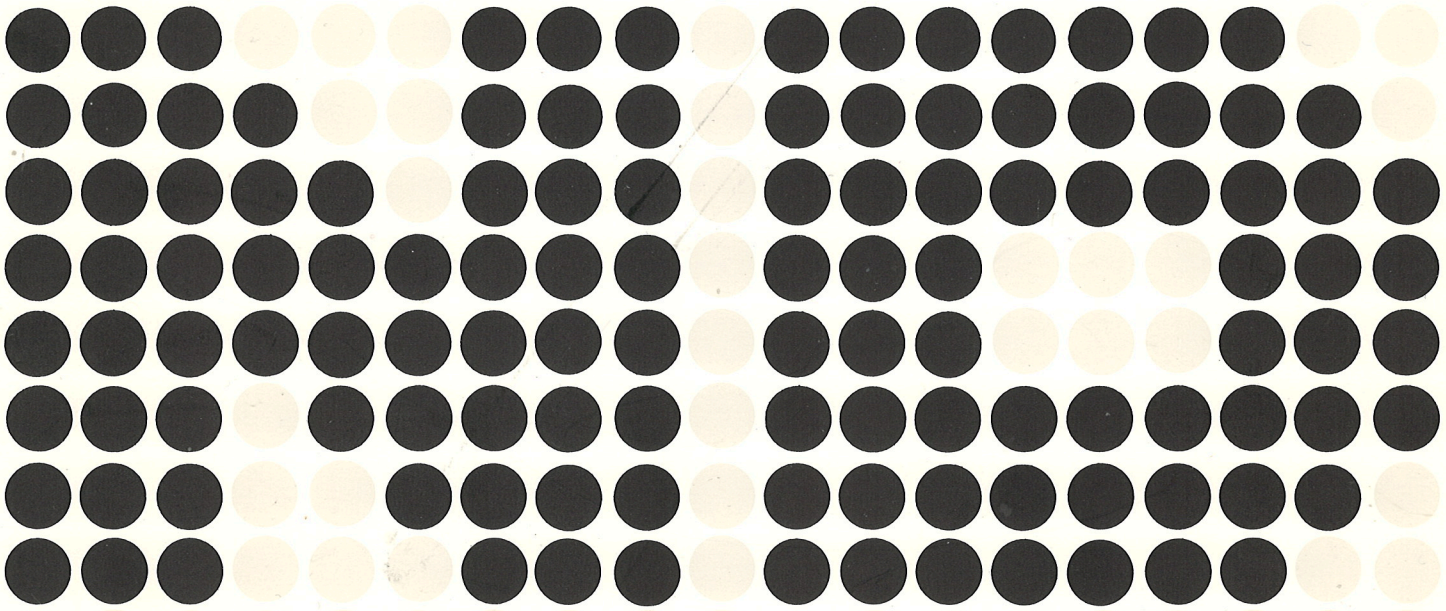
SINTRAN III
INTRODUCTION

NORSK DATA A.S



SINTRAN III
INTRODUCTION

NOTICE

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

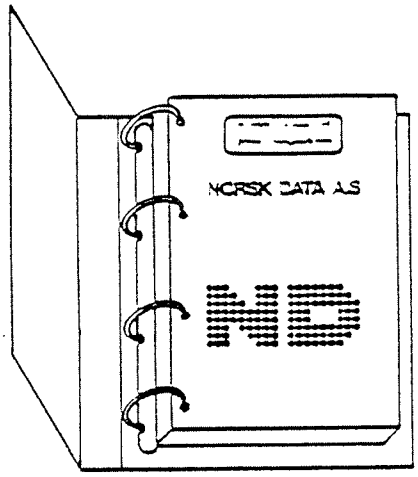
The information described in this document is protected by copyright. It may not be photocopied, reproduced or translated without the prior consent of Norsk Data A.S.

Copyright (C) 1979 by Norsk Data A.S.

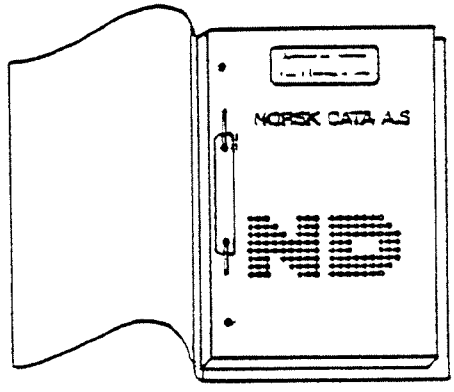
This manual is in loose leaf form for ease of updating. Old pages may be removed and new pages easily inserted if the manual is revised.

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

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



A Ring Binder



B Plastic Cover

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

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

ORDER FORM

I would like to order

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

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

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

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

PRINTING RECORD

Printing	Notes
09/79	Preliminary Issue
10/81	Second Version (Corresponds to the F Version of SINTRAN III)

ND-60.125.02 - SINTRAN III Introduction
October 1981

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

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

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

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

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

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

PREFACE

THE PRODUCT

SINTRAN III is the operating system presently used with ND computers from Norsk Data A.S. This manual corresponds to the F version of SINTRAN III.

THE READER

The manual is intended for the user who needs an introduction to SINTRAN III. It will be of special interest to those who have little knowledge of computers in general, but it will also be useful to other users of ND systems as an introduction to the more advanced manuals in the SINTRAN III series mentioned below.

PREREQUISITE KNOWLEDGE

The reader should preferably have some basic knowledge of computers for the greater part of the manual. However the first chapters can be understood by those without such experience.

THE MANUAL

Anyone new to the SINTRAN III operating system should read this introductory manual before attempting to use the other SINTRAN III manuals. It gives an overview of the most basic functions available in SINTRAN III, such as communicating with the system using a terminal, the basic concepts of the file storage system, and creating and running programs.

For the more experienced user, detailed information about SINTRAN III can be found in

SINTRAN III REFERENCE MANUAL ND-60.128.

SINTRAN III TIMESHARING BATCH GUIDE ND-60.132.

SINTRAN III REAL TIME GUIDE ND-60.133.

SINTRAN III COMMUNICATIONS GUIDE ND-60.134.

TABLE OF CONTENTS

1. THE SINTRAN III OPERATING SYSTEM

1.1. TO THE USER

If you have some basic knowledge of computers, we suggest that you turn straightway to chapter 2. This chapter is written for non-programmers who are about to use an ND computer terminal for the first time and who may want to have a basic explanation of how the system works.

So that you can understand what is happening at each stage, and what facilities are available to help you run your program, what follows will describe some fundamental terms and concepts. It will also be shown how SINTRAN fits into the system and why you need it.

SINTRAN recognises several types of user. This manual is written for the time-sharing user who will access the system through a terminal. The names of the other types and what these names mean will be found in chapter 7.

1.2. WHAT IS A COMPUTER?

A computer is nothing magical or superhuman. It only does what it has been instructed to do. But it is very flexible, and hence very many types of instructions can be given to it. Some of these instructions are short and simple, but some are not.

The process of converting the user's requirements into instructions the computer can understand, is called programming.

Example:

```
A = B + C

PRINT A
```

When Norsk Data delivers a machine, it provides some programs to help people get the most out of it. The most important of these standard programs form the operating system.

1.3. WHAT IS AN OPERATING SYSTEM?

The computer will appear to be able to do several things at once. The operating system is a program which enables the machine to handle these different tasks and to share out the facilities available. With ND machines the operating system is called SINTRAN.

1.4. WHO CAN USE SINTRAN?

Some of the users at any one time may be, for example:

A typist, needing to update a document.

An accountant, wanting to look at the financial data base.

A stock controller in a warehouse who needs to know how many of certain items he still has.

A programmer writing and testing his program.

One of the things SINTRAN does is to apportion computer time between such users, i.e., it schedules work.

2. USING THE TERMINAL

2.1. LOGGING-ON

When you want to use the computer and hence the operating system, you must first gain its attention. This is done by pressing the ESC (escape) button on the terminal, a process known as logging-on.

SINTRAN has been supplied with a list of all those people who can use the machine. If you are not yet one of these, see the system supervisor who will give you a user name.

After you have pressed ESC, the computer will respond with

```
ENTER
```

displayed on the terminal. On a display screen terminal, the cursor, a small blinking line, will show you where you are positioned to write.

Now type in your user name and press RETURN (or CR).

These keys you have just used need some explanation:

ESC (escape) is used initially to set up a connection with the operating system, i.e., when logging-on. Once this has been achieved, pressing ESC will terminate whatever is going on at the terminal, and it therefore can act as a panic button.

On some terminals you can also use a button called BREAK as a panic button.

RETURN, usually abbreviated to CR (carriage return), must terminate all lines which you enter into the system. The cursor will then move to the leftmost position.

The terminal now asks for a password. If you have one, type it in, and then press CR. Note that the system does not display the password you have entered. If you do not have a password, merely press CR.

If the accounting system is running you will also be asked for a project password. This is optional, its function is to identify who is paying for the computer time. Enter it if you have one, and press CR. Otherwise just press CR.

This completes the logging-on procedure and from now on the terminal will display the 'commercial at' prompt:

```
@
```

indicating that the system is waiting for your instructions.

2.2. LOGGING-OUT

When you have finished your session at the terminal, the way to log out is to enter logout, or just log, following a displayed @ prompt (see chapter 5.2).

```
@LOG
```

2.3. COMMANDS AND ABBREVIATIONS

Instructions or requests you give to the system are called COMMANDS.

There is a large number of commands which you can issue which request information from or give directives to the computer. These are described in detail in the SINTRAN III TIME SHARING/BATCH GUIDE. It is also possible to start one of the programs provided by Norsk Data. To do this you merely type in the program's name.

Whether you type in a command or a program name, you now press CR as described earlier in chapter 2.1. SINTRAN will then process the command or start up the program. When the task is complete, the command or program will return control to SINTRAN, which will now display @ once more to indicate that it is waiting for further instructions.

In general, a command has a command name, and most commands have parameters.

For example:

```
@WHO-IS-ON
```

WHO-IS-ON is underlined to indicate that it is entered by the user. This practise is followed throughout the manual.

WHO-IS-ON is a command needing no parameters. It returns the names of all other users besides yourself, together with the terminal numbers of the terminals they are using. Your terminal is marked by an arrow.

Most commands can be abbreviated so long as the abbreviation does not cause ambiguity between it and the abbreviation of another command, i.e., it must be unique. WHO-IS-ON can be abbreviated to WHO.

A more rigorous explanation of abbreviations is as follows:

A command name usually has several subfields separated by hyphens (minus signs). Thus WHO-IS-ON has three subfields, WHO, IS, and ON. When a command is abbreviated, the individual subfields may be shortened independently. Thus WHO-IS-ON could be written as

W-I-O, or WHO-I-O, or WH—ON

Subfields can be omitted completely from the end. Thus WHO-IS, or WHO, are also good abbreviations. The key point is that the overall name you write is an abbreviation of only one command. Otherwise SINTRAN will respond with the message AMBIGUOUS COMMAND and redisplay its expectant @.

Another command is:

@SET-TERMINAL-TYPE < terminal no. >,< terminal type >

which can be abbreviated to S-T-T.

This command has a parameter list. Parameters are shown inside angled brackets and are separated by commas. The lower case used here implies that you are to supply a value. If you try out this command you can enter the number of your own terminal which was given to you by the WHO-IS-ON command above.

@S-T-T 35,3

A command name is separated from its parameter list by a comma or a space. Either of these separators must also appear between each parameter which you specify. However, if one or more parameters are omitted, SINTRAN will use a default value in each case.

When a parameter is omitted, an extra separator must indicate its place. Thus, in the previous example you might enter:

@S-T-T ,3 or @S-T-T,,3

This will use the default value for terminal number, which happens to be your terminal (i.e. the terminal on which the command was given). If the last parameter of a list is omitted, the list must end with two commas.

If you omit any parameters and also omit the separators which indicate that the default value is to be taken, then the system will prompt you to specify the missing values, one at a time.

```
@S-T-T
TERMINAL NUMBER: 35
TERMINAL TYPE: 3
@
```

A list of useful commands is given in chapter 5.

2.4. ALTERATIONS

Mistakes in typing-in can be easily corrected. Another key to become familiar with is CTRL (control) which is operated like a shift key, i.e., it is held down while another key is pressed.

A mistyped character is erased by pressing the A key while holding down CTRL. (Known as CTRL A). If, for example, you wish to delete the last three characters, press CTRL A three times.

If you want to delete the whole line, press CTRL Q.

3. FILES

3.1. FILE NAMES

Before you can do any work you will probably need some data to work on. Such data is kept on what is called a file. A file is just a collection of related data held on a piece of computer hardware, generally a disk.

Files are referred to by a file name. This name can contain quite a lot of detail, and when you enter a file command the complete file name will be displayed for the file(s) involved, together with the information about the command's result.

However, for most purposes it is only necessary to know about two of the parts; usually you will use only the first.

The first part can contain up to 16 letters, digits, or hyphens and it can be anything you like, for instance:

```
MYFILE99
BEST-SO-FAR
```

3.2. FILE TYPES

The second part is the file type and it is separated from the first part by a colon. This part is not mandatory and consists of up to four letters or digits. A two-part file name might look like:

```
MYFILE99:DATA
```

If you do not specify the file type, then a default type will be supplied which is context dependent. A file type makes it possible to hold the same information in different ways. Some common types and their uses are:

:SYMB - This used for some human-readable files e. g. output from commands. It is the normal default and it is also the default for editors (see chapter 4.3).

:TEXT - The type used by the NOTIS text processing system

:BRF - This type is not required by ordinary users. It is the type for compiler output by default.

:PROG - A type used for programs that can be loaded and executed.

:DATA - The default when creating or deleting files.
For user data of various kinds.

:BPUN - The default for reentrant systems, etc.
Not required by ordinary users.

3.3. FILE COMMANDS

An example of a file command is:

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

which copies one file (the source file) to another (the destination file).

Entering:

```
@COPY-FILE DESTFILE,MYFILE1
```

copies the contents of MYFILE1:SYMB (note the default file type) to DESTFILE:SYMB.

Besides referencing you own files you can also reference those of another user. To do this you insert the other user's name, enclosed in parentheses, immediately in front of the name of the file in the file name parameter. Thus, the same command as above:

```
@COPY-FILE DESTFILE,(JOE)WORKFILE
```

copies the file WORKFILE:SYMB belonging to the user name JOE, to DESTFILE :SYMB. When you enter a file command you will receive extra information from SINTRAN in the response appearing on your screen which you can ignore at this stage. For instance the command:

```
@LIST-FILES MYFILE,,
```

will provide information about any file whose name has MYFILE as an abbreviation. Note the pair of commas denoting that a parameter is missing. In this case the default taken causes the ouput to be printed on the terminal. If such a file exists on the system the result will look rather like this:

```
FILE 0 : (PACK-1:USERA)MYFILE1:SYMB;1
```

USERA is the name of the owner of MYFILE1. If this file belongs to you, then your user name would have appeared in this position. PACK1 is the name of a file directory and although this information in the output is not important to you at the moment, it is helpful to know the purpose of a directory. This will now be described together with other file concepts.

3.4. FILE ORGANIZATION

When files are stored on a disk they are kept in an area associated with their particular user name. All the user areas together form a directory. Usually one disk pack or one floppy disk contain one directory.

A file held on disk occupies some space. This space is allocated (i.e.,reserved) in units called pages. The name page is deceptive however, since these units of storage may be scattered around the disk for reasons of flexibility. When the pages are thus distributed on the disk, one page is kept as an index to all the others. The files stored by this particular method are termed indexed. It is also possible to hold files on pages which are stored contiguously. Data retrieval may be faster in this case since there is no index to look up. Files stored by this method are called contiguous.

More terminology that you will read and hear about includes mass storage files which are those held on disk. Peripheral files are all the others, including magnetic tape, terminals, line-printers, Diablos, card readers etc.

3.5. USERS AND FILE ACCESS

Access to a file can be made for different purposes. For instance, you may want to find out what it contains, i.e., read it, or you may want to put something on it, i.e., write onto it, etc. The different kinds of access permitted to a file are:

R - read permitted
W - write permitted
A - append (expansion of the file) permitted
C - common access permitted
D - directory access (delete) permitted
N - no access permitted

However, not all users are normally allowed every type of access. The user categories and the default access types for each of these are as follows:

OWN : For the owner of a file the default is all access.

FRIENDS : The owner may have friends associated with him. For these the default access is read, write, and append.

PUBLIC : All other users. The only default access permitted

is read.

It is up to the owner of files to tell the system which other users are to be classed as friends. The owner can also define the access types for any friend, as well as change the default access for any category of user. The commands enabling him to do this are described in chapter 5.1.

4. RUNNING PROGRAMS

The most important function of SINTRAN III is to control the creation and execution of programs.

In earlier days, when computers were large, monstrous machines located in a room of their own, the machines were tended by operators in white overcoats. A program was created as a stack of cards and submitted to one of the white overcoats who ran the program while the programmer waited anxiously outside the computer room. After two or three hours of waiting (if lucky), the programmer was rewarded with a computer printout showing the results of the computer run. If there were mistakes in the program, he had to repeat the procedure over and over again until the program ran correctly. One programmer has vivid memories of working in a batch environment during the wintertime. The procedure included running between two buildings with the stack of cards under the arm. One day the road was particularly icy and ...

Today this is computer history. SINTRAN III allows us to complete the whole program development cycle while sitting at the terminal. We type the program on the keyboard and view the results on the screen. The program is stored for us on a disk. We can run the program when we want to, starting it from the terminal and seeing the results on the screen.

4.1. STARTING A READY-TO-USE PROGRAM

This section describes how to run a program which has already been created. It may originally have been received on a "floppy" before being copied onto a file in the system. This might be the case, for instance, with a "package" designed for a particular function. The program could also have been written by a terminal user and sent to one of the user's files.

A ready-to-use program (of type :PROG - see below) is started simply by entering the name of the program after the @ prompt (not forgetting to terminate the entry by pressing CR).

```
@PROGRAM-A
```

The system has a record of all names of programs available to it. The name you type in is first assumed to be a command. Only if SINTRAN does not know of such a command will it look for a program of the same name. Thus there is a search order which continues with examination of your own files for one with the specified name and type :PROG. Failing that, SINTRAN tries the files under the user name SYSTEM.

When the program has been found, it will be started and the instructions in the program will be carried out. The program will also have control of your terminal, the terminal entering "program mode". This means that all input from your terminal is now handled by this program instead of by SINTRAN. In program mode the programs will not

provide an @ prompt to request input. Instead they each provide their own characteristic symbol, like the editor QED (chapter 4.3) which uses * (asterisk).

The program will terminate itself when it reaches the end. If you wish to terminate the program before it finishes running, you can press ESCAPE as also described in chapter 2.1. This then cancels the program, and when the @ prompt is displayed on your terminal you will know you are back in "command mode".

4.2. CREATING A NEW PROGRAM

If you want to create your own program which can be run at a later stage, then you must first write your program into the computer using one of the editors provided by Norsk Data. This program will be written in a symbolic programming language such as Basic, Fortran, Cobol, Pascal, etc.

After the program has been written, it must be translated into the language the computer itself understands, called machine language, before it can be run. This translation, called compiling, is done by a special program called a compiler.

Finally the machine-language program must normally be loaded into the memory before it can be started (not always necessary for Basic programs). Another special program called a loader is used for this.

After writing, compiling, and loading, programs will be ready to start as described above.

These steps will be described in some more detail.

4.3. THE FUNCTION OF EDITORS (PED and QED)

As far as SINTRAN is concerned, editors are just programs which can be requested ("invoked") in the usual way. That is, by entering the name of the editor following the @ prompt.

Not only does an editor enable you to create a program, but it allows you to alter it or correct it, either during creation or sometime later. Hence the name.

PED and QED are editors commonly used. PED is the newest and easiest to use, but it is incompatible with some types of terminals. This chapter will first describe PED, and then QED.

4.3.1. PED

In PED, the entering procedure is:

```
@PED
```

which will produce the response:

```
*

          ****   *****  ****
          *   *  *      *    *
          ****   ***    *    *
          *      *      *    *
          *      *****  ****

     NORD PROGRAM EDITOR - SUT 2399F
```

Pressing CR will then move the cursor to the first line below the heading

```
LINES:01 - 20 / PED / hh.mm.ss (current time)

(...:....1....:....2....:....3....:....4....:....5.  etc.

-
```

You can now start writing in your program, using the cursor control keys with arrows and the CR key to place the cursor at the position where you want to write. If you make a mistake when typing in your program, you can move the cursor back and correct the mistake.

There are many commands and control functions in PED to help you write in your program. You can add text, delete text, and change text simply and quickly. See the PED User's Guide, ND-60.121, for more details.

Your program may look something like this:

```
LINES:01 - 20 / PED / hh.mm.ss

(...:....1....:....2....:....3....:....4....:....5.  etc.

      PROGRAM MARY

      WRITE (1,5)

5     FORMAT (1X,'MARY HAS BEEN EXECUTED')

      END
```

When you have typed in the whole program, press the "home" key (indicated by a slanted arrow on some terminals). The cursor will hop up to the upper lefthand corner. This is the "home position" where most commands must be given. You can now save your program on a file by entering W, which is the WRITE command. PED will ask you for the name of the file, and you answer the name followed by CR. If you use an already existing file (overwriting the previous contents), just give the file name. If you want to create a new file, enclose the file name in double quotes. End with CR.

```
*WRITE-FILE:"MYPROG"

LINES:01 - 20 / PED / hh.mm.ss

(...:....1....:....2....:....3....:....4....:....5.  etc.
```

If you want to change a program which already exists on a file, you will start out by reading the file with the READ command in home position:

```
*READ-FILE:MYPROG

LINES:01 - 20 / PED / hh.mm.ss

(...:....1....:....2....:....3....:....4....:....5.  etc.
```

Press CR to move the cursor down again. You may now change the program as you wish, move the cursor to home position, and write it back on the file with the WRITE command without quotes:

```
*WRITE-FILE:MYPROG

LINES:01 - 20 / PED / hh.mm.ss

(...:....1....:....2....:....3....:....4....:....5.  etc.
```

To leave PED and return to SINTRAN, enter E (for EXIT) in home position.

4.3.2. QED

In QED, the entering procedure is:

```
@QED
```

which will produce the response:

```
QED
*
```

The * (asterisk) is the symbol used by QED to indicate that it is ready to accept a command. If you now enter the command A, which is the command for "adding" text, you will be allowed to enter all the lines of your program. (Once again, remember to press CR after each line). After you have typed in the last line, press the CTRL button, and while holding this down press the "L" key as well. (The CTRL (control) button was previously mentioned in chapter 2.4).

This returns you to the QED command mode as you will see by the display of an asterisk on you screen. At this point you could enter W which is another QED command as shown in the example. (A description of QED commands is given in the QED Users Manual ND-60.031).

```
@QED

QED 4. 0
*A
        PROGRAM MARY

        WRITE (1,5)

5       FORMAT (1X,'MARY HAS BEEN EXECUTED')

        END

*W "MYPROG"

28 WORDS WRITTEN
*EX

@
```

In the example, W is the QED command for "write". It requests the program text just entered to be written on a file, here to be given the name "MYPROG". If, as in this case, you are creating a file which did not exist before, you must enclose the file name in double quotes (see the COPY-FILE command in chapter 5).

The command W "MYPROG" writes the program text on the file MYPROG:SYMB, where :SYMB is the default file type as explained in chapter 3.2.

QED, after telling you the size of your program, now asks for further instructions by displaying another *.

To return to SINTRAN you can use a further QED command: EX (for exit), as shown in the example. You will know you are back in SINTRAN command mode from the @ prompt now displayed on the terminal.

If, instead of creating a program you want to alter one which already exists on a file, then the filename should be referred to as follows, i. e. , without quotes:

```
@QED

QED
*R MYFILE
```

Here, another QED command R (or read) will make the text of MYFILE:SYMB available to this editor. Upon receiving the next asterisk you may enter further QED commands according to how you wish to modify the text.

4.4. COMPILING AND LOADING

After you have written your program on a file with an editor, you must compile (translate) the program from the programming language to a form of machine language called binary relocatable format (BRF format). After this has been done, the program still cannot be run until it has been loaded into the computer memory with the loader.

If your program is written in Fortran, as is the sample program in the previous section, the Fortran compiler must be started and given the command to compile your program, as follows:

```
@FTN

NORD 10/100 FORTRAN COMPILER FTN-2090H

$COM MYPROG,TERMINAL,"MYPROG"

    1*          PROGRAM MARY

    2*          WRITE(1,5)

    3*  5       FORMAT(1X,'MARY HAS BEEN EXECUTED')

    4*          END

4 LINES COMPILED , OCTAL SIZE=  55

CPU-TIME USED IS 0.1 SEC.

$EX

@
```

Typing FTN calls up the Fortran compiler; a response is given on the second line. On the third line you give the command to compile the program. (The $ sign is the Fortran compiler prompt character.) The input is MYPROG:SYMB (SYMB is the file type put there by PED or QED). The program list file is TERMINAL and the compiled output file is MYPROG:BRF. Again, BRF is the file type, put there this time by the Fortran compiler.

Note that MYPROG:SYMB and MYPROG:BRF are two different files, one containing the Fortran language program and one containing the compiled BRF program. It is not necessary to specify the type as part of the name however, since the correct type is used automatically. It is also not necessary to use the same name for the SYMB and BRF files.

Your BRF-file is now ready to be loaded and run.

```
@NRL

RELOCATING LOADER LDR-1935H

*LOAD MYPROG

FREE: 022203-177777

*RUN


MARY HAS BEEN EXECUTED

@
```

Typing NRL calls up the NORD relocating loader; the response is given on the second line. On the third line you give the command to load the program. After the program is loaded, you get a message indicating the amount of free space as shown on the fourth line.

On the next line you type RUN to execute the program. The program starts executing, prints the message MARY HAS BEEN EXECUTED as it should, and stops. You are now back in SINTRAN command mode.

This is a different way of starting an execution than the one described above under "starting a ready-to-use program". This is because your program was not ready-to-use, but had to be loaded first. It is possible to start a program directly from the loader, so we did it here to see if the program worked all right. However, when the execution finished, the loaded program disappeared.

If you want to keep your loaded program so it can be used again without reloading, you must "dump" it onto another file type, a PROG file. This is done in the loader with the DUMP command.

```
@NRL

RELOCATING LOADER LDR-1935H

*LOAD MYFILE

FREE: 022203 - 177777

*DUMP "MARY"

FREE: 034554 - 177777

*EX

@
```

The program is now ready to use with the name MARY:PROG. (The free area after the DUMP command is smaller than after the LOAD command because some extra routines were loaded from the Fortran library.) You can now start the program as described at the beginning:

```
@MARY


MARY HAS BEEN EXECUTED

@
```

5. COMMONLY USED SINTRAN COMMANDS

This chapter contains descriptions of some of the more commonly-used commands.

5.1. FILE SYSTEM COMMANDS

This is an important area since files contain our working data. More commands are associated with this area than any other.

Note that you can control where the output is to appear when "output file" is used as a parameter in a command. This output file can be an output device such as your terminal, or it can be a line printer, a diablo, card punch, etc. Each installation can have its own ouput file name for the various output devices and therefore these names are termed installation-dependent. However, it is probable that your installation will be using easy-to-remember names like TERMINAL (abbr.TERM) for the terminal, LINE-PRINTER (abbr. L-P) for the line-printer etc. These output devices are examples of peripheral files mentioned in chapter 3.

For commands which give us information about the system we have:

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

which will list names of files matching the parameter 'file name' on the 'output file'. Matching file names are those which include what you have entered in the first parameter. For example, if you type in:

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

then if files with file names WORKAREA, WORKING, WORKHABITS, etc. exist, information about them will be listed on your terminal, since TERMINAL is the default for the output file as mentioned above.

```
FILE 8 : (PACK-ONE:USER-X)WORKAREA-A:TEXT;1
FILE 12 : (PACK-ONE:USER-X)WORKAREA-A:OUT;1
FILE 18 : (PACK-ONE:USER-X)WORKING:TEXT;1
FILE 24 : (PACK-ONE:USER-X)WORK-HABITS:SYMB;1
FILE 28 : (PACK-ONE:USER-X)WORKING:OUT;1
FILE 49 : (PACK-ONE:USER-X)WORK-HABITS:BRF;1
FILE 68 : (PACK-ONE:USER-X)WORK-PROG:PROG;1
```

If you have just created a file, LIST-FILES can be used to check that it actually exists. If it does exist, you will receive information about it. If it does not exist, the message : NO SUCH FILE NAME will result.

You will get more complete information about files if you use:

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

Information not supplied by LIST-FILES includes how old and how big the file is, what kinds of access are permitted to the three user types : OWN, FRIENDS, and PUBLIC, and the number of times the file has been opened.

The command:

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

will change the name and/or type of a file.

For example:

@RENAME-FILE JOESFILE,FREDSFILE:SYMB

will change the name of JOESFILE to FREDSFILE and give it the file type SYMB. Again:

@RE-FI TOMSFILE:SYMB,:DATA

will change the file type of TOMSFILE to DATA.

When you are certain that one or more of your files are no longer required, it is good practise to delete them. This saves space on the system. The command to use is:

@DELETE-FILE < file name >

If you want to create a new file on which to store some output, then the technique for doing this can be illustrated by the use of the command:

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

Here the source file is copied onto the destination file. Where the destination file is to be a new one, its name is enclosed in double quotes as in the following:

@CO-FI "NEWMASTER",WORKFILE

and the name of the new file, NEWMASTER:SYMB, will be entered in the file directory.

Permitted access to files was described in chapter 3.5. The permitted access to an individual file can be set by:

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

where the access specification is any combination of those listed in chapter 3.5. For example:

```
@S-FI-ACC MYFILE:SYMB,R,W,RWAC
```

User types and associated access types were also mentioned in chapter 3.5. You can create a user as a friend (but note that you cannot declare yourself the friend of another user; that must be done by the other user) by writing:

```
@CREATE-FRIEND < user name >
```

The access to your files permitted to your friend will default to read, write, and append (provided the file permits this for friends) unless you change it with:

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

5.2. SOME OTHER COMMANDS

Should you need to know the date and current time of day there is:

```
@DATCL
```

a command which will list them both on your terminal:

```
10.36.10    14 SEPTEMBER   1981
```

If you need to know how much computer (CPU) time you have used and the duration of time since you logged on, this will be given by entering:

```
@TIME-USED
```

```
TIME USED IS  1 MINS  3 SECS  OUT OF  75 MINS  26 SECS
```

An obviously useful command is:

```
@HELP < command >,< output file >
```

Use this to aid your memory when you have forgotten an exact command name. Enter the part of the name you do remember as the first parameter. For example, assuming you only remember FRIEND as the second word, type in:

```
@HELP -FRIEND,,
```

Whereupon you will receive a list of all command names which match, in this case:

```
CREATE-FRIEND
DELETE-FRIEND
SET-FRIEND-ACCESS
```

Finally, to log off, use:

```
@LOGOUT  (or just @LOG)
```

The terminal will display the time and date followed by EXIT:

```
10.42.34      14 SEPTEMBER  1981

—EXIT—
```

and your terminal session is now finished.

5.3. THE USE OF MODE FILES

It is possible to do repetitive work by creating a so-called "mode file". This file contains a list of commands which you would normally enter at the terminal every time you wished to do a routine piece of processing. If you have these commands on a mode file then by merely giving the file name as a parameter to the MODE command, the series of commands on it will be executed automatically.

For instance, when taking back-ups (see chapter 6), suppose you need to make a back-up of all the symbolic files of a currently logged-on user, and to put the back-up on a magnetic tape.

Entering

```
@MODE BACK-UP
```

would perform all the commands on the file having the file-name "back-up" which you might have created as follows, using the editor PED:

```
*WRITE-FILE: BACK-UP
LINES:01 - 20 / PED / hh.mm.ss
(...:....1....:....2....:....3....:....4....:....5.  etc.
@BACKUP-SYSTEM
CREATE-VOLUME VOL-1,MAG-TAPE-1,0
C-U-F VOL,VOL-1,MAG-TAPE-1,0,1,DIR,MAIN-PA,,:SYMB,N
EXIT
```

Note that when entering the lines in the MODE file, you must insert the @ prompt at the beginning of SINTRAN commands.

Whenever this type of back-up is required from now on, only one MODE command needs to be entered. Other repetitive tasks can be similarly handled.

5.4. MAIL

It is possible for the central operator to send messages to other users. These messages can either be displayed on the screen immediately, or may be saved for access at the user's leisure.

If a message appears on your screen then it has no effect on the program you are executing, it is only for your information.

The messages which have been saved will be indicated when you attempt to log off, or immediately after you have logged on, by the message "YOU HAVE MAIL".

To read these messages, use the command:

```
@MAIL < filename >
```

As the default filename is TERMINAL, entering

```
@MAIL ,,
```

will cause the message to appear on your screen.

Following the message a prompt of * (asterisk) will appear. By typing EXIT you will return to SINTRAN.

It is also possible to send messages using the mail system. You can send a message to one particular user or broadcast to all users. For example:

```
@MAIL

*SEND-MESSAGE

TO USER: JOE-HANSEN

TYPE YOUR MESSAGE, TERMINATED BY CONTROL L:

HI, JOE, HOW ARE YOU TODAY?    (control L)

*EX

@
```

6. TAKING BACKUPS

No matter how reliable a computer is, it will go wrong at some point. In order to protect ones files from machine malfunction (e.g., a "disk crash") it is advisable to take a back-up periodically.

A disk-to-disk copy is usually done once a week by the computer installation staff. So, in the worst case, you might lose a week's work. However, you can take your own back-up, of all or part of your files, when you have something that you particularly do not want to lose.

Back-ups can be taken on a large disk, magnetic tape, or floppy.

A special program, called BACKUP-SYSTEM, is available for taking backup. This program has several functions, but most users will use it mainly for copying files. The COPY-USERS-FILES command can be used both for taking the back-up and for recovering, should you have to. Two sets of parameters are required, one to describe what you are copying to, the other to describe the source, i. e. , what you are copying from. The files you have been working on are in a directory, on disk. The back-up files will also be on a directory if they are on a large disk or floppy. On tape they will be on a volume.

The backup is taken by starting the BACKUP-SYSTEM program and giving the command COPY-USERS-FILES:

```
@BACKUP-SYSTEM

BA-SY:COPY-USERS-FILES < destination >,< source >
```

For example, to copy all of your files of type :TEXT onto a magnetic tape:

```
@BACKUP-SYSTEM
BA-SY:COPY-US-FI
   DESTINATION TYPE: VOL
   DEST. VOLUME-NAME: BACKUP
   DEST. DEVICE-NAME: MAG-TAPE-1
   DEST. UNIT-NUMBER: 0
   DEST. FILE-GENERATION: 1
   SOURCE TYPE: DIR
   SOURCE DIRECTORY-NAME: PACK-ONE
   SOURCE USER-NAME: JOHN-SMITH
   SOURCE FILE-NAME: :TEXT
   MANUAL CHECK: Y
BA-SY:EX
@
```

The first parameter (VOL) indicates the type of the destination, here it is a volume.

The second parameter (BACKUP) is the name of the volume. (The system will make sure that the magnetic tape on the designated unit has the name you specify).

The third parameter (MAG-TAPE-1) is the device on which the magnetic tape is mounted.

The fourth parameter (0) is the unit number on that device.

The fifth parameter (1) is the generation number, which allows several different copies of the same files to be keep on one volume.

The sixth parameter (DIR) starts the description of the source. Here it is a directory.

The seventh parameter (PACK-ONE) is the name of the directory.

The eighth parameter (JOHN-SMITH) is the name of the user whose files are to be copied.

The ninth parameter (:TEXT) names all the files you wish to be copied, in this case all files of the type TEXT.

The tenth parameter (Y(es)) means that for each file to be copied, the file name will be written on the terminal and the program will wait until you indicate whether the file really should be copied (press C) or whether it should be skipped (press CR).

To recover these files one may write:

```
BA-SY:C-U-F DIR,PACK-ONE,JOHN-SMITH,
                VOL,BACKUP,MAG-TAPE-1,0,1,:TEXT,N
```

where the first three parameters describe the destination, and the remainder describe the source.

Note: Volumes are created on new tapes by the command:

```
@BACKUP-SYS
BA-SY:CREATE-VOLUME < volume name >,< device name >,< unit >
```

Floppies are best used for maintaining copies of a few crucial files. If you need to create a directory and users on a floppy, refer to the SINTRAN III Time-sharing/Batch Guide, ND-60.132.

To copy a single file, you can use the COPY-FILE command in SINTRAN:

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

For example:

```
@COP-FIL (FLOPPY:USER)FILEA:DATA,FILEA:DATA
```

which will copy your file called FILEA:DATA onto the floppy disk. Note that the floppy disk contains a normal directory and that you are conforming to the normal file-naming conventions.

Note: Before you use a floppy you must use the command ENTER-DIRECTORY. Similarly, when you have finished using the floppy you must use the command RELEASE-DIRECTORY. See the Time-Sharing/Batch Guide quoted above.

7. SINTRAN USER CATEGORIES

Apart from the time-sharing user there is also the batch user. Instead of entering commands and data at a terminal, he defines all his input ahead of time as a file or a deck of cards. This job can be run independently of any terminal or user interaction. It is controlled by a 'batch processor'.

Time-sharing and batch users are also called 'background' users of SINTRAN. They are known to the system by names which must be presented either at the beginning of each terminal session or as the first command in a batch job. By choosing a password to accompany the user name, the user can be protected from unauthorised access to the system.

Time-sharing and batch programs are given processing time on an equal, time-shared basis. A set of priorities are assigned to them and these priorities are varied on a round-robin basis by a time scheduler. These programs are normally executed with lower priorities than the real time processes described below.

The real-time user is recognised by SINTRAN by the special user name of RT. The user RT has access to some special SINTRAN commands but is otherwise a normal time-sharing user.

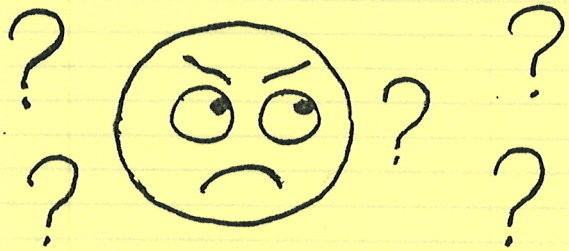
One of the most important of these special commands is the RT command. This starts the execution of a real time (RT) process. Real time in this context usually means that the process is doing time critical work, either requiring rapid processing, periodic or time-dependent execution or special facilities available only to RT programs. RT processes may be responding to or controlling external events, and they can be given higher priority than those of time-sharing and batch programs. RT processes are independent of terminals and batch processors, they have no log-in procedure to identify a user, and processing time is distributed according to a fixed priority assigned to the process.

The system-supervisor user is in a category of his own. Another special user name, SYSTEM, identifies him to SINTRAN. The system supervisor has responsibility for the installation and he can do such things as allocate disk space, create and delete users, and update the operating system itself.

INDEX

INDEX

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

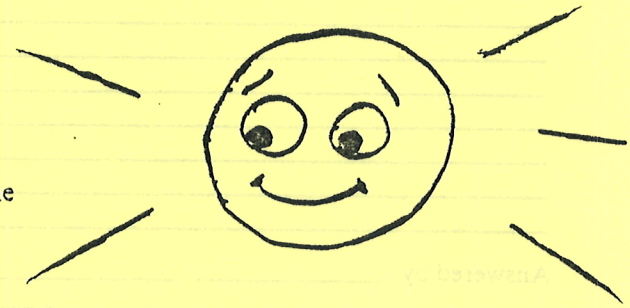


Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card - and an answer to your comments.

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: Manual number:

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

Do you have suggestions for improving this manual? _____

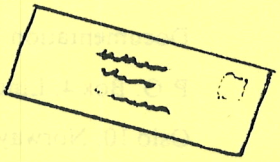
Your name: _____ _____ _____ Date: _____

Company: _____ _____ _____ Position: _____

Address: _____

What are you using this manual for? _____

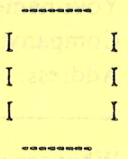
Send to: Norsk Data A.S.
Documentation Department
P.O. Box 4, Lindeberg Gård
Oslo 10, Norway



Norsk Data's answer will be found on reverse side

Answer from Norsk Data _____

Answered by _____ _____ _____ _____ Date _____



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

– we make bits for the future

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