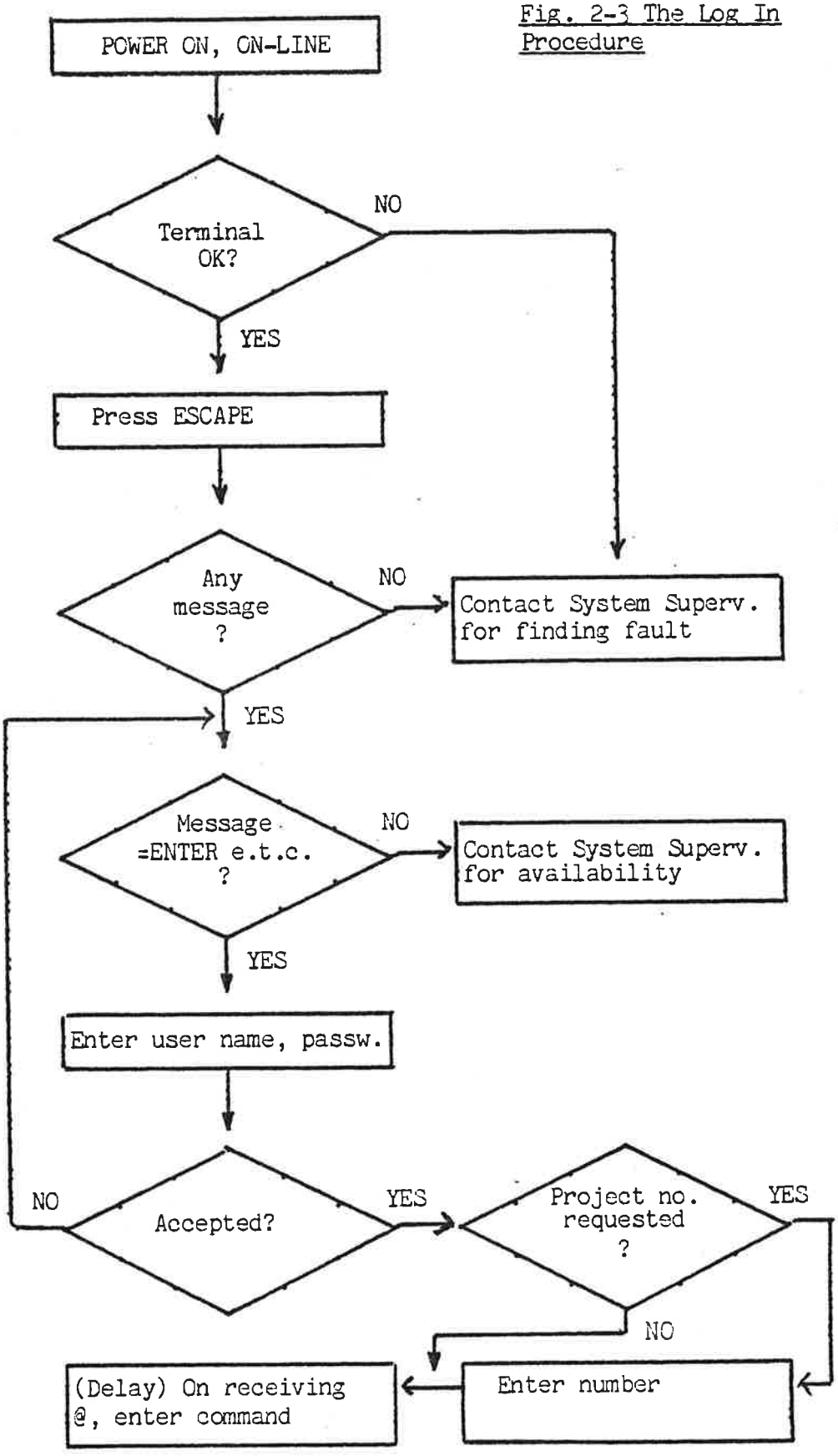


Fig. 2-2 A Console Terminal (hard-copy)

Fig. 2-3 The Log In Procedure



2.3. Executing Commands and Programs

The different states of the interactive terminal work are shown in fig. 2-4. After you have logged in, you enter the command mode as indicated by the @ on the screen.. The terminal is now waiting for a command or the name of a program. If you enter a command (consisting of name and parameters) you will enter the comand execution mode. The command may be written as upper or lower case letters. It may terminate by itself or you can in most cases terminate by pressing ESCAPE. In the command mode you can also enter the name of a program. This can be either a system program (i.e. subsystem, a program supplied wth the machine) or a user program (a program written by the user). The program is loaded and the terminal enters the user mode. All terminal input (except ESCAPE) is now handled by the user program. In a way similar to command execution, the program may terminate by itself or you may at any time press ESCAPE to cancel the execution. When you receive @, you are back in command mode.

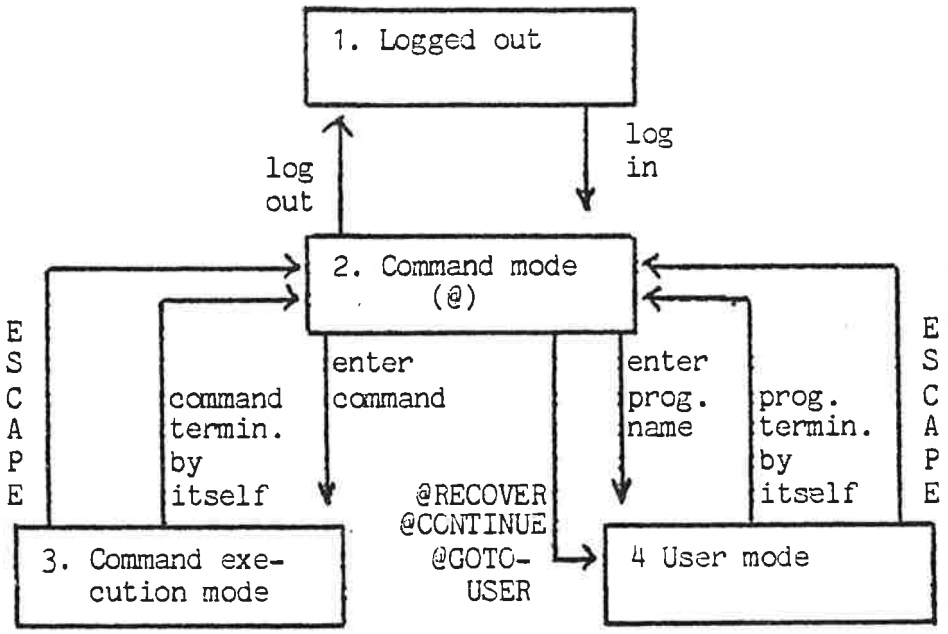


Fig. 2-4 The states of a user terminal

2.4. Writing SINTRAN III Commands

2.4.1. Single Line Commands

A SIII command is normally written on one line. As an example, consider the command below.

Definition:

@LIST-FILES <file name>,<output file>

Example:

@LIST-FILES OLE,TERMINAL

The command elements are: command name (LIST-FILE) and parameters (OLE and TERMINAL). The elements are seperated by space or comma. The command name consists of one or more subfields (LIST and FILES) seperated by hyphen (-). Parameters are defined in the documentation by enclosing the name in angular brackets (<file name>). A parameter may be one of four types.

ONE, a numeric parameter is either octal or decimal. If you specify just the digits, the radix is determined by the command. The radix can be specified explicitly by appending a B for octal or D for decimal. For ex.

@GET-TERMINAL-TYPE 35

will use 35 as a decimal number. The command could have been written,

@GET-TERMINAL-TYPE 43B

TWO, a parameter may be a file name. This is either the name of a data file or a peripheral.

THREE, the parameter can be a selection, i.e. an answer to a question, such as YES or NO.

FOUR, the parameter can be a character string terminated by an apostrophe (') before the comma or space.

2.4.2. Abbreviation in Commands

Command names, file names and selections can be abbreviated. Usually we want only one value of the command element. The element can then be abbreviated as long as it is unique. For ex. @LIST-FILES may be written as @L-FIL or @LI-FI. However, @LI-F would give the message,

AMBIGUOUS COMMAND

since it can mean both @LIST-FILES and @LIST-FRIENDS. The same applies to file names and selections. When creating a file, however, the name must be given in full.

Sometimes we want to refer to several objects with one abberviation. For ex. the command,

@LI-FI OLE,TERM

will list all files matching the name OLE, such as OLE-1, OLE-2, OLEANNA, e.t.c.

2.4.3. Use of Asterisk (*)

If an asterisk (*) appears in a command element, it will match any character. For ex.,

@LI-FI *COPY,TERM

will list files such as ACOPY, BCOPY e.t.c.

2.4.4. Default Values

Many parameters have default values. If the parameter is omitted (as indicated by two subsequent element seperators), the default value will be selected. For ex.,

@LI-FI,,L-P

will list all files for the own user in default directory on the printer. (Default directory will be explained in chapter 3.) The command

@LI-FI OLE,,

will list on the TERMINAL, which is the default <output file>. Note that two commas must end this command or else the last parameter will be asked for.

2.4.5. Prompting Line Commands

Until now, all command elements have been entered on the same line. It is also possible to enter each command element on a seperate line. For ex. you may type,

```
@LI-FI~                    (~ = RETURN)
FILE NAME:OLE~
OUTPUT FILE:TERMINAL
```

Lines containing parameters are called prompting lines. Just hitting RETURN as an answer to a parameter indicates a default value. For ex.,

```
@LI-FI~
FILE NAME:~
OUTPUT FILE:~
```

has the same function as,

```
@LI-FI,,,
```

If you do not specify all parameters in a command, the rest will be asked for (prompted).For ex.,

```
@LI-FI OLE,~
OUTPUT FILE:~
```

The last parameter is prompted, since the first line ended with only one comma.

2.4.6. Correcting Typing Errors

To correct errors while you are typing, SINTRAN III has available control characters. They are the same characters used by the EDIT command in the QED subsystem. You can correct the line you are currently typing, or edit the previous command and enter it one more time. In the latter case, the previous command must have been a single line command. This section demonstrates a minimum set of control characters. For a more complete list, refer to appendix A.

2.4.6.1. Correcting the Current Line

CTRL/A deletes the last character in the current line. Assume the cursor is positioned after the last character in the (erroneous) line,

```
@TERMINAL-STATUUS
```

To eliminate the last U, type CTRL/A twice and resume the typing from the error.

```
@TERMINAL-STATUUS^^S,,,
```

The up-arrows are echoes to CTRL/A and verify that you have eliminated the previous two characters.

CTRL/Q deletes the current line. For ex. if you have typed erroneously (cursor is positioned after U),

@TARMINAL-STATU

you might rather want to start the line all over again. Hit CTRL/Q and the cursor will advance to the beginning of the next line.

2.4.6.2. Editing the Previous Line

The basic commands for this purpose are CTRL/D, CTRL/C, CTRL/S and CTRL/E.

CTRL/D copies the rest of the old command to the new command. You can use it to enter the previous single line command one more time. Or, you can use it to terminate the correction. For ex. if you by mistake type,

```
@TARMINAL-STATUS,,
NO SUCH FILE NAME
@
```

you can correct it by typing T, E, and CTRL/D,

```
@TERMINAL-STATUS,,
  ...
```

The command is now entered correctly.

CTRL/C copies one character from the old command to the new command. For ex.,

```
@LU-FI,,,
NO SUCH FILE NAME
@
```

Now, type CTRL/C once, then the correct character (I instead of U) and then CTRL/D to terminate the command.

```
@LI-FI,,,
```

Your command will now be accepted.

CTRL/S skips one character in the old command. For ex.,

```
@LII-FI,,,
NO SUCH FILE NAME
@
```

To corrrect the command, type CTRL/C twice, then CTRL/S and finally CTRL/D.

@LI%-FI,,,

The echo of CTRL/S is percent (%).

CTRL/E is used for insert mode. Hitting it the first time sets the editing in insert mode. Hitting it again, will set the mode back to overstrike. In insert mode, all printing characters will be inserted in the old command. For ex.,

@L-FI,,,
AMBIGUOUS COMMAND
@

Type CTRL/C, CTRL/E and the missing character. Finish the editing by CTRL/E and CTRL/D.

@L<I>-FI,,,

When you enter insert mode a start angle bracket (<) i echoed. When you leave this mode (and enter overstrike mode) an end angle bracket (>) is echoed. There is no restriction to how many times you may type CTRL/E on one line.

2.5. Error Messages from SINTRAN III

This section gives an overview of the error messages you may get at the user terminal. A complete list of error messages are included in appendix D of the SINTRAN III REFERENCE MANUAL.

ONE, in commands there are two types of error messages.

a. Run time errors are errors directly from SINTRAN III kernal. (The central part of the operating system.) The message format is as follows,

ERROR nnn AT aaaaaa; <error text>

The meaning of the number nn and the <error text> is shown in appendix D.1.2 which lists the different error messages.

b. File system errors are displayed only as <error text>. The different texts are shown in appendix D.2.1 of the SINTRAN III REFERENCE MANUAL under the column "error text".

c. Command processor errors are displayed as <error text>.

TWO, errors may also occur when you are running user programs. Error messages are then either displayed directly by SINTRAN III on the terminal, or returned with an error indicator from a monitor call.

a.Run time errors may occur in the same way as for commands. The message is displayed directly on the terminal.

b.File system errors are indicated on return from monitor calls. For ex. OPEN (MON 50) has an error exit with the A-register containing the error number. The numbers are listed in appendix D.2.1 of the SINTRAN III REFERENCE MANUAL. The monitor calls ERMSG (MON 64) or QERMS (MON 65) are normally used to display the <error text> on the user terminal. A call may then look as follows,

```
MON  50              %OPEN
MON  65              %QERMS, TERMINATE ON RETURN
<normal return>
```

c. Command processor errors may also be caused by monitor calls. The <error text> is displayed. The messages can not be produced by ERMSG or QERMS.

d. FORTRAN run time errors are written out directly from a FORTRAN library routine according to the format,

```
FORTRAN RUN-TIME ERROR nnn AT ADDRESS aaaaaa
<error text>
```

The number nnn and the corresponding <error text> is shown in appendix D.5.2 of the SINTRAN III REFERENCE MANUAL.

THREE, errors may occur in SINTRAN III subsystems. These system programs have their own set of messages. Consult the user manual of the subsystem for documentation of error messages.

Finally, error messages from Real Time programs are routed to the error device (normally the console terminal.) This also applies to input/output messages from the batch output device.

2.6. Listing Information about the System

At this point you should be familiar with how to write SINTRAN III commands. To exercise your skill, we can try out some commands which list information about the system. The first group lists information about the other users. Each command below is explained with its definition and an example.

@LIST-USERS <user name>,<output file>
@LIST-USERS,,,

The example lists all users recognised in the system (as defined in default directory). Listing is done on TERMINAL. You will get a list similar to the one below.

```
USER 0 : PACK-ONE:SYSTEM
USER 1 : PACK-ONE:GUEST
USER 2 : PACK-ONE:TPS-SYSGEN
USER 3 : PACK-ONE:HAP-SB
USER 4 : PACK-ONE:BACK-SIBAS
USER 5 : PACK-ONE:ACEM-BOH
USER 6 : PACK-ONE:KIRSTI-HOLLOKKEN
USER 7 : PACK-ONE:LEITA-VOLD
USER 8 : PACK-ONE:PRISCILLA-HAUGE
USER 9 : PACK-ONE:JERRY-HAMILTON
USER 10 : PACK-ONE:LASSE-LEIRO
```

@WHO-IS-ON

This command will show which of the users have logged on the system. It will be a subset of the previous list and may look as follows,

```
 1  SYSTEM
35  LASSE-LEIRO
48  GUEST
```

The first column shows the logical device number (see sec. 3.) of the terminal where the user is logged in.

@TERMINAL-STATUS <logical dev. no.>,<interval>

```
@TERMINAL-STATUS,,10
LOG.NO  USER            MODE      CPU-MIN OUT CF  LAST COMMAND
     1  SYSTEM          COMMAND         0     15  SYS
    35  LASSE-LEIRO     USER            0     50  CC <TED> CC
    48  GUEST           COMMAND         4     54  FTN
```

It is possible to get a periodic report of the activity on one or all terminals logged in. The example produces a report on all terminals every 10 seconds. The reporting will continue until you press ESCAPE to get back to command mode.

The last set of commands is used to check the time.

@DATCL

The command gives you the current date and time of day.

@TIME-USED

The command lists the amount of time you have used the CPU and your connect time (the duration of time since you logged in).

2.7. @LOGOUT

When you want to finish the terminal session, you log out from the system by giving the command,

```
@LOGOUT                                    (or just @LOG)
10.09.18      28 APRIL    1979
--EXIT--
```

If the accounting system is running, the "TIME-USED" information is also displayed. At this point the terminal session is finished.

It is a good habit to always log out, if you are leaving the terminal for a while.

2.8. @CHANGE-PASSWORD

@CHANGE-PASSWORD <old password>,<new password>
In order to preserve privacy of your own programs and data, the password can be changed at any time. If you do not want to have the password displayed on the screen, enter the command as multiple lines. For ex.,

```
@CHANGE-PASSWORD
OLD PASSWORD:          (old password is entered and is not echoed)
NEW PASSWORD:          (new password is entered and is not echoed)
@
```

2.9. Dialed up Terminals

Some users access the system through dialed up telephone lines. These users must establish phone connection with the computer in addition to the normal log in procedure. When logging out the phone must be disconnected.

Most terminals work in the full duplex mode. This implies that you can input to the computer at the same time the computer is typing back, like you can when you are using a normal terminal connection.

There are two types of couplings at the terminal site, acoustical coupling and wired coupling.

2.9.1. Acoustical Coupling

A terminal with this type of coupling has two rubber cups where the telephone receiver is mounted. Logging in proceeds as follows.

1. Turn the terminal on.

2. Dial the computer to get the high pitch tone.

3. Mount receiver in acoustical coupler.

4. Type ESCAPE to start the normal log in procedure.

The log out procedure is as follows.

1. Type the @LOGOUT command.

2. Put the receiver back on the hook to disconnect the line.

2.9.2. Wired Coupling

A terminal with this type of coupling is wired to the telephone modem. Logging in proceeds as follows.

1. Turn the terminal on.

2. Turn switch on modem to "PHONE" position.

3. Dial the computer to get the high pitch tone.

4. Turn switch to "DATA" position and put receiver back on the hook.

5. Type ESCAPE to start the normal log in procedure.

The log out procedure is as follows.

1. Type the @LOGOUT command.

2. Turn switch to "PHONE" position.

3. Lift the receiver momentarily from the hook to ensure that the line is disconnected.

2.10. User Programs in SINTRAN III

2.10.1. Monitor Calls from FORTRAN Programs

The previous sections have demonstrated the various commands the user can enter at the terminal. When running programs, it is possible to use the facilities of SINTRAN III by means of monitor calls. In FORTRAN the call is made to a subroutine or function. The routine then checks the parameters and makes the corresponding monitor call in machine code.

A small program to demonstrate the use of FORTRAN monitor call will be shown here. It gets the time used since you logged in by calling TUSED. First we will use QED to enter the symbolic code.

For explanation of QED, see the manuals SINTRAN III INTRODUCTION or QED USER'S MANUAL.

```
@QED
QED 4.2
*A
          DOUBLE INTEGER TD,TDD,TUSED
          TD=TUSED(I)
          TDD=TD/50
          WRITE(1,10)TDD
10        FORMAT(' TIME USED IS ',I5,' SECONDS')
          STOP
          END                              (Type RETURN and CTRL/L)
*W "TEST"
64 WORDS WRITTEN
@
```

The program file created is TEST:SYMB. It can now be compiled by the FORTRAN compiler.

```
@FTN
NORD 10 FORTRAN COMPILER FTN-2090F
$COM TEST,,"TEST"
7 STATEMENTS COMPILED OCTAL SIZE = 130
CPU-TIME USED IS 0.1 SEC.
$EX
@
```

The input to the compiler was the file TEST:SYMB and a new file of object code, TEST:BRF, was created. The program is ready to be loaded and run by the NRL, Nord Relocating Loader.

```
@NRL
RELOCATING LOADER LDR-1935E
*LOAD TEST,FTNLIB
FREE : 026527-177777
*RUN

TIME USED IS     3 SECONDS

013740 STOP 0
@
```

2.10.2. Monitor Calls from MAC Assembly Programs

Monitor calls are made directly in MAC programs by using the MON n instruction, where n is the monitor call number. We will make a program similar to the one in the previous section. The monitor call for TUSED is MON 114. IOUT (MON 35) will be used to print the number.

In order to enter the program we will make use of the direct program entry feature of MAC. The program can be entered directly without creating a symbolic code, as was demonstrated in "SINTRAN III INTRODUCTION".

```
@MAC

- MAC -
10/054100 _                set location counter to 10
MON 114                    TUSED
SAT 62                     divide AD by 50
RDIV ST                       ...
SAT 12                     print in decimal
MON 35                        ...
MON 0                      exit to command mode
10!       2                start execution in loc. 10
@
```

In this case, we have used 2 seconds of CPU time. The result can be verified by using the command corresponding to TUSED,

```
@TIME-USED

TIME USED IS    2 SECS OUT OF    9 MINS   44 SECS

@
```

2.10.3. Monitor Call Formats

The monitor calls described in the previous two sections are all available from time-sharing or batch programs. Some calls are only available from RT programs. Refer to the SINTRAN III REFERENCE MANUAL for the availability of each monitor call.

In general, passing parameters in MAC monitor calls is done in two different ways,

ONE, through hardware registers. Normally the T, A, and X registers are used. The input parameters must be loaded by the user program before the call is made. An example is INBT (MON 1),

```
SAT  1          %device no. is 1
MON  1          %INBT - read one character
JMP  ERR        %error return, A = error no.
STA  CHR        %skip return, A = character in bits 7-0
```

TWO, by using a standard call format. The A register points to a parameter list containing the addresses of the values of each parameter. An example is CLOCK (MON 113),

```
      LDA  (PAR .        %A reg. points to parameter list
      MON  113           %CLOCK - read date and time
      ...                % next instruction
)FILL
PAR,  ARRAY              %pointer to 7 word array
      ...
ARRAY,0                  %basic units
      0                  %seconds
      0                  %minutes
      0                  %hours
      0                  %day
      0                  %month
      0                  %year
```

Most monitor calls use the two following instructions as return points, the first if an error occurs, the second when the function is performed correctly. In case of the error return, the A register usually contains the file system error number. Appendix D.2.1 of the SINTRAN III REFERENCE MANUAL lists the different errors.

When executing a MON instruction within a user program, an interrupt is generated, transfering control to a system routine on interrupt level 14. This routine activates the required function on a lower interrupt level (1, 3, 4, or 5) on behalf of the calling program.

2.10.4. Aborting a User Program

As mentioned in section 2.3, a user program or subsystem can be aborted by pressing the ESCAPE key. The terminal will then display the message

USER BREAK AT aaaaaa

where aaaaaa is the address to be executed next after the point of interruption. All opened files will be closed, except the ones which are permanently opened (Sec. 3.8.2).

Two commands are useful after aborting a program,

@STATUS

Display the contents of the program registers at the point of interruption.

@CONTINUE

Restart the program at the restart address if any (Sec. 4.2.2).

@GOTO-USER <address>

Restart the program at <address>. Default address is the next one (aaaaaa, as shown above). If the program uses files when it is interrupted, it will help to set them permanently opened before the program is started the first time. They will then still be open when the program is restarted.

3. The File System

3.1. General

The SINTRAN III File System is designed to manipulate files on disks, magnetic tapes, floppy-disks, standard peripherals, e.t.c. A "file" in this context is a collection of records or blocks, ordered randomly or sequentially. The file system is designed to operate as part of SINTRAN III.

Each file in the file system is named with a character string, and these strings are used in all commands to the file system. The file has a owner, which must be one of the users of the file system. Each user may have several of the other users as friends. The file system provides individual protection of files, with seperate protection modes for the owner, the owner's friends and the public access to the file.

The structure of the physical devices connected to a NCRD computer is shown in fig. 3-1. The lowest element of the hierarchy is a device or a data storage medium. (Note that this is just a logical organization of devices by functional category as opposed to how they are physically interconnected.) A peripheral file name is associated with each of the character devices. A directory is associated with each of the mass storage media. The data area of a directory is divided into user areas and each area will contain a set of mass storage files (see fig. 3-2). A file name may thus be either a peripheral file name or a mass storage file name. For example, if you want to copy a file to OLE (a mass storage file) from TAPE-READER (a peripheral file) you just write,

```
@COPY OLE,TAPE-READER
```

If you want to copy between two mass storage files, you can use the same command,

```
@COPY OLE,PER
```

The command copies "to mass storage file CLE from the mass storage file PER. The system will itself make the distinction between the two types of files.

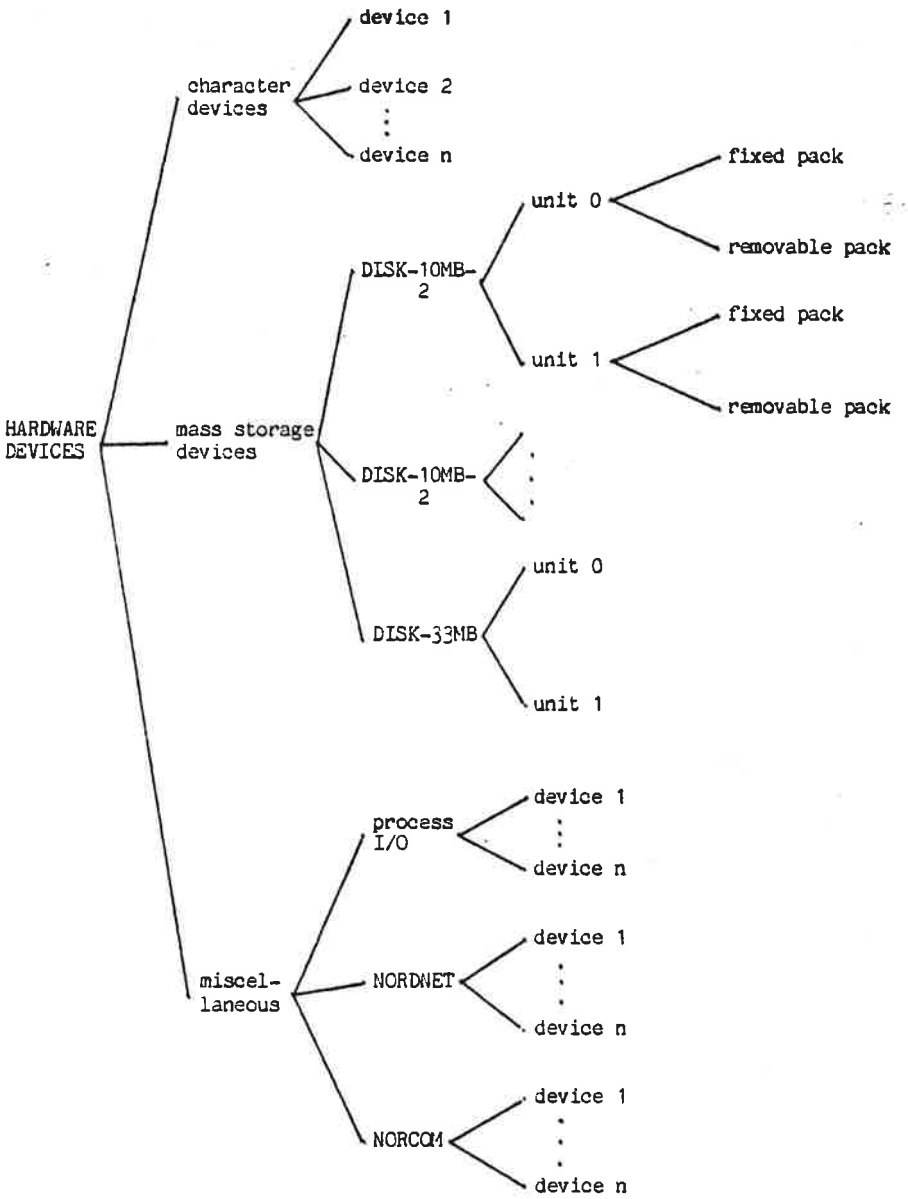


Fig. 3-1 Categories of Hardware Devices

The user of the file system may treat files on disks and magnetic tapes in an uniform manner. The storage unit on disk and tape is always 1024 words (2K bytes), but the user may in most cases address a file with any other block size.

When a file is opened for data access it must first be connected to a number. When the file is accessed for data manipulation, the file number is used. The number is then used in the access routines. In commands and MAC programs we use the logical device number (LDN). In FORTRAN programs we use FORTRAN unit number (unit no.).

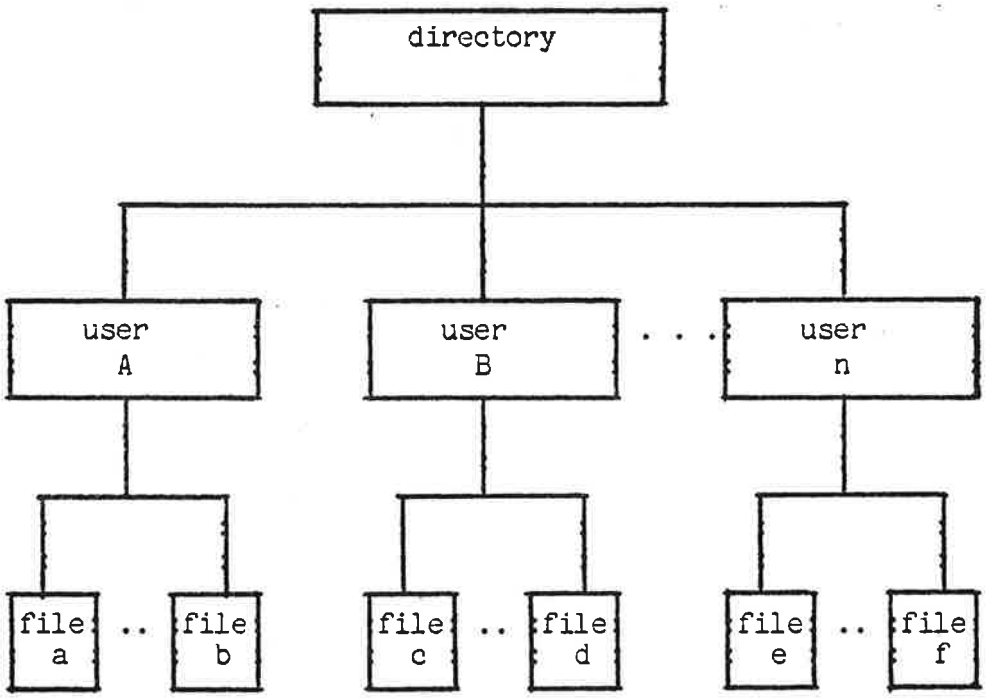


Fig. 3-2 Data Organisation on a Mass Storage Medium.

3.2. Files

3.2.1. General

A file is a collection of data under a given name. The term file in SINTRAN III applies both to mass storage files (fig. 3-2) and peripheral files. A mass storage file may be OLE:DATA. A peripheral file may be LINE-PRINTER, the name associated with the line-printer peripheral. The smallest unit of readable data in a file is a character or byte which is always 8 bits. In mass storage files the bytes are grouped into equal sized blocks. The normal block size is 512 decimal bytes (or 256 words). The block consept is only used for direct addressing of data within the file.

The allocation of data on a mass storage device is in terms of 1K word pages. Blocks and pages are independent collections of the same data. Random reading and writing of data to a mass storage file (see below) are made in terms of extents. An extent is a collection of bytes where the length is determined by the I/O command or monitor call. FORTRAN I/O is transmitted in terms of records. The length of a record is data-defined.

Mass storage files are of two types. The type mostly used is the indexed file. Each file has its 1K word pages scattered around the mass storage device. It also has an index with pointers pointing to where the pages are located. The size of the file may expand dynamically as the user writes onto the file. The index is kept in a separate page belonging to the file so that the file always needs at least one more page than it has data.

Not all pages in a indexed file may be allocated. If no extents or records of a page is written into the index table, then the page is not allocated. The entry in the index table is then said to contain a hole.

The other file type is contiguous file. The pages of the file are then allocated on a contiguous area of the mass storage device. The file has a fixed size during all data access. The advantage of a contiguous file is that data access may be faster since there is no need to look up an index.

The manual SINTRAN III SYSTEM SUPERVISOR (ND-60.103) shows a detail description of the file layout of different devices.

Read and write to files are performed in two ways, by sequential access and by random access. Sequential access implies that read or write starts at the first extent or record and proceeds through the last record in the file. Reading and writing to random extens in the file is called random access. All extents in random files

must be of the same size. Sequential access is normally done on indexed files, while random access is normally done on contiguous files.

3.2.2. File Names

The format of a file name is defined as follows,

(<file directory name>:<owner name>)<object name>:<type>;
<version>

The permissible characters used are numeric for <version>, alphanumeric for <type> and alphanumeric and hyphen (-) for all the others. <file directory name>, <owner name> and <object name> can contain at maximum 16 characers each. <type> can at maximum contain 4 characters. The maximum value of <version number> is 256.

Forunately we do not always have to specify the file name in full. This can best be demonstrated by an example. Assume you want to create an indexed data file in your own user area on the default directory. Type the following,

```
@CRE-FI OLE,,
```

In this case you need only type the <object name>. Default values are used for the rest of the name. Let us now get a list of the full name,

```
@LI-FI OLE,,
FILE 16 : (BIG-PACK:GUEST)OLE:DATA;1
@
```

You will get a message similar to the one above. SINTRAN III finds that you are logged in as user GUEST and that GUEST's default directory is BIG-PACK. File type is DATA and its version number is 1. (If there were more than one version, each one would be listed seperately with its version number.)

Also, the character * will match any character. See section 2.4.3 for further details.

3.2.3. File Types

The purpose of file type is to make it possible to have several different types of the same information. For example, most programs are created as SYMB files containing symbolic code. When it is compiled, normally a new file is created with the same <object name> but of type BRF (Binary Relocatable Format). An absolute program has the type PROG. Subsystems and commands assume

a default type when the file name is specified.

3.2.4. File Versions

Files may also differ only in the version number. If a file is created in more than one version, then no. 1 will be the last version, no. 2 the previous one, and so on. When a file is opened for read, no. 1 will be accessed. When it is opened for write, the version with the highest no. will be selected. When writing is finished and the file is closed, the numbers will be updated so that the accessed file is version no. 1.

It is also possible to access a specific version of a file by specifying its version no. For ex.,

```
@LI-FI OLE,,
```

will list all versions of file OLE, but,

```
@LI-FI OLE;1,,
```

will list only version no. 1.

3.2.5. Logical Device Number

As mentioned in 3.1, a file name must be connected to a number for data access. A file may be a mass storage file (OLE:SYMB;1) or a peripheral file (LINE-PRINTER:;1). When opening the file from a command or a MAC program, a number is supplied by the file system on output. This number is called the logical device number (LDN). Note that this number is in general different from the FORTRAN unit number as used by a FORTRAN OPEN statement. The Unit no. will be explaind in the next section.

For example, consider the command,

```
@OPEN-FILE OLE,RW
FILE NUMBER IS 000101
@
```

We have opened a mass storage file, and the file system gave us a LND = 101 (octal). We can also open a character device,

```
@OPEN-FILE L-P,W
FILE NUMBER IS 5
@
```

LINE-PRINTER is a peripheral file and has 5 permanently assigned as the LDN.

The number 101 was an example of a dynamic LDN which is assigned on demand as a file is opened. A dynamic file no. was also allocated for L-P above, but only the static LDN was used in the open command. A static LDN is global for the whole system. A dynamic LDN, however, is local to the own user. I.e. if another user opens another file, he may also get the number 101.

The structure of logical devices is shown in fig. 3-3. The two most important main categories are dynamic devices, which are the opened files, and character devices. To the right in the figure is shown how the LDNs are assigned. (See appendix C of SINTRAN III REFERENCE MANUAL for a more detailed list.) Static file numbers are universal for all terminals but dynamic numbers are local to your own terminal. (An open file is open only for your own terminal.)

On the bottom of the hierarchy are the individual logical devices, each device being a device element. For the character device TERMINAL, this is terminal 0, terminal 1, or terminal 2, e.t.c. It is important to distinguish element no. from LDN. To each device element of a character device (except terminal 0) there is also a peripheral file name. For example, terminal no. 2 has peripheral file name TERMINAL-2, element no. 2, and LDN = 9. LDN for terminals are described seperately in sec. 3.2.7.

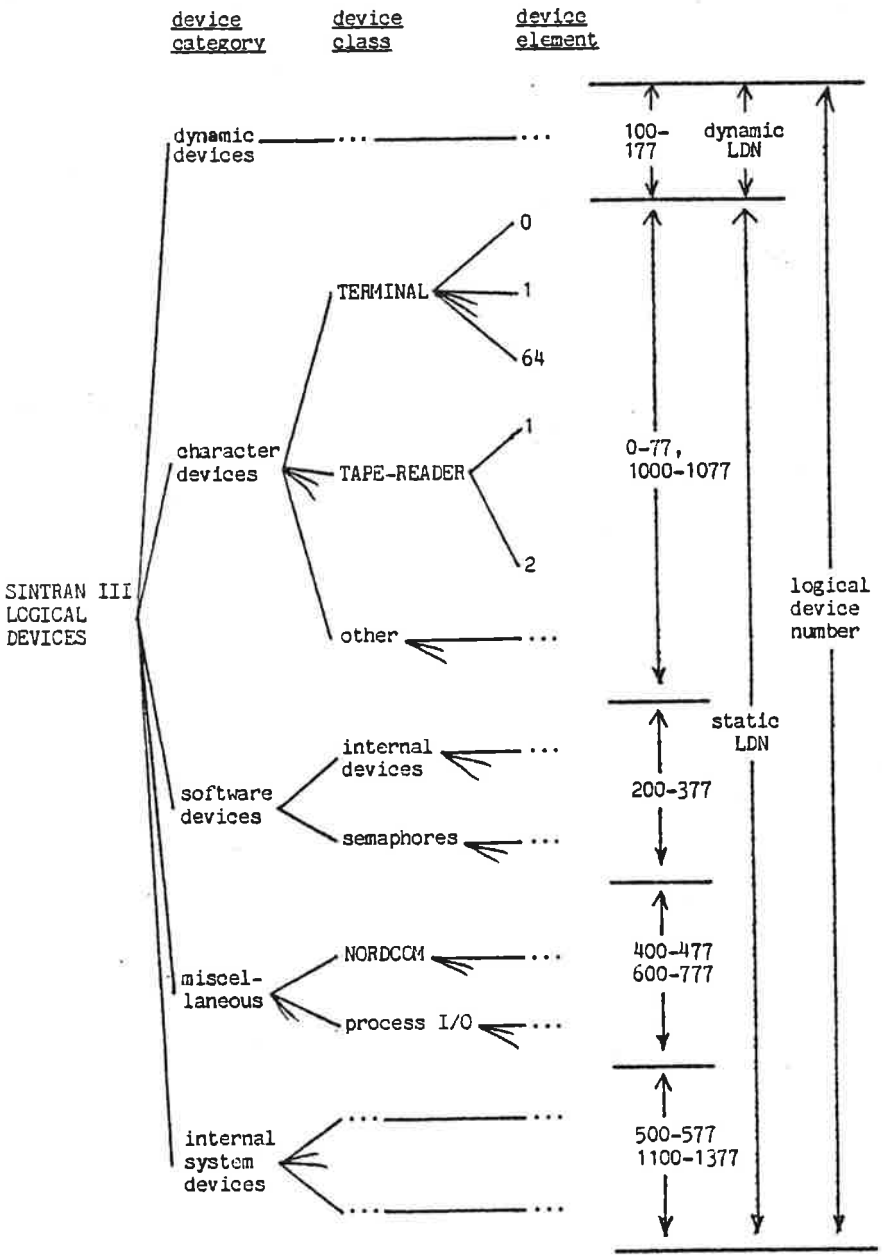


Fig. 3-3 Categories of Logical Devices

3.2.6. FORTRAN Unit Number

The unit no. used in FORTRAN programs is similar to the LDN described in the previous section. In an OPEN statement it is used for associating a value with the file name for later data access. The value is in the range 1 to 99. As opposed to the LDN, a value must be supplied on input to the OPEN statement. (In RT programs, the value is assigned by the OPEN statement.) This value is used throughout the rest of the data access. It is an advantage to use the same value as the LDN whenever possible.

Let us consider two cases of output from a program. First we will write on the line printer. This we can do directly by a statement such as,

```
WRITE(5,10)I,J
```

5 is the standard LDN of the LINE-PRINTER. If 5 is not opened with @OPEN, then 5 will be assumed to be a LDN.

Second, we will write to a file. The file must first be opened with a unit no. Procedure 17 in "SINTRAN III INTRODUCTION" shows an example of opening a file, using the statements,

```
IFIL=65
OPEN(IFIL,FILE='TIME:DATA',ACCESS='W')
```

The file TIME:DATA will now be open for data access with the unit no. 65 (=101 in octal). It can now be written to by a statement such as,

```
WRITE(IFIL,10)TD(7),TD(6),TD(5),TD(4),TD(3),TD(2),TD(1)
```

In general, the terminal user should avoid mixing LDN and unit no.s whenever possible.

3.2.7. Logical Device Numbers for Terminals

From a background program it is possible to write to or read from any terminal except the console terminal (TERMINAL-1). Mostly, we want to communicate with the "own" terminal and LDN = 0 and 1 are used for this purpose.

LDN = 0

This is only permitted for input. It specifies edited input from the own terminal. Edited input means that the terminal user may use the control characters.

LDN = 1

This is permitted on both input and output. It specifies that all characters are input from or output to the own terminal (except ESCAPE on input which terminates the program).

LDN = 9, 34, 35, e.t.c.

This specifies input from or output to TERMINAL-2, TERMINAL-3, TERMINAL-4, e.t.c. (Device element no. 2, 3, 4, e.t.c.) Appendix C of SINTRAN III REFERENCE MANUAL lists the LDN for all terminals.

3.2.8. Terminal Types

A large number of different terminals are used with the NORD computers. They are either display terminals (VDT units) or hard copy terminals from different manufacturers. To make it possible to distinguish between the different types, SINTRAN III will store a terminal type number as part of the information about a LDN, provided it is a terminal. A standard terminal type name and corresponding number for each type is shown in appendix B of the REFERENCE MANUAL. If a terminal is sold as part of the system it is also listed with its ND-number (supplied by ND marketing).

If a user wants to standardize a particular terminal, he should contact his ND representative to get it into the list. The terminal type name is not stored in the operating system but the standard is used by some of the SINTRAN III subsystems.

Two commands are used to manage the terminal type, @SET-TERMINAL-TYPE and @GET-TERMINAL-TYPE. The corresponding monitor calls are MSTTY (MON 17) and MGTTY (MON 16). They are explained in section 5.3.

3.2.9. Scratch Files

Each terminal (with a background program) has been assigned a scratch file. The file is indexed and is kept permanently open while the terminal user is logged in. When the user logs out the file is reduced to a maximum of 32 pages. LDN = 100 is permanently assigned to this file.

The scratch file may be used for temporary data on mass storage. Some subsystems use this file and it is then not available for the terminal user. However, the terminal user can specify 100 as BRF output from the FORTRAN compiler. The same number is then specified as input to the Nord Relocating Loader.

3.3. File Directories

3.3.1. General

A directory is a set of files and has the structure as shown in fig. 3-2. There may only be one directory per mass storage medium. Mass storage may be hard disks or floppy-disks. Each directory may contain files for one or several users. If the directory is contained on a removable medium, it may be moved to other installations and used there. Each mass storage medium to be used as a directory must be created with a name and entered before it can be used by the system. When a directory is no longer needed, it may be relesed and physically dismounted from the system. The next time it is needed, it must be mounted and entered again.

The first directory entered containing SINTRAN III (and related subsystems) is regarded as the main directory. This directory can not be released.

Any directory in the system can be a default directory. If a user leaves out the directory name in a file name, the default directory where the user has space is refered to. A user must have space in only one default directory.

The system supervisor (user SYSTEM) is responsible for entering the main and default directories, setting them up as default, creating the users and allocating space to the different users. Time-sharing users can enter and release all other directories. Time-sharing users can also create and administrate floppy-disks.

3.3.2. Directory Commands

The following commands are used to establish and maintain directories. They are all generally available for time-sharing users, except when noted otherwise.

3.3.2.1. @CREATE-DIRECORY

@CREATE-DIRECTORY <dir. name>,<device name>,<unit>,<bit-file address>

Establish a new directory on a mass storage medium. The only device permitted for time-sharing users is floppy-disk. <device name> must then be FLOPPY-DISK-1 or FLOPPY-DISK-2. Usually the user chooses the default value of <bit-file address>. The location will then be choosen by the system. When allocating large contiguous files on a floppy-disk the bit-file should be allocated at the end of the disk. The value of the parameter may then be set to 150.

For ex.

@CRE-DIR FLOPPY-1,F-D-1,0

Create a new directory in FLOPPY-DISK-1, unit 0. It is given the name FLOPPY-1. Section 3.12 shows how to initiate a floppy-disk.

3.3.2.2. @ENTER-DIRECTORY

@ENTER-DIRECTORY <dir. name>,<device name>,<unit>

Make a directory available to the system. The parameters are the same as for @CREATE-DIRECTORY. For ex.

@ENTER-DIRECTORY F-1,F-D-1,0

Make the directory on FLOPPY-DISK-1 known to the system. It must have the name F-1.

3.3.2.3. @CREATE-USER

@CREATE-USER <dir. name>:<user name>

Enter a new user name on the specified directory. For time-sharing users the directory must be on a floppy-disk. The user name must already be known in the main directory. For ex.

@CRE-US F-1:GUEST

Create the new user named GUEST on directory F-1.

3.3.2.4. @GIVE-USER-SPACE

@GIVE-USER-SPACE <dir. name>:<user name>,<no. of pages>

Reserve the no. of pages for the given user and directory. For timesharing users the directory must be on a floppy-disk. The maximum number of pages on a floppy-disk is 148. For ex.

@G-U-S F-1:GUEST,148

3.3.2.5. @RELEASE-DIRECTORY

@RELEASE-DIRECTORY <dir. name>

Make the specified directory unavailable for the system. It may now be physically removed from the system. For ex.

@REL-DIR F-1

3.4. File Creation and Deletion

"SINTRAN III INTRODUCTION" showed examples of implicit file creation and explicit file creation. An example of an implicit file creation is the command

*W "AAA"

in the subsystem QED. This command creates the file AAA:SYMB and writes the text-buffer to this file. Files created in this way are always indexed files. Files can be created implicitly both in subsystems and by commands. An example of an explicit creation of a file is shown in Procedure 15 by the usage of the command,

3.4.1. @CREATE-FILE

@CREATE-FILE <file name>,<no. of pages>

Create a new file with the given name and number of pages. If <no. of pages> is zero, an empty indexed file will be created. If it is greater than zero, an empty contiguous file will be created with the indicated <no. of pages>. If the file name contains a version number, it specifies the no. of versions to be created. For ex.,

@CRE-FI FILE-FOUR,10

Create the file FILE-FOUR:DATA as an contiguous file with 10 pages (20K bytes). Note that quotes are not used here.

3.4.2. @EXPAND-FILE

@EXPAND-FILE <file name>,<no. of pages>

This command must be used in order to expand a contiguous file. The parameter specifies the additional number of pages to be added to the file. For ex.:

@EXPAND-FILE FILE-FOUR,9

FILE-FOUR:DATA is expanded by nine pages (provided the contiguous space is available).

3.4.3. @CREATE-NEW-VERSION

@CREATE-NEW-VERSION <file name>,<no. of pages>

Create one or more new versions of the specified file and give them the number of pages as indicated. The rule for <no. of pages> is the same as for @CREATE-FILE. The no. of new versions to be created depends on how many versions exist and the version no. on the file name of the command. For ex. assume that there are two versions of the file F-1.

F-1:DATA;1
F-1:DATA;2

If we give the command,

@C-N-V F-1:;4

This command will create new versions up to and including 4 (F-1:DATA;3 and F-1:DATA;4). Assume that we instead had given the command,

@C-N-V F-1:;1,10

The new version we have created (with an empty set of data) would be named as version 1. The previous version 1 would get the number 2 and the old version 2 would get the number 3. New version of a file must always be created by this command.

3.4.4. @ALLOCATE-FILE

@ALLOCATE-FILE <file name>,<page address>,<no. of pages>

In the previous commands we have let the file system decide where to physically locate the file. It is possible to determine the location of a file by @ALLOCATE-FILE. This command is similar to @CREATE-FILE, but the file will be located at the <page address>. In order for the file to be created, the area in the directory must not be in use. It is also possible to allocate more versions of the file, in which case they will be located one after the other in the directory. For ex.,

@ALL-FI FILE-THREE,100,8

A contiguous file, FILE-THREE:DATA containing 8 pages, is allocated in default directory starting at page 100 (octal).

3.4.5. @ALLOCATE-NEW-VERSIONS

@ALLOCATE-NEW-VERSIONS <file name>,<page address>,<no. of pages>

This command is similar to @CREATE-NEW-VERSION, but instead one or more new versions are created with the first version starting at <page address>. For ex.,

@ALL-N-V FILE-THREE;2,140,8

Version 2 of the contiguous file FILE-THREE:DATA is created starting at location 140. The file size is 8 pages.

3.4.6. @RENAME-FILE

@RENAME-FILE <old file name>, <new object name> :<new type>

Change object name and/or type of a file. For ex.,

@RE-FI (PACK-TWO:GUEST)F-1:SRC,:SYMB

Change the type of the file from SRC to SYMB.

3.4.7. @DELETE-FILE

@DELETE-FILE <file name>

Delete a single file. The full name must be given. If a version no. is given, only that version is deleted. if not, all versions will be deleted. The corresponding monitor call MDLFI (MON 54) is explained at the end of this section. For ex.,

@DEL-FI F-1:DATA

Delete all versions of the file F-1:DATA.

3.4.8. @DELETE-USERS-FILES

@DELETE-USERS-FILES <file name>,<MANUAL CHECK?>

Delete all files matching <file name>. If the last parameter is YES each file name will be printed out. The user can then decide whether the file should be deleted or not. If NO, all matching files will be deleted at once. For ex.,

@D-U-F ,NC

Delete immediately all files in the users default directory.

3.4.9. @SET-TEMPORARY-FILE

@SET-TEMPORARY-FILE <file name>

Define the contents of a file as temporary. The pages of the file will be deleted after it has been read from. The commmand should be given immediately after the file is created. This command is mostly used for spooling files. For ex.,

@S-TEM-F L-P;2

Declare the file L-P, version 2 to be a temporary file.

Two commands related to file creation are declared as system commands. They are @SET-TERMINAL-FILE and @SET-PERIPHERAL-FILE. They will be found in the manual SINTRAN III SYSTEM SUPERVISOR.

3.4.10. MDLFI (MON 54)

Delete a single file. The same rule applies to the file name as for @DELETE-FILE. For ex.,

```
      LDX  (FILE            % X REG POINTS TO FILE NAME STRING
      MON  54               % MDLFI
      MON  64               % ERMSG ON ERRCR RETURN
      ...                   % NORMAL RETURN
FILE, 'FILE-THREE:DATA'
```

Delete the file FILE-THREE:DATA.

3.5. File Access and Reservation

3.5.1. User Types

A file is accessed for read, write, append (i.e. expanding the file), common (i.e. more than one user having access at the same time), random or indexed, and directory modification.

Access is determined by what kind of access you request, what user category you belong to and what kind of access the owner of the file has permitted for your category. The users of a file are divided into three categories,

CWN: The owner of the file. Normal default access type permitted for the own user is all access.

FRIENDS: A group of users associated with each user. (They are friends relative to a users own files.) Normal default access types permitted are read, write, and append (expanding the file).

PUBLIC: All other users. The only normal default access type permitted is read.

You get the default access type associated with a file when the file is created (For ex. @CREATE-FILE) The term "normal default" signifies the default when the system is started. It can be changed later on by the command @SET-DEFAULT-FILE-ACCESS (to be explained below).

The protection around a file is illustrated in fig. 3-4. It is normally organised such that the own user has the most access types while public has the least.

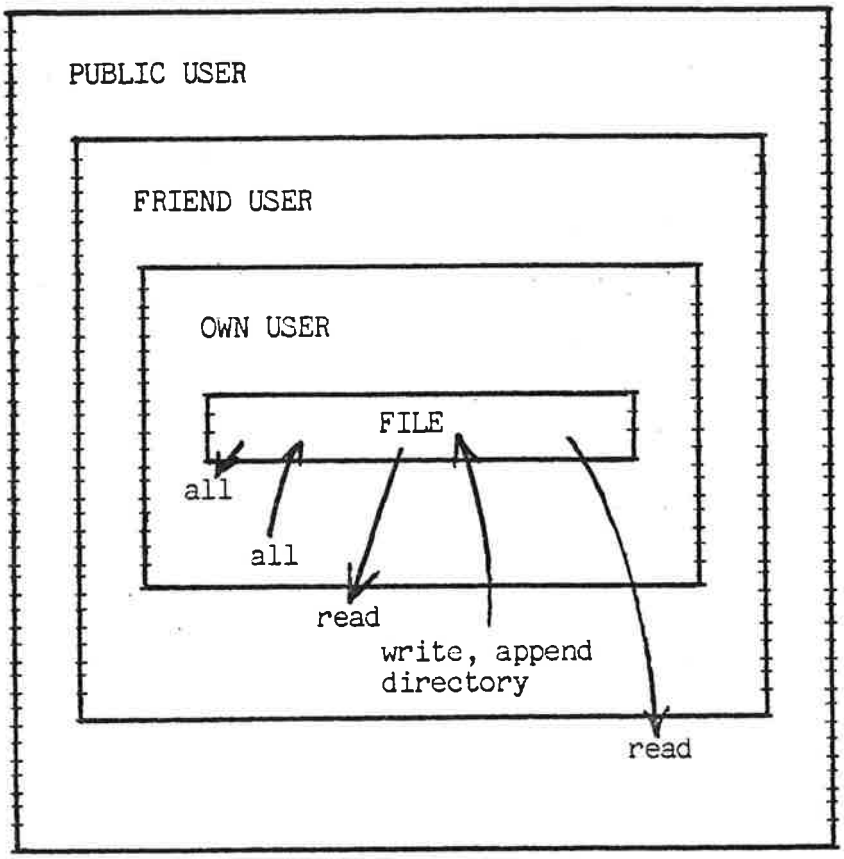


Fig. 3-4 Normal File Protection

3.5.2. Permitting Access to Files

The permission is stored in the directory as part of the information about each file. In addition, for each user there is a special list of permissions for each of his friends. This is shown in fig. 3-5. For the own and public requests, only the permission stored with the file is checked. For friend access, the permission for the friend is checked in addition to the permission stored with the file. The access is granted only if it is permitted in both lists.

We will first look at commands that set the permitted access for individual files.

3.5.3. @SET-FILE-ACCESS

@SET-FILE-ACCESS <file name>,<public access>,<friend access>,

<own access>

Set the access mode for a specific file. The access specification is any relevant combination of,

R - read permitted

W - write permitted

A - append permitted

C - common access permitted

D - directory access permitted (the file may be deleted,legal access mode changed and new versions created.)

N - no access permitted

For ex.,

@S-FI-ACC F-1:DATA N,RW,RWACD

Public users are not permitted to access the file. Friends have read and write access. The own user has all types of access. The result of the command can be checked by the command

@FILE-STATISTICS f-1:DATA

3.5.4. @SET-DEFAULT-FILE-ACCESS

@SET-DEFAULT-FILE-ACCESS <public access>,<friend access>,
<own access>

Change the default file access for the own user. The default file access is the values the file will get when it is opened. The values of default file access when the system is started was mentioned in the previous section ("normal default"). For ex.,

@S-D-F-A N,,,

The default file access of public is changed to no access.

3.5.5. @CREATE-FRIEND

@CEATE-FRIEND <user name>

Create a user as a friend. Note that a user can only declare other users as friends. He can not declare himself as friend of other

users. Thus, the friendship may not be mutual. The friend will now normally get permission to read, write, and append to your file (provided also that the particular file permits this for friends). For ex.,

@CRE-FR GUEST

Declare the user GUEST as your friend.

3.5.6. @DELETE-FRIEND

@DELETE-FRIEND <user name>

Remove the user name from the list of your friends. He will now be a public user. For ex.,

@DEL-FR GUEST

The user GUEST is removed from the list of your friends.

3.5.7. @SET-FRIEND-ACCESS

@SET-FRIEND-ACCESS <user name>,<access type>

Specify access from a friend to the own users files. Fig. 3-5 shows how this access permission is used together with the file permission. For ex.,

Friend access for file F-1:DATA is WA.

@S-FR-A GUEST,RW

Access to file F-1:DATA for user GUEST is now only W (It may only be opened with write access.)

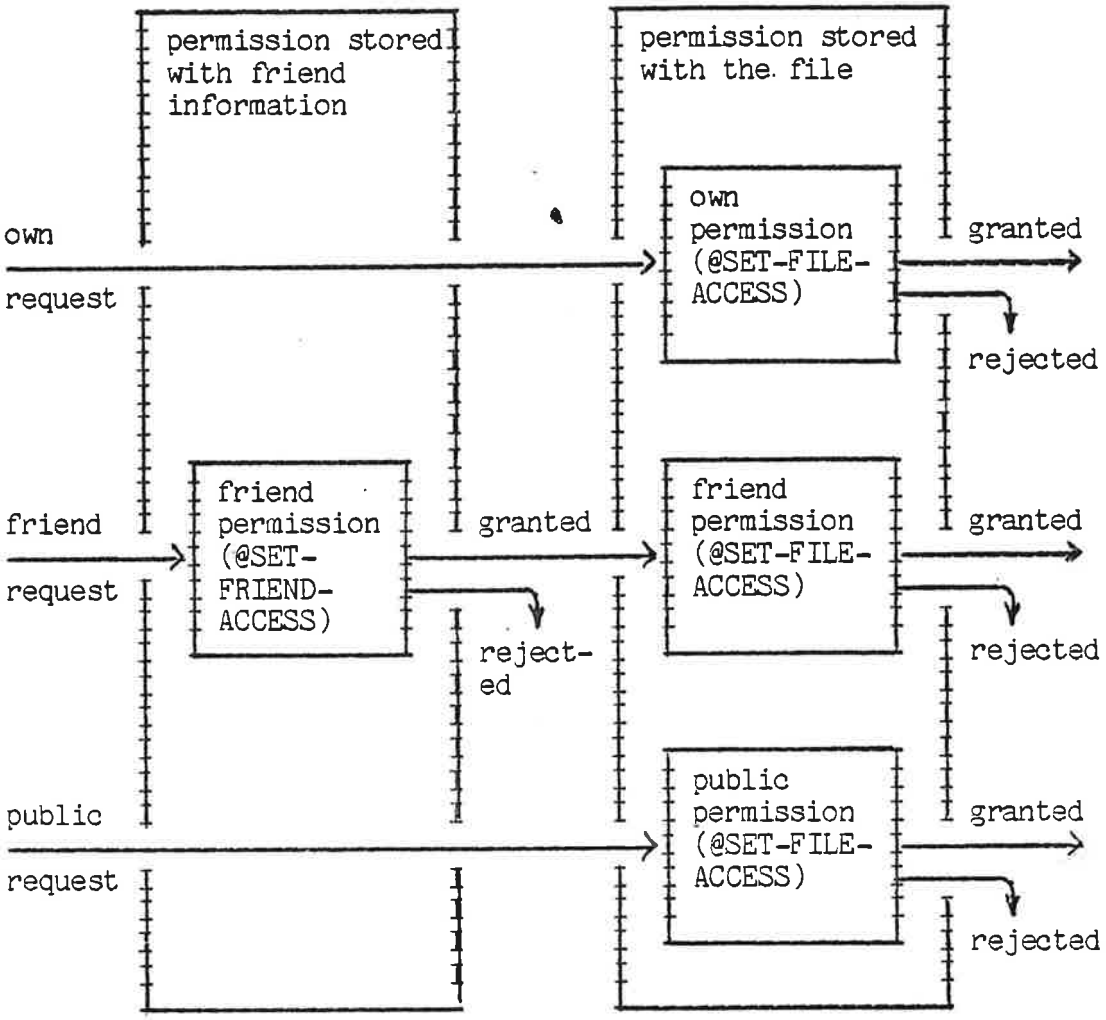


Fig. 3-5 Granting Access to Files.

3.6. Requesting Access to Files

Files are accessed both from commands and monitor calls. The access is only accepted if the corresponding file (and friend) access is set properly, as shown in the previous section. There are two ways of requesting access,

Explicit access mode applies both to commands and monitor calls. It means that you specify by a character combination what type of access you want. The characters are,

R - read request

W - write request

X - random access request

A - append request

C - common request (only for contiguous files)

D - direct transfer (only for foreground programs)

An example is @OPEN-FILE, FORTRAN OPEN or OPEN (MON 50).

@OPEN-FILE OLE,RX

means that the file is opened for random read only. In FORTRAN a similar statement is,

```
OPEN(IFIL,FILE='OLE',ACCESS=RX)
```

The file OLE is opened for random write only.

Implicit access mode applies only to commands. It means that the access mode is implied by the command. for ex.,

@COPY-FILE (SYSTEM)OLE,(SYSTEM)PER

means that you are requesting read access to (SYSTEM)PER:DATA and write access to (SYSTEM)OLE:DATA.

3.7. Reserving and Releasing Files

The previous sections showed how common access to a file was restricted when the file was opened. It is also possible to reserve the access to a file for the users terminal. (Note that it is the terminal and not the user name.) From commands it is possible to reserve peripheral files by name and whole mass storage devices. From monitor calls it is possible to reserve all static LDN (see fig. 3-3).

It is not possible to reserve mass storage files and dynamic LDNs.

3.7.1. @RESERVE-FILE

@RESERVE-FILE <peripheral file name>

Reserve a peripheral for the exclusive use of the own terminal. The corresponding monitor call RESRV (MON 122) is explained at the end of this section. For ex.,

@RES-FI L-P

The peripheral file L-P (LINE-PRINTER) is reserved for the

exclusive use of the own terminal.

3.7.2. @RELEASE-FILE

@RELEASE-FILE <peripheral file name>

Permit a peripheral to be used from other terminals.

3.7.3. @RESERVE-DEVICE-UNIT

@RESERVE-DEVICE-UNIT <device name>,<unit>,<'F' or 'R'>,<subunit>

Reserve part of a device for special use, such that the directory on the device can not be entered. For time-sharing users this should be restricted to FLOPPY-DISK. For ex.,

@R-D-U F-D-1 0

Do not allow FLOPPY-DISK-1, unit 0 to be used from other terminals.

3.7.4. @RELEASE-DEVICE-UNIT

@RELEASE-DEVICE-UNIT <device name>,<unit>,<'F' or 'R'>,<subunit>

This is the release command corresponding to the reserve command above.

The monitor calls for reserving and releasing logical devices are RESRV and RELES.

3.7.5. RESRV (MON 122)

```
INTEGER RESRV
ISTAT=RESRV(<LDN>,<read/write>,<wait-flag>)
```

Reserve a static LDN for the exclusive use of this program. If <read/write> is zero, then the input part is to be reserved, if one, the output part is reserved. <return flag> specifies the return strategy to be used if the LDN is already reserved. If zero, the program is to be set in a waiting state. If one, the call will return with a function value of -1. Successful reservation always gives a function value of zero.

If the user program exits without releasing the LDN, then it will not be released until the user logs out. (User RT and SYSTEM can

always force it to be released by the command @PRLS.)

For ex.,

```
INTEGER RESRV
ISTAT=RESRV(5,1,0)
```

Reserve the output part of LDN = 5 (usually LINE-PRINTER). If the device is already reserved the program will be set in a waiting state. It will be started again when the device is released.

3.7.6. RELES (MON 123)

```
CALL RELES(<LDN>,<read/write>)
```

This is the release routine corresponding to RESRV. A time-sharing user can only release a LDN from the terminal where it was originally reserved.

For RT-programming there are two commands for reserving and releasing, @PRSRV and @PRLS. They will force the reserving or releasing independently of its current state.

In RT-programs the monitor call WHDEV (MON 140) can be used to find out who has reserved a LDN.

3.8. Reading and Writing Data

3.8.1. General

This section describes commands and monitor calls for transmitting data between files and the user's memory. Mass storage files must always be opened before data can be accessed. Peripheral files do not have to be opened.

Most of the functions are monitor calls since reading and writing data is mainly done by programs. Some of the calls have corresponding commands (for ex.OPEN and @OPEN-FILE).

3.8.2. Opening Files

The most used command and monitor call is @OPEN-FILE and OPEN (MON 50). These functions use LDN (file number). In the FORTRAN OPEN statement we use the unit no. as was explained in sec. 3.2.6. The other functions are commands to open a file with a specific LDN, open it as a scratch file, and to set a file permanently open.

RT users have two special commands for opening a file for RT-programs. They are @RTOPEN-FILE and @RTCONNECT-FILE.

3.8.2.1. @OPEN-FILE

@OPEN-FILE <file name>,<access type>

Make a file available for data access and return the LDN to be used for accessing the file. The <access type> requested is checked with the legal access types for the file, as described in sec. 3.5.3. Only certain combinations are legal. The file can be created with the @OPEN-FILE command. The corresponding monitor call is explained at the end of this section. For ex.,

```
@OPEN-FI "PERSONNEL-1",WA
FILE NUMBER IS 000101
@
```

Create and open the file PERSONNEL-1:DATA for write and append access. The file is given the number 101 (octal)

```
@OPEN-FI (SYSTEM)OLE:SYMB,RW

NOT READ AND WRITE ACCESS
@
```

Open the existing file OLE:SYMB for read and write access. The command was not successful due to access restrictions.

3.8.2.2. @CONNECT-FILE

@CONNECT-FILE <file name>,<file no.>,<access type>

Open a file with a predetermined dynamic LDN. The call is similar to @OPEN-FILE except that the user specifies the file no. For ex.,

```
@CON-FI OLE,110,RW
@
```

Open OLE:DATA for sequential read and write access. Its file no. will be 110 (octal).

3.8.2.3. @SCRATCH-OPEN

@SCRATCH-OPEN <file name>,<access type>

Open the file as a scratch file. Such a file is only closed when logging out, or by the command @CLOSE-FILE -1 which closes all files. When closed, a maximum of only 32 pages are retained in the file. In other respects, the command is similar to @OPEN-FILE. For ex.,

```
@SC-OP TEMP,RWA
FILE NUMBER IS 000104
@
```

Open the file TEMP:DATA as a scratch file for read, write, and append access. The file number 104 is allocated to the file.

3.8.2.4. @SET-PERMANENT-OPENED

@SET-PERMANENT-OPENED <file no.>

The already opened file can not be closed by the command @CLOSE-FILE -1. For ex.,

```
@OPEN-FILE OLE,RW
FILE NUMBER IS 101
@S-P-O 101
   ...
@CLOSE -1
   the file is still open
   ...
@CLOSE 101
   the file 101 is now closed
```

3.8.2.5. OPEN (MON 50)

Open a file for data access. The file name can be read both from the user program and the user terminal. For ex.

```
      LDX  (FILE          % X = addr. to name
      SAT  4              % T = sequential read
                          %           and write
      MON  50             % OPEN
      MON  64             % error return
       ...                % normal return
FILE, 'PER:SYMB'          % file name string
```

Open the file PER:SYMB for sequential read and write.

3.8.3. Sequential Input/Output

These are all monitor calls for user programs. Selecting a call depends on how many characters you want to be read by one call, what type of return strategy you want, e.t.c. There is a corresponding output call for each input call (but not vice versa).

The 8-byte calls (M8INB, M8OUT, B8INB, B4INW and B8OUT) may only be used for character devices and software devices (fig. 3-3) and NORDNET (SINTRAN III SPECIAL I/O GUIDE).

3.8.3.1. INBT (MON 1, FORTRAN call: INCH)

Read one byte from a device. (Read one word if the device is a data link or word oriented device.) For ex.,

```
ICH=INCH(2)
IF(ERRCODE.NE.0)GO TO 1000
```

Read one character to ICH from LDN = 2 (TAPE-READER). The read is successful if ERRCODE is zero, if not it will contain the error number.

3.8.3.2. OUTBT (MON 2, FORTRAN call: OUTCH)

This is the output call corresponding to INBT above. For ex.,

```
CALL OUTCH(5,ICHAR)
IF(ERRCODE.NE.0)GO TO 1000
```

Output one character in ICHAR to LDN = 5 (LINE-PRINTER). ERRCODE contains error status on return.

3.8.3.3. M8INB (MON 21)

Read up to 8 bytes from a device. Less bytes may be read if the intenal input buffer contains less than 8 characters. The echo and break strategy of the device apply. For ex.,

```
SAT  2          % T = LDN
MON  21         % M8INB
MON  64         % error return
...             % normal ret., T = no. of
                %                 bytes read
```

Read at maximum 8 bytes from LDN = 2 (an opened file).

3.8.3.4. M8OUT (MON 22)

This is the output call corresponding to M8INB above. At maximum 8 bytes are output. A byte = 0 is not output. For ex.,

```
LDA   DAT12          % A = byte 1 and 2
LDX   DAT34          % D =
COPY  SX DD          %     byte 3 and 4
LDX   DAT56          % L =
COPY  SX DL          %     byte 5 and 6
LDX   DAT78          % X = byte 7 and 8
SAT   3              % T = LDN
MON   22             % M8OUT
MON   64             % error return
...                  % normal return
```

An attempt is made to write 8 bytes to LDN = 3.

3.8.3.5. B8INB (MON 23)

Input up to 8 characters from a device. The call is similar to M8INB except that the echo and break strategy of the device do not apply. Also, this call can not be used on terminals.

3.8.3.6. B4INW (MON 63)

Read 8 bytes from a device. Same call as B8INB but the number of bytes read is always 8.

3.8.3.7. B8OUT (MON 24)

Output 8 characters to a device. The call is similar to M8OUT except that also a character = 0 is output. For ex.,

```
LDA   DAT12          % A = byte 1 and 2
LDX   DAT34          % D =
COPY  SX DD          %     byte 3 and 4
LDX   DAT56          % L =
COPY  SX DL          %     byte 5 and 6
LDX   DAT78          % X = byte 7 and 8
SAT   3              % T = LDN
MON   24             % B8OUT
MON   64             % error return
...                  % normal return
```

All 8 bytes in A, D, L, and X are output to LDN = 3.

3.8.3.8. INSTR (MON 161)

Read a string of characters from a peripheral device. This call is an option when ordering SINTRAN III. For ex.,

```
DIMENSION MTXT(50)
   ...
ISTAT=INSTR(1,MTXT,100,15B)
IF(ISTAT.NE.0)GO TO 1000
```

Read a string of characters from the own terminal to MTXT. Terminate after 100 characters read or after receiving CARRIAGE RETURN (15B), whatever occurs first.

3.8.3.9. OUTST (MON 162)

Write a string of characters to a peripheral device. This call is an option when ordering SINTRAN III. For ex.,

```
INTEGER OUTST
DIMENSION MOUT(50)
   ...
ISTAT=OUTST(1,MOUT,15B)
```

Write a string of characters from MOUT to the own terminal. Terminate after the first CARRIAGE RETURN (15B).

3.8.3.10. MSG (MON 32)

Write a character string on the own terminal. For ex.,

```
      LDX  (TEXT              % X = addr. of character
                              %                 string
      MON  32                 % MSG
      ...                     % normal return
TEXT, 'ERROR IN INPUT DATA'
```

The text ERROR IN INPUT DATA is written on the own terminal.

3.8.3.11. IOUT (MON 35)

Print a number in octal or decimal format on the own terminal. For ex.,

```
      SAT  12                 % print in decimal format
      LDA  (NUMB              % A = number
      MON  35                 % IOUT
      ...                     % normal return
```

The value of NUMB is written in decimal format.

3.8.3.12. NOWT (MON 36)

Set wait mode or no wait mode of I/O calls to character devices.

All character I/O is normally static I/O or in wait mode. That is, the program will wait (enter a wait state) until the transmission is finished. It is possible to change to dynamic I/O or no-wait mode. There will then always be an immediate return from the I/O call. If the character was put in or taken from the I/O buffer the error code is zero. If the buffer is empty on input or full on output there will be an immeditate return with error code =3 (end of file). The call must then be attempted at a later time. HOLD is used to wait for I/O completion. For ex.,

```
      DIMENSION MTXT1(1000),MTXT2(1000)
          ...
C INITIATE OUTPUT TO LINE PRINTER 1 AND 2
      ISTAT=NOWT(5,1,1)
      IF(ISTAT.NE.0)GO TO 999
      ISTAT=NOWT(13,1,1)
      IF(ISTAT.NE.0)GO TO 999
      I=1
      J=1
C
C OUTPUT ARRAY MTXT1 AND MTXT2
100   IF(I.GT.1000)GO TO 200
C
C         OUTPUT TO LINE PRINTER 1
          CALL OUTCH(5,MTXT1(I))
          IF(ERRCODE.EQ.0)I=I+1
          IF(ERRCODE.NE.0.OR.ERRCODE.NE.3)GO TO 999
C
C         OUTPUT TO LINE PRINTER 2
200       IF(J.GT.1000)GO TO 300
          CALL OUTCH(13,MTXT2(I))
          IF(ERRCODE.EQ.0)J=J+1
          IF(ERRCODE.NE.0.OR.ERRCODE.NE.3)GO TO 999
             ...
300       CALL HOLD(1000,4)
          GO TO 100
C
C CONTINUE IF ANY PRINTER HAS MORE DATA
      STOP
C
C ERROR EXIT
999       ...
      STOP
      END
```

The program will output the array MTXT1 containing 1000 characters to line printer 1 and the array MTXT2, also containing 1000 characters, to line printer 2. The first statements set the peripherals in no-wait-mode and initiates indexes to the arrays. The program loop consists of two sets of output statements. The call to OUTCH attempts to put a character in the output buffer.If successful, the index can be incremented. If ERRCODE is 3 then the index is not incremented and we will attempt to output the same character at the next call to OUTCH for the same peripheral. If ERRCODE is neither 0 nor 3 there is an error and we exit through statement 999.

HOLD works in a special way for NO-wait-mode. If a break condition is detected (i.e. a peripheral signaled that a break character has been transmitted), there will be an exit from the call. The call to HOLD above then works as a "wait for break" statement. (The hold time is virtually infinity.) When any of the peripherals generate a break, HOLD will exit and the program will attempt to output the next character or the same character one more time. The output will continue until there is no more data for any device.

3.8.4. Random Input/Output

Random I/O can be made to both indexed and contiguous files. A random file consists of a set of equal sized records. The records are addressed in terms of blocks. Standard block size is 256 words. (The size can be modified by @SET-BLOCK-SIZE.)

The two most important functions for I/O are RFILE and WFILE. They are mostly used as monitor calls, but are also available as commands. One call transmitts a unit of data called extent. It is independent of the block size.

3.8.4.1. @RFILE

@RFILE <file no.>,<memory address>,<block no.>,<no. of words>

Transfer an extent from one or more random file blocks into the user's memory. <no. of words> is the extent length. The number is independent of the block size. Therefore, one or more blocks may be read. For ex.,

@RFILE 101,400,0,1000

(All parameters are in octal.) Read 1000 words from block 0 of file 101 to memory address 400. The block size is 400 (standard), so two blocks are read. Also, a <no. of words> less than the block size may be specified.

3.8.4.2. @WFILE

@WFILE <file no.>,<memory address>,<block no.>,<no. of words>

Transfer an extent from the user's memory to one or more blocks in a random file. <no. of words> is the extent length. For ex.,

@WFILE 101,400,0,1000

(All parameters are in octal.) Write 1000 words from memory address 400 to block 0 of file 101. The block size is 400 (standard), so two blocks are read.

These two commands also exist as monitor calls.

3.8.4.3. RFILE (MON 117)

Read an extent from a file to the user's memory. For ex.,

```
DIMENSION MDATA(512)
   ...
CALL RFILE(101B,0,MDATA,0,512)
```

Read an extent from file 101 to array MDATA. The extent starts in block 0 and is 512 decimal words long. The second parameter specifies that there will be no return from the call until all input is complete. It is the only permissible value for timesharing users.

3.8.4.4. WFILE (MON 120)

Write an extent to a file from the user memory. For ex.,

```
DIMENSION MDATA(512)
   ...
CALL WFILE(101B,0,MDATA,0,512)
```

Write an extent to file 101 from array MDATA. The extent starts in block 0 and is 512 decimal words long. The second parameter must be zero for time-sharing users.

RT-programs may use WAITF (MON 121) to wait for transmission to be completed when the second parameter is not zero.

Four other monitor calls are used for random file access. They are RDISK (MON 5), WDISK (MON 6), RPAGE (MON 7), and WPAGE (MON 10). These calls are only included in SIII in order to be compatible with the old TSS operating system. They should

perferably not be used by "new" users.

3.8.5. Closing Files

Files can be closed both from commands and monitor calls. If a file is open when the program terminates, it will be closed automatically. This applies to both normal and abnormal termination. (The user can avoid this by setting the file permanently open, see next section.)

If the file is opened by a command, it will always be closed when the user logs out.

3.8.5.1. @CLOSE-FILE

@CLOSE-FILE <file no.>

Close one or more files. If <file no.> = -1 then all files not permanently opened are closed. If = -2 then all files are closed. For ex.,

@CLOSE 101

Close the file number 101.

3.8.5.2. CLOSE (MON 43)

The monitor call can close files in a manner similar to the command. For ex.,

```
SAT  -1              % close all files not
                     %    permanently opened
MON  43              % CLOSE
JMP  ERROR           % error return
...                  % normal return
```

This call has a function similar to the command @CLOSE-FILE -1.

3.8.6. Data Access Parameters

This section deals with the parameters used for sequential and random data access of a file. For sequential access there is a byte pointer which points to the next byte to be accessed. The maximum byte pointer is one less than the number of bytes in the file. The block size may be modified and the byte pointer may be set to the beginning of a block.

The block size is the only parameter relevant for random access.

ECHOM (MON 3) and BRKM (MON 4) affect data access for terminals. The calls are documented in chapter 5.

3.8.6.1. @SET-BYTE-POINTER

@SET-BYTE-POINTER <file no.>,<byte no.>

Set the byte pointer to be used by the next sequential file access. The corresponding monitor calls are SETBT and REABT. For ex.:

@S-BYTE-P 101,0

The byte pointer is reset to the beginning of the file.

3.8.6.2. @SET-BLOCK-SIZE

@SET-BLOCK-SIZE <file no.>,<block size>

Set the block size to be used by the next file access. The standard block size is 256 decimal words. The corresponding monitor call is SETBS. For ex.:

@S-BL-S 101,512
Set the block size to 512 decimal words.

3.8.6.3. @SET-BLOCK-POINTER

@SET-BLOCK-POINTER <file no.>,<block no.>

Set the byte pointer to the beginning of a block. The position is then dependent on the block size. The corresponding monitor call is SETBL. For ex.:

@S-BLOCK-S 101,512
@S-BLOCK-P 101,1

Set the byte pointer to byte 1024 in the file. (The first byte has address 0.)

3.8.6.4. SETBT (MON 74)

Set the byte pointer of a file. For ex.:

```
DOUBLE INTEGER IBYTE
   ...
IBYTE=0
CALL SETBT(10,IBYTE)
```

Reset the byte pointer of LDN = 10 to 0.

3.8.6.5. REABT (MON 75)

Read the byte pointer of a file. For ex.:

```
DOUBLE INTEGER IBYTE
   ...
CALL REABT(10,IBYTE)
```

Read the byte pointer of LDN=10.

3.8.6.6. SMAX (MON 73)

Set the maximum byte pointer of a file. For ex.:

```
DOUBLE INTEGER MAXBY
   ...
MAXBY=377777B
CALL SMAX(10,MAXBY)
```

Set the maximum byte pointer of LDN=10 to 256K-1. The maximum size of the file will then be 256K bytes.

3.8.6.7. RMAX (MON 62)

Read the maximum byte pointer of a file. For ex.:

```
DOUBLE INTEGER MAXBY
   ...
CALL RMAX(10,MAXBY)
```

Read the maximum byte pointer of LDN = 10.

3.8.6.8. SETBS (MON 76)

Set the block size of an open file. The standard block size is 256 decimal words. For ex.:

```
CALL SETBS(101b,512)
```

Set the block size of LDN = 101 (octal) to 512 (decimal) words. The block size can also be set by including RECL= ... in the OPEN statement.

3.8.6.9. SETBL (MON 77)

Set the byte pointer to the beginning of a block. The position in the file is then dependent on the block size. For ex.:

```
CALL SETBS(101B,512)
CALL SETBL(101B,1)
```

Set the byte pointer to byte 1024 (decimal) in the file. (The first byte has address 0.)

3.8.6.10. CIBUF (MON 13)

Clear the input buffer of a device. For ex.:

```
CALL CIBUF(9)
```

Clear input buffer of terminal 2.

3.8.6.11. COBUF (MON 14)

Clear the output buffer of a device. For ex.:

```
CALL COBUF(9)
```

Clear the output buffer of terminal 2.

3.8.6.12. ISIZE (MON 66)

Get the current number of bytes in an input buffer. For ex.:

```
NUMB=ISIZE(9)
```

On return NUMB will contain the number of bytes in the input buffer of terminal 2.

3.8.6.13. OSIZE (MON 67)

Get the current number of bytes in an outut buffer. For ex.:

```
INTEGER OSIZE
  ...
NUMB=OSIZE(9)
```

On return NUMB will contain the number of bytes in the output buffer of terminal 2.

3.9. Commands for Copying Files

Data can be copied from a source file to a destination file. The methods of copying are: one character at a time or one page at a time. It is also possible to copy more than one file by a single command.

User SYSTEM has available commands for copying all files in a directory (@COPY-DIRECTORY) or copying all pages on a device (@COPY-DEVICE).

3.9.1. @COPY

@COPY <destination file>,<source file>

Copy the contents of the source file one byte at a time to the destination file. For ex.:

@COPY OLE,TAPE-READER

copy from the tape-reader to OLE:SYMB.

3.9.2. @COPY-FILE

@COPY-FILE <destination file>,<source file>

Copy the contents of the source file one block at a time to the destination file. This command should be used if both files are mass storage files. For ex.:

@CO-FI OLE,PER

Copy the contents of PER:SYMB to OLE:SYMB.

3.9.3. @COPY-USERS-FILES

@COPY-USERS-FILES <destination parameters>,<source parameters>

Where
<destination parameters> are either
DIRECTORY,<destination directory>
or
VOLUME,<d. device name>,<d. unit>,<d. volume name>,
<d. file generation>
<source parameters> are either
DIRECTORY,<source name>
or
VOLUME,<s. device name>,<s. unit>,<s. volume name>,
<file name>,<s. file generation>

This command copies a set of mass storage files. The set can reside on either an ANSI standard magnetic tape or a directory.

For ex.:

@C-U-F DIR,P-3,DIR,A:SYMB

Copy all files starting with A on the directory P-3.

@CREATE-VOLUME VOL1,MAG-TAPE-1
@C-U-F VOL,MAG-TAPE-1,VOL1,0312,DIR,:SYMB

Make a backup of all symbolic files on the currently logged on user on a mag. tape volume.

3.9.4. @CREATE-VOLUME

@CREATE-VOLUME <volume name>,<device name>,<unit>

Create a new volume on a magnetic tape. A volume is a labeled collection of files recorded on a reel of magnetic tapes according to the ANSI standard.

For ex.:

@CREATE-VOLUME LABEL1,M-T-1,1

The volume named "LABEL1" is created on MAG-TAPE-1, unit 1.

3.10. Commands for Listing File System Information

Commands are available for listing information about files, directories, users and friends. Some commands are similar to others, except that they give more details. (@FILE-STATISTICS give more details than @LIST-FILES.) The command @LIST-OPENED-FILES gives

information about the files open by the own user. However, user RT and SYSTEM have available a similar command for files opened by RT-programs. (@LIST-RTOPENED-FILES).

Some monitor calls also give information about the file system. Reading the byte pointer and its maximum value (REABT and RMAX) was shown in section 3.8.6. For RT-programs a monitor call is available for checking whether a device is reserved or not (WHDEV, see section 3.5).

3.10.1. @LIST-FILES

@LIST-FILES <file name>,<output file>

List names of files matching <file name> on <output file>. The number listed (FILE ...) is the file object number. For ex.:

```
@LI-FI PROG,TERM
FILE 0 : (PACK-ONE:GUEST)PROG-1:SYMB;1
FILE 12 : (PACK-ONE:GUEST)PROG-2:SYMB;1
FILE 13 : (PACK-ONE:GUEST)PROG-2:BRF;1
@
```

The files matching PROG are listed on TERMINAL.

3.10.2. @FILE-STATISTICS

@FILE-STATISTICS <file name>,<output file>

List complete information about files matching <file name> on <output file>. For ex.:

```
@FI-ST PROG-2:SYMB,TERM
FILE 12 : (PACK-ONE:GUEST)PROG-2:SYMB;1
          (INDEXED FILE)
          PUBLIC ACCESS : READ
          FRIEND ACCESS : READ, WRITE, APPEND, COMMON, DIRECTORY
          OWN ACCESS : READ, WRITE, APPEND, COMMON, DIRECTORY
          OPENED 79 TIMES
          CREATED 16.10.32 APRIL 23, 1979
          OPENED FOR READ 21.59.18 AUGUST 15, 1979
          OPENED FOR WRITE 10.45.15 APRIL 25, 1979
@
```

3.10.3. @LIST-OPENED-FILES

@LIST-OPENED-FILES <output file>

List dynamic file number and full name of all files opened for the terminal user. Note that this is the dynamic and not the static number. For ex.:

```
@L-O-F,,
FILE NUMBER 000100 : (BIG-PACK:SCRATCH)SCRATCH08:DATA;1
@
```
Opened files are listed on TERMINAL.

3.10.4. @LIST-DIRECTORIES-ENTERED

@LIST-DIRECTORIES-ENTERED <directoryname>,<output file>

List names of entered directories matching <directory name> and devices where the directories are mounted. For ex.:

```
@L-DIR-E,,,
DISC-66MB-1 UNIT 0: PACK-ONE
DISC-66MB-1 UNIT 1: PACK-TWO

DISC-288MB-2 UNIT 0: PACK-THREE
@
```
All directories are listed on the TERMINAL.

3.10.5. @DIRECTORY-STATISTICS

@DIRECTORY-STATISTICS <directory name>,<output file>

List information about directories matching <directory name> on <output file>. For ex.:

```
@DIR-STAT B-P,,
DISC-66MB-1 UNIT 0 : BIG-PACK
            (MAIN AND DEFAULT DIRECTORY)
            343 PAGES UNRESERVED AND 6665 PAGES UNUSED BY THE SYSTEM
@
```

3.10.6. @LIST-USERS

@LIST-USERS <directory name>:<user name>,<output file>
List names of all users matching <directory name>:<user name>. For ex.:

```
@LI-US PACK-TWO:,,
USER 0 : PACK-TWO:SYSTEM
```

```
USER 1 : PACK-TWO:JFB
USER 2 : PACK-TWO:GUEST
@
```

3.10.7. @USER-STATISTICS

@USER-STATISTICS <directory name>:<user name>,<output file>

List complete information about all users matching <directory name>:<user name> on <output device>. The default access used when creating a file for the user is listed if the directory is main directory. For ex.:

```
@US-ST P-O:SYS,,
USER 0 : PACK-TWO:SYSTEM
         CREATED 10.01.04 AUGUST 24, 1979
         DEFAULT PUBLIC ACCESS : READ
         DEFAULT FRIEND ACCESS : READ, WRITE, APPEND, COMMON, DIRECTORY
         DEFAULT OWN ACCESS : READ, WRITE, APPEND, COMMON, DIRECTORY
         13782 PAGES USED OUT OF 14511 PAGES
@
```

3.10.8. @LIST-FRIENDS

@LIST-FRIENDS <user name>,<output file>
List names of friends to the terminal user matching <user name>. The list is produced on the <output file>. For ex.:

```
@LI-FR,,,
FRIEND 0 : SYSTEM
              ACCESS : READ, WRITE, APPEND
FRIEND 1 : GUEST
              ACCESS : READ, WRITE, APPEND
@
```

All friends are listed together with their file access.

3.10.9. @LIST-VOLUME

@LIST-VOLUME <device name>,<unit>,<output file>

List the identification of a file, i.e., The reel label and the file labels.

3.10.10. @WHERE-IS-FILE

@WHERE-IS-FILE <file name>
Check whether a file is opened and/or reserved. Also, list the user or RT-program who has opened or reserved the file. For ex:.

```
@W-I-F (SCR)SCRATCH08:D
SCRATCH08:DATA : OPENED BY USER GUEST ON TERMINAL 39
@
```

3.10.11. @LIST-DEVICE

@LIST-DEVICE <LDN>,<read/write>
List name of the RT program having reserved the LDN. <read/write> = 0 for read part, 1 for write part. For ex.:

```
@LIST-DEVICE 1,1
RESERVED BY PROGRAM : BAK01
@
```

The output part of device 1 is reserved by the program BAK01.

3.11. File System Maintenance

It is possible to read the user entry and (file) object entry by two monitor calls. This may be desirable for obtaining information not provided by @USER-STATISTICS and @FILE-STATISTICS. The corresponding commands for printing user entry and object entry (for ex. on the TERMINAL) are @DUMP-USER-ENTRY and @DUMP-OBJECT-ENTRY. They are both SYSTEM commands and are explained in the manual, SINTRAN III SYSTEM SUPERVISOR. The manual also defines the terms "user entry" and "object entry".

3.11.1. RUSER (MON 44)

Read a user entry from a directory to a data area in the user memory. For ex.:

```
        LDA (USEN       % address of data area
        LDX (NAME       % user name
        MON 44          % RUSER
        ...             % normal return
USEN,   *+40/           % data area
NAME,   'GUEST'         % user name
```

The user entry of user GUEST is read to the data area USEN.

3.11.2. ROBJE (MON 41)

Read an object entry of an opened file to a data area in the user memory. For ex.:

```
        (file 101 is opened)
        LDT (101            % open file no. =101
        LDA (OBJEN          % read to data area
        MON 41              % ROBJE
        JMP ERROR           % error return
        ...                 % normal return
OBJEN,  *+40/               % data area
```

The object entry of file 101 is read to the data area OBJEN.

3.12. Initiating a Floppy-Disk

Most time-sharing users will employ floppy-disks for backup of programs and data. A floppy-disk is normally used with a directory similarly to the hard disks. Before files can be written to the disk, a directory must be established. The procedure is as follows,

1. Turn on power on the floppy-disk drive.

2. If it is a "virgin" floppy-disk it must first be formatted by the command

@DEVICE-FUNCTION <peripheral file name>,FORMAT-FLOPPY

For ex.,

@DEV-FU F-2-0,FORM-FLOP

The command takes 1-2 minutes to complete. If there are any unusable pages on the disk, they will be marked by @DEVICE-FUNCTION and listed during the formatting. The disk can still be used.

Note that the first parameter is <peripheral file name> as opposed to <device name>. A more detailed guide to @DEVICE-FUNCTION is found in SINTRAN III SPECIAL I/O GUIDE (ND-60.134).

3. The directory can now be created by the command,

@CREATE-DIRECTORY <dir. name>,<device name>,<unit>,
<bit-file address>

For ex.,

@CRE-DIR BACKUP,F-D-2,0,,

4. Entering the directory is done by the command,

@ENTER-DIRECTORY <dir. name>,<device name>,<unit>

For ex.,

@ENT-DIR ,F-D-2,0

5. Establishing the user is done by the two commands,

@CREATE-USER <user name>
@GIVE-USER-SPACE <user name>,<no. of pages>

For time-sharing users the only permissible <user name> is his own.

The total <no. of pages> available on a floppy-disk is 148. If there are bad pages, the ammount must be reduced by the number indicated in step 2 above.

For ex.,

@CRE-US BACKUP:GUEST
@G-U-S BACKUP:GUEST,148

The user area BAKUP:GUEST is now ready to be used. Before the disk can be removed the @RELEASE-DIRECTORY command must be used.

When entering the floppy-disk the next time only steps 1 and 4 are folowed.

4. Executing User Programs and Subsystems

4.1. General

The manual "SINTRAN III INTRODUCTION" shows how to create and execute programs under SINTRAN III. The symbolic code is created as a SYMB-file. By compiling it, a BRF-file is created, containing relocatable code. NORD Relocatable Loader is used to load and execute the program.

Once the program is loaded, SINTRAN III has available commands for

- creating a PROG-file. This is the program in absolute format.

- loading, starting, and restarting a program.

- examining the user's registers and memory.

- loading binary programs. A binary program is loaded from a BPUN-file. It is similar to a PROG-file except that it can also be loaded as a stand-alone program. (I.e. independently of SINTRAN III.)

Also, there are monitor calls available for extending the address space from 64K to 128K words.

Time-sharing users have available a command for listing programs defined as reentrant subsystems, @LIST-REENTRANT. It is explained in chapter 6.

4.2. Creating a PROG-File

When a program has been loaded by Nord Relocating Loader the program area in the user's memory may be dumped to a PROG-file. The file will contain the program in absolute format.

4.2.1. @MEMORY

@MEMORY <low address>,<high address>

Define the area of the user's memory to be dumped (by @DUMP, see next section). For ex.:

@MEM 10,1777

Set dump limits to 10 to 1777 (octal), limits inclusive.

4.2.2. @DUMP

@DUMP <file name>,<start address>,<restart address>

Dump the user memory area (as defined by @MEMORY) to a file. When the program is started by @RECOVER, the execution will begin at <start address>. When restarted by @CONTINUE, the execution will begin at <restart address>. For ex.:

```
@MEMORY 10,1777
@DUMP MYPROG,10,11
```

The file MYPROG:PROG in default directory will contain a dump of the user area 10-1777. Program start address is 10 and restart address is 11. (All values in octal.)

4.3. Controlling Execution

This section describes commands for starting user programs and subsystems. A general description of how to execute programs is found in section 2.3. The commands described here are used for the transfer between Command Mode and User Mode as shown in fig. 2.4. The command type determines the start address to be used.

4.3.1. @RECOVER

@RECOVER <program name>

Load a PROG-file to the user's memory and start it at the <start address> as specified by @DUMP or @DUMP-REENTRANT. (@DUMP-REENTRANT is explained in the SINTRAN III System Supervisor Manual.) @RECOVER is a default command. I.e. if only the program name is specified, the command is asumed to be @RECOVER. For ex.:

@RECOVER MAC

is equivalent to

@MAC

@RECOVER MYPROG

where MYPROG:PROG is a user defined program, is equivalent to

@MYPROG

4.3.2. @CONTINUE

Restart a program at the <restart address> as specified by @DUMP. (The program is not loaded, only restarted.)

4.3.3. @GOTO-USER

@GOTO-USER <address>

Start a program at a specific address. Default address is the point where the program was interrupted. For ex.:

```
(The user presses ESCAPE during execution.)
@STATUS
P=    1723
X=       5
   ...
@GOTO
```

The program is restarted in address 1723 (octal, @STATUS is explained in the next section.)

4.4. Examining User's Registers and Memory

User's registers and memory may be both examined and modified by the following commands.

4.4.1. @STATUS

Display the contents of the user's registers. For ex.:

```
@STATUS
P=   3157
X=  13302
T=    105
A=  12663
D=     47
L=    605
S=     40
B=    101
@
```

4.4.2. @LOOK-AT

@LOOK-AT <space reference>
This is a general command for modifying user's registers and memory. The <space reference> permitted for time-sharing users are

MEMORY - the time-sharing user's virtual memory (0-64K)

ALT-MEMORY - the time-sharing user's alternative virtual memory (see also section 4.6). Th addresses are specified relative to the 64K boundary.

The references SEGMENT, RTCOMMON, IMAGE, RESIDENT, and REGISTERS are all restricted to users RT and SYSTEM. Addresses and registers are specified on the line following the command. Typing . (dot) causes return to command mode. For ex.:

```
@LOOK-AT REG
X/ 13302  13000
.
@
```

Change the contents of the X register from 13302 to 13000.

```
@LOOK-AT MEM
13000/    10  100
   11  .
@
```

Change the contents of location 13000 from 10 to 100. (11 is the contents of location 13001.)

4.4.3. @SET-MEMORY-CONTENTS

@SET-MEMORY-CONTENTS <contents>,<low address>,<high address>

Set all locations of an area of the user's memory to the same value. This is more convenient then using @LOOK-AT MEMORY. For ex.:

@SET-M-C 124000,0,177777

Set all locations of the user's memory to 124000 (i.e. the instruction JMP *).

4.5. Loading Binary Programs

The following two commands are used for loading programs created by the)BPUN command in MAC or by the BPUN command in NRL.

4.5.1. @LOAD-BINARY

@LOAD-BINARY <file name>

Load a program in BPUN format to the user's memory and start executing it. The command is similar to @RECOVER except that the program file is in BPUN format. For ex.:

@L-BIN PROG-1

The program in the file PROG-1:BPUN is loaded and executed.

4.5.2. @PLACE-BINARY

@PLACE-BINARY <file name>

Load a program in BPUN format to the user's memory. The program is not started. For ex.:

@P-BIN PROG-1

The program in the file PROG-1:BPUN is loaded.

4.6. Extending the Address Space or User Memory

The address space is normally restricted to 64K words for program and data. However, another 64K can be added for data by using the Alternative Page Table of the NORD-100 (and NORD-10). The NORD-100 Reference manual (ND-06.014) describes this feature in detail.

By including the monitor call ALTON (MON 33) in the user program all further X and B relative data references will be to the 64K alternative area. P relative addressing will be to the normal address space. ALTOFF (MON 34) sets the addressing back to the normal mode.

All non-file-system monitor calls plus RFILE and WFILE are permitted after the ALTON call. RFILE and WFILE uses the normal page table for call parameters and data transfer.

For the alternative area to be used, the System Supervisor must increase the background segment from 64 pages to 128 pages (@CHANGE-BACKGROUND-SEGMENT-SIZE).

4.6.1. ALTON (MON 33)

Select alternative page table. For background programs, the normal page table is 2 and the only permitted alternative page table is 3. For ex.:

```
      LDA (PAR          % get parameter list
      MON 33            % ALTON
      ...
PAR,  (3                % alternative page table no. 3
```

Page table no. 3 is selected. Any other number will not give an error message, but the call will be ignored.

4.6.2. ALTOFF (MON 34)

Select normal page table. The call has no parameters.

4.6.3. Sample Program Using ALTON and ALTOFF

the following program moves 1K words from the normal area to the alternative area.

```
NRMAD=40000
ALTAD=0
%
START,  BSET ZRO 0      % set normal addressing
        LDA  (PAR       % get parameter list
        MON  33         % ALTON
        LDA  (-2000     % 1K
        STA  COUNT      % set counter
        LDA  (NRMAD     % data address in normal area
        STA  GET        % set base address of data source
        LDA  (ALTAD     % data address in alt. area
        STA  PUT        % set base address of data destination
LOOP,   LDX  GET        % get source address
        BSET ZRO 0      % reset status bit 0
        LDA  ,X         % get data from normal area
        LDX  PUT        % get destination address
        BSET ONE 0      % set status bit 0
        STA  ,X         % put data in alternative area
        MIN  GET        % increment source address
        MIN  PUT        % increment destination address
        MIN  COUNT      % increment loop count
        JMP  LOOP       % loop while COUNT not eq. to 0
        MON  34         % ALTOF
        MON  0          % EXIT
%
PAR,    (3              % alternative page table no. 3
COUNT,  0               % loop counter
GET,    0               % data source address
PUT,    0               % data destination address
)FILL
)LINE
```

4.7. Error Messages from User Programs

Section 2.5 gives an overview of error handling in SINTRAN III. By selecting one of two monitor calls the user can decide whether an error should just be displayed or whether it should also cause program termination.

4.7.1. ERMSG (MON 64)

Display the error message for a given error number. This monitor call is mostly used in connection with the error return of some other call. For ex.:

```
        MON 50          %OPEN
        MON 64          %ERMSG, error return
         ...            %normal return
```

After displaying the error message the execution resumes at the next location.

4.7.2. QERMS (MON 65)

Display the error message for a given error number and exit from the program. The monitor call is used similarly to ERMSG:

```
MON 50          %OPEN
MON 65          %QERMS, error return
 ...            %normal return
```

After printing the error message the program exits.

4.8. Communicating with the Own Terminal

LDN = 0 specifies the own terminal in input/output monitor calls. On input the text line is transfered to the SINTRAN III input buffer on receiving a carriage return. While the line is typed, the QED control characters are available. In order to use the 'old line' for editing, the user program must first read the line and then use SETCM to transfer the line back to the input buffer.

By using LDN=0 the program may start with reading the rest of the @RECOVER command. This is done with a REWIND statement. Assume the @RECOVER command is

@USPROG ANY TEXT

where USPROG is the name of the user program and 'ANY TEXT' is text to be read by the program. USPROG must then look as follows:

```
      REWIND 0
      READ(0,1)AA
10    FORMAT(A16)
```

Finally, a SINTRAN III command may be executed by specifying it as a text string (COMND).

4.8.1. SETCM (MON 12)

Transfer a string to the command input buffer. It is used for 'old line' editing.

4.8.2. COMND (MON 70)

Execute a character string as a SINTRAN III command. For ex.:

```
      CHARACTER DELFI*50
```

```
DATA DELFI/'DELETE-FILE XXX:SYMB'''/
 ...
CALL COMND(DELFI)
```

The command

@DELETE-FILE XXX:SYMB

is executed. If any error occurs, the program is aborted.

4.9. Terminating Programs

Normally only the monitor call LEAVE is used for program termination, but batch jobs need two methods of termination. Either the user program is to be terminated or, if a serious error, the whole job should be terminated. RTEXT is used for the latter case.

4.9.1. LEAVE (MON 0)

Terminate the executing program and return control to the operating system. In batch jobs, the next command of the job is executed.

4.9.2. RTEXT (MON 134)

Terminate the executing program and return control to the operating system. In batch jobs the job is terminated.

5. Terminal Characteristics

5.1. General

With the wide variety of computer terminals on the market, it is necessary for SINTRAN III to treat the different models individually. This treatment includes adapting to the communication characteristics (@TERMINAL-MODE, @DISABLE-ESCAPE-FUNCTION, @ENABLE-ESCAPE-FUNCTION) and identifying the model (@SET-TERMINAL-TYPE, @GET-TERMINAL-TYPE). In addition, the system supervisor has available commands for selecting which terminal is to receive the system error messages (@SET-ERROR-DEVICE, @GET-ERROR-DEVICE) and commands for selecting the value of the ESCAPE (ESC) character (@DEFINE-ESCAPE-CHARACTER). These commands are explained in the system supervisor manual.

The terminal type is identified by a 16 bit terminal number associated with the LDN. The current numbers defined by Norsk Data A.S is shown in appendix B of the SINTRAN III Reference Manual. If a terminal is sold as part of the system it is also listed with its ND-number.

If a user wants to standardize a particular terminal, he should contact ND to get it into the list. The terminal type name is not stored in SINTRAN III but is used by some subsystems.

5.2. Terminal Communication

5.2.1. @TERMINAL-MODE

This command sets the communication mode of the terminal. The mode determines how SINTRAN III should handle the terminal. The functions are:

- converting characters to upper case on input.

- causing delay after CARRIAGE RETURN (CR) on output by printing filler (dummy) characters.

- stopping after 20 lines of output in order to increase the readability of fast displays.

- logging out the terminal on receiving a "missing carrier" signal.

The command parameters are answers to questions. Default values causes no change in the function. For ex.:

```
@TERM-MO
CAPITAL LETTERS?N
DELAY AFTER CR?
STOP ON FULL PAGE?Y
LOGOUT ON MISSING CARRIER?
@
```

Capital letters should not be converted. Output should stop after 20 lines of output. The other functions are not changed.

5.2.2. @DISABLE-ESCAPE-FUNCTION

@DISABLE-ESCAPE-FUNCTION <LDN>

Disable the function of the ESCAPE character. It is disabled until the command @ENABLE-ESCAPE-FUNCTION is given or the user logs out. Time-sharing users are only permitted to use the default value of <LDN> which is the own terminal.(Only user SYSTEM can specify a terminal other than his own.)

5.2.3. @ENABLE-ESCAPE-FUNCTION

@ENABLE-ESCAPE-FUNCTION <LDN>

Enable the function of the ESCAPE character. Time-sharing users are only permitted to use the default value of the parameter which is the own terminal. (Only user SYSTEM can specify a terminal other than his own.)

5.2.4. TERMO (MON 52)

Set communication mode for any terminal. The call corresponds to @TERMINAL-MODE. For ex. in a background program,

```
INTEGER TERMO
ISTAT=TERMO(0,4)
```

Set "stop on full page" for the own terminal. All other modes are set to NO.

5.2.5. DESCF (MON 71)

Disable escape function.The call corresponds to @DISABLE-ESCAPE-FUNCTION above. For ex.:

```
CALL DESCF(ITERM)
```

Disable the escape function for the own terminal. The LDN in ITERM is ignored for background programs.

5.2.6. EESCF (MON 72)

Enable escape function. It corresponds to @ENABLE-ESCAPE-FUNCTION above. For ex.:

```
CALL EESCF(ITERM)
```

ITERM is ignored for background programs.

5.2.7. ECHOM (MON 3)

Set the echo strategy of a terminal. A description of this strategy is found in SINTRAN III SYSTEM DOCUMENTATION, section 3.4.6.4 (ND-60.062). For ex.:

```
LDN=9
CALL ECHOM(LDN,1)
```

On terminal 2 set echo on all characters except control characters.

5.2.8. BRKM (MON 4)

Set the break strategy of a terminal. A description of this strategy is found in SINTRAN III SYSTEM DOCUMENTATION, section 3.5.6.4 (ND-60.062). For ex.:

```
LDN=9
CALL BRKM(LDN,0)
```

On terminal 2 set break on all characters.

5.3. Terminal Identification

5.3.1. @SET-TERMINAL-TYPE

@SET-TERMINAL-TYPE <LDN>,<terminal number>

Associate a terminal type number with a terminal. Time-sharing users are only permitted to use the default value of the parameter which is the own terminal. For ex.:

@SET-TERMINAL-TYPE ,160007B

The own terminal is identified as TANDBERG-TDV2000.

5.3.2. @GET-TERMINAL-TYPE

@GET-TERMINAL-TYPE <LDN>

Display the terminal number. (Time-sharing users may use any LDN.) For ex.:

```
@G-T-T 52
TERMINAL NUMBER IS -8189

@
```

The terminal number of LDN = 52 (decimal) is displayed as a decimal signed integer.

5.3.3. MSTTY (MON 17)

Set the terminal type. For ex.,

```
CALL MSTTY(0,4)
```

Set the terminal number of the own terminal to 4.

5.3.4. MGTTY (MON 16)

Get the terminal number. For ex.,

```
CALL MGTTY(0,ITY)
```

The terminal type of the own terminal is stored in ITY.

5.4. Suspending the Terminal

5.4.1. @HOLD

@HOLD <no. of time units>,<time unit>

Keep the terminal waiting a specified amount of time. For ex.:

```
@HOLD 5,3
@
```

The last @ occurs after 5 minutes.

5.4.2. HOLD (MON 104)

Set the calling program in the waiting state for a specified amount of time. For ex.:

```
CALL HOLD(10,2)
```

The calling program will wait for 10 seconds before continuing with the next statement.

6. Measurement and Statistics

6.1. General

This chapter describes commands and monitor calls for measuring the performance and obtaining information on programs.

6.2. CPU Histogram

A CPU histogram is a recording of the time spent in different parts of a program for a given time period. The user defines a contiguous set of 64 intervals in the user memory. As a program is running the contents of the P-register is sampled every basic time unit. At the end of the sampling period the user can request a listing of the samples including a percentage of time spent in each interval.

User SYSTEM has available a command to make a histogram of the operating system (@DEFINE-SYSTEM-HISTOGRAM). Also, user SYSTEM has available commands for making a program log of an RT-program. This is a continuous recording of the current CPU- and input/output-time used by the RT-program.

6.2.1. @DEFINE-HISTOGRAM

@DEFINE-HISTOGRAM <program name>,<start address>,<interval>

Define a histogram for an RT program. Time-sharing users can only specify the default value of <program name> which is the own background program For ex.:

@DEF-HI ,0,2000

Define a histogram for the own background program. All 64K words will be sampled, dividing it into 1K intervals.

6.2.2. @START-HISTOGRAM

This command is used to start the samping for the histogram.

6.2.3. @STOP-HISTOGRAM

This command stops the sampling for the histogram.

6.2.4. @PRINT-HISTOGRAM

@PRINT-HISTOGRAM <output file>

This command prints the resulting histogram and sets the "histogram mode" to "undefined". (i.e. in order to print another histogram the user must issue a @DEFINE-HISTOGRAM command.) For ex.:

```
@PR-HI
                    PERCENT SAMPLES
OUTSIDE:              0       0
SYSTEM:               3       6
     0-  1777         5       10
          ...
176000-177777         0       0

@
```

A histogram is printed, corresponding to the previous example of @DEFINE-HISTOGRAM. "OUTSIDE" is the number of samples found outside the sampling area but not within SINTRAN III.

6.3. Information about RT Programs and Segments

Management of RT programs is restricted to users RT and SYSTEM. However, listing information about RT programs is permitted for time-sharing users. Commands are available for obtaining information on scheduling queues, RT-descriptions, and segments.

6.3.1. @LIST-EXECUTION-QUEUE

List the names of programs in the execution queue. The list contains names or addresses of the RT-descriptions. For ex.:

```
@L-E-Q
RCOM1
RTERR
RU051
RU041
BAK05
BAK06
 ...
BAK02
DUMMY
@
```

6.3.2. @LIST-TIME-QUEUE

List the names of the RT programs waiting in the time queue. For ex.:

```
@L-T-Q
RTSLI
TIMRT
@
```

6.3.3. @LIST-RT-DESCRIPTION

@LIST-RT-DISCRIPTION <program>

List the information in the RT description on the terminal. The parameter is RT program name or address of an RT description. For ex.:

```
@L-R-D SIBI1
I/O-WAIT  RING:1 PRIORITY:     24
LAST STARTED: 1060 MINS    49 SECS
START ADDRESS:  32002, SEGMENTS:      0     236
P=  14245
X=  33747
T=    203
A=    203
D=  33752
L=  10036
S=     40
B=  31261
READY
ACTUAL SEGM.:    176    236
REENTRANT SEGMENT:    176
RESERVED DATAFIELDS:
  30751
  30722

@
```

6.3.4. @LIST-SEGMENT

@LIST-SEGMENT <segment no.>

List information about the segment. For ex.:

```
@LI-SEG 176

FIRST PAGE:      124 LENGTH:       24
SEG.FILE:          2 MASS. ADR:    556
WPM RPM FPM RING1 DEMAND
```

@

Segment 176 starts at logical page 124 (i.e. page 24 on page table 1) and has a length of 24 pages. It is stored on segment file "SEGFIL2" at address 556 relative to the start of the segment file. Writing, reading, and fetching is permitted. The segment belongs to ring 1 and is of demand type.

6.3.5. GETRT (MON 30)

Get the RT-description address of the own program. For time-sharing programs this is the address of the background program. For ex.:

```
INTEGER GETRT
IRTADR=GETRT(0)
```

IRTADR will contain the address of the own RT-description.

6.4. The SINTRAN III Calendar

NORD computers are equiped with hardware for measuring time. The date and time of day are stored in SINTRAN III in basic time units. One basic time unit is normally 20 milliseconds. The calendar can be read either in basic time units or to an array in decoded form.

The two commands for displaying the calendar (@DATCL) and the CPU time used (@TIME-USED) were explained in chapter 2. Monitor calls for modifying the calendar are only permitted for foreground programs (UPDAT and CLADJ). The corresponding commands are only permitted for users RT and SYSTEM (@UPDAT and @CLADJ).

6.4.1. @DATCL

See section 2.6.

6.4.2. @TIME-USED

See section 2.6.

6.4.3. CLOCK (MON 113)

Read the current setting of date and time of day. For ex.:

```
INTEGER CLDAT(7)
CALL CLOCK(CLDAT)
```

CLDAT will receve the following data:

CLDAT(1) : basic time units

CLDAT(2) : seconds

CLDAT(3) : minutes

CLDAT(4) : hours

CLDAT(5) : day

CLDAT(6) : month

CLDAT(7) : year

6.4.4. TIME (MON 11)

Read current internal time. For ex.:

```
DOUBLE INTEGER TIME,TI
TI=TIME(0)
```

TI will receive the internal time in basic time units.

6.4.5. TUSED (MON 114)

Read the CPU time used since the user logged on or the batch job was started. For ex.:

```
DOUBLE INTEGER TD,TUSED
TD=TUSED(0)
```

TD will receive the CPU time used in basic time units.

6.5. Miscellanous

6.5.1. @LIST-REENTRANT

List the reentrant subsystems established by the @DUMP-REENTRANT command (user SYSTEM).For ex.:

```
@LI-RE
 START RESTART SEGMENT
     0       1     112   QED
177777  177775     114   MAC
          ...
```

6.5.2. @LIST-TITLE

List the system identification normally displayed when logging on. For ex.:

```
@LI-TI
NORD 10/S - 781012
@
```

6.5.3. @HELP

@HELP <command>,<output file>
List command names matching <command>. For ex.,

```
@HELP EX,,
RT: EXECUTE-IOX
FILSYS: EXPAND-FILE
@
```

Two commands start with "EX", one RT and one file-system command.

6.5.4. @TERMINAL-STATUS

See section 2.6.

6.5.5. @WHO-IS-ON

See section 2.6.

7. Batch- and Mode-Jobs

7.1. General

SINTRAN III commands can be processed interactively on the terminal by the user, in a mode-job under terminal control or in a batch-job independently of any terminal.

Mode- and batch-jobs are demonstrated in SINTRAN III INTRODUCTION. A mode-job normally resides in a mass storage file and command output is routed to the terminal. A batch-job normally resides on a mass storage file and the batch output is routed to some other file or to the line-printer.

The SINTRAN III batch processing system comprises one or more batch processors. The number is determined at system generation time. Internally, each processor is implemented as a background program similarly to the ones that run the terminals.

User SYSTEM has available commands for starting and stopping each batch processor (@BATCH and @ABORT-BATCH).

A batch processor may be in one of three states:

PASSIVE: the batch processor has not been started.

IDLE: the batch processor has been started but its queue is empty.

ACTIVE: the batch processor is working on a batch job.

In the IDLE state, when any user appends a batch file to a queue, the processor will be activated. Batch files will be taken from the queue until it becomes empty. The processor then goes back to the idle state.

Remote job entry is documented in the manual SINTRAN III SPECIAL I/O GUIDE (ND-60.134).

7.2. Definitions

A batch job is a collection of SINTRAN III and subsystem commands, arranged in the sequence they would be entered from a terminal keyboard. The first command of the job must be @ENTER. It has the function of "logging on" a batch user. The batch job will then "belong to" this user. The commands in the job will be regarded as comming from this user.

A mode-job is equivalent to a batch-job, only it is used in a different way. The mode-job is started by the @MODE command and the terminal is dedicated to the mode processing until all commands are

executed. Any @ENTER-command is ignored. The mode-job belongs to the user giving the @MODE command.

A job input file is either a batch input file or a mode input file. It contains one or more jobs to be processed. The boundaries between the jobs are determined by @ENTER or the end of the job input file.

A job output file is either a batch output file or mode output file. It is the destination of the command output normally displayed on the terminal.

A batch queue is a list of batch files to be input to the batch processor. Each entry in the queue consists of a batch input file and a corresponding batch output file.

7.3. Sample Batch File

The example will do the following:

Compile a FORTRAN program included in the job.

Load the program together with subprograms and library routines.

Execute the program with data from the batch file.

```
@ENTER USER-ONE,1814,100,2
@CC    *************************
@CC    *** SAMPLE BATCH FILE ***
@CC    *************************
@FTN
COM 1,1,PROGA
      PROGRAM A
      I1=0
      I2=0
      LL=10
       ...
      END
      EOF
EX
@NRL
LOAD TEST,SUBLIB
LOAD FTNLIBR
ENTRIES-DEFINED
ENTRIES-UNDEFINED
DUMP PROGA
EXIT
@PROGA
10 20 30
11 15 -1
```

Note that SINTRAN III commands must include the attention character (@). The subcommands must not have the attention character.

The job can also be run as a mode-job. The @ENTER-command will then be ignored. It must be run by USER-ONE.

7.4. Commands for Running Batch Jobs

These are commands for appending a batch file, aborting the currently running job and deleting an entry from the batch queue. It is also possible to list the contents of a queue and the status of the batch processor.

7.4.1. @APPEND-BATCH

@APPEND-BATCH <batch no.>,<input file>,<output file>

Append a batch file to the queue of a batch processor. For ex.:

@APP-B 1,JOB-1,L-P
The batch-file JOB-1:SYMB is appended to the batch processor. Output will be appended to the LINE-PRINTER.

7.4.2. @ABORT-JOB

@ABORT-JOB <batch no.>,<user name>

Abort the current batch job being processed. The next job will be initiated. A time-sharing user can only abort a job if it belongs to the own user. For ex.:

(own user is GUEST)
@AB-J 1,GUEST

The current batch job for batch processor 1 is aborted provided its owner is GUEST.

7.4.3. @DELETE-BATCH-QUEUE-ENTRY

@DELETE-BATCH-QUEUE-ENTRY <batch no.>,<input file>,<output file>

Delete a batch file in the batch queue waiting to be started. Time-sharing users can only delete a file belonging to himself. <input file> and <output file> must be spelled the same way as in the corresponding @APPEND-BATCH command. (Use @LIST-BATCH-QUEUE to find the correct spelling.) For ex.:

@D-B-Q-E 1,JOB-1,L-P
The batch file JOB-1 is deleted from the queue of batch processor 1, provided the second and third parameters match exactly an entry

in the queue.

7.4.4. @LIST-BATCH-QUEUE

@LIST-BATCH-QUEUE <batch no.>

List the contents of the batch queue. For ex.:

```
@L-B-Q 1
1 CARD-READER LINE-PRINTER
2 JOB-1 L-P
@
```

There are two entries in the queue. The file "CARD-READER" is the next one to be processed.

7.4.5. @LIST-BATCH-PROCESS

List the state of each batch process defined in the system. For ex.:

```
@L-B-P
      1 IDLE,     NO USE LOGGED ON
      2 ACTIVE    USER GUEST LOGGED ON
      3 PASSIVE
@
```

Three processors are defined. The second one runs a job belonging to user GUEST.

7.5. Commands for Running Mode-Jobs

7.5.1. @MODE

@MODE <input file>,<output file>

Substitute the current command input (normally TERMINAL) with commands from the <input file>. Command output is routed to the <output file> (normally TERMINAL). When receiving end-of-file on the <input file> the command input and output are routed to the previous destinations. For ex.:

```
(user is logged on as USER-ONE)
@MODE JOB-1,TERM
   ...                    (command output)
@
```

If the command is issued interactively at the terminal, the next commands are taken from the mass storage file "JOB-1:SYMB". The

command output is routed to the terminal.

(the following command resides in the file JOB-1:SYMB)
@MODE JOB-2,TERM

The next commands will be taken from JOB-2:SYMB and the output is routed to the <output file> specified by the @MODE-command previously given at the terminal. This is the <output file> at the "top" of the @MODE nesting and may not necessarily be TERMINAL.

7.6. Commands Within Mode- or Batch-Jobs

The command @ENTER has already been mentioned as the first command in a batch file. @SCHEDULE is used to reserve devices.

Each SINTRAN III command must start with the attention character @. Subcommands do not use any attention character. All command parameters must be specified, either on the same line or on subsequent lines. @LOGOUT and @MODE are not permitted within batch jobs. Also, some commands are only relevant for interactive use. Mode-jobs may contain @MODE commands for nesting jobs.

If an error occurs in a job, the error message is written to the job output file together with the message

*** BATCH JOB ABORTED ***

The job is then aborted. If an error occurs in accessing the job input or output file, an error message is routed to the error device.

7.6.1. @ENTER

@ENTER <user name>,<password>,<project no.>,<max. time>

Start a new batch job. The parameters are similar to the ones used when logging on. The job is terminated after the last command or by being aborted when <max. time> has expired. An example of @ENTER was shown in the section "SAMPLE BATCH JOB".

7.6.2. @SCHEDULE

@SCHEDULE <logical dev. no.>,<logical dev. no.> ...
<<logical dev. no.>

Reserve a set of devies for the batch job. If any device is already defined, the batch processor will enter a waiting state until all devices are released. For ex.:

@SCHEDULE 2,5

Reserve the tape-reader and line-printer.

7.6.3. @CC

@CC <text>

This command functions as a comment card in batch or mode jobs. Section 7.3 shows an example of three @CC commands.

7.7. Monitor Calls for Mode- and Batch-Jobs

7.7.1. RSIO (MON 143)

This is a call for determining the execution mode of the calling program. The modes are: Interactive, batch-job or mode-job. For ex.:

```
CALL RSIO(IEXEC,IINP,IOUTP,IUSER)
```

IEXEC will on return contain the execution mode which is 0 for interactive, 1 for batch and 2 for mode. IINP, IOUTP and IUSER will contain the file number of the input job file, the output job file and the user number, respectively. (The user number is the value listed by @LIST-USERS.)

8. Spooling

8.1. General

The term spooling is derived from the acronym SPOOL which stands for Simultaneous Peripheral Output On Line. In SINTRAN III it means that files to be printed can be put in a queue waiting to be output. This is obviously more convenient for the user than having to wait for the printer to be available, then reserving it, copy the file (@COPY-FILE, etc.), and finally releasing the printer. Files are being output from the spooling queue as an independent process in the computer.

A peripheral with the spooling feature will have a peripheral file with more than one version. Version 1 is the normal peripheral file. The other files are mass storage files, called peripheral spooling files (PSF). They will receive the data when the user writes to the peripheral (@COPY, @COPY-FILE, etc.). After data is written the PSF is appended to the spooling queue. The user can also append a user file directly to the spooling queue by means of the command @APPEND-SPOOLING-FILE. This is faster than a copy command since it bypasses the PSF, but the user can not access the file until it is printed. Thus, the spooling queue may contain two types of files, PSFs and users files.

Normally the user wants the output to be started as soon as his file becomes the first one in the queue and the preceeding file is finished. However, the user may also specify a stop condition, i.e. the output will stop when the file is ready to be printed. A user message will appear on the error device and the output must be started by a @START-PRINT command. Only user SYSTEM and the user who appended the file to the queue are permitted to issue this command. The stop condition is useful for printing on special forms.

In general there are three ways of specifying a stop condition.

1. User SYSTEM may specify stop condition for all files in the queue. (@DEFINE-SPOOLING-CONDITIONS)

2. A time-sharing user may specify a stop condition for all PSFs he has appended. (@SPOOLING-FILE-MESSAGE)

3. A time-sharing user may specify a stop condition for the particular file he appends. (@APPEND-SPOOLING-FILE)

The output may also be stopped while a file is being printed (@STOP-PRINT).

Files are normally printed in the order in which they occur in the queue. User SYSTEM may specify that only files with a specific user message should be printed (@SET-SPOOLING-FORM). The other files will

then get a lower priority in the queue and only be printed if no files in the queue have the specific message. Time-sharing users may use @LIST-SPOOLING-FORM to check the message.

Fig. 8-1 shows the different states of the spooling system while it is activated by user SYSTEM (@START-SPOOLING). In state 1 there are no files in the queue. (The message from @LIST-SPOOLING-QUEUE is 'QUEUE IS EMPTY'.) The system will leave this state as soon as a file is put in the queue. Normally no stop condition is specified and the system is transfered to state 2 where the output starts immediately. (@LIST-SPOOING-QUEUE now returns the message 'FILE CURRENTLY PRINTED IS ...'). The system will be in this state as long as the queue contains files to be printed without stop condition. When the queue becomes empty the system returns to state 1.

In state 2 the file may be aborted (@ABORT-PRINT) causing the print to be terminated and the next file to be started, if any. The file may also be restarted (@RESTART-PRINT) causing the print to start over again from the first page.

A file with a stop condition will cause the system to enter state 3 when it is ready to be printed. (This state is indicated by a user message on the error device.) Note that in this state the file is still part of the queue. On the @START-PRINT command the system is transfered to state 2 where the file is taken from the queue and printed. Special concern must be given to deleting the file in state 3. Since it is still part of the queue then @ABORT-PRINT and @FORWARD-SPACE-PRINT do not work. The file must be removed by @DELETE-SPOOLING-FILE. The user should note that the spooling must then be started either by user SYSTEM or the user who appended the deleted file and not the one who appended the next file in the queue.

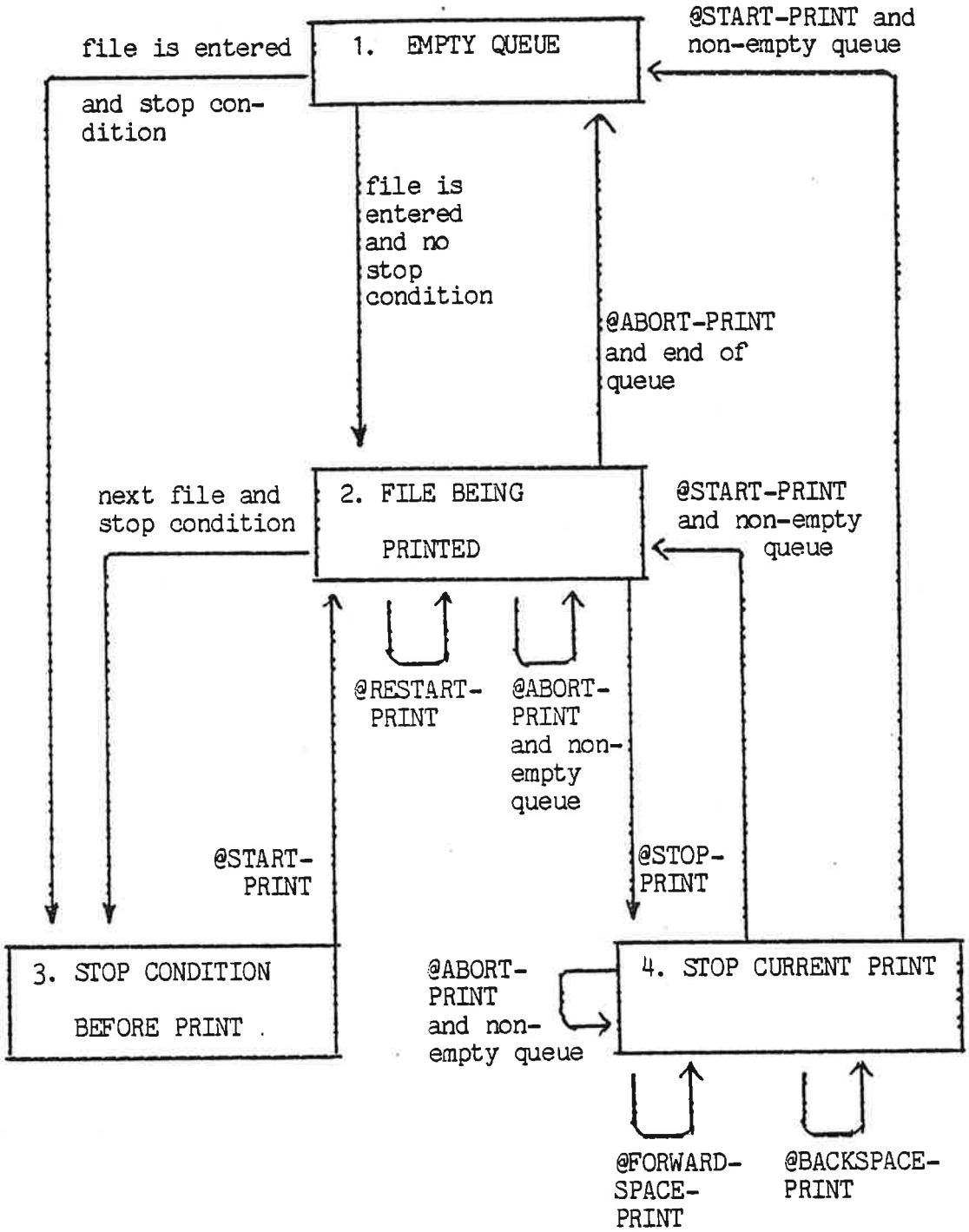


Fig. 8-1 Spooling system for time-sharing users

In state 2, @STOP-PRINT transfers the system to state 4, stop current print. (It can be verified by @LIST-SPOOLING-QUEUE. The message 'FILE CURRENTLY PRINTED ...' is deleted.) Stopping the print is useful should something unexpected occur during the output (paper crash etc.). @RESTART-PRINT, @ABORT-PRINT, @FORWARD-SPACE-PRINT, and @BACKSPACE-PRINT is accepted, but no output occurs until the next @START-PRINT. If the file is aborted the print must be started by user SYSTEM or the user who appended the aborted file, similarly to the rule for state 3.

Time-sharing users will normally only need to copy or append a file to the queue and maybe check the proceedings of the output. The basic commands used for this purpose is @COPY-FILE, @APPEND-FILE and @LIST-SPOOLING-FILE.

8.2. Appending and Changing the Queue Entry

As mentioned previously, a queue entry may be a PSF or a user file. The queue entry contains information about the file name, the number of copies to be printed, and, if used, a text to be printed out on the error device before the user starts the printout.

8.2.1. @APPEND-SPOOLING-FILE

@APPEND-SPOOLING-FILE <peripheral file name>,<file name>,
<no. of copies>,<text>,<PRINTING MESSAGE INDEPENDENTLY OF SPOOLING CONDITIONS?>

Append one or more copies of a file to a spooling queue. For ex.:

```
@APP-S-F L-P,OLE,'.,
```

The user file OLE is put in the spooling queue to line-printer. It is started directly (state 2) unless the own user has specified a stop condition.

```
@APP-S-F L-P,OLE,OLE IS READY',Y
```

The command is similar to the one above except that when the file is ready to be output, the message 'OLE IS READY' is output to the error device (state 3). The output is independent of any stop condition.

8.2.2. @SET-NUMBER-OF-PRINT-COPIES

@SET-NUMBER-OF-PRINT-COPIES <peripheral file name>,<file name>,
<no. of copies>

Change the number of copies to be printed for a specific file in the queue. For ex.:

@SET-N-O-P-C L-P,OLE,2

Two copies of the file OLE will be printed.

8.2.3. @DEFINE-SPOOLING-FILE-MESAGE

@DEFINE-SPOOLING-FILE-MESSAGE <text>,
<PRINTING MESSAGE INDEPENDENTLY OF SPOOLING CONDITIONS?>
Associate a text with a spooling-close. This means that if a user file is copied to a PSF the text will be attached to the peripheral spooling file. When the file is ready to be printed, the text is output to the error device and the spooling system will wait for a @START-PRINT. For ex.:

@DEF-S-F-M READY FOR USER GUEST',Y

Print the text 'READY FOR USER GUEST' and wait for @START-PRINT before printing any PSF containing text from the own user's files.

@DEF-S-F-M ',Y

Reset to normal operation.

8.3. Changing the Order in the Queue

The time-sharing user is permitted to remove one of his files in the queue or to move it backwards. User SYSTEM may also select a subset of his files to be printed (@SET-SPOOLING-FORM).

8.3.1. @MOVE-SPOOLING-QUEUE-ENTRY

@MOVE-SPOOLING-QUEUE-ENTRY <peripheral file name>,<file name>,
<INSERT OR APPEND?>,<before/after file name>

Change the order of a file in the spooling queue. Only user SYSTEM may move entries forward in the queue. For ex.:

@M-S-Q-E L-p,F-1,F-2

The file F-1 is moved in front of F-2 in the spooling queue.

8.3.2. @REMOVE-FROM-SPOOLING-QUEUE

@REMOVE-FROM-SPOOLING-QUEUE <peripheral file name>,<file name>

Remove a file from the spooling queue. The contents of the file is retained. For ex.:

@R-F-S-Q L-P,F-1

If found in the spooling queue to L-P, the file F-1 is removed.

8.3.3. @DELETE-SPOOLING-FILE

@DELETE-SPOOLING-FILE <peripheral file name>,<file name>

Remove a file from the spooling queue. If it is a PSF its pages are released and returned to the pool of free spooling pages. For ex.:

@DEL-S-F L-P,L-P::10

If found in the spooling queue to L-P, the PSF L-P::10 is removed and its pages deleted. It will then exist as a mass storage file with zero pages.

8.4. Starting and Stopping the Print

The time-sharing user may start and stop the printing of his own user files or the PSFs containing his own text.

8.4.1. @START-PRINT

@START-PRINT <peripheral file name>
Continue the printing of the current file, or, if there is no current file, start the printing of the next file.

8.4.2. @STOP-PRINT

@STOP-PRINT <peripheral file name>
Stop the current printout. The command may take some time to complete because of data buffering.

8.4.3. @RESTART-PRINT

@RESTART-PRINT <peripheral file name>
Restart the file currently being printed on the spooling device. If the printing is stopped, the command will only reset the file. (It must be started by @START-PRINT).

8.4.4. @ABORT-PRINT

@ABORT-PRINT <peripheral file name>

Abort the current printout. A time-sharing user may only delete a file appended by himself. For ex.:

@ABORT-PRINT L-P

If the own user appended the current printout, it is aborted and the next file is taken from the queue.

8.5. Adjusting the Print

Sometimes it is necessary to repeat the printing of the last few pages or to skip the printing of a few pages in the file currently being printed. This is normally done while the spooling is in the stop print mode within the file. When the printing is started the output will continue at the new position. The following two commands are used for this purpose.

8.5.1. @FORWARD-SPACE-PRINT

@FORWARD-SPACE-PRINT <peripheral file name>,<no. of pages>,
<no. of lines>

Skip the specified number of pages and lines in the current file. For ex.:

@F-S-P L-P,10,0
Skip 10 pages in the current file.

8.5.2. @BACKSPACE-PRINT

@BACKSPACE-PRINT <peripheral file name>,<no. of pages>,
<no. of lines>

Repeat the printing of the specified number of pages and lines in the current file. For ex.:

@BACK-P L-P,10,0
Repeat the printing of the last 10 pages.

8.6. Statistics

8.6.1. @LIST-SPOOLING-QUEUE

@LIST-SPOOLING-QUEUE <peripheral file name>,<output file>

List information of a spooling queue on the output device. The listing contains information on the file currently being printed and the files in the spooling queue. For ex.:

@L-S-Q L-P,,

```
FILE CURRENTLY BEING PRINTED ON:  LINE-PRINTER:;1
(PACK-THREE:USER1)N500-MIC:SYMB;1 , APPENDED BY USER1 , 1 COPY LEFT
      APPROX. 283434 BYTES LEFT TO PRINT

(PACK-THREE:USER1)N500-MIC-OCTAL:SYMB;1 , APPENDED BY USER1 , 1 COPY
      307602 BYTES IN FILE
(PACK-THREE:ELI-S)GRUFS:LIST;1 , APPENDED BY ELI-S , 1 COPY
      51005 BYTES IN FILE
(PACK-THREE:GUEST)JAN:SYMB;1 , APPENDED BY GUEST , 1 COPY
      73 BYTES IN FILE
*** USER MESSAGE: JAN IS READY
(PACK-THREE:GUEST)FILE1:SYMB;1 , APPENDED BY GUEST , 1 COPY
      0 BYTES IN FILE
*** USER MESSAGE: FILE1 IS READY
```

The first three lines show the name of the file currently being printed, showing how many bytes are left to be printed. The remaining lines show the files in the spooling queue. The last two files will cause a stop condition to be generated before the file is printed (state 3). The user message will be printed on the error device.

8.6.2. @SPOOLING-PAGES-LEFT

List the remainding number of pages that can be used by the spooling files. For ex.:

```
@S-P-L
533 SPOOLING PAGES LEFT
@
```

8.6.3. @LIST-SPOOLING-FORM

@LIST-SPOOLING-FORM <peripheral file name>

List the spooling identification key. The key is defined by user SYSTEM (@SET-SPOOLING-FORM) and is compared to the user message of every file to be printed out. The file is only printed if there is a match. It may also be printed if there is no file in the queue with a matching message. Thus, user SYSTEM may give a subset of the files in the spooling queue a higher priority.

8.7. Monitor Calls for Spooling

8.7.1. SPCLO (MON 40)

Close a PSF. The call corresponds to @APPEND-SPOOLING-FILE except that the file must be a PSF. The file is put on the spooling queue and a user message can be associated with the file. For ex.:

```
CALL SPCLO(101B,INSERT FORM B',1,1)
```

The PSF 101B is closed and put on the spooling queue. Before it is printed the message 'INSERT FORM B' is printed on the error device and the spooling system waits for a @START-PRINT command.

8.7.2. RSPQE (MON 55)

Read next spooling queue entry and remove it from the queue. This call is meant for user defined spooling systems. For ex.:

```
      LDX  (QENT        % Queue entry destination
      LDT  (5           % LDN of spooling device
      MON  55           % RSPQE
      MON  65           % error return
       ...              % normal return
QENT=*
*=*+200
```

9. Sending Messages to Other Terminals

The SINTRAN III mail system (@MAIL) makes it possible for time-sharing users to send a message to a specific terminal. The full set of subcommands to @MAIL is only available to user SYSTEM. He must initiate the mail system and is the only user permitted to send a message to all terminals by one command(broadcast).

Time-sharing users may also send a special message to the operator in order to request some service (@OPERATOR). He can finally wait to be restarted by the operator (@WAIT-FOR-OPERATOR).

9.1. @MAIL

@MAIL <output file>
<subcommands>
Enter the mail system. <output file> specifies the destination of the own user's mail. Time-sharing users can send a message to a specific terminal by using one of two subcommands:

*SEND-MESSAGE <user name>
<message>

The message is terminated by CTRL/L. It is routed to the user's mailbox and he will be notified the next time he logs out or logs on the system. He must then give a @MAIL command in order to read the message.

*SEND-DIRECT-MESSAGE <LDN>
<message>

After typing CTRL/L, the message is routed directly to a specific terminal. It is displayed independently of the terminal activity. (The message will not interfere with the receiver's program execution.)

A list of the subcommands is made available by using the subcommand *HELP. *EXIT returns the terminal to SINTRAN III command mode.

9.2. @OPERATOR

@OPERATOR <text>

Send a message to the error device. The message is terminated by carriage return. (not ' or CTRL/L). For ex.:

(The command is issued on terminal 52)
@OPER MOUNT TAPE 1

On the error device the following message is received:

*** 13.25.15 TERMINAL 52:

MOUNT TAPE 1

9.3. @WAIT-FOR-OPERATOR

Wait for the operator to restart the user (@RESTART-USER). The message received on the error device may look as follows:

---- 13.27.05 WAITING TERMINAL 52

APPENDIX, Editing Control Characters for Commands

The following list contains the subset of QED control characters which are relevant for SINTRAN III commands.

CORRECTING THE CURRENT LINE

CTRL/A Backspace one character (echo is up-arrow or under-line)

CTRL/Q Restart the command on a new line (echo is left-arrow)

COPY FROM PREVIOUS LINE

CTRL/C Copy one character

CTRL/D Copy the rest of the line including carriage return

CTRL/H Copy the rest of the line not including carriage return

CTRL/Zx Copy the old command up to and including x

CTRL/Ox Copy the old command up to but not including x

SKIP CHARACTERS IN OLD COMMAND

CTRL/S Skip one character

CTRL/Xx Skip characters up to and including x

CTRL/Px Skip characters up to but not including x

(Carriage return skips the rest of the line.)

INSERT CHARACTERS

CTRL/E Change to insert mode (echo is <) or overstrike mode (echo is >)

This index includes terms which are not complete headings. For names of time-sharing commands and monitor calls, the reader should make himself familiar with the table of contents. Some commands and monitor calls are only referenced in this manual and are therefore included in the index.

* indicates a definition of the term. It occurs only in a list of multiple references.

* * * * * * * * * * 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: Sintran III TIME-SHARING/Batch Guide Manual number: ND-60.132.01

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 _____

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