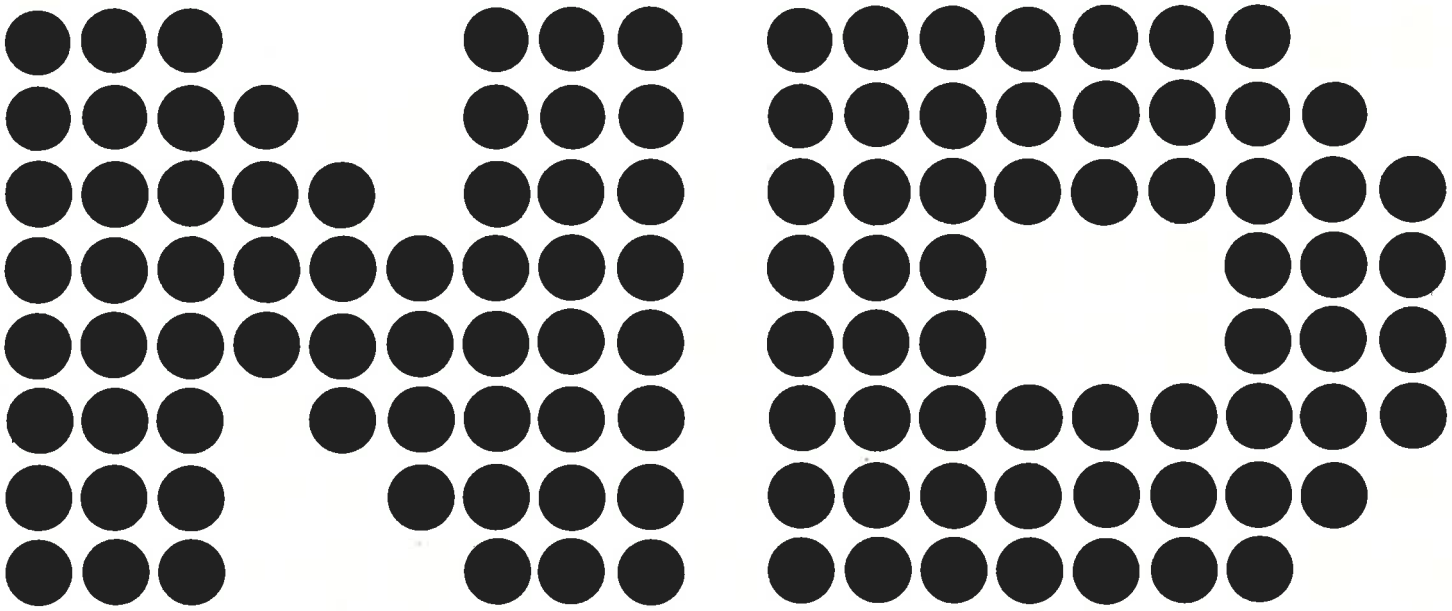
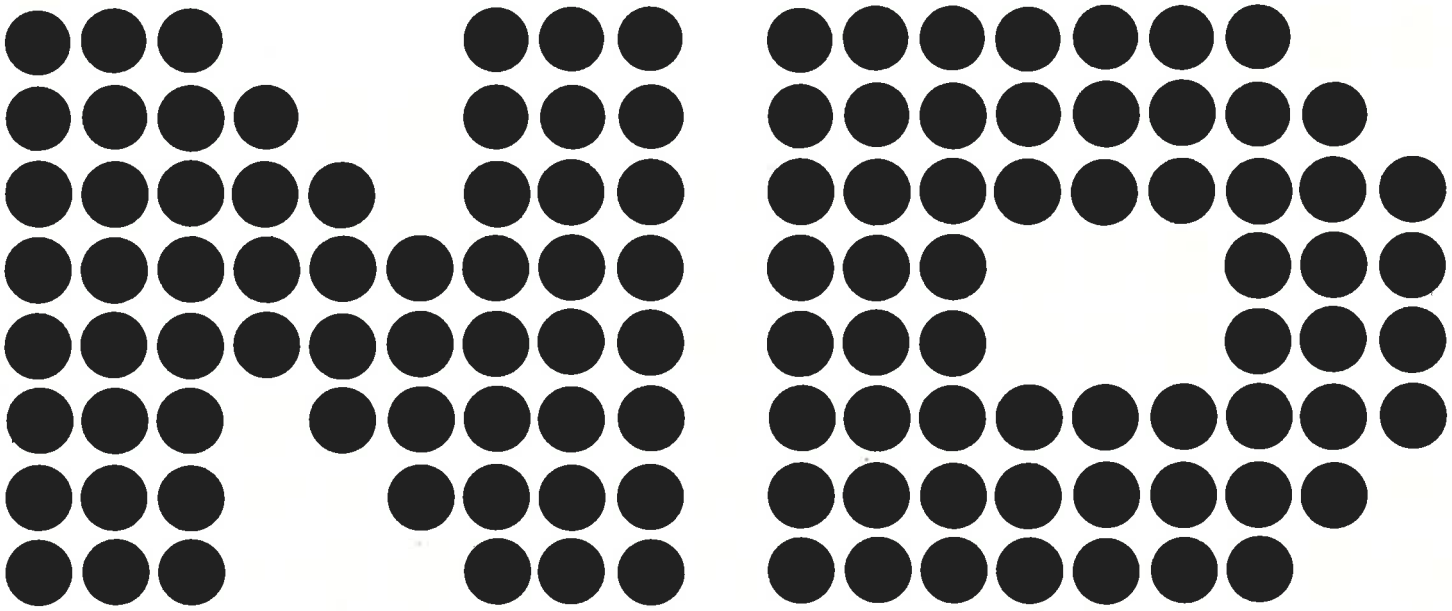
NORD FILE SYSTEM

NORSK DATA A.S



NORD FILE SYSTEM

NORSK DATA A.S



NORD FILE SYSTEM

REVISION RECORD

Revision	Notes
10/74	Version two, superseding the original printing
08/76	Version three, superseding previous editions
04/77	Version four, superseding previous editions
09/77	Revision A
	The following pages have been revised: 3--7, 3--8.

NORD FILE SYSTEM
Publication No. ND-60.052.04

NORSK DATA A.S.

Lørenveien 57, Postboks 163 Økern, Oslo 5, Norway

TABLE OF CONTENTS

+ + +

Section:		*Page:*
1	**INTRODUCTION**	1–1
2	**FILE SYSTEM USE**	2–1
2.1	Commands and Monitor Calls	2–1
2.2	File Creation and Use	2–2
2.3	File Types	2–4
2.4	File Versions	2–7
2.5	Owners and Friends	2–10
2.6	File Protection and Access Modes	2–11
2.7	Directories	2–16
2.8	Peripherals	2–19
2.9	Tapes	2–20
2.10	The Spooling System	2–21
3	**DEVICE LAYOUT**	3–1
3.1	Disk and Drum	3–1
3.1.1	Master Block	3–2
3.1.2	Bit File	3–2
3.1.3	User File	3–2
3.1.4	Object File	3–4
3.1.5	Data Files	3–5
3.1.6	File Layout	3–5
3.2	Magnetic Tape	3–7
3.2.1	Magnetic Tape Directions	3–7
3.2.2	Magnetic Tape Volumes	3–7
3.3	Peripherals	3–9
4	**SYNTAX**	4–1
4.1	Symbolic Names	4–1
4.2	File Names	4–2

Section:		*Page:*
4.2.1	Directory Names	4–3
4.2.2	User Name	4–3
4.2.3	Object Name	4–3
4.2.4	Type	4–4
4.2.5	Version	4–4
4.3	Device Names	4–5
4.4	Access Strings	4–6
4.4.1	Legal Access String	4–6
4.4.2	Open Access String	4–6
5	**COMMANDS**	5–1
5.1	Directory Commands	5–2
5.1.1	Create Directory	5–2
5.1.2	Rename Directory	5–2
5.1.3	Enter Directory	5–2
5.1.4	Release Directory	5–2
5.1.5	Set Default Directory	5–3
5.2	User Commands	5–4
5.2.1	Create User	5–4
5.2.2	Rename User	5–4
5.2.3	Delete User	5–4
5.2.4	Give User Space	5–4
5.2.5	Take User Space	5–5
5.2.6	Enter User	5–5
5.2.7	Change Password	5–5
5.2.8	Clear Password	5–5
5.2.9	Log Out	5–5
5.3	Friend Commands	5–6
5.3.1	Create Friend	5–6
5.3.2	Delete Friend	5–6
5.3.3	Set Friend Access	5–6
5.4	File Creation and Deletion	5–7
5.4.1	Create File	5–7
5.4.2	Create New Version	5–7
5.4.3	Allocate File	5–7

Section:		*Page:*
5.4.4	Allocate New Version	5–8
5.4.5	Expand File	5–8
5.4.6	Rename File	5–8
5.4.7	Set Terminal File	5–8
5.4.8	Set Peripheral File	5–8
5.4.9	Delete File	5–9
5.4.10	Delete-Users-Files	5–9
5.5	File Protection and Access	5–10
5.5.1	Set File Access	5–10
5.5.2	Open File	5–10
5.5.3	Connect File	5–10
5.5.4	Open RT File	5–11
5.5.5	Connect RT File	5–11
5.5.6	Close File	5–11
5.5.7	Close RT File	5–11
5.5.8	Set-Permanent-Opened	5–11
5.5.9	Open-Scratch-File	5–11
5.5.10	Set-Block-Size	5–12
5.5.11	Set-Byte-Pointer	5–12
5.5.12	Set-Block-Pointer	5–12
5.5.13	Reserve File	5–12
5.5.14	Release File	5–12
5.5.15	Where-Is-File	5–12
5.5.16	Reserve Device Unit	5–12
5.5.17	Release Device Unit	5–13
5.6	Backup and Test Commands	5–14
5.6.1	Copy File	5–14
5.6.2	Copy Directory	5–14
5.6.3	Save Directory	5–14
5.6.4	Copy Device	5–14
5.6.5	Regenerate Directory	5–14
5.6.6	Back-up on Labeled Magnetic Tapes	5–15
5.7	Statistics Commands	5–19
5.7.1	List Directories Entered	5–19
5.7.2	Directory Status	5–19
5.7.3	List Users	5–19
5.7.4	User Statistics	5–19
5.7.5	List Friends	5–19

Section:		*Page:*
5.7.6	List Files	5–20
5.7.7	File Statistics	5–20
5.7.8	List Opened Files	5–20
5.7.9	List RT-Opened Files	5–20
5.8	Maintenance Commands	5–21
5.8.1	Dump Directory Entry	5–21
5.8.2	Change Directory Entry	5–21
5.8.3	Dump User Entry	5–21
5.8.4	Change User Entry	5–21
5.8.5	Dump Object Entry	5–21
5.8.6	Change Object Entry	5–22
5.8.7	Dump Bit-File	5–22
5.8.8	Change Bit-File	5–22
5.8.9	Dump Page	5–22
5.8.10	Change Page	5–22
5.9	Spooling Commands for the User System	5–23
5.9.1	Start Spooling	5–23
5.9.2	Stop Spooling	5–23
5.10	Spooling Commands for Public Users	5–24
5.10.1	Append Spooling File	5–24
5.10.2	Delete Spooling File	5–24
5.10.3	List Spooling Queue	5–24
5.10.4	Abort Print	5–24
5.10.5	Restart Print	5–24
5.10.6	Give Spooling Pages	5–25
5.10.7	Take Spooling Pages	5–25
5.10.8	Spooling Pages Left	5–25
6	**MONITOR CALLS**	6–1
6.1	Input One Byte	6–2
6.2	Output One Byte	6–3
6.3	Read Disk	6–4
6.4	Write Disk	6–5
6.5	Read Page	6–6
6.6	Write Page	6–7
6.7	Read Object Entry	6–8
6.8	Close File	6–9
6.9	Read User Entry	6–10

Section:		*Page:*
6.10	Open File	6–11
6.11	Read Max Pointer	6–12
6.12	Write Error Message	6–13
6.13	Write Error Message and Leave	6–14
6.14	Set Max Pointer	6–15
6.15	Set Byte Pointer	6–16
6.16	Read Byte Pointer	6–17
6.17	Set Block Size of a File	6–18
6.18	Set Block Pointer of a File	6–19
6.19	Read File	6–20
6.20	Write File	6–21
6.21	Wait File	6–22

Appendix:

A	**ERROR CODES RETURNED FROM MONITOR CALLS**	**A–1**

1 INTRODUCTION

The NORD-10 File System is designed to manipulate files on disks, drums, magnetic tapes, cassette tapes or standard peripherals. A "file" in this context means a collection of records or blocks, ordered randomly or sequentially.

The file system is designed to operate as a 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. When a file is accessed, the file name must be connected to a file number, and this number is used in the access routines.

Each file in the file system has one owner, which must be defined as a user of the file system. Each user may have several other users as friends. The file system provides individual protection of files, with separate protection modes for the owner, the owner's friends and the public's access of the file.

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

The next chapter in this manual is an introduction to the file system, and the remaining chapters contain reference information.

2 FILE SYSTEM USE

2.1 *COMMANDS AND MONITOR CALLS*

The file system is activated in two ways:

- through commands or
- through monitor calls.

File system commands are used to organize files, for instance, to create files, delete files, etc. A command is a string of characters separated in fields. Field separators are space or comma. The first field is the command name, and the next fields are the parameters to this command. The file system will always ask for missing parameters to a command.

Example:

```
@CREATE-FILE  FILE-ONE, 10
```

This is a command to create a file. The command string has three fields, the first field is the command name (CREATE-FILE), the second field is the name of the file to be created (FILE-ONE), and the last field is the file size (10). The @ is the sign from the command monitor that it is ready to accept a new command, and is not written by the user. (Underlined characters are written by the system.)

The fields in a command may be abbreviated in many ways. A field abbreviation is accepted by the file system as long as the field is not ambiguous.

Example:

The command in the example above could have been abbreviated in the following ways:

```
@CREATE-FI    FILE-ONE, 10
@CRE-FI       FILE-ONE, 10
@C-FILE       FILE-ONE, 10
```

The - serves as a subfield separator, and each subfield may be abbreviated separately.

File system monitor calls are used to access files from user programs. A monitor call is the NORD-10 instruction MON, and parameters to the file system are transferred in the registers.

Example:

```
SAT 1
MON 1
```

This is a monitor call to read one byte from file number 1. (File number is transferred in the T register.)

2.2 *FILE CREATION AND USE*

A file is always accessed through monitor calls, but this may be transparent to the user. If, for instance, he uses a subsystem under TSS or SINTRAN III, he may access a file by means of its symbolic name.

Example:

A QED user wants to read a file called FILE-ONE with symbolic information. He may then write:

```
*R  FILE-ONE
```

as a command to QED. The file name may be abbreviated, so he can also write:

```
*R  F-O
*R  FILE
*R  -ONE
```

etc. as long as the name is not ambigious.

If a subsystem user wants to create a new file, he may do this by putting the file name in quotes (").

Example:

A QED user wants to write on a new file which he wants to call FILE-TWO. He may then write:

```
*W "FILE-TWO"
```

as a command to QED.

Note that in this case, the file name should not be abbreviated.

If the subsystem does not accept symbolic file names, only file numbers, the file must be opened before the subsystem is used. A file command called OPEN-FILE opens the file and returns a file number which the user may use for file access. The OPEN-FILE command has two parameters. The first parameter is the file name and the second parameter is the access mode. The file may be opened for sequential or random access, and for read or write access.

Example:

```
@OPEN-FILE     FILE-ONE, R
FILE NUMBER IS 5
```

The file may now be read sequentially with file number 5 (i.e., with *R 5 in QED). From an assembly coded user program, the file may be accessed with the INBT monitor call (MON 1):

```
SAT     5
MON     1
JMP     ERROR
STA     BYTE
```

The INBT call reads one byte from a file opened for sequential read access. The file number is transferred in the T register and the byte is returned in the A register. Return to first instruction after MON 1 means error, return to second instruction means correct transfer.

Similarly, a file may be opened for sequential write and accessed with the OUTBT monitor call (MON 2):

```
@OPEN-FILE    FILE-TWO, W
FILE NUMBER IS 6
```

Assembly program:

```
LDA     BYTE
SAT     6
MON     2
JMP     ERROR
```

A new file can be created with the OPEN-FILE command with the file name in quotes.

Example:

```
@OPEN-FILE    "FILE-THREE", W
FILE NUMBER IS 7
```

No space is allocated for the file, but space is allocated when something is written on the file.

If a user wants a file with a fixed amount of space, he can create the file with the CREATE-FILE command. This command has two parameters; file name and number of pages (2K bytes) in the file. The file space is allocated as a continuous area on the disk or drum.

Example:

```
@CREATE-FILE        FILE-FOUR, 10
```

Create a file with name FILE-FOUR with 10 pages (20K bytes). Note that quotes are not used here.

2.3 *FILE TYPES*

Two files cannot have the same file name. It is illegal to create a file with the same name as an existing file.

Example:

```
@CREATE-FILE        FILEXX, 10
@CREATE-FILE        FILEXX, 20
FILE ALREADY EXISTS
```

But two files with the same name may be distinguished with another identification, called the file type. The file type is a character string added to the file name string. The file name and the file type are separated with a:

Example:

```
@CREATE-FILE        FILEXX: BIN, 10
@CREATE-FILE        FILEXX: DATA, 20
```

A file type is always inserted by the file system if not specified by the user. Default file type in CREATE-FILE is DATA and default file type in OPEN-FILE is SYMB.

Example:

```
@CREATE-FILE        FILEXX, 10
```

gives a file which may be referred to as FILEXX: DATA, i.e., in an OPEN request:

```
@OPEN-FILE          FILEXX: DATA, W
FILE NUMBER IS 101
```

If the file is created in an OPEN-FILE command, the file should be referred to as FILEXX:SYMB.

```
@OPEN-FILE          "FILEXX", W
FILE NUMBER IS 101
@OPEN-FILE          FILEXX:SYMB, R
FILE NUMBER IS 102
```

or, as SYMB is default for OPEN-FILE

```
@OPEN-FILE          FILEXX, R
FILE NUMBER IS 102
```

Various subsystems under SINTRAN III may have other default values for file type.

Example:

In the subsystem MAC, file names may be used in)9.ASSM command. The first file is input file with default type SYMB. The second file is list file, also with SYMB as default type. The third file is object file with default type BRF. Thus, writing

```
)9ASSM  FILE1, FILE2, FILE3
```

is the same as writing

```
)9ASSM  FILE1:SYMB, FILE2:SYMB, FILE3:BRF
```

The type identification serves another purpose, other than the file name extension. It ensures that correct file type is accessed and protects the user from writing on wrong files.

Examples:

Suppose a user has two files identified by FILE1:DATA and FILE2:SYMB. If, from the subsystem QED, he wants to write on the first one, he writes:

```
*W  FILE1
```

he will receive an error message, as SYMB is default type in QED. If he wants to access FILE1, he must write

```
*W  FILE1:DATA
```

or an abbreviation of this, ie.,

```
*W  FILE1:DA
```

or even

```
*W  FILE1:
```

as long as he has only one file called FILE1.

If a file is no longer needed, it may be deleted from the file directories with a file command called DELETE-FILE. Parameter to DELETE-FILE is the file name.

The command DELETE-FILE has no default type value. Thus, the type must be specified.

Example:

```
@DELETE-FILE        FILE1:DATA
```

or an abbreviation:

```
@DELETE-FILE        FILE1:D
```

2.4 *FILE VERSIONS*

The file version concept in the file systems is a complex concept with different purposes and should be used with care. Versions should only be used when needed and not to distinguish between independent files.

A file may be created in one or more versions. If this is not specified, the file is created in one version.

Example:

```
@CREATE-FILE        FILEXX, 10
```

will create one version of the file FILEXX.

If more than one version is needed, a version identifier must be specified in the file name. This identifier is a number following the file name and type, separated from the name and type with a semicolon (;). The maximum number of versions of one file is 256.

Example:

```
@CREATE-FILE        FILEXX; 3, 10
```

Create three versions of the file FILEXX, each version with 10 pages.

```
@CREATE-FILE        FILEYY:SYMB; 4, 10
```

Create four versions of the file FILEYY:SYMB, each version with 10 pages.

If more than one version of a file is created, the user must select one version of the file when he wants to use it. This selection is also done with the version number identifier.

Example:

Open second version of the file FILEXX:DATA for read:

```
@OPEN-FILE          FILEXX:DATA; 2, R
```

Open third version of the file FILEYY:SYMB for write:

```
@OPEN-FILE          FILEYY; 3, W
```

If a file is created in only one version, this is version one and may be accessed as version one.

Example:

```
@CREATE-FILE        FILEWW, 2
@OPEN-FILE          FILEWW:DATA; 1, W
```

If a file is created in more than one version, the file may be accessed without specifying a version. The file system will then select one of the existing versions. This automatic selection of file versions is the only advantage to files with more than one version.

The actual version selected in this case depends on the kind of use. The version normally selected is version one.

Example:

Suppose FILEYY:SYMB exists in four versions and a QED user reads this file:

```
*R  FILEYY
```

In this case, version one is selected, so this is the same as:

```
*R  FILEYY; 1
```

or

```
*R  FILEYY:SYMB; 1
```

There are important exceptions from the "select version one" rule. These exceptions will be described later. (Sections 2.6, 2.8 and 2.10.).

If a file is created with the ". . . ." notation, it is also possible to create many versions, however, only one is opened for access.

Example:

```
@OPEN-FILE          "DUMMYFILE; 3", R
FILE NUMBER IS 101
```

The first three versions of the file DUMMYFILE:SYMB are created, then version one is opened. This is the same as:

```
@CREATE-FILE        DUMMYFILE:SYMB; 3, 0
@OPEN-FILE          DUMMYFILE; 1, R
FILE NUMBER IS 101
```

It is also possible to create new versions of a file, either with a special command, CREATE-NEW-VERSIONS, or by putting the version number only in quotes.

Example:

Suppose a FILE1:DATA exists in two versions. The version three may be created with:

```
@CREATE-NEW-VERSIONS        FILE1; 3, 10
```

or with

```
@OPEN-FILE                  FILE1:DATA; "3", W
FILE NUMBER IS 101
```

Note that, in this case, version three is also opened.

If

```
@CREATE-NEW-VERSIONS        FILE1; 4, 10
```

was written, both version 3 and version 4 will be created, each with 10 pages. This could also be done by writing

```
@OPEN-FILE                  FILE:DATA; "4", R
FILE NUMBER IS 102
```

Again, the specified version (version four) will be opened.

It is also possible to insert new versions of a file.

Example:

Suppose versions one, two and three exist of the file FILEXX:DATA. A new version two may be created with

```
@CREATE-NEW-VERSIONS        FILEXX; 2, 5
```

Old version two is now version three and old version three is version four.

2.5 *OWNERS AND FRIENDS*

Each file in the file system has an owner. If a user is logged in on a terminal and creates a file, he is defined as the owner of the file. One user may obtain access to other users' files. To do this, he must specity the owner of the file. The owner name is specified as a prefix to the file name. The prefix consists of the owner name in parenthesis.

Example:

A file called FILE-ONE, created by user USER-TWO, may be accessed as

(USER-TWO) FILE-ONE

A new user of the file system is introduced with a command called CREATE-USER. Parameter to this command is the name of the new uer.

Example:

@CREATE-USER USER-THREE

Before the user can create any files, he must be given space on the disk. This may be done with a GIVE-USER-SPACE command with parameters, user name and number of pages on the disk to be reserved for this user.

Example:

@GIVE-USER-SPACE USER-THREE 53

will give USER-THREE 53 pages (106K bytes) of disk space.

The space reserved for user will be used both for data pages and pages for file directories.

The two commands CREATE-USER and GIVE- USER-SPACE are examples of restricted commands. A restricted command is a command that may only be issued by a user called SYSTEM. User SYSTEM should be the very first user created.

A user may define up to 8 other users as friends. A friend may have access to files that are not accessible for non-friend (public) users.

A friend is created with the CREATE-FRIEND command.

Example:

@CREATE-FRIEND USER-ONE

Create user USER-ONE as friend of the currently logged in user.

2.6 *FILE PROTECTION AND ACCESS MODES*

A file, in the file system, may be accessed in many different ways. The access mode of a file is defined when the file is opened. That means that one file may be accessed in many different modes, but it must be closed and opened again to change access mode.

Example:

```
@OPEN-FILE                    FILE-ONE, R
```

Open file FILE-ONE for sequential read access.

The access mode is always specified as last parameter in OPEN-FILE as a character string. The character string must be a combination of the following characters:

R	read access
W	write access
X	random access
A	append
C	common access (write allowed for more than one user simultaneously on continuous or allocated files)

A list of legal combination of the characters is found in Section 4.4.2.

The access mode of a file may be restricted, either due to physical reasons or because the file is protected against some access modes.

Example:

The line printer may be accessed through the file system with file name LINE-PRINTER. The line printer may only be opened for write. Thus, commands like:

```
@OPEN-FILE                    LINE-PRINTER, R
(sequential read access),
```

or

```
@OPEN-FILE                    LINE-PRINTER, RX
(random read access)
```

are illegal.

The legal access modes of a file may be changed any time by a user with sufficient access to the file. Legal file access is defined for three groups of users: one legal access mode is defined for the owner of the file, one mode for the owner's friends, and one mode for all other users (public).

The legal access to a file is changed with a command, SET-FILE-ACCESS, with four parameters. First parameter is file name, second parameter is legal access for public, third parameter is legal access for friends and last parameter is legal access for the owner.

The legal access is defined as a combination of the following characters:

R	read permitted
W	write permitted
A	append permitted
C	common access permitted
D	directory access permitted

Append permitted means that the file may be expanded when used, and directory access means the file may be deleted, legal access mode may be changed or new versions of the file may be created.

If a legal access parameter is omitted, the current access allowed for this user group is not changed.

Example:

```
@CREATE-FILE          FILEXX, 10
@SET-FILE-ACCESS      FILEXX, RW,, DWR
```

FILE XX may now be read or written by all users, but not expanded by the file owner. Friend access is not changed.

Default access when a file is created is:

For public	read permitted
For friends	read, write and append permitted
For owner	all access permitted

The default file access may be changed for each individual user by the @SET-DEFAULT-FILE-ACCESS command.

Example:

Suppose a file is created by USER-ONE, which has USER-TWO as friend. This is done by the following command (issued by USER-ONE):

```
@CREATE-FILE          FILE-ONE
@CREATE-FRIEND        USER-TWO
```

USER-TWO may now open the file, i.e., for sequential write append (insert new information at the end of the file):

```
@OPEN-FILE                    (USER-ONE) FILE-ONE, WA
```

But he cannot delete the file or create a new version:

```
@OPEN-FILE                    (USER-ONE) FILE-ONE; "2", W
NOT DIRECTORY ACCESS
```

The eight different friends of a user may have different file access restrictions. The restrictions for each friend is defined with the SET-FRIEND-ACCESS command, with parameters friend name and a legal access string (consisting of a combination of R, W, A, C and D).

Example:

```
@SET-FRIEND-ACCESS            USER-TWO, RCD
```

The friend USER-TWO is given read, common and directory access to the files of the logged in user.

Thus, a friend's access to a file is defined both by his access allowed in general (with the SET-FRIEND-ACCESS command) and by the friend access permitted to the file in question (defined with the SET-FILE-ACCESS command).

Some special considerations apply for files with more than one version, because access restrictions may be defined separately for each version. Then, if a user opens such a file without specifying version number, he usually will get access to the version he has access to with the lowest version number.

Example:

A user creates a file in two versions and gives them different kinds of protection.

```
@CREATE-FILE                  FILEXX;2, 40
@SET-FILE-ACCESS              FILEXX;1, R, RW,,
@SET-FILE-ACCESS              FILEXX;2, RW, RW,,
```

Then, if a public user opens this file for write, he will obtain access to version 2. If a friend does the same, he will obtain access to version 1.

An important exception to the "select lowest version number" rule is files opened for sequential write (W). When a one-version file is opened for sequential write, the old contents of the file are lost. If more versions of the file exist, the version with sufficient access with highest version number is always selected. When the file is closed, this version is inserted as version one. Thus, for sequential files, the last updated version is always version one.

Example:

A QED user has a file with two versions called QEDDATA:SYMB. If he edits the file:

```
*R  QEDDATA          (read version 1)
    .
    .                (edit commands)
    .
*W  QEDDATA          (write on version 2 and set as version 1)
```

he may recover the unedited version of the file as version 2:

```
*R  QEDDATA;2        (read version 2)
    .
    .                (correct edit commands)
    .
*W  QEDDATA          (write on version 2 and set as version 1)
```

with three versions of the file he could have maintained two backup copies, etc.

An owner of a file may protect one or more versions of the file for use from other users (friends with write access). Consider the following example:

```
@CREATE-FILE         SYMBFILE;3, 40
@SET-FILE-ACCESS     SYMBFILE;3,,R,,
```

Version three of the file can only be read by a friend, but version one and version two may be read or written. Suppose a friend is changing the file with QED. Note that the friend does not have to know that more than one version exists.

```
*R  SYMBFILE         (read version 1)
    .
    .                (edit)
    .
*W  SYMBFILE         (write on version 2, and set as
                      version 1)
```

Version three is still intact and may be accessed by the file owner. If the owner uses the file, the following will happen:

```
*R  SYMBFILE         (read version 1, updated by the
                      friend)
    .
    .                (edit)
    .
*W  SYMBFILE         (write on version 3, and set as
                      version 1)
```

Note that the protection of this version is also moved, so that version 1 is protected for write access. If the friends use the file again, the following will occur:

*R SYMBFILE	(read version 1)
.	
.	(edit)
.	
*W SYMBFILE	(write on version 3, and set as version 1)

Version 2 is now the version without write access for the friend.

Note: If a file exists in more than one version, and no version is specified in the SET-FILE-ACCESS command, the legal access of all versions are changed with this command.

2.7 *DIRECTORIES*

A complete file system may consist of one or more directories. Each mass storage device maintained by the file system has its own directory and is completely independent of all other devices. Devices (disk packs, magnetic tapes, etc.) may be moved to other installations and used there.

Each device directory is given a name of up to 16 characters. Before a device can be used by the file system, the directory must be created, then entered. When the device is not needed anymore, the directory may be released. Then, the next time the device is needed, it must be reentered, etc.

A directory is created with a command called CREATE-DIRECTORY with parameters, directory name and device name. The directory name is saved on the device, and the device is initialized if it is a disk or a drum.

Example:

```
@CREATE-DIRECTORY     PACK-TWO    DISK-10 F
```

Create a directory called PACK-TWO on the fixed cartridge on disk unit 0.

```
@CREATE-DIRECTORY     TAPE-ONE    MAG-TAPE-1 1
```

Create a directory called TAPE-ONE on the tape mounted on magnetic tape unit 1.

A directory is entered with the command ENTER-DIRECTORY with parameters, directory name, and device name.

Example:

```
@ENTER-DIRECTORY      PACK-TWO    DISK-1 0 R
@ENTER-DIRECTORY      TAPE-ONE    MAG-TAPE-1 1
```

Enter the two directories PACK-TWO and TAPE-ONE.

When a directory is entered, users and files may be created, and files may be accessed. The various directories entered may be distinguished by specifying the directory name as a prefix to the user name, separated from the user name with colon.

Examples:

```
@CREATE-USER          PACK-TWO:USER-ONE
```

Create a user called USER-ONE in directory PACK-TWO.

@CREATE-FILE (PACK-TWO:USER-ONE) FILE-ONE, 10

Create a file for USER-ONE in directory PACK-TWO

@OPEN-FILE "(TAPE-ONE:) TAPEFILE", W

Create and open a file for the currently entered user in directory TAPE-ONE.

When all files in a directory are closed, the directory may be released with the RELEASE-DIRECTORY command.

Example:

@RELEASE-DIRECTORY PACK-TWO

Release directory called PACK-TWO.

When a directory is released, another directory may be entered on the corresponding device unit.

The first directory entered, regardless of device type, is defined as the main directory in the file system.

All file system users have to be users in main directory. All information about friends and friend access is kept in the main directory.

Example:

@CREATE-USER PACK-TWO:USER-ONE
NO SUCH USER IN MAIN DIRECTORY

This means that no user with name USER-ONE exists in main directory and therefore, it cannot be created in directory PACK-TWO.

A directory normally used as main directory may, without any changes, be entered and used as a normal directory.

If no directory name is specified in the commands, the main directory is usually assumed.

Example:

@CREATE-USER USER-XXXX

This command will always create a user in main directory.

@OPEN-FILE "(USER-XXXX) FILE-ONE", W

This file will be created in main directory, if USER-XXXX has space in main directory.

Any number of directories entered may be defined as default directories. Default directories are used when no directory name is specified in the commands for creating and accessing files. Main directory is always default directory.

Which default directory to select depends upon the user logged on. The file system will always select the default directory where the user is given space. One user should not be given space in more than one default directory.

A directory is set a default directory with the SET-DEFAULT-DIRECTORY.

Example:

```
@SET-DEFAULT-DIRECTORY  PACK-TWO
@SET-DEFAULT-DIRECTORY  PACK-THREE
@GIVE-USER-SPACE  PACK-TWO:USER-ONE, 30
@GIVE-USER-SPACE  PACK-THREE:USER-TWO, 40
@CREATE-FILE  (USER-ONE) FILE-ONE, 10
@CREATE-FILE  (USER-TWO) FILE-TWO, 20
```

The file FILE-ONE will now exist in directory PACK- TWO, and the file FILE-TWO in directory PACK-THREE.

2.8 *PERIPHERALS*

Peripherals, such as, card readers, line printers, etc., may be treated by a user very much like sequential disk files. Before a peripheral can be used, it must be created by the command SET-PERIPHERAL-FILE with parameters, file name and logical device number. The logical device number used depends on the actual SINTRAN III system. The file name is the name of an empty file.

Example:

```
@CREATE-FILE          LINE PRINTER, 0
@SET-PERIPHERAL-FILE          LINE-PRINTER, 5
```

Now the line printer may be opened by any user with:

```
@OPEN-FILE            LINE-PRINTER, W
```

Peripherals can only be used by one user at a time. When a peripheral is opened, it is reserved for the logged in user and not released before the file is closed. If another user tries to open the same peripheral, he will get the information that the peripheral is occupied.

If two or more peripherals are of the same type, they may be treated as two or more versions of the same file, and the file system will always select the on unoccupied.

Example:

An installation has two line printers with device numbers 5 and 25. They may be treated as two versions of the same file:

```
@CREATE-FILE          LINE-PRINTER; 2, 0
@SET-PERIPHERAL-FILE          LINE-PRINTER; 1, 5
@SET-PERIPHERAL-FILE          LINE-PRINTER; 2, 25
```

If a user opens the file LINE-PRINTER, he will get the first of the line printers not reserved:

```
@OPEN-FILE            LINE-PRINTER, W
```

If he wants the one with device number 25, he may write:

```
@OPEN-FILE            LINE-PRINTER; 2, W
```

2.9 *TAPES*

Magnetic tapes are supported by the file system. Files on tapes may be treated very much like files on disk or drum, but with some restrictions.

Files on tapes may only be opened for sequential read or write. When a file is opened for write, a new version of the file is always created at the end of the tape, except when the file is the last one on the tape.

Example:

Suppose TAPE-ONE is a magnetic tape directory. After the following sequence of commands:

```
@OPEN-FILE    (TAPE-ONE:) FILE-ONE, W
    .
    .
@OPEN-FILE    (TAPE-ONE:) FILE-TWO, W
    .
    .
@OPEN-FILE    (TAPE-ONE:) FILE-ONE, R
    .
    .
@OPEN-FILE    (TAPE-ONE:) FILE-TWO, R
    .
    .
@OPEN-FILE    (TAPE-ONE:) FILE-ONE, W
    .
    .
```

The last three files on the tape are:

FILE-ONE:SYMB; 2 , FILE-TWO:SYMB; 1 and
FILE-ONE:SYMB; 1

2.10 *THE SPOOLING SYSTEM*

A peripheral file may be created in more versions than the existing number of corresponding peripherals. All versions of the file not connected to a device number, will be treated as spooling files.

Example:

```
@CREATE-FILE LINE-PRINTER; 10, 0
@SET-PERIPHERAL-FILE LINE-PRINTER, 5
@SET-FILE-ACCESS LINE-PRINTER WA RWA RWAD
```

There are now ten versions of the file LINE-PRINTER. The first version is a peripheral file with device number 5. The remaining files are spooling files.

Spooling files may be utilized for output spooling if the actual SINTRAN III system is generated with an optional spooling program for the peripheral in question.

If the system is generated with a spooling program, output spooling may be initiated with the command START-SPOOLING, with the peripheral file name as parameter.

Example:

The system is generated with a spooling program for device number 5 and the file LINE-PRINTER has ten versions: nine spooling files and one peripheral file with device number 5. Output spooling on the line printer may then be initiated with the command:

```
@START-SPOOLING LINE-PRINTER
```

When output spooling is started with the START-SPOOLING command, the peripheral (line-printer with device number 5 as in the above examples) is reserved by the spooling program and cannot be used directly.

Example:

The file LINE-PRINTER;1 is a peripheral file and spooling is initiated on this device as in the above examples. The command:

```
@OPEN-FILE LINE-PRINTER;1 W
```

will give the error message

```
FILE ALREADY RESERVED
```

When spooling is initiated, all output to the peripheral must go to the spooling files. When a user tries to open the peripheral he will not get the peripheral itself but the first free spooling file of that peripheral. When the file is closed, the file is linked to a spooling queue for the peripheral and eventually emptied on the peripheral.

Example:

```
@START-SPOOLING LINE-PRINTER
@COPY-FILE LINE-PRINTER USER-FILE-ONE
```

The file USER-FILE-ONE is copied onto a spooling file version of the file LINE-PRINTER. The spooling file is linked to the spooling queue when the COPY-FILE command is finished. The file is emptied while the user continues with other commands.

If more than one spooling file exists, then more than one user may open the peripheral at the same time or the same user may open the peripheral several times.

The spooling queue may be examined with the command LIST-SPOOLING-QUEUE with parameter peripheral file name.

Example:

```
@COPY-FILE LINE-PRINTER FILE-ONE
@COPY-FILE LINE-PRINTER FILE-TWO
@COPY-FILE LINE-PRINTER FILE-THREE
@LIST-SPOOLING-QUEUE LINE-PRINTER
(PACK-ONE:SYSTEM) LINE-PRINTER:;3, etc.
                  :
                  :
                  :
```

The first spooling file version (version 2 – with a copy of FILE-ONE) is not in the queue because printing of this file has already started.

A user may also insert his own file in the spooling queue and get a desired number of copies of the file. This is accomplished with the command @APPEND-SPOOLING-FILE with parameters, peripheral file name, name of the file to be appended to the spooling queue, and the number of copies desired.

Example:

```
@APPEND-SPOOLING-FILE LINE-PRINTER FILE-ONE 2
@APPEND-SPOOLING-FILE LINE-PRINTER FILE-TWO 1
```

A file may be removed from the queue with the @DELETE-SPOOLING-FILE command. The current print-out may be aborted or restarted with the commands @ABORT-PRINT or @RESTART-PRINT.

The spooling program may be discontinued with the @STOP-SPOOLING command. The spooling output is always discontinued after the end of the current print-file. The peripheral is released and may be accessed directly. The spooling files may still be used. These files, and any user file, may be inserted in the spooling queue. The spooling program will resume with the first file in the queue when the @START-SPOOLING command is eventually used.

The number of pages on the disk which may be used by the spooling files are limited to 500. The spooling files are usually files in the main directory that belong to user SYSTEM. The above mentioned limit is used to secure that no more than 500 user SYSTEM's pages are used for spooling. This limit may be changed with the following two commands:

@GIVE-SPOOLING-PAGES, and
@TAKE-SPOOLING-PAGES

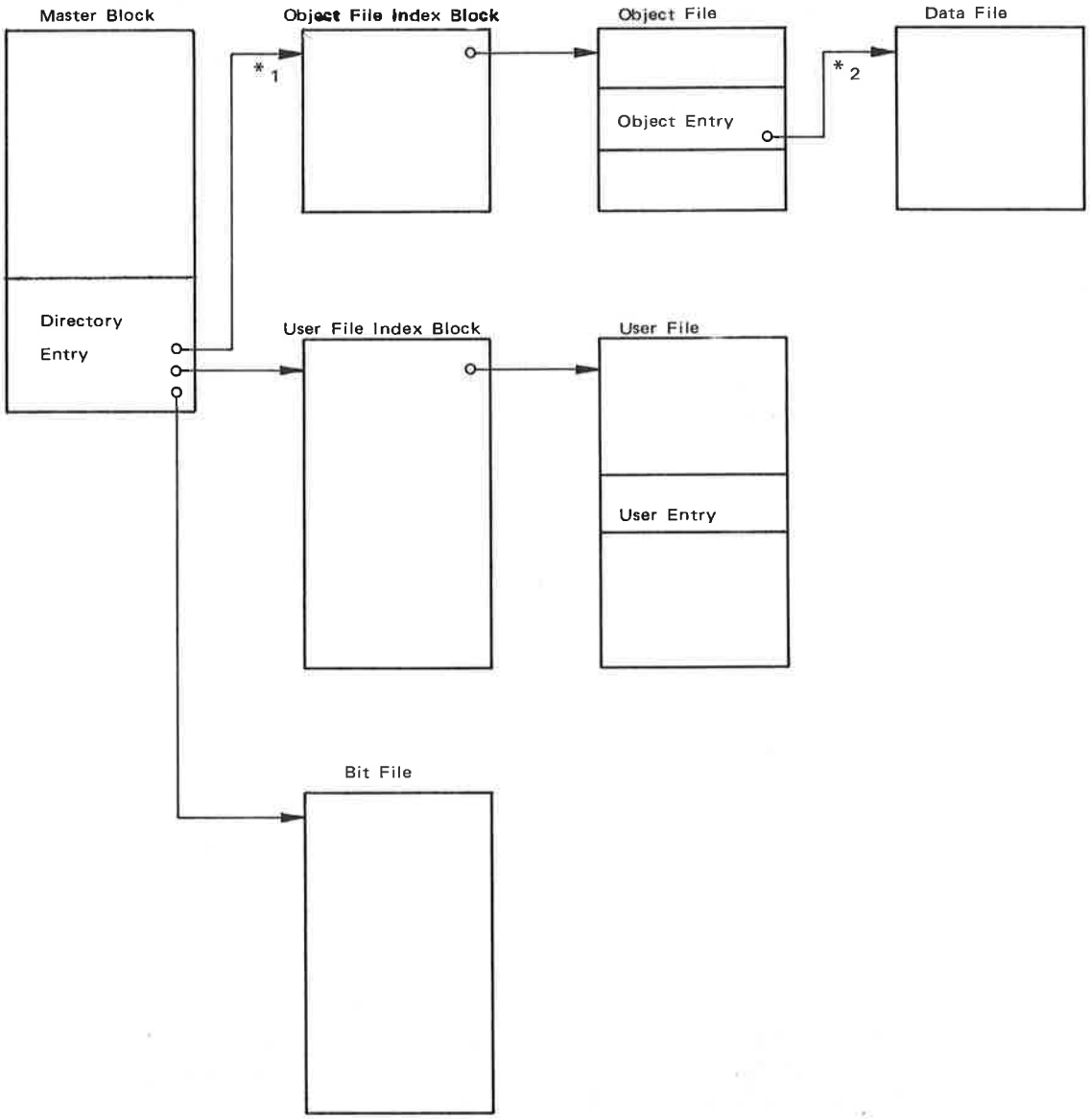
These commands will increase or decrease the limit with the number of pages specified. The command @SPOOLING-PAGES-LEFT will return the number of pages still available.

If the system runs out of spooling pages (either by reaching the maximum limit of 500 or because user SYSTEM has no pages left), all user programs currently doing output to spooling files will enter a waiting state. The spooling program will then start printing one of the spooling files, return the pages to the pool of free spooling pages, and restart the waiting user programs.

3 DEVICE LAYOUT

3.1 *DISK AND DRUM*

A disk or drum unit is organized by the file system in the following way:



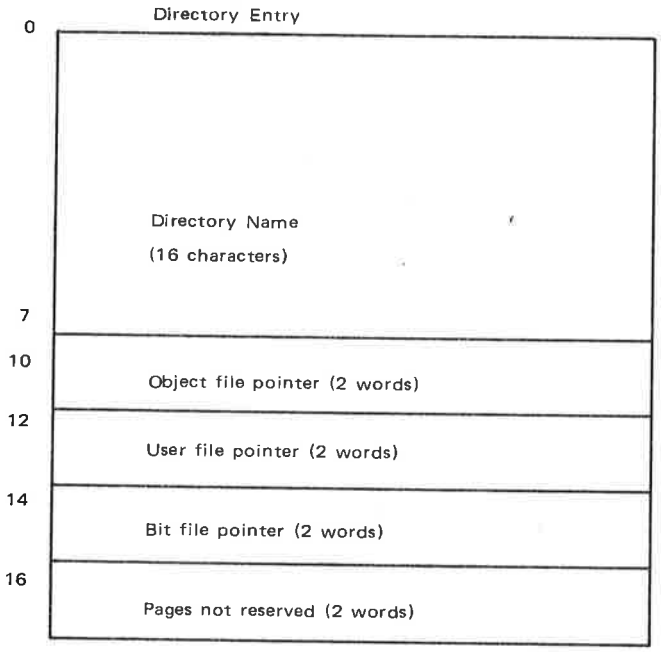
*1: The system will expand itself with a subindex block where necessary.

*2: Files are indexed when necessary.

3.1.1 *Master Block*

The mast block is a 1K block located at lowest address on the unit. Only the last part of the master block is used by the file system. The remaining part of the master block may be used, for instance, for a restart program.

The last part of the master block is called the directory entry. The directory entry contains the name of the device unit (the "directory name"), pointers to three files, the bit file, the user file and the object file, and number of pages not reserved for any users.



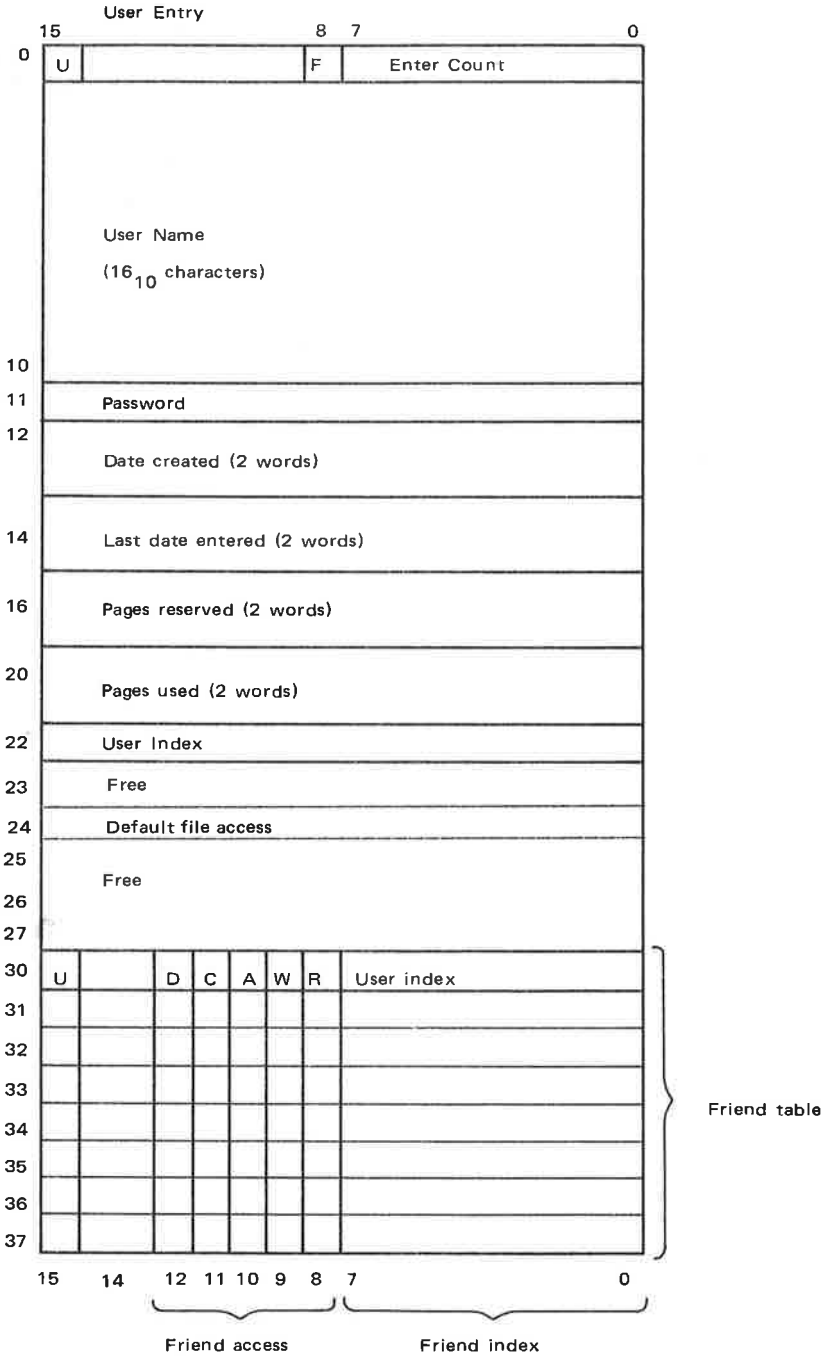
3.1.2 *Bit File*

The bit file contains a free/reserved map of the device unit. Each bit in the bit file corresponds to one page (1K) of the device unit. The page is free if the bit is 0, and occupied if the bit is one.

3.1.3 *User File*

The user file contains information about all users of the device unit. Each user has one entry in the user file. This entry contains the name of the user and a table of friends. Each user may have eight friends, and each friend may have different access restrictuions to the users file.

Each device unit may have 256 users.

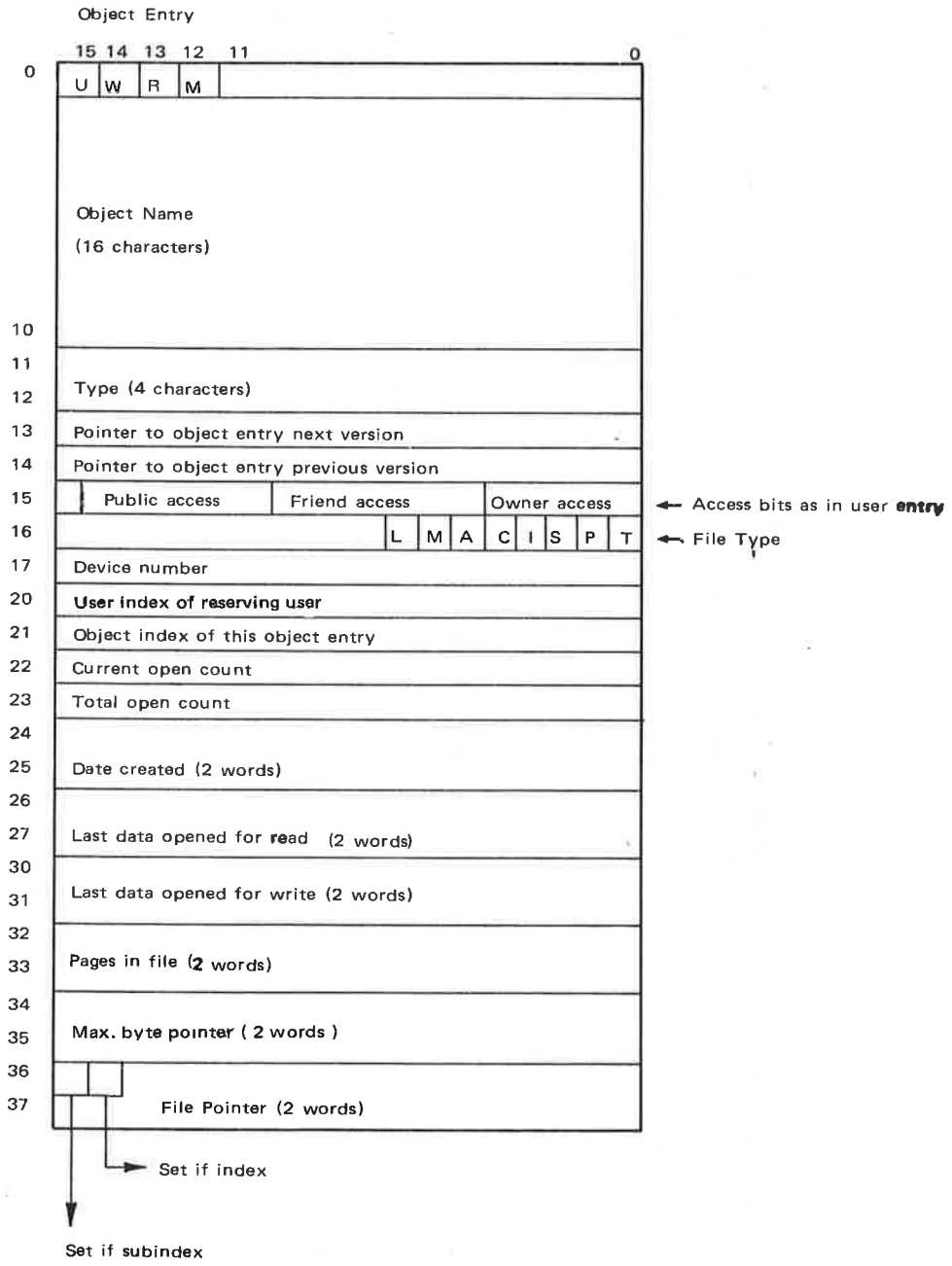


U - entry used
D - directorv access
C - common access
A - append access
W - write access
R - read access
F - user/object entry flag (1 for user entry)

3.1.4 *Object File*

The object file contains information about all user's files on the device unit. Each file has one entry in the object file. This entry contains the name and type of the object, access restrictions, file length, version pointers, and a point to the file.

Each device unit may have 64K files, divided into 256 files for each user. User zero has object entry 0-255, user one object entry 256-511, etc.



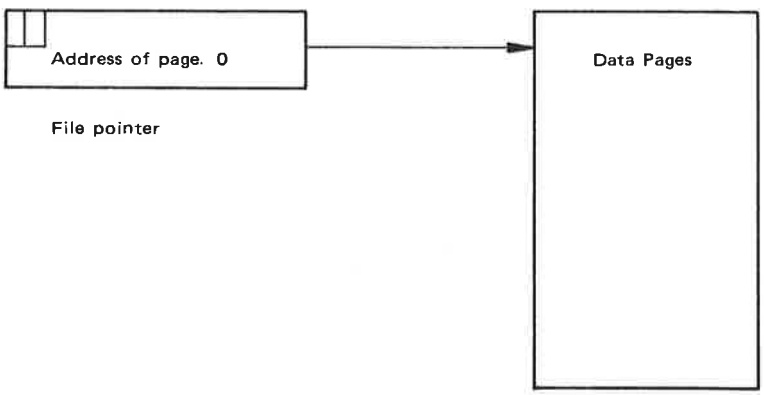
U entry used
M file modified (opened for write)
R file reserved
W currently opened for write
F user/object entry flag (0 for object entry)
L library file
M magnetic tape file
A allocated file
C continuous file
I indexed file
S spooling file
P peripheral file
T terminal file

3.1.5 *Data Files*

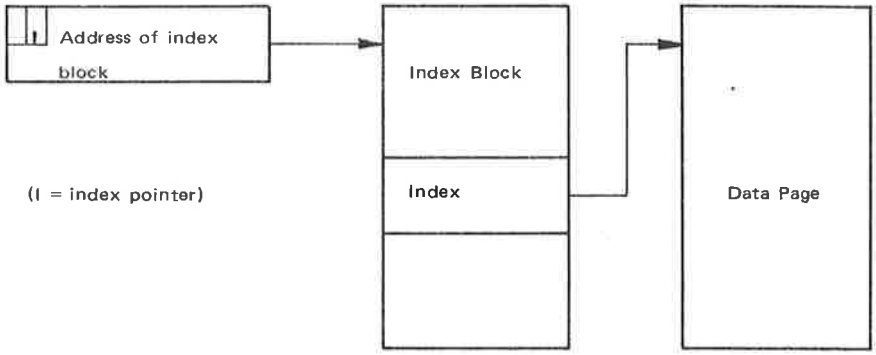
Each object entry corresponds to a data file. The data files contain the user's data. The file user may access this data in sequential or random mode by means of a set of file access routines. Indexed files may be automatically expanded in write operations.

3.1.6 *File Layout*

A file on disk or drum is either organized as a continuous or an index-sequential file. A continuous file may be of any length and is defined by start address and length:



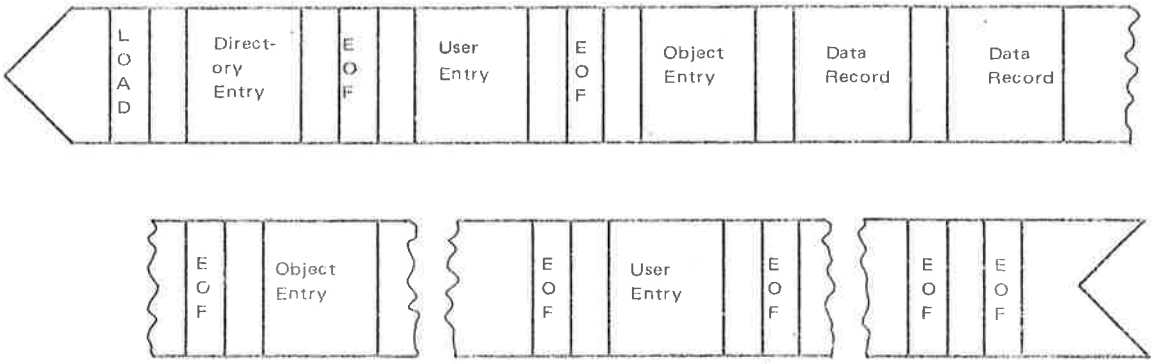
An index-sequential file is defined by a 1K index block, which contains pointers to the 1K data pages of the file:



3.2 *MAGNETIC TAPE*

3.2.1 *Magnetic Tape Directions*

A tape directory with files is organized in the following way:



One tape may have a total number of 64K user and object entries. Each file consists of data records of 1K each and may contain 64K data records. Thus, the maximum file size on tape is 64 M words (128 Mbytes).

The directory entry, user entry and object entry contain information similar to the corresponding entries of disk or drum.

The tape may also be accessed freely, disregarding any file layout on the tape.

3.2.2 *Magnetic Tape Volumes*

Files on magnetic tape volumes are recorded according to the ANSI standard.

The general tape layout is as follows:

VOL1 HDR1 HDR2* – FILE1 – *EOF*HDR1 HDR2–FILE2–*EOF1**

Where VOL1, HDR1, HDR2 and EOF1 are tape labels, and asterisks (*) are end-of-file marks.

All labels are 80 charactersrecords, while file data is recorded as 1024 words records (2048 characters).

All information in the labels are recorded as ASCII characters with the parity bit cleared. All unused character positions will contain spaces. The first position is position 1 and the last position is position 80.

The volume header label, VOL1:

Position	Contents
1-4	VOL1
5-10	Volume name. Any alphanumeric characters.
38-51	User name of the owner of this volume. Any alphnumeric characters.
80	1

If the owner name exceeds 14 characters, the first 14 characters of the owner name will be used.

The first file header label, HDR1:

Position	Contents
1-4	HDR1
5-21	File name
22-27	File type
28-31	0001
32-35	File sequence number in volume. The contents of this field will be 0001 for the first file in a volume, 0002 for the next and so on.
36-39	File generation. Any alphanumeric characters.
55-60	000000

The second file header label, HDR2:

Position	Contents
1-4	HDR2
5	U
6-10	02048
16-31	User name of the owner of this file.

The end of the file label, EOF1:

Position	Contents
1-4	EOF1
55-60	Number of data records (pages) in this file

The second file header label (HDR2) may be missing. In this case, the system will assume that all files in the volume is owner by the owner of the volume.

3.3 *PERIPHERALS*

Peripherals may have symbolic names and may be treated like sequential disk or drum files. They may have object entries on disk or drum. Such an object entry will not contain a data file pointer, but the device number used to access the peripheral.

4 SYNTAX

4.1 *SYMBOLIC NAMES*

All names used by the file system may consist of the characters A-Z and the numbers 0-9. The name may be divided into fields with hyphens.

Examples:

LINE-PRINTER
DISK-FILE-1
DK01

To recognize a name, the file system does not need the complete name. Any or all fields of the name may be abbreviated. A name is recognized if an exact match is found or if the abbreviation is not ambigious. An * in a character position is defined to match any character.

Example:

The name DISK-FILE-1 of the three names above may be abbreviated to DISK, D1, D-F-1, D-F, -FILE, DI-FI-1, –1, etc. The abbreviation D will not be accepted, as it may be confused with DK01.

4.2 *FILE NAMES*

The general format of a filename is:

<file name>:=

(<directory name>:<user name>) <object name>:<type>; version

Default names exist for all parts of the <file name> except the <object name>.

Example:

```
(PACK-1:USER-1) DATAFILE:DATA; 5
(USER-2) SYMBFILE
(PACK-2:) BRFFILE:BRF
BINFILE; 1
```

If [ ] denotes parts of the name that may be omitted, the syntax for a file is:

<file name>:=

[([<directory name>:] [<user name>])] <object name> [:<type>] [;version]

A new version is created by putting the <version> in quotes:

<file name>:=

[([<directory name>:] [<user name>])] <object name> [:<type>] [;"version"]

A new file is created by putting the whole <file name> in quotes:

<file name>:=

"[([<directory name>:] [<user name>])] <object name> [:<type>] [;version]"

4.2.1 *Directory Names*

A directory name may consist of 16 characters and it is used to identify a device unit.

Examples:

DRUM
PACK-ONE
TAPE-10034

Default directory name is the directory defined as default directory for specified use.

4.2.2 *User Name*

A user name may consist of 16 characters and it is used to identify a user.

Examples:

OLE-PEDER
SYSTEM

Default user name is the name of the user currently entered or, in some instances, the user SYSTEM.

4.2.3 *Object Name*

An object name may consist of 16 characters, and it is used to identify an object entry. Object entries for the defined user and user SYSTEM are searched for ambiguous names.

Examples:

SYMBFILE
FORTRAN
LINE-PRINTER

4.2.4 *Type*

The type part of the file name may consist of four characters and is used to specify the purpose the file is to be used. The type is not necessary to identify a file, but may be used to separate, otherwise equal, file names. The type is usually one of the following:

SYMB	Symbolic file
BRF	Binary relocating format file
PROG	Program file
DATA	Data file
CORE	Core image file
BPUN	MAC assembler BPUN format

Default for type depends upon the command used.

4.2.5 *Version*

The version part of the file name consists of a decimal number. This part is used to find the correct version of a file. A file may exist in up to 256 versions within one device unit, and the last version written is usually version 1.

Default version number is usually 1, except when a file is opened for write. Then, the highest version number is used and is later regarded as version 1.

During write operations on tape, a new version is always created at the end of the tape and version specifications are not allowed.

4.3 *DEVICE NAMES*

In some file system commands, device names for disk, drum and tape drivers are used. The following names are usually used:

DRUM-n
DISK-n
MAG-TAPE-n
BIG–DISC

where 'n' is an integer number.

4.4 *ACCESS STRINGS*

A couple of the file commands have access strings as parameters. An access string is a string of characters where each character has one specific meaning. The characters may be specified in any order, and one character may be repeated any number of times.

Example:

RW
WR
RWW
RWRW
RRRRW

All these access strings have the same meaning.

The access strings are of two types, legal access strings and open access strings.

4.4.1 *Legal Access String*

Legal access strings are strings specifying the legal access to a file or the legal access of a friend. A legal access string may consist of the following characters:

R	read permitted
W	write permitted
A	append permitted
C	common access permitted (continuous or allocated files)
D	directory access permitted
N	no access permitted

Append access means that the file may be expanded. Common access means that the file may be opened for write by more than one user at the same time. Directory access means that the file may be deleted, or that the access may be changed.

4.4.2 *Open Access String*

Open access strings are strings specifying the current access to a file when the file is opened. The following characters may be used.

R	read
W	write
A	append to file
X	random access
C	common access (continuous and allocated files)

The following combinations are legal:

- R sequential read
- W sequential write
- RW sequential read or write
- RX read random
- WX read and write random
- WA sequential write append
- RC read random with read and write access from other users allowed (continuous and allocated files)
- WC read and write random with read and write access from other users allowed (continuous and allocated files)

5 COMMANDS

A set of file system routines are available as commands. The file system will contain a command interpreter which allows symbolic command names and parameters. This interpreter will allow abbreviation of the command names as described in Section 4.1.

The available commands are listed below, along with parameters and a description of the effect of the command. The syntax described in Chapter 4 is used.

Some of the commands are restricted, which means that they can only be used by the user called 'SYSTEM'. User SYSTEM should always be the very first user created.

5.1 *DIRECTORY COMMANDS*

5.1.1 *Create Directory*

CREATE-DIRECTORY (Restricted) <directory name>, <device name> [,unit] [,<'F' or 'R'>]

Create a directory entry on the device unit specified by <device name> and unit. If the device is a disk or drum, the device is tested for bad spots, and a bit table is generated. 'F' or 'R' is used on 8 Mbytes disks to separate the fixed and the removable disk pack. If the device unit has been used before, all information will be lost if this command is used.

5.1.2 *Rename Directory*

RENAME-DIRECTORY (Restricted) <old directory name>, <new directory name>, <device name> [,unit] [,<'F' or 'R'>]

Change the name of the directory on the specified device. Renaming is not allowed if the device is entered.

5.1.3 *Enter Directory*

ENTER-DIRECTORY <directory name>, <device name> [,unit] [,<'F' or 'R'>]

The enter command must be used before the device unit can be accessed by the file system. The system will check that the correct directory is found on the specified device.

5.1.4 *Release Directory*

RELEASE-DIRECTORY <directory name>

The release command will release the device unit that corresponds to the specified directory. The directory may be removed and another directory entered on the same unit. All files must be closed before the release command is accepted.

5.1.5 *Set Default Directory*

SET-DEFAULT-DIRECTORY<directory name>
(Restricted)

This command will set the specified directory as a default directory. A default directory is searched for files if no directory name is specified in the file commands. Only the default directory where the user in question is granted space is searched. A user should not be given space in more than one default directory.

If a user has space in more than one default directory, only one will be searched if no directory name is specified.

The first directory entered is called main directory. Main directory is always a default directory.

5.2 *USER COMMANDS*

5.2.1 *Create User*

CREATE-USER [<directory name>:] <user name>
(Restricted)

Create one user with the specified name in the specified directory. If no directory name is specified, the user is created in the main directory. If a user is created in another directory, he has to exist in the main directory.

5.2.2 *Rename User*

RENAME-USER [<directory name>:] <old user name>,
(Restricted) <new user name>

This command is used to change the name of a user in the specified directory, or in the main directory if no directory name is specified. A user cannot be renamed if he has opened files in the specified directory or if he is entered.

5.2.3 *Delete User*

DELETE-USER [<directory name>:] <user name>,
(Restricted) number of pages

This command will delete the specified user from the specified directory. If no directory name is specified, the main directory is assumed. A user cannot be deleted from a directory if he has created files in the directory. A user cannot be deleted from the main directory if he is entered (logged in).

5.2.4 *Give User Space*

GIVE-USER-SPACE [<directory name>:] <user name>,
(Restricted) number of pages

When a user is created with the CREATE-USER command, he has no space available. The user is given space with the GIVE–USER–SPACE command. This command does not apply for magnetic tapes. The user is only given space in the specified directory and must exist in that directory. One user should be given space in only one default directory. The space given is used both for data pages and directory pages for this user.

5.2.5 *Take User Space*

TAKE-USER-SPACE (Restricted)	[<directory name>:] <user name>, number of pages

The disk space given to a user with the GIVE-USER-SPACE command may be taken back with this command. It is only possible to take back unused pages from a user.

5.2.6 *Enter User*

'ESC'	<user name>, <password>

After 'ESC' the system writes ENTER and waits for a user name. After 'CR', the system writes PASSWORD and waits for the user's password.

5.2.7 *Change Password*

CHANGE-PASSWORD	<old password>, <new password>

Change the password of the currently entered user. Old password has to be correctly specified to change the password.

5.2.8 *Clear Password*

CLEAR-PASSWORD (Restricted)	<user name>

Clear the password of the specified user in the main directory.

5.2.9 *Log Out*

LOGOUT

Release the currently entered user on this terminal. All files opened from the terminal are closed and the terminal may be used by another user.

After a LOGOUT command, no commands, other than the Enter-User command may be issued.

5.3 *FRIEND COMMANDS*

5.3.1 *Create Friend*

CREATE-FRIEND <friend name>

Create one friend of the currently entered user. The friend has to be a user in the main directory. One user can have a maximum of eight friends.

5.3.2 *Delete Friend*

DELETE-FRIEND <friend name>

Delete a previously created friend.

5.3.3 *Set Friend Access*

SET-FRIEND-ACCESS <friend name>, <legal access string>

This command is used to define the legal file access modes of a friend. The defined access regards only the creating (entered) users files. The legal access modes are defined by the access string (see Section 4.4.1).

5.4 *FILE CREATION AND DELETION*

5.4.1 *Create File*

CREATE-FILE <file name>, number of pages

Create one or more versions of a file, each version with the specified number of pages.

The directory must be entered, and if a user name is specified, the currently entered user must be a friend of this user with directory access.

The file space is allocated as a continuous area on the device. If number of pages is zero or not specified, the file is created as an empty indexed file.

If there is not enough file space for all versions specified, the system will create a subset that fits into the remaining space and give an error message. The number of versions created may be found by using the LIST-FILE command.

5.4.2 *Create New Version*

CREATE-NEW-VERSION <file name>, number of pages

Create one or more new versions of a file, each version with the specified number of pages, just like the previous command. If the version number specified is equal to a version already existing, one new version is created and inserted as this version. If the specified version number is higher than the highest version existing, new versions are created until the total number of versions is equal to the specified version.

5.4.3 *Allocate File*

ALLOCATE-FILE <file name>, page address, number of pages

This command acts the same as the CREATE-FILE command, except that the start address of the file is specified. The file space is allocated as a continuous area from this address. The page size is always 1K words (2K bytes) and the page address is given as page number from the start of the device, which is always page 0.

If more than one version is created, they are allocated one after the other, with version 1 from the specified page address.

5.4.4 *Allocate New Version*

ALLOCATE-NEW-VERSION <file name>, number of pages, page address

This command acts like the CREATE-NEW-VERSION command, except that the start address of the new version or versions may be specified.

5.4.5 *Expand File*

EXPAND-FILE <file name>, number of pages

Expand the file with the specified number of pages. The file must be a continuous file, an allocated file or an empty file created with one of the CREATE- or ALLOCATE-commands.

5.4.6 *Rename File*

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

Change the name or type of a file. The name is changed if a new object name is specified, and the type is changed if a new type is specified. If no version is specified in the old file name string, all versions are renamed. If a version is specified, only that version is renamed. Note that directory name, user name and version number should not be specified in the new file name.

5.4.7 *Set Terminal File*

SET-TERMINAL-FILE <file name>
(Restricted)

The specified file is defined as the name of the terminal. New files or new versions may be specified by the quote notation.

5.4.8 *Set Peripheral File*

SET-PERIPHERAL-FILE <file name>, device number
(Restricted)

The specified file is defined as the name of a peripheral. The device number of the peripheral must be specified. New files or new versions may be specified by the quote notation. If a peripheral file is created with more than one version, the additional versions will be created as spooling files.

5.4.9 *Delete File*

DELETE-FILE <file name>

Delete the specified file from the corresponding directory. If a version is specified, only that version of the file is deleted. If no version is specified, all versions are deleted. The file must be closed before deletion. The file type must be specified.

5.4.10 *Delete-Users-Files <manual check> <file name>*

Delete all files matching the string <file name>. The parameter <manual check> may be specified as YES or NO. If YES is specified each file name is listed on the terminal before it is deleted and the user may type YES or NO after the file name if he wants it deleted or not.

5.5 *FILE PROTECTION AND ACCESS*

5.5.1 *Set File Access*

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

Define the various access modes for the file. Separate access modes must be specified for owner, friend and public users. If no version is specified in the file name, all versions will be given the specified access. If a version is specified, only this version will get new access modes. If an access string is empty, the access for the corresponding user group is not changed. Legal access strings are defined in Section 4.4.1.

5.5.2 *Open File*

OPEN-FILE <file name>, <open access>

The specified file is opened for access. The current access wanted is given in the open access string (Section 4.4.1), and the file is opened if the entered user has sufficient legal access to the file.

If a version is specified, the file system will try to open that version. If no version is specified, the following rules apply:

1. If the file is a peripheral file, with spooling files, the lowest unused version of the file is opened. If both the peripheral and the spooling files are used, an error message will be given.

2. If the file is opened for write, the highest version with sufficient access is selected. When the file is closed, the version is restored as version one.

3. Otherwise, the lowest version with sufficient access is selected.

A file number used by the access routines is returned by the file system.

5.5.3 *Connect File*

CONNECT-FILE <file name>, file number, <open access>

This command acts exactly like the previous command, except that the file number to be used is supplied in the command. If the file number is out of range or already used, the file is not opened.

5.5.4 *Open RT File*

RTOPEN-FILE <file name>, <open access>

This and the next command are specifically for real-time programming. The command acts like the OPEN-FILE command, except that the file is opened for use by real-time programs and not for the entered user. Entered user must be user 'RT' or user 'SYSTEM'.

5.5.5 *Connect RT File*

RTCONNECT-FILE <file name>, file number, <open access>

This command may be used instead of the CONNECT-FILE command for files that shall be accessed by real-time programs. Entered user must be user 'RT' or user 'SYSTEM'.

5.5.6 *Close File*

CLOSE-FILE file number

Close the file with the specified file number. If file number is -2, all files for the entered user are closed. If file number is -1, all files that are not permanently opened, are closed.

5.5.7 *Close RT File*

RTCLOSE-FILE file number

This command will close files opened for real-time programs.

5.5.8 *Set-Permanent-Opened*

SET-PERMANENT-OPENED file number

Set the specified file as a permanent opened file. That means that the file is not closed by a CLOSE-FILE -1 command.

5.5.9 *Open-Scratch-File*

SCRATCH-OPEN <file name>, <open access>

Open the specified file as a scratch file. A scratch file is a permanently opened file, partially deleted when the file is closed. Only 32K words of a scratch file remain in the directory when the file is closed.

5.5.10 *Set-Block-Size*

SET-BLOCK-SIZE file number, block size

Set the block size of the specified opened file. The block size may be any number greater than or equal to 1. Default block size for all opened files is 256 words.

5.5.11 *Set-Byte-Pointer*

SET-BYTE-POINTER file number, byte number

Set the byte pointer to the specified byte number.

5.5.12 *Set-Block-Pointer*

SET-BLOCK-POINTER file number, block number

Set the byte pointer of the file to the first byte in the specified block.

5.5.13 *Reserve File*

RESERVE-FILE <peripheral file name>

Reserve the specified peripheral file for exclusive use from the current terminal.

5.5.14 *Release File*

RELEASE-FILE <peripheral file name>

Release a previously reserved peripheral file.

5.5.15 *Where-Is-File*

WHERE-IS-FILE <file name>

Find out if a file is reserved or opened by another user.

5.5.16 *Reserve Device Unit*

RESERVE-DEVICE-UNIT <device name>, [unit]

Reserve a device for special use. A directory cannot be entered on a reserved device.

5.5.17 *Release Device Unit*

RELEASE-DEVICE-UNIT <device name>, [unit]

Release a device reserved for special use by the previous command.

5.6 *BACKUP AND TEST COMMANDS*

5.6.1 *Copy File*

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

Copy the source file to the destination file.

5.6.2 *Copy Directory*

COPY-DIRECTORY <destination directory name>, <source directory name>

Copy all files in the source directory to the destination directory. The user and file names will be the same in the destination directory as in the source directory.

5.6.3 *Save Directory*

SAVE-DIRECTORY <destination device name>, [<destination unit>], <source device name>, [<source unit>]

Save the entire contents of the source device on the destination device. Bot the used and unused pages are saved.

5.6.4 *Copy Device*

COPY-DEVICE <destination device name>, [<destination unit>], <source device name>, [<source unit>]

Same as Save Directory.

5.6.5 *Regenerate Directory*

REGENERATE-DIRECTORY <directory name>
(Restricted)

Regenerate the specified directory. Conflicting pages are deleted and bit file regenerated.

5.6.6 *Back-up on Labeled Magnetic Tapes*

Files may be copied to labeled magnetic tape reels following the ANSII standard of magnetic tape labels.

A mag. tape volume is, in this context, a collection of files recorded on a reel of magnetic tape according to this ANSII standard.

A new mag. tape volume is created by the following command:

@CREATE-VOLUME <volume name> <device name> [<unit>]

where

<volume name>

. is written onto the tape reel label.

<device name>

must be MAG-TAPE-1 or MAG-TAPE-2.

<unit>

is the unit number of the magnetic tape station (only necessary when more units are connected to the same interface).

To list the contents of a volume, i.e., the reel label and the file labels, the following command is available:

@LIST-VOLUME <device name> [<unit>] <output file>

To copy some or all files of a user from one directory to another directory, from a directory to a mag. tape volume or from a mag. tape volume to a directory, the following command is available:

@COPY-USERS-FILES <destination type>

{<destination name>
<d. device name> [<d. unit>] <d. volume name> <d. file generation>}

<source type>

{<source name>
<s. device name> [<s. unit>] <s. volume name> <file name>
s. file generation>}

where

<destination type>

is DIRECTORY or VOLUME. If DIRECTORY, <destination name> must be specified instead of the alternative four parameters. If VOLUME, the alternative four parameters must be specified instead of <destination name>.

<destination name>

is the name of the destination directory. Only if <destination type> is DIRECTORY.

<d. device name>

is MAG-TAPE-1 or MAG-TAPE-2. Only if <destination type> is VOLUME.

<d. unit>

is the unit number of the destination magnetic tape station when more units are connected to the same interface. Only if <destination type> is VOLUME.

<d. volume name>

is the name of the destination tape reel label. Only if <destination type> is VOLUME.

<d. file generation>

is a four character identification of this back-up generation. Only if <destination type> is VOLUME.

<source type>

is DIRECTORY or VOLUME. If DIRECTORY, <source name> must be specified instead of the alternative five parameters. If VOLUME, the alternative five parameters must be be specified instead of <source name>.

<source name>

is an identifier matching the name(s) of the file(s) to be taken back-up of. Only if <source type> is DIRECTORY.

<s. device name>

is MAG-TAPE-1 or MAG-TAPE-2. Only if <source type> is VOLUME.

<s. unit>

is the unit number of the source magnetic tape station when more units are connected to the same interface. Only if <source type> is VOLUME.

<s. volume name>

is the name of the source tape reel label. Only if <source type> is VOLUME.

<file name>

is an identifier matching the name(s) of the file(s) to be copied. Only if <source type> is VOLUME.

<s. file generation>

is an identifier matching the back-up generation(s) to be copied. Only if <source type> is VOLUME.

Note: If only one unit (number 0) is connected to an interface, the unit number is neither necessary nor legal in the command.

By copying files from a mag. tape volume to a directory, all files giving full or partial match with both the <file name> and the <s. file generation> specified are copied.

If a volume is owned (created) by user SYSTEM, it may contain files for several users. But only files for one user may be copied in one @COPY-USERS-FILES command. If a volume is owned by any other user, it may only contain files for this user.

When files are copied to a volume they are always appended at the end of the volume.

When files are copied to a directory, the user must be created and have space in the destination directory. If a file does not exist in the destination directory, it will be created.

Examples:

@COPY-USERS-FILES DIR, P—3, DIR, A:SYMB

Copy all symbolic files starting with A to the directory P—3.

@CREATE-VOLUME VOL1 MAG-TAPE-1
@COPY-USERS-FILES VOL, MAG-TAPE-1, VOL1, 0312, DIR, :SYMB

Make a back-up of all symbolic files of the currently logged on user on a mag. tape volume.

Assuming user SYSTEM is logged on:

@CREATE-VOLUME VOLSY MAG-TAPE-1;
@COPY-USERS-FILES VOL, M—T—1, VOLS, 0312, DIR, (OLE) :SYMB
@COPY-USERS-FILES VOL, M—T—1, VOLS, 0312, DIR, (PER) :SYMB

This command sequence will make a back-up of the symbolic files of users OLE and PER on a mag. tape volume.

@COPY-USERS-FILES DIR, P—3, VOL, M—T—1, VOl 1, A:SYMB, 0312

This command will replace all symbolic files starting with A in P—3 for the current user with the corresponding files in the mag. tape volume VOL1, from back-up generation 0312.

5.7 *STATISTICS COMMANDS*

The <output file> in the following commands is the file name of the list device. If no name is given, the terminal is used as output file.

5.7.1 *List Directories Entered*

LIST-DIRECTORIES-ENTERED <directory name>, <output file>

List the names of the entered directories with names that match the specified directory name.

5.7.2 *Directory Statistics*

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

List names and some statistics of the entered directories with names that match the specified directory name.

5.7.3 *List Users*

LIST-USERS [<directory name:>] <user name>, <output file>

List the names of all users with names that match the specified user name.

5.7.4 *User Statistics*

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

List the names and some statistics of all users with names that match the specified user name.

5.7.5 *List Friends*

LIST-FRIENDS <user name>, <output file>

List the names and legal access of all friends of the entered user with names that match the specified name.

5.7.6 *List Files*

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

List the names of all files with names that match the specified file name.

5.7.7 *File Statistics*

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

List the names and some statistics of all files with names that match the specified file name.

5.7.8 *List Opened Files*

LIST-OPENED-FILES <output file>

List the file number and names of all currently opened files.

5.7.9 *List RT-Opened Files*

LIST-RTOPENED-FILES <output file>

List the file number and names of all files currently opened for use by real-time programs.

5.8 *MAINTENANCE COMMANDS*

5.8.1 *Dump Directory Entry*

DUMP-DIRECTORY-ENTRY <device name> [,unit] [,<'F' or 'R'>]
(Restricted) <output file>

Gives an octal dump of the directory entry on specified device unit.

5.8.2 *Change Directory Entry*

CHANGE-DIRECTORY-ENTRY <device name> [,unit] [,<'F' or 'R'>]
(Restricted)

Change the contents of the directory entry on the specified device unit. The directory is addressed from 0 to 17 and octal input is accepted. Addresses are specified as octal numbers, terminated by /. After the /, the old contents are written out. New contents may be specified as an octal number, terminated by carriage return. If only carriage return is written, the contents of next location is written out. A @ or . will terminate the change command.

5.8.3 *Dump User Entry*

DUMP-USER-ENTRY <directory name>, user number, <output file>
(Restricted)

Gives an octal dump of the user entry of the specified user number in the specified directory.

5.8.4 *Change User Entry*

CHANGE-USER-ENTRY <directory name>, user number
(Restricted)

Change the user entry of the specified user number in the specified directory. The entry is addressed from 0 to 37 (octal). Otherwise, the command acts like the CHANGE-DIRECTORY-ENTRY command.

5.8.5 *Dump Object Entry*

DUMP-OBJECT-ENTRY <user name>, object number, <output file>
(Restricted)

Gives an octal dump of the object entry specified by the user name and object number.

5.8.6 *Change Object Entry*

CHANGE-OBJECT-ENTRY <user name>, object number
(Restricted)

Change the specified object entry. The entry is addressed from 0 to 37 (octal). Otherwise, the command acts like the CHANGE-DIRECTORY-ENTRY command.

5.8.7 *Dump Bit-File*

DUMP-BIT-FILE <directory name>, block number, <output file>
(Restricted)

Gives an octal dump of one 16 words block of the bit file.

5.8.8 *Change Bit-File*

CHANGE-BIT-FILE <directory name>, block number
(Restricted)

Change the specified bit file block. The block is addressed from 0 to 17 (octal). Otherwise, the command acts like the CHANGE-DIRECTORY-ENTRY command.

5.8.9 *Dump Page*

DUMP-PAGE <directory name>, page address, <output file>
(Restricted)

Gives an octal dump of one 1K page.

5.8.10 *Change Page*

CHANGE-PAGE <directory name>, page address
(Restricted)

Change the specified page. The page is addressed from 0 to 1777 (octal). Otherwise, the command acts like the CHANGE-DIRECTORY-ENTRY command.

5.9 *SPOOLING COMMANDS FOR THE USER SYSTEM*

5.9.1 *Start Spooling*

START-SPOOLING <peripheral file name>
(Restricted)

Starts the spooling program for the specified peripheral. The peripheral will be reserved for the spooling program and the spooling program will print every file linked to the spooling queue for that device until the @STOP-SPOOLING command is used.

If more than one version of the file is a peripheral, the spooling programs for all peripheral versions of the file are started. One specific peripheral may be selected by including a version number in the file name.

An error message will appear if the specified file name is not the name of a peripheral or if no spooling program exists for a specified peripheral.

5.9.2 *Stop Spooling*

STOP-SPOOLING <peripheral file name>
(Restricted)

Stops the spooling program for the specified peripheral and releases the peripheral from the spooling program. Any file currently being printed by the spooling program will be completed before the spooling program is stopped. The spooling queue is unaffected by the command and files may still be appended to the queue. The spooling program will resume printing the files in the spooling queue when the @START-SPOOLING command is used.

5.10 *SPOOLING COMMANDS FOR PUBLIC USERS*

5.10.1 *Append Spooling File*

APPEND-SPOOLING-FILE <peripheral file name>, <file name>, number of copies

The file specified in the second parameter is appended to the spooling queue for the specified peripheral. The specified number of copies of the file is printed in due time on the peripheral.

5.10.2 *Delete Spooling File*

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

The file specified in the second parameter is removed from the spooling queue for the specified peripheral. Only user SYSTEM and the user which appended the file to the queue can delete the file from the queue.

5.10.3 *List Spooling Queue*

LIST-SPOOLING-QUEUE <peripheral file name>

List all the entries in the spooling queue for the specified peripheral.

5.10.4 *Abort Print*

ABORT-PRINT <peripheral file name>

Aborts the current print-out on the specified peripheral and lets the spooling program continue with the next file in the queue. The command has no effect if the spooling program for a specified peripheral is not started, or if no file is being printed. Only user SYSTEM and the user which appended the file to the queue can abort the printing of the file.

5.10.5 *Restart Print*

RESTART-PRINT <peripheral file name>

Restart the printing of the file currently being processed by the spooling program. The command has no effect if the spooling program for the specified peripheral is not started, or if no file is being printed. Only user SYSTEM and the user which appended the file to the queue can restart the printing of the file.

5.10.6 *Give Spooling Pages*

GIVE-SPOOLING-PAGES <number of pages>

Only a certain number of pages of the disk can be used by the spooling files. This number may be increased with this command. Note that the command does not guarantee that the disk space is available. This command is only intended to limit the number of disk pages used by the spooling system. Five hundred pages are initially given to the spooling system.

5.10.7 *Take Spooling Pages*

TAKE-SPOOLING-PAGES <number of pages>

This command may be used to decrease the number of pages the spooling files may use. The number of pages to be taken must be unused.

5.10.8 *Spooling Pages Left*

SPOOLING-PAGES-LEFT

Lists the number of pages left that can be used by the spooling files. Note that the number given is an upper limit and that the disk space may be used by other files.

6 MONITOR CALLS

A set of monitor calls are available for access of files from user programs. The monitor call instruction is the MON instruction. Parameters to and from the monitor call routines are transferred in the T, A, D and X registers.

6.1 *INPUT ONE BYTE*

INBT (MON 1)

T register	: File number
Return - A register	: Error code
Skipreturn - A register	: Byte

Read one byte from a sequential file.

6.2 *OUTPUT ONE BYTE*

OUTBT (MON 2)

T register	: File number
A register	: Byte
Return - A register	: Error code
Skipreturn - OK	

Write one byte on a sequential file.

6.3 *READ DISK*

RDISK (MON 5)

T register	: Block number
X register	: Core address
Return - A register	: Error code
Skipreturn - OK	

Read a block of 256 words from the first opened scratch file.

6.4 *WRITE DISK*

WDISK (MON 6)

T register	: Block number
X register	: Core address
Return - A register	: Error code
Skipreturn - OK	

Write a block of 256 words on the first opened scratch file.

6.5 *READ PAGE*

RPAGE (MON 7)

T register	: File number
A register	: Block number
X register	: Core address
Return - A register	: Error code
Skipreturn - OK	

Read one block from the specified file, which must be opened with random read access. Default block size is 256 words. The block size may be changed with the set-block-size monitor call or command.

6.6 *WRITE PAGE*

WPAGE (MON 10)

T register	: File number
A register	: Block number
X register	: Core address
Return - A register	: Error code
Skipreturn - OK	

Write one block on the specified file, which must be opened with random write access. Default block size is 256 words. The block size may be changed with the set block size monitor call or command.

6.7 *READ OBJECT ENTRY*

ROBJE (MON 41)

T register	: File Number
A register	: Address Memory
Error return- A register	: Error number
Skipreturn	: OK

This monitor call reads the object entry of an opened file into the buffer area given in the A register. 40_8 words are read.

6.8 *CLOSE FILE*

CLOSE (MON 43)

T register : File number

Return - A register : Error code
Skipreturn - OK

This monitor call acts exactly like the CLOSE-FILE command.

6.9 *READ USER ENTRY*

RUSER (MON 44)

A register	: Memory address
X register	: User name pointer
Error return - A register	: Error number
Skipreturn	:OK

This monitor call reads the user entry of a user into the buffer area given in the A register. 40_8 words are read. The user name is a character string terminated by a single apostrophe.

6.10 *OPEN FILE*

OPEN (MON 50)

X register	: Pointer to file name string
A register	: Pointer to default type string
T register	: Access code –

0 sequential write
1 sequential read
2 random read or write
3 random read
4 sequential read or write
5 sequential write append
6 random read or write common
7 random read common

Return - A register	: Error code
Skipreturn - A register	: File number

This monitor call acts exactly like the OPEN-FILE command.

6.11 *READ MAX POINTER*

RMAX (MON 62)

T register	: File number
Return - A register	: Error code
Skipreturn - A & D	: Maximum file pointer

Read the number of bytes in a file. Note that the value returned will be greater by one than the value set by the Set Max Pointer monitor call (MON 73) because the byte pointer starts from 0.

6.12 *WRITE ERROR MESSAGE*

ERMSG (MON 64)

A register : Error code

Write a symbolic error message on the terminal and return to user program.

6.13 *WRITE ERROR MESSAGE AND LEAVE*

QERMS (MON 65)

A register : Error Code

Write a symbolic error message on the terminal and return to command processor.

6.14 *SET MAX POINTER*

SMAX (MON 73)

T register	: File number
A & D register	: Byte printer
Return - A register	: Error code
Skipreturn - OK	

Set the end pointer of a file.

6.15 *SET BYTE POINTER*

SETBT (MON 74)

T register	: File number
A & D registers	: Byte pointer
Return - A register	: Error code
Skipreturn - OK	

Set the byte pointer of a file.

6.16 *READ BYTE POINTER*

REABT (MON 75)

T register : File number

Return - A register : Error code
Skipreturn - A & D registers : Byte pointer

Read the byte pointer of a file.

6.17 *SET BLOCK SIZE OF A FILE*

SETBS (MON 76)

T register	: File number
A register	: Block size (words)
Return - A register	: Error code
Skipreturn - OK	

Set the block size used by the RPAGE, WPAGE, RFILE and WFILE monitor calls.

6.18 *SET BLOCK POINTER OF A FILE*

SETBL (MON 77)

T register	: File number
A register	: Block number
Return - A register	: Error code
Skipreturn - OK	

Set the byte printer of a file to point to the first byte in specified block.

6.19 *READ FILE*

RFILE (MON 117)

A register	: Pointer to parameter list
T register	: File number
Return - A register	: Error code A = 0 if no error

Parameter list:

(File number	
(Wait flag	0 - wait for transfer completed 1 - initiate transfer (only from RT)
Core address	
(Block number	
(Number of words	

Transfer the specified number of words from the specified block of the file to the given core address. Default block size is 256, but may be changed with the set-block-size monitor call or command. The file must be opened for random read.

6.20 *WRITE FILE*

WFILE (MON 120)

A register	: Pointer to parameter list
T register	: File number
Return - A register	: Error code A = 0 if no error

Parameter list:

(File number	
(Wait flag	0 - wait for transfer completed 1 - initiate transfer (only from RT)
Core address	
(Block number	
(Number of words	

Transfer the specified number of words from the specified core address to the specified block on the file. The file must be opened for random write.

6.21 *WAIT FILE*

WAITF (MON 121)

A register	: Pointer to parameter list
T register	: File number
Return - A register	: Error code 0 - if no error and transfer completed 1 - if no error and transfer not completed

Parameter list:

(File number	
(Continue flag	0 - wait until transfer completed 1 - return with completed/uncompleted code in the A register

Wait or test for a file transfer completed. The file transfer must be initiated by a RFILE or WFILE monitor call with wait flag = 1. This monitor call is illegal from background programs.

APPENDIX A

APPENDIX A

ERROR CODES RETURNED FROM MONITOR CALLS

Error Code:	Meaning:
000	Not used
001	Not used
002	Bad File Number
003	End of File
004	Card Reader Error (Card Read)
005	Device Not Reserved
006	Not Used
007	Card Reader Error (Card not read)
010	Not used
011	Not used
012	End of device (time-out)
013	Not used
014	Not used
015	Not used
016	Not used
017	Not used
020	Not used
021	Illegal character in parameter
022	No such page
023	Not decimal number
024	Not octal number
025	You are not authorized to do this

Error Code:	Meaning:
026	Directory not entered
027	Ambiguous directory name
030	No such device name
031	Ambiguous device name
032	Directory entered
033	No such logical unit
034	Unit occupied
035	Master block transfer error
036	Bit file transfer error
037	No more tracks available
040	Directory not on specified unit
041	Files opened on this directory
042	Main directory not last one released
043	No main directory
044	Too long parameter
045	Ambiguous user name
046	No such user name
047	No such user name in main directory
050	Attempt to create too many users
051	User already exists
052	User has files
053	User is entered
054	Not so much space unreserved in directory
055	Reserved space already used

Error Code:	Meaning:
056	No such file name
057	Ambiguous file name
060	Wrong password
061	User already entered
062	No user entered
063	Friend already exists
064	No such friend
065	Attempt to create too many friends
066	Attempt to create yourself as friend
067	Continuous space not available
070	Not directory access
071	Space not available to expand file
072	Space already allocated
073	No space in default directories
074	No such file version
075	No more pages available for this user
076	File already exists
077	Attempt to create too many files
100	Outside device limits
101	No previous version
102	File not continuous
103	File type already defined
104	No such access code
105	File already opened

Error Code:	Meaning:
106	Not write access
107	Attempt to open too many files
110	Not write and append access
111	Not read access
112	Not read, write and common access
113	Not read and write access
114	Not read and common access
115	File reserved by another user
116	File already opened for write by you
117	No such user index
120	Not append access
121	Attempt to open too many mass storage files
122	Attempt to open too many files
123	Not opened for sequential write
124	Not opened for sequential read
125	Not opened for random write
126	Not opened for random read
127	File number out of range
130	File number already used
131	No more buffer space
132	No file opened with this number
133	Not mass storage file
134	File used for write
135	File used for read

Error Code:	Meaning:
136	File only opened for sequential read or write
137	No scratch file opened
140	File not reserved by you
141	Transfer error
142	File already reserved
143	No such block
144	Source and destination equal
145	Illegal on tape device
146	End of tape
147	Device unit reserved for special use
150	Not random access on tape files
151	Not last file on tape
152	Not tape device
153	Illegal address reference in monitor call
154	Source empty
155	File already open by another user
156	File already open for write by another user
157	Missing parameter
160	Two pages must be left unreserved
161	No answer from remote computer
162	Device cannot be reserved
163	Overflow in read
164	DMA error
165	Bad datablock

Error Code:	Meaning:
166	CONTROL/MODUS word error
167	Parity error
170	LRC error
171	Device error (read-last-status to get status)
172	No device buffer available
173	Illegal mass storage unit number
174	Illegal parameter
175	Write-protect violation
176	Error detected by read after write
177	No EOF mark found
200	Cassette not in position
201	Illegal function code
202	Time out (no datablock found)
203	Paper fault
204	Device not ready
205	Device already reserved
206	Not peripheral file
207	No such queue entry
210	Not so much space left
211	No spooling for this device
212	No such queue
213	Queue empty
214	Queue full
215	Not last used by you

Error Code:	Meaning:
216	No such channel name
217	No remote connection
220	Illegal channel
221	Channel already reserved on remote computer
222	No remote file processor
223	Formatting error
224	Incompatable device sizes
225	Remote processor not available
226	Tape format error
227	Block count error
230	Volume not on specified unit
231	Not deleted record

NORSK DATA A.S.
Lørenvn 57 - Postboks 163, Økern
OSLO 1

COMMENT AND EVALUATION SHEET

NORD FILE SYSTEM
April 1977

Publication No. ND-60.052.04

In order for this manual to develop to the point where it best suits your needs, we must have your comments, corrections, suggestions for additions, etc. Please write down your comments on this pre-addressed form and post it. Please be specific wherever possible.

FROM: _____

_____

_____

– we make bits for the future

NORSK DATA A.S LØRENVEIEN 57 OSLO 5 NORWAY PHONE: 21 73 71 TELEX: 18284

– we make bits for the future

NORSK DATA A.S LØRENVEIEN 57 OSLO 5 NORWAY PHONE: 21 73 71 TELEX: 18284