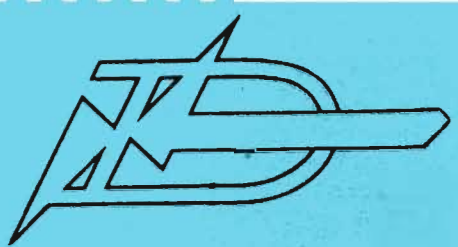
NORD
COMPUTER SYSTEMS

NORD-OPS
Reference Manual



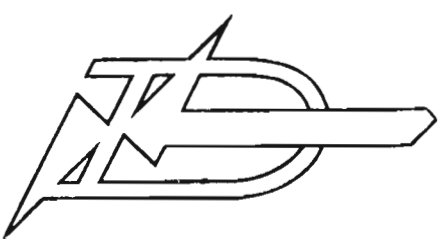
A/S NORSK DATA-ELEKTRONIKK
Økernveien 145, Oslo 5 - Telefon (02) 21 73 71

NORD

COMPUTER SYSTEMS

NORD-OPS

Reference Manual



A/S NORSK DATA-ELEKTRONIKK

Økernveien 145, Oslo 5 - Tlf. 21 73 71

REVISION RECORD

Revision	Notes
2/72	Total revision superceding all previous versions.
8/73	Complete revision. All previous versions obsolete.

Publication No. ND-60.026.02
August 1973

A/S NORSK DATA-ELEKTRONIKK
Økernveien 145, Oslo 5 - Tlf. 21 73 71

TABLE OF CONTENTS

+++
+

Chapters: Page

Chapters:

Chapters:

Chapters:

Appendices:

1 INTRODUCTION

1.1 General Description

NORD-OPS is a mass storage oriented operating system for the NORD-1, NORD-10 and NORD-5 computer systems. The main design objectives for NORD-OPS have been to make a modular, flexible and reliable operating system, which then should make it easy to use and utilize the NORD computers in their different configurations and environments.

To reach these design objectives, it was for several reasons natural to base the operating system on the SINTRAN Monitor System:

- SINTRAN is an advanced real-time multi-programming monitor that is especially designed for an easy extension to a general operating system.

- SINTRAN will presumably make a basis for further development on real-time systems for NORD-1, and a duplication of standard software should be avoided.

Thus, NORD-OPS is organized as a set of tasks running under SINTRAN, providing the user with the possibility of running batch jobs and/or activities in "conversational mode" while they use SINTRAN for real time applications.

Special features of NORD-OPS to be pointed out are:

1.1.1 Control Language

The user presents his work to NORD-OPS as a sequence of control commands. Each command normally specifies a processor or a program to be started. For instance,

- start the FORTRAN compiler

- load and execute the compiled program

The basic function of NORD-OPS is the capability of reading and executing these commands. This function is performed by the Job Supervisor (JBSUP), the basic task of NORD-OPS.

1.1.2 Buffering of Standard Input/Output

To give a better utilization of the hardware resources, NORD-OPS may buffer the standard input and output (normally card reader input and line printer output). This is performed by providing two other tasks running under SINTRAN: the System Input Processor (SIP) and the System Output Processor (SOP). SIP transfers the job decks from the card reader to mass storage. Here the jobs are organized

in a queue according to their priority; thus the jobs are served on a priority basis. The jobs will then read their standard input from mass storage. Similar, the jobs put their standard output to mass storage. After a job is finished, its standard output is transferred to the printer by means of SOP.

Normally, the JBSUP (and thus the jobs) will run with the lowest (SINTRAN) priority, in particular with lower priority than SIP and SOP. Thus the jobs pick up the CPU resources left free by the other activities.

1.1.3 Security

One design objective of NORD-OPS has been to make it impossible for a batch job to destroy the rest of the system, including the operating system. This is possible due to the hardware protection system.

Normally, the batch jobs are running in restricted mode, i.e., with the rest of the system protected from them. The system is then said to be in "USER MODE".

However, special tasks in a job may run in unrestricted mode, for instance the FILE system, which needs access to central tables. The system is then said to be in "SYSTEM MODE".

1.1.4 Communication between the Tasks in a Job and the System

The jobs do all their communication with the system by jumping to the Request Processor in the JBSUP. Normally the jobs are running in restricted mode, however, so such a jump will cause a protection violation interrupt to occur. An interrupt routine in SINTRAN will then identify the jump as a "valid" one and route control to the Request Processor.

JBSUP has a dedicated area, the Task Parameter Area, into which it puts the control command before calling a job task. The task may then pick up the parameter string from this area.

1.1.5 Mass Storage

In all mass storage oriented systems, the handling of files has a central position. The NORD FILE SYSTEM, a sub-system of NORD-OPS, is used for definition and maintenance of mass storage files.

1.1.6 Multicomputer Configurations

NORD-OPS may utilize more than one computer. Any number of task computers may be connected to the original operation system computer. As task computer may serve one or more NORD-1, NORD-10 or NORD-5 computers.

1.1.7 Software

Most NORD software (especially that written at ND) will be available under NORD-OPS.

1.2 Versions of NORD-OPS

NORD-OPS will be available in several different versions, each version satisfying specific requirements. The kernel of NORD-OPS - the Job Supervisor and all auxiliary software (loader, processor a.s.f.) - will, however, be equivalent for all versions, thus unifying the use of NORD-OPS and making transitions of jobs and data between different installations easy. The differences will primary be upon the actual handling of I/O and in particular in the handling of standard input and output, and in the number of computers included in the system.

All versions of NORD-OPS require a minimum hardware configuration of a standard NORD-1 (NORD-10) computer, 24 - 32K of core storage and a mass storage device. Although not required, a card reader and a line printer are recommended for any extensive use of NORD-OPS.

NORD-OPS may be regarded as an extension to the SINTRAN Monitor System, thus providing a powerful real-time and batch operating system. NORD-OPS runs as one low priority RT-program (direct mode version), or as a set of low and medium priority RT-programs, utilizing relevant functions of SINTRAN (for example the I/O system).

Four different versions are available, the last three being extensions of the first. The last two are multicomputer versions, and may be combined in one.

1.2.1 Direct Mode Version

The basic SINTRAN version is intended for those who use their NORD-1 computer installation primarily for process control, but also have a limited need for batch. As NORD-OPS runs as a low priority RT-program, it will consume background time without interfering with urgent process controllers. This version needs at least 24K core memory.

1.2.2 Buffered Mode Version

The buffered mode version is intended for typical batch installations. This version includes, in addition to all features available with the basic SINTRAN version, buffering of standard input and output (see Section 1.1.2). This version needs at least 32K core memory.

1.2.3 Multicomputer Version

The multicomputer version uses an additional NORD-1 (NORD-10) as task computer. This is an extension of the buffered mode version.

1.2.4 NORD-5 Version

The NORD-5 version may use a NORD-5 computer for special compute bound tasks. The user decides which computer (NORD-1/10 or NORD-5) he wants to use for each task. The NORD-5 version is an extension of the buffered mode version, or the multicomputer version.

2 THE CONTROL LANGUAGE

The control language is the language in which the users of NORD-OPS present their work to the system. The language is composed of several control commands, each of which directs NORD-OPS to perform a spesific task.

The control language may be used from a variety of input devices, for instance card reader or Teletype. On cards the natural bounds for a command is the card: a command must be kept within one card, and a card cannot hold more than one command. On Teletype a command is framed by LINE FEED and CARRIAGE RETURN. Note also that some characters are reserved when using the control language; these characters are

$$\$ \ (\ , \ ) \ {}^{E}S_{C}$$

where ${}^{E}S_{C}$ is the multipunch ${}^{7}8_{9}$ on cards and the ESCAPE key on Teletype.

The NORD-OPS control commands are separated into two groups:

- external control commands
- internal control commands

The external control commands are directions to the system which must occur between jobs in the standard input stream, as opposed to the internal control commands which are processed during the actual execution of the job. In the complete SINTRAN version of NORD-OPS, when buffering of standard input is in effect, the external control commands are processed by the Standard Input Processor, while internal commands are just read and queued for later interpretation by the Job Supervisor. When buffering of standard input is not in effect, there is, however, no real difference in the processing mode of the two types of control commands; both are processed in succession when they are encountered in the input stream.

2.1 External Control Commands

The external control commands include commands for

- identifying a job to the system,
- controlling the processing of standard input and output,
- terminating NORD-OPS as an RT-program.

2.1.1 JOB Command

The JOB command has to be the first control command in a job, a job being defined as a collection of internal commands preceded by a JOB command and succeeded by an END-OF-JOB command (see below). The command gives the operating system a description of the job, such as priority, maximum run-time, etc.*

The general format of the JOB command (JOB card) is:

$JOB(accno, name, priority, maxsec, maxlines)

where

- accno — is the users accounting number (1-8 characters)
- name — is a name identifying the job (1-8 characters)
- priority — is the priority of the job (a number between 1 and 16 where 16 is the highest assignable priority); note that this parameter has no effect when used without buffering of standard input
- maxsec — is the maximum run-time for the job expressed in seconds
- maxlines — is the maximum number of lines which the job may print (on the standard output device)

Some examples of JOB commands:

$JOB(1000A, P. HANSEN, 10, 5, 1000)
$JOB(1001A, O. OLSEN, 13, 4, 2000)

2.1.2 End of Job

The END-OF-JOB command has to be the last control command in a job. It simply informs the operating system of the physical end of the job.

The command consists of the single character $^{E}S_{C}$ (see page 2-1) which is recognized whereever placed in the standard input stream.

---

* In the direct mode version of NORD-OPS, the parameters in the JOB command are not processed. They may be regarded as operator information only and may even be completely omitted.

2.1.3 MODE Command

The MODE command is used for controlling the processing of standard input and output. The command is used for two purposes:

- to vary the standard input/output devices,
- to control the buffering mode of standard input/output.

The general format of the command is:

$MODE(bufferflag, stip, stop)

where

- bufferflag — is a mnemonic controlling the buffering mode of standard input/output:
  - B — full buffering of standard input/output,
  - I — buffering of standard input only,
  - O — buffering of standard output only,
  - D — no buffering of standard input/output,
  - omitted — do not change buffering mode,
- stip — specifies the dsi* of the standard input device,
- stop — specifies the dsi* of the standard output device.

Note that the "bufferflag" has no effect in the direct mode version of NORD-OPS; in this version it must therefore be completely omitted.

Some examples of use:

Example 1:

Suppose current assignment of standard input/output devices are card reader/line printer, and you want to enter (and execute) a job conversationally. Assume the current version of NORD-OPS to be the direct one. Then the following card should be placed in the card reader as the last card:

$MODE(1, 1)

---

* data set identifier - see Chapter 3

When this card is encountered in the standard input stream, control is switched to the Teletype, and you may enter your job. The job entry will be "conversational" in the sense that each control command is interpreted and executed before next control command is read.

The last command entered via the Teletype may be

$MODE(4,5)

which switches the standard input/output streams back to card reader and line printer.

Example 2:

Another example may be that a job has so much standard input and/or output that buffering of it may not be feasible (assuming the buffered mode version of NORD-OPS). Then the following card should be placed immediately before the JOB card:

$MODE(D,4,5)

The effect of this card will be that NORD-OPS empties its standard input/output queues, and only when the queues are empty, reading of cards will be resumed. Processing will then continue in unbuffered mode. A

$MODE(B,4,5)

card will eventually switch back to buffered mode.

As the examples indicate, the NORD-OPS operator should have a complete control of the use of the MODE command. In particular, each batch of jobs through the card reader should be terminated with a

$MODE(D,1,1)

command, switching control to the Teletype.

2.1.4 EXIT Command

The EXIT command is used to terminate all NORD-OPS RT-programs under SINTRAN. In buffered mode, the queues are emptied before the corresponding RT-programs are terminated.

The command should be reserved for exclusive operator use.

The format of the EXIT command is

$EXIT

2.2 Internal Control Commands

Internal control commands are used for starting execution of absolute programs and processors (tasks: see Section 4.2.1). The programs must have been tailored by the BRF loader into a file in XQT format (see Chapter 5).*

The general form of an internal control command is:

$name,dsi(parameter string)

or

$name(parameter string)

where

- name — identifies the program or processor being wanted,
- dsi — is the data set identifier (see Chapter 3) of the mass storage file where the program is kept,
- parameter string is a string of characters transferred to the program (see also Chapter 4: Task Management); the string may consist of any characters except $ and $^{E}S_{C}$, although parentheses must be nested in pairs.

If 'dsi' is omitted, a search for 'name' is performed in the central directory.

Examples of internal control commands:

$FTN(4,5,80) — The FORTRAN compiler is started.

$MAC(4,5,80) — The MAC II assembler is started.

$MYPROG,50 — A program named MYPROG is loaded from the file identified through dsi=50 and started. No parameter information is provided.

$FILE(OP,,,52,JEA,FILE,T1,W)
The NORD FILE SYSTEM is loaded and started.

$YOURPROG,59(A,(B,C),13)
A program named YOURPROG is loaded from the file identified through dsi=59 and started. The parameter string contains one level of parentheses.

---

* Certain system tasks may, however, be permanently core resident or reside on a SINTRAN coreload.

2.2.1 MSG Command

The MSG command is a special purpose internal control command, used for giving messages to the operator. The format of the command is:

$MSG(message,flag)

where

- message — is a string of alphanumeric characters (, is not allowed) to be written on the operator's console,
- flag — is an optional parameter ('W' for 'wait') specifying that processing may not continue until the operator answers on the console.

When the MSG command is executed, the following string appears on the operator's console:

hh.mm.ss <job name> <message>
:

where hh.mm.ss is the current time in hours, minutes and seconds, <job name> is the name specified in the job command, and <message> is the message specified in the MSG command. The ':' will only appear if the flag W is specified.

For operator answer, see Section 8.1.3.

2.2.2 BIN and RBIN Commands

The BIN command may be used to accept binary input from the standard input device. This means that the operating system no longer tests for the ES_C character (End-of-job). This command thus has to be used with the greatest care. The format of the BIN command is

$BIN

The RBIN command resets the effect of the BIN command. The RBIN command must be used after a BIN command and before the END-OF-JOB command. The format of the RBIN command is

$RBIN

2.2.3 Data

Data may be specified along with the internal control commands in a job. Data belonging to a specific task must immediately follow the internal control command specifying that task. The program may then get access to its data through SYSIO (see Chapter 4), specifying the dsi of the standard input device. It is transparent to the program whether the standard input is buffered.

It is the task of the programs to check that all data belonging to them has been processed, and, on the other side, that no internal commands have been read as data! Thus each and every program has to establish its own end-of-data convention.

One should note, however, that the END-OF-JOB command is recognized by the input processor in SYSIO; thus it is not possible for a job to destroy another job, by reading its control commands as data.

Example:

```
$FTN
      1 FORMAT (* 1 A SIMPLE FORTRAN PROGRAM *)
        WRITE (5,1)
        END
        EOF
(another internal control command, or END-OF-JOB)
```

The FORTRAN compiler uses the EOF statement as an end-of-data indication.

3 NORD FILE SYSTEM

3.1 Introduction

As mentioned in the Introduction, NORD-OPS is a mass storage oriented operating system. The NORD FILE SYSTEM is the sub-system of NORD-OPS that organizes and supervises the usage and utilazation of the mass storage.

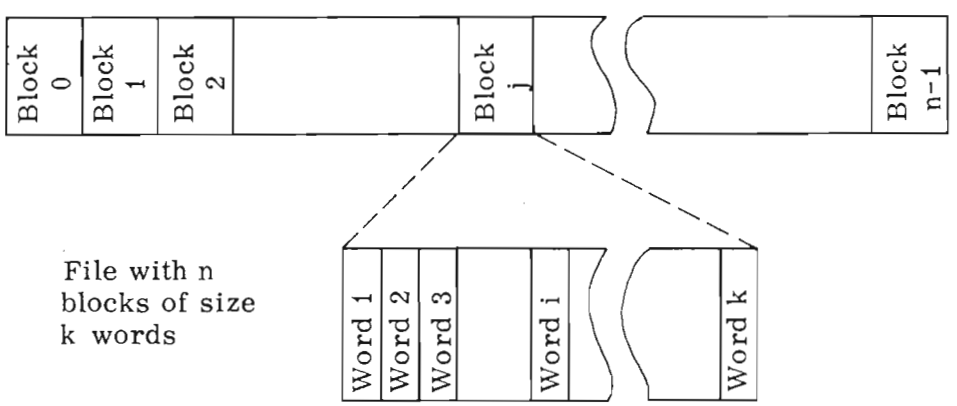
As the name should indicate, the NORD FILE SYSTEM structures the mass storage into files. (Thus, one may say that NORD-OPS is a file oriented operating system.) This is a suitable way of organizing storage of data and programs in a multi user environment.

Logically, a file is a structured collection of data, where data may be a program, card images from the card reader, listing from a job ready for printing, data written by a program, etc. A program file, for instance, consists of an absolute program in XQT format (see Section 5.3).

The logical file structure may be defined on different levels (see for instance Appendix C).

Basically, however, all files consist of an ordered sequence of storage units called blocks, all blocks being of the same size, a size which may be specified by the user at file allocation time. The block is thus the basic addressable unit of a file, the block sequence number constituting the unique address of a block.

FIGURE: BASIC LOGICAL LAYOUT OF A FILE



Physically, a (mass storage) file consists of storage space on a disc pack or a drum. The space for a file is organized from non-contiguous segments. A segment itself must (by definition) be a contiguous storage element. At present, this is a track or 2048 words. The transformation from logical file addresses to physical disc or drum addresses is done by the file access routines.

Now, the NORD FILE SYSTEM consists of three parts:

- The File Label System (FLS)

  The file label system is the part of NORD-OPS that supervises the organization of mass storage into files. A number of directories are maintained for this purpose. The file label system is organized as a package of utility routines (FILE), and it is described in Section 3.2.

- The File Service System (FSR)

  This system supplies the users with the necessary routines to make backup copies of files, obtain listings of the central directories, etc. It too is organized as a package of utility routines and is in fact included in the FILE package. The file service system is described in Section 3.3.

- The Block Input/Output System (BLCIO)

  BLCIO is the system through which the basic accesses to files have to pass. BLCIO is part of the core resident part of NORD-OPS, and is called through a system service request (see Section 4.4). BLCIO handles both mass storage and magnetic tape.

3.2 The File Label System

3.2.1 Directories

The FLS makes use of two (types of) directories to keep record of the physical appearance of files and to provide information on the files for the access routines.

3.2.1.1 The Volume Table of Contents

Unlike many other systems the NORD FILE SYSTEM makes no use of centralized directories. Instead, each mass storage volume is selfcontained with control information regarding the files on that volume. Thus, the system is prevented from breaking down if a spesific disc pack should crash. Also, the exchange of disc packs between different installations is facilitated.

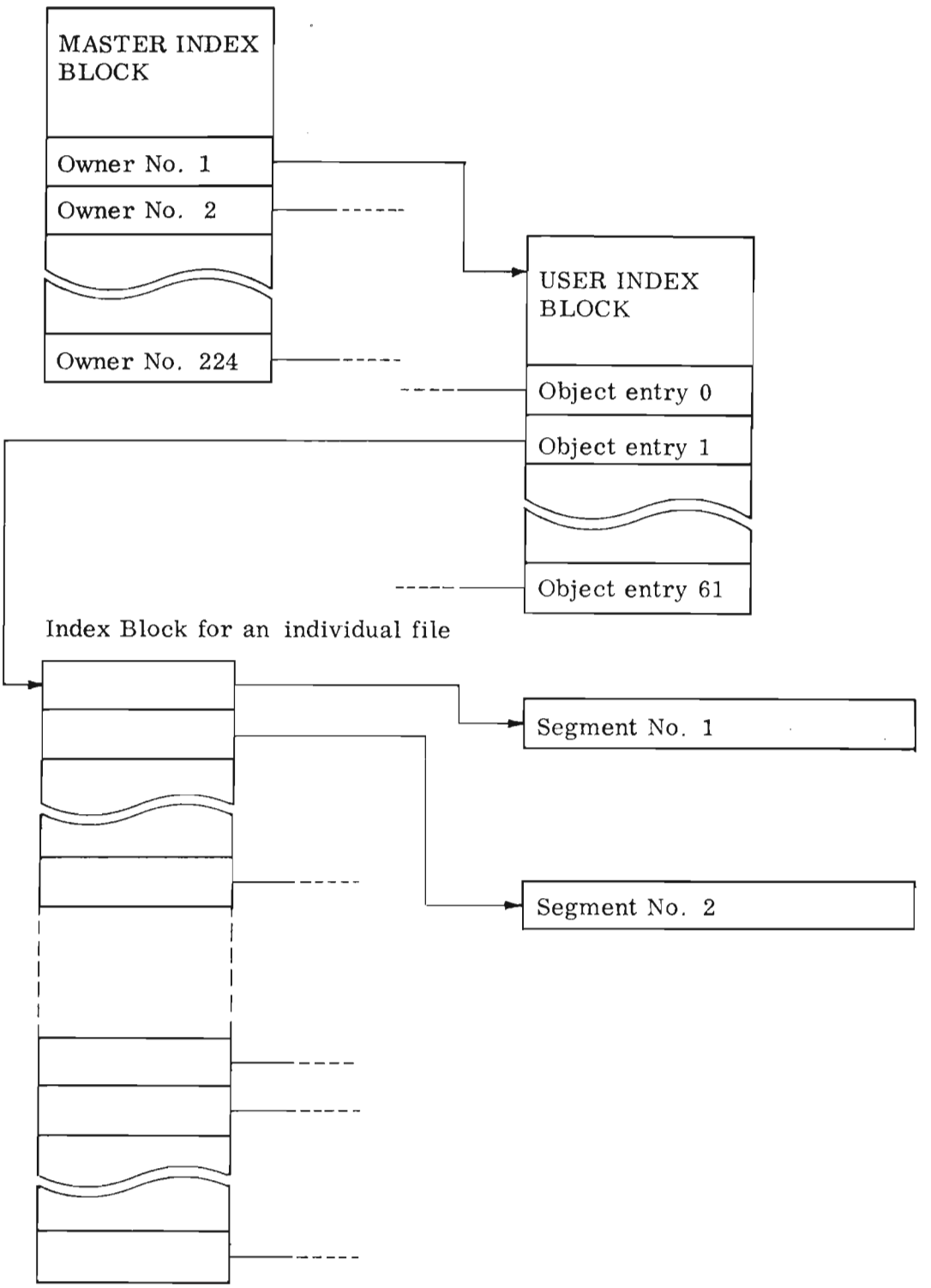
The control information residing on a mass storage volume is collected in a directory called the volume table of contents (VTOC). The VTOC is organized as an indexed sequential file, and below each type of index will be explained. The structure of the VTOC may also be seen from the accompanying figure.

Each mass storage volume in the system is identified by a master index block (MIB), which is stored in a fixed location on the volume. The MIB constitutes the root of the VTOC and contains, besides the volume identification, volume dependent information such as physical layout of the storage medium and a map showing used/unused parts. Finally, the MIB contains a directory of all "owners" (see Section 3.2.2.1) of files on the volume, pointing to the next level in the VTOC, the user index blocks.

Associated with each "owner" of files on the mass storage volume concerned, a user index block (UIB) is found, the main contents of which are a directory of all files (on that volume!) belonging to the "owner". The entries in this directory, called object entries, are the bearers of the file dependent information, such as file identification, physical layout and other control information.

For further details the interested reader is referred to Appendix B where the detailed volume layout is shown, including a complete description of the VTOC.

FIGURE: THE VOLUME TABLE OF CONTENTS



3.2.1.2 The Open File Table

The open file table (OFT) is a core resident table keeping information about all files currently in use, both permanent and scratch (see Sections 3.2.5 and 3.2.6). This information includes a map of the physical layout of each file, and also the necessary control information. The OFT entries are created by the FLS; it is invoked by BLCIO each time a block transfer is requested.

3.2.2 File and Device Identification

3.2.2.1 File Identification

A file is referenced by its identification; thus, the file identification has to be uniquely defined. The format of the file identification is:

owner , file name , edition

All fields are alphanumeric; "owner" from 1 to 8 characters, "file name" up to 16 characters and "edition" up to 4 characters.

Example:

5352B, PRDATA, A41

3.2.2.2 File Classification

In addition to the file identification described above, a file classification code ("type") is provided. It is not used by the file system itself, but may be used by installation dependent utility routines. For instance, it may contain an expiration date.

The file classification code is a field containing up to four characters.

3.2.2.3 Device Identification

All mass storage volumes and unit record devices in the system will have an identifier of two-by-two characters. The first two are a code representing the device type, and the last two are a number representing the specific mass storage volume or unit record device.

Valid device type codes are:

DK	- disc
DR	- drum
TT	- Teletype
PR	- paper tape reader

PP - paper tape punch
CR - card reader
LP - line printer
MT - magnetic tape
PL - plotter

Examples:
DK, 04 - disc pack number 4
TT, 02 - Teletype number 2
MT, 01 - magnetic tape volume number 1

3.2.3 File Security

A password of maximum four characters is associated with each file at allocation time. This password must be specified with all FLS requests that modify fixed information in the mass storage resident directories of the file system.

Also, with each file a usage mode ("um") is associated. The usage mode may be one of the following:

W - the file may be both read and written into,
R - the file may be read only,
L - the file is locked, i.e., it may not be accessed at all.

Usage mode is always set to W at file allocation time. (It may, of course, be changed later - see Section 3.2.5.5.)

3.2.4 Data Set Identifiers

The file identification scheme described in Section 3.2.2.1 is quite relevant when identifying a file externally, i.e., to the outside world, because uniqueness is easy to achieve. It would, however, be quite bothersome if we should be forced to use those lengthy names for referencing files within an application program. Also, writing file independent programs, although not impossible, would be quite troublesome.

Therefore, a quite different naming scheme is used within the programs. Instead of a lengthy character string, each file is identified by a number, the so called data set identifier ("dsi"). The correspondance between the external (unique) identification and the internal name (the dsi), may be established and broken freely by the user (see Section 3.2.5.6 and Section 3.2.5.7). Thus, a flexible mapping of external identifications onto dsi's is available, and file independence is easily achieved at no cost on the uniqueness.

It may be noted that the dsi notation is used also for unit record devices (i.e., Teletype, card reader, mag. tape unit, etc.). Here a fixed relationship holds between devices and dsi's. Also, some dsi's are reserved for exclusive system use. The present usage is as follows:

dsi =		0	dummy
		1	Teletype
		2	paper tape reader
		3	paper tape punch
		4	card reader
		5	line printer
		7	display or Teletype 2
79	to	89	system files
		90	standard input
		91	standard output

However, these are only default assignments, made at job initiation time. During job execution, the user may freely reassign any dsi between 1 and 78 to any unit record device or mass storage file. Thus, not only file independence, but even device independence of programs may easily be achieved (see also Chapter 4 on SYSIO and BLCIO).

3.2.5 Permanent Files

Permanent files are files which are permanently resident on mass storage. They may be scratched, i.e., the mass storage space which they occupy may be released, only as the effect of an explisit command.

The various commands related to permanent files will now be treated. By the use of these commands the user may

- allocate files
- release files partly or totally
- expand existing files
- modify identification, password and usage mode
- open files for data transmission
- close files

Some frequently used command parameters are:

mst	- mass storage type
msn	- mass storage number (mst and msn together constitute the mass storage volume identification; the system pack DK, 01* is default),
owner filename edition	- file identification, see Section 3.2.2.1 Note, however, that default for "owner" is the accounting number in the JOB command.
type	- file classification code, see Section 3.2.2.2,
password	- see Section 3.2.3,
blocksize	- logical block size in words; default value is 256,
nblocks	- number of blocks,
dsi	- data set identifier, see Section 3.2.4.

Where nothing else is noted, 0 is default value.

3.2.5.1 Allocate a File

When mass storage space is required for a file, you may specify to the system the amount of space needed, or you may not. In case you specify insufficient space, or even no space at all, the system will automatically reserve new mass storage space, one segment at a time, each time a previously unassigned block is attempted written** Thus, effective utilization of the mass storage may be achieved.

The general form of the allocate request is:

```
ØFILE(AL, mst, msn, owner, filename, edition, type, password,
      blocksize, nblocks, allocate mode, track addr)
```

The allocate routine reserves mass storage space, if requested, and generates an entry for the file in the appropriate VTOC. The usage mode is set to W.

Note that if the block size is not an integral multiple of the sector size for the mass storage volume on which the file is stored, part of every block is waisted.

---

* For those having drum as system volume, DR, 01 is the system "pack".

** But see Section 3.2.5.6 - Open

Now, if mass storage space was requested in the allocate request (i.e., "nblocks" was specified), the number of segments required to accomodate the desired number of blocks will be reserved.

The manner in which this space will be physically distributed is controlled by the "allocate mode" parameter, and one of the following three modes may be specified:

- Absolute Allocation ('ABS')

  When absolute allocation is requested, the system will put the file into the contiguous area starting at the track address specified by "track addr", provided this area is unused and contains no bad tracks.

- Cylinder Allocation ('CYL')

  The cylinder allocation mode is spesific for discs; it has no meaning for drums. When cylinder allocation is requested, the system will distribute the file over the smallest number of cylinders possible. It will also assign contiguous number sequences to the segments of the file that is contained within each cylinder. For instance, suppose we have a file consisting of n segments and find that we may accomodate the file on two cylinders, R segments on the one and the rest on the other. Then the segments on the first cylinder will be numbered from 1 to R, and those on the second from R+1 to n. Note, however, that the system may freely distribute the segments within each cylinder; physical contiguity is not implied.

- Free Allocation ('FRE')

  The free allocation mode should be selfexplanatory in the sense that the system has got free hands.

  Default allocation mode is 'FRE'.

  Some examples:

  $FILE(AL,DK,02,5352B,JEA,1,2,3,300,4,FRE)

  A file is allocated on disc pack number 2 (logical description of pack, it has nothing to do with the physical position of the pack in the system). owner 5352B, filename JEA, edition 1, type 2, password 3, blocksize 300, number of blocks 4, and free allocation mode).

  $FILE(AL,,,5352B,JEA,1,2,3,300,4,FRE)

  The same file as above, with the exception that the file is placed on the system pack.

$FILE(AL, , , 5352B)

A file with owner 5352B is allocated on the system pack. No space will be reserved for it at the moment. Filename, edition, type and password equals 0.

$FILE(AL, DR, 01, 5352B)

The same file as above is generated on drum 01.

3.2.5.2 Release a File

$FILE(RL, mst, msn, owner, filename, edition, password, delete code, lower bound, upper bound)

The release routine releases partly or totally the mass storage space occupied by a file, according to the "delete code" parameter which may take on the following values:

- A - The file and its associated VTOC entry are scratched.
- P - All segments contained in the area enclosed by the lower and upper bound parameters (block numbers!) are released.

The file invoked may not be in current use, i.e., it must be closed.

Note that for delete code P, the released space will still exist logically. A future attempt to write into this area will thus make the system reassign the space, one segment at a time (see also Section 3.2.5.1), though most likely at different physical locations. The released space may not be read, however.

Some examples:

$FILE(RL, DK, 01, 5352B, JEA, 2, 3, A)

$FILE(RL, DK, 01, 5352B, JEA, 2, 3, P, 2, 4)

$FILE(RL, , , 5352B, , , , A)

3.2.5.3 Expand a File

$FILE(EX, mst, msn, owner, filename, edition, nblocks, allocate mode)

The file will be expanded with the number of blocks specified in the "nblocks" parameter. The invoked file may or may not be in current use (i.e., opened).

The way of expansion is specified by the "allocate mode", which can take on one of the two values "FRE" and "CYL". The interpretation of these parameters is as for the corresponding ones in the allocate command.

Some examples:

```
$FILE(EX, DK, 01, 5352B, JEA, 2, 5, CYL)
$FILE(EX, , , 5352B, , , 6, FRE)
```

which is the same as

```
$FILE(EX, , , 5352B, , , 6)
```

as FRE is default value of "allocate mode".

3.2.5.4 Modify File Identification, Password and File Classification

```
$FILE(MD, mst, msn, owner, filename, edition, password,
        new filename, new edition, new type, new password)
```

This command enables the user to rename his files. The omitted "new" fields are taken as zero fields, i.e., the contents of the corresponding new entries will be zero.

Some examples:

```
$FILE(MD, DK, 01, 5352B, JEA, 2, 3, HANS, 4, 5, 6)
$FILE(MD, , , 5352B, JEA, 2, 3, , 4, 5, 6)
```

Here the new file name takes on zero value; in order to rename this file again we may submit the following command:

```
$FILE(MD, , , 5352B, , 4, 6, HANS, 4, 5, 6)
```

3.2.5.5 Modify File Usage Mode

```
$FILE(UM, mst, msn, owner, filename, edition, password,
        usage mode)
```

This command is used to change the usage mode of a file (see Section 3.2.3).

Some examples:

```
$FILE(UM, DK, 01, 5352B, JEA, 2, 3, L)
$FILE(UM, , , 5352B, JEA, 2, 3, R)
$FILE(UM, DR, 01, 5680K, HANS, TO, TRE, W)
```

3.2.5.6 Open File for Data Transmission

$FILE(OP, devt, devn, dsi, owner, filename, edition, usage, pflag, density)

Here, 'devt' and 'devn' contain 'device type' and 'device number' specifications, thus identifying a mass storage volume or a unit record device (see Section 3.2.2.3).

The open routine prepares a mass storage file or a unit record device for data transfer, and, in the case of a mass storage file, a core resident OFT entry to be used by the I/O system is generated.

The file (or the unit record device) is opened for a usage, which may be:

R - The file is opened for input; all files, provided they are not locked, may be opened for input.

W - The file is opened for both input and output; read-only files may not be opened for this usage.

For mass storage files you may further specify the optional parameter 'pflag' (mnemonic 'L') to designate that you do not want automatic expansion of the file (see Section 3.2.5.1)! If the 'pflag' is omitted, automatic expansion may occur in the normal way.

For magnetic tape units the 'pflag' and 'density' parameters must be specified. The 'pflag' specifies whether even ('E') or odd ('O') parity should be used, and the 'density' is the density of the tape (200, 556 or 800 bpi).

Note that some unit record devices do not have to be opened. Those occurring in the standard list (see Section 3.2.4) have to be opened only when you want to change the default assignments.

Some examples:

$FILE(OP,DK,01,40,5352B,JEA,2,R)

$FILE(OP,DR,01,30,5352B,JEA,2,W)

The following example

$FILE(OP,DR,01,30,5352B,JEA,2,W,L)

is the same as above, with suppression of the automatic expansion mechanism.

$FILE(OP,CR,,10)

The (only) card reader is assigned a dsi of 10.

$FILE(OP,MT,01,33,,,W,E,800)

The last command indicates the opening of magnetic tape volume 1, usage mode read-write, even parity and 800 bpi.

3.2.5.7 Close a File

$FILE(CL, dsi)

The close command releases the core resident OFT entry associated with the file and updates some fields in the relevant VTOC. Also, a possible system buffer associated with the file (see Section 4.3.3) is released.

Example:

$FILE(CL, 50)

The file associated with dsi = 50 is closed; dsi = 50 may now be assigned to another file.

3.2.6 Scratch Files

A scratch file is a temporary file existing for only a short interval of time, at most as long as the job allocating it is running. Thus, normally, a scratch file and its contents will be lost when the file is closed. However, a special command is provided (see Section 3.2.6.3), which may be used to save a scratch file, making it a permanent file.

The commands related to scratch files are described in the following. The parameters used have the same meaning as those used with permanent file commands (see Section 3.2.5).

3.2.6.1 Allocate and Open a Scratch File

$FILE(AS, mst, msn, dsi, blocksize, blocks, allocate mode)

This command makes the system allocate and open a scratch file of the desired size. It is analoguous to the combined effect of the allocate and open commands for permanent files.

Note that a scratch file is directly identified by the dsi, indicating that its existence is limited to the time it is open. Also, a scratch file may not be opened with absolute allocate mode!

Some examples:

$FILE(AS, , , 30, 300, 10, FRE)

$FILE(AS, DK, 05, 25, , 20, CYL)

$FILE(AS, , , 40)

The last command causes the system to allocate a scratch file on the system pack, with dsi = 40, blocksize = 256 and no room initially reserved for it.

3.2.6.2 Close and Release a Scratch File

$FILE(CS,dsi)

This command closes and releases a scratch file, thereby making its mass storage space available for other files before job termination. Also, the "dsi" is made available. The command is quite analoguous to the combined effect of the close and release commands for permanent files.

Example:

$FILE(CS,30)

3.2.6.3 Save a Scratch File

$FILE(SS,dsi,file name,edition,type,password)

A scratch file is allocated by the file system as a permanent file with a special identification. The save scratch command enables the user to change this identification to his own specifications, and thereby making the file permanent. The file is afterwards in every respect treated as an ordinary permanent file.

Example:

$FILE(SS,30,JEA,1,2,3)

3.3 The File Service System

3.3.1 General

The file service system (FSR) will supply the necessary routines to

- obtain information about the (mass storage) files in the system;
- dump backup copies of mass storage files to mass storage, magnetic tape (9 track only) or other suitable output media;
- restore backup copies of mass storage files from mass storage, magnetic tape or other suitable input media.
- enter and delete mass storage volumes.

Most of the file service system routines are only intended for the system operators and maintenance personnel, but two of them (CP and BT) may be used by ordinary users.

3.3.2 The Commands

The file service system is used by submitting commands to FILE as follows:

$FILE(code,parameters)

where code identifies the specific function called upon.

3.3.2.1 List Groups of File Labels

$FILE(LI,mst,msn,owner,file name,edition,password)

Here the parameters are interpreted as in the file label system, except the password. The password may be specified as a special "system password" to list the passwords of the files.

The LIst command lists pertinent information about the file specified. If any fields are omitted, information on a group of files for which the remaining parameters fit may be obtained.

For example, if only the owner is specified, information on all files belonging to that owner will be listed. If no parameters are specified, information on all files on the system pack will be listed.

The list will appear on the standard output device. The following information will be listed for each file:

owner
file name
edition - file identification, see Section 3.2.2.1.

type - file classification code, see Section 3.2.2.2.

address of
index block - the sector address of the index block of the file. The address is given relative to the beginning of the actual volume, and the sector size is always 256 words.

file length - number of data tracks (1 track = 2K) in the file. The index blocks are not included.

logical block
size - the logical block size of the file.

number of
times used - the number of times the file has been opened.

number of times accessed - the number of transfers to or from the file.

date - the date of allocation.

password - the password of the file. This information is only listed if the system password is specified on the LIst command.

The LI routine is optional.

3.3.2.2 Make a Copy of a File

$FILE(CP,idsi,odsi)

where

- idsi is the data set identifier of the input file (or unit record device);
- odsi is the data set identifier of the output file (or unit record device).

The CoPy routine copies the contents of the input file onto the output file. At least one of the parameters must be a mass storage file.

If the odsi refers to a magnetic tape, the last information on the tape will be an end-of-file mark.

If the odsi refers to a unit record device, the first two words output will contain*

- number of blocks in the file (negated),
- block size words.

Then, immediately following these two words, the file will be output as a continuous stream of binary information.

If the idsi refers to a unit record device, it is assumed that the input file is organized as described above. Thus, the copy command may be used to obtain backup copies of mass storage files on, e.g., paper tape, which may then also be reloaded by the same copy command.

---

* This information is currently only used to record the total size of the file. The "block size" need not have relationship with any physical block size (e.g., on mag.tape); in fact it is always set to 100_8.

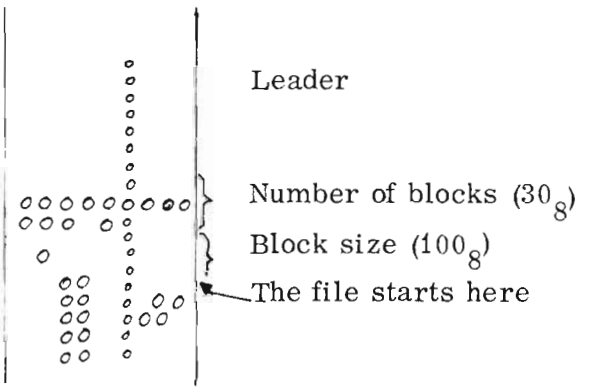


Figure: Format of file on paper tape (example).

3.3.2.3 Make a Copy of a Mass Storage Volume

$FILE(CD,mst1,msn1,unit1,mst2,msn2,unit2,file marks,password)

where

- mst1,msn1 is the source mass storage volume.
- unit1 is the physical unit of the source volume.
- mst2,msn2 is the destination mass storage volume.
- unit2 is the physical unit of the destination volume.
- file marks is the number of end-of-file marks to skip before copy starts if one of the volumes is a magnetic tape.
- password is the "system password" that must be specified.

The Copy Disc/Drum routine copies the contents of one mass storage volume to another mass storage volume. At least one of the volumes must be a disc or drum. The two volumes may have the same logical mass storage identification.

Examples:

$FILE(CD,DK,01,0,DK,01,1,,....)

Make a new copy of disc volume DK,01 on unit 0 on the DK,01 on unit 1.

$FILE(CD,DK,01,0,MT,93,1,....)

Make a backup copy of DK,01 on mag.tape MT,93. Skip one 1 end-of-file mark before the copy starts. The copy will end with an end-of-file mark on the tape.

Only the used part of the volume is copied, and the destination volume has to be initialized by the INit routine (see Section 3.3.2.4) before the copy can start. The CD routine is optional.

3.3.2.4 Initialize a Mass Storage Volume

$FILE(IN,mst,msn,unit,flag,password)

where

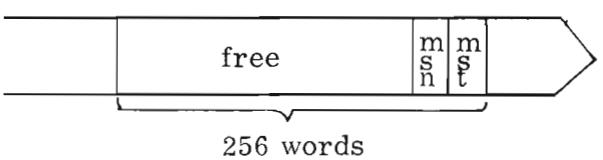
- mst,msn — are the mass storage volume identification
- unit — is the physical unit
- flag — is a initialize/update flag
- password — is the "system password" that must be specified.

The INitialize routine may be used to label and search for bad tracks on new mass storage volumes, or search for new bad tracks on old volumes. The 'flag' parameter may have two volumes:

- I — Initialize volume. That means to search for bad track and label volume.
- U — Update volume. That means to search for new bad tracks. The volume label is unchanged.

If the mass storage volume is a magnetic tape, the 'flag' is ignored, and the function is always initialize.

After the initialization, the volume has been given the logical identification mst,msn. The volume is now ready for use by the file system. If the volume is a magnetic tape, the following information is written on the tape, as the first record after the load point mark:



3.3.2.5 Search and list bad Tracks in a File

$FILE(BT,dsi,wflag)

where

- dsi — is the data set identifier of the file.
- wflag — is an optional parameter, that will, if specified, cause a test pattern to be written onto the file.

The file 'dsi' is searched for bad tracks, and the page addresses of such bad tracks are listed. Note that the page size is always 256 words. A new track is allocated, and the contents of the bad track are copied to the new track, if possible. The BT routine is optional.

3.3.2.6 Enter Mass Storage Volume

$FILE(EN,mst,msn,unit)

The ENter routine must be used before allocation and opening of files on a mass storage volume. The routine gives the file system the connection between the mass storage identification 'mst,msn' and the physical unit 'unit'. The ENter routine checks that the correct volume is entered on the given unit. This check may be skipped in two ways:

- The connection between mass storage identification and physical unit may be defined at system generation time.
- A magnetic tape may be entered as mass storage number 65535. Then the check of the first record on the tape is skipped.

3.3.2.7 Delete Mass Storage Volume

$FILE(DL,mst,msn,unit)

The DeLete routine is the opposite of the ENter routine. After deletion, the connection between the identification 'mst,msn' and the physical unit 'unit' no longer exists. The DeLete routine checks that no open files exist on the actual volume.

3.3.2.8 Release Scratch Files

$FILE(RS,mst,msn)

The Release Scratch routine may be used after an abnormal stop of the operating system. The main task of RS is to remove any scratch files that exist on the volume. A side effect of RS is to remove users without files.

4 SYSTEM SERVICE REQUESTS

4.1 General

4.1.1 Introduction

NORD-OPS provides a set of subprograms or routines which a user may call from within his programs*. These routines, called the system service routines, cover roughly three application areas:

- block oriented input/output (mass storage and magnetic tape);
- stream oriented input/output;
- task management.

Each of these application areas will be treated in detail in subsequent sections. In addition there is a number of miscellaneous routines.

4.1.2 The Request Processor

All system service requests are scheduled by the request processor, a routine in the core resident part of NORD-OPS. Its entry point is in a fixed location: 501_8; thus, system service is called upon by a subroutine jump to location 501_8.

Normally, however, the user programs are run in restricted mode. A jump to the request processor will then cause a protection violation interrupt to occur. An interrupt routine in SINTRAN will recognize the jump as a "valid" one, and in due time route control to the request processor as if the jump was direct.

4.1.3 Calling Conventions

The standard for calling system service under NORD-OPS is as follows:

```
P-1     LDA    *+4
P       JPL    I *+2
P+1     'code'
P+2     5018
P+3     'parameter list pointer'
P+4     return here
```

Here 'code' identifies the specific function called upon. At return the X and B registers will have the same values as before the subroutine jump.

---

* The routines are directly callable from assembly programs only, but provisions are made for the direct use of certain routines also from FORTRAN. For details, however, the appropriate FORTRAN manual should be consulted.

The parameter lists are organized according to SINTRAN specifications, that is, the parameter lists are lists of pointers to the actual variables. There are several reasons for adopting this standard:

- Compatibility between NORD-OPS parameter lists and SINTRAN parameter lists means that certain functions in SINTRAN (e.g., byte oriented I/O) may be offered to NORD-OPS users at only a light cost.

- The SINTRAN standard is well suited for FORTRAN subroutine calls, thereby making it easy to transform FORTRAN calls to NORD-OPS system service requests.

(See also Section 4.2 - Task Management.)

In subsequent sections and subsections the various functions callable by system service requests will be described. The calling sequences will then be represented in the following way:

/ function ('code') /

(parameter 1
(parameter 2

(parameter n

where 'code' is the identifier of the function described above, and '(' is the customary literal symbol used in the MAC assembler.

Note: The constant 501_8 should be represented symbolically in case of changes to the system.

4.2 Task Management

4.2.1 Definition

The unit of work established by an internal control command is called a task. Thus, a task may be a user's program, a system processor (such as the FORTRAN compiler) or a utility package.

4.2.2 Entrance to and Exit from Tasks

4.2.2.1 Entrance to Tasks

A task established by NORD-OPS is always entered in its primary entry point, i.e., the entry point specified by)9BEG in MAC or main program in FORTRAN.

On entrance, the A register points to a main storage area containing the parameter string discussed in Section 2.2. The parameter string is packed, two characters per word, starting with the first left parenthesis and terminating with the last right parenthesis.

4.2.2.2 Normal Exit

A normal exit from a task routes control back to the job supervisor. The JBSUP will then assume the next record in the standard input stream to be a control command (see, however, Section 4.2.3 - Subtasks).

A normal exit is provided by the following system service request:

/ JBSUP (20_8) /

4.2.2.3 Abnormal Exit

Due to errors detected by a program, the program may want to terminate its processing abnormally. (For instance, if the FORTRAN compiler is directed to read its source input from the file assigned to dsi = 50, but no such file is provided, then it may as well abort its processing.)

Such an abnormal or abort exit is provided for in NORD-OPS. It consists of the following system request:

/ ABORT (22_8) /

The ABORT request results roughly in two system actions.

1) The following message will be printed on the standard output device:

   USER ERROR 'number'

   where 'number' is the contents of the A register at the time of the ABORT request.

2) Normally, the standard input stream is flushed until EOJ is detected (the job is aborted). If, however, control commands are entered conversationally, then control is routed to the job supervisor and processing continues as if the return was normal (see, however, Section 4.2.3 - Subtasks).

Errors may also be detected during control command processing, or during processing of system service requests. In such cases the message

SYSTEM ERROR 'number'

will be printed; in all other respects the system will behave as though a user abort request was generated. The various system error numbers and their meaning are listed in Appendix A.

4.2.3 Subtasks

In certain cases it might be convenient for a task to use facilities provided for in other programs.

Example:

A task may want to give a message to the operator, telling him to perform some external action. Only when the action is completed, it does want to continue processing. Note that this function is provided for in the MSG command (Section 2.2.1).

To meet this demand, NORD-OPS provides a facility for a task to establish a new task, called a subtask of the former task. This facility is called by the ESTAB and CTASK system service requests.

4.2.3.1 Call Subtask with Return

/ ESTAB(23_8) /

m.a.

Here 'm.a.' is the address of a main storage area containing an internal control command. The control command must be packed, two characters per word, and terminated by CR (carriage return).

The ESTAB request results in the following system actions:

1) The task currently executing, together with sufficient control information, is written onto mass storage.

2) Then the task described in the internal control command pointed to, the subtask, is established and entered. The rules for entrance to ordinary tasks, treated in Section 4.2.2.1, are valid for entrance to subtasks as well and will not be repeated here.

When the subtask has finished its execution, it returns control to the job supervisor, either normally, or abnormally, in exactly the same way as ordinary tasks (Sections 4.2.2.2 - 3). The job supervisor will then:

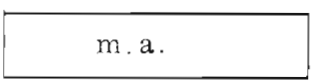
- If the return was normal, the previous task will be re-established together with its environment. The original contents of the registers will be lost.

- If the return was abnormal, the re-establishment of the previous task(s) is suppressed, and processing continues as if it were the original task that aborted the job (see Section 4.2.2.3).

Subtasking may be performed in eight levels.

4.2.3.2 Call Subtask without Return

/CTASK(21)/



Like ESTAB, the 'm.a.' is the address of a main storage area containing an internal control command.

The CTASK request results in the following system actions:

- The task described in the internal control command pointed to, the subtask, is established and entered, exactly like action 2) in the ESTAB request.

After subtask execution, the control is returned to the job supervisor, which will perform one of two actions:

- If the return was normal, the next command will be taken from the standard input device, as if it were the original task that returned.

- If the return was abnormal, processing continues as if it were the original task that aborted (see Section 4.2.2.3).

4.2.4 Subroutines

A word may be said about subroutine calls under NORD-OPS.

When the user calls his own subroutines, he may, of course, establish his own calling conventions. When using system library routines, however, certain general guiding lines should be pointed out, and, at least in principle, the user should adopt these standards to ease the transition of subroutines among users.

The standard generally adopted in system library routines is the standard SINTRAN calling sequence (see also Section 4.1.3). This standard is illustrated by the following example:

```
JPL I (SUBR
```

where SUBR is the name of the subroutine called. The A register contains the address of a parameter list. This parameter list contains the addresses of the actual parameters. Return is to the location following JPL. Also, normally the B and X registers remain unchanged by the subroutine.

Note, however, that certain sets of library routines do not conform to this standard. As an example the Re-entrant FORTRAN Mathematical Library may be mentioned. Therefore, the above considerations should not prevent the reader from carefully studying the appropriate manuals.

4.3 Stream Oriented Input/Output

4.3.1 Introduction

A generalized stream oriented I/O system is provided for under NORD-OPS. The system is labeled System Input/Output System (SYSIO), and its I/O commands are generalized in the sense that they are device (or hardware) independent. Thus, the same commands apply for unit record devices, mass storage and magnetic tape*.

A special buffer system takes care of data to be transferred to and from mass storage.

4.3.2 File Structure

To SYSIO all files look like an unstructured sequence of (8 bits) bytes or groups of bytes called records. In the buffers and on the word oriented storage media (magnetic tape, mass storage) the bytes are packed two bytes per word. The bytes are so located in the words that the lowest numbered byte occupies the 8 most significant bits.

---

* Under the present version of NORD-OPS, SYSIO treats magnetic tape more or less as a mass storage file. The block size is a system generation parameter.

A record is a sequence of contiguous bytes. Generally, the files consist of an arbitrary number of records. The record structure is, however, not inherent in the file, but is rather defined a d h o c by the user, and may even be changed from one call to another.

4.3.3 Buffer System

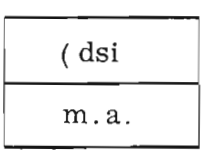
A program using SYSIO for communication with mass storage* files, is, in general, responsible for allocating intermediate main storage buffers for mass storage* I/O. This is done by a "define buffer" system service request.

For unit record devices, however, the buffer system of SINTRAN is used. Therefore, to make device independence in the use of SYSIO possible, NORD-OPS maintains a pool of mass storage* buffers. SYSIO automatically allocates one of these if an access request to an open mass storage* file is submitted, and a buffer has not previously been provided. Nevertheless, this automatic buffer assignment mechanism should be used with care, because congestion may arise if the user depends on too many buffers being automatically assigned to him. This is due to the limited amount of available buffer space.

The various routines in the buffer system, together with their calling sequences, will now be described. Requests for service by these routines, referring to unit record devices, are dummy.

4.3.3.1 Define Buffer

/DBUF(5)/

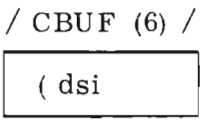


The DBUF system service request defines the main storage area starting at 'm.a.' to be used as an intermediate buffer for the mass storage file referred to by 'dsi'. The size of the buffer area must at least be equal to the block size of the file.

---

* Under the present version of NORD-OPS, SYSIO treats magnetic tape more or less as a mass storage file. The block size is a system generation parameter.

4.3.3.2 Clear Buffer



The CBUF system service request is used when you want to switch the current use of the mass storage file, referred to by 'dsi', from input to output, or vice versa. It may also be used to simply terminate processing of current block.

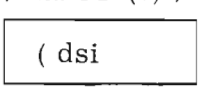
If you are currently outputting on the file, the contents of the non-filled part of the buffer is zeroed and the buffer contents are then transferred to the file.

If inputting, the rest of the buffer is skipped.

Thus, in any case the next I/O request will refer to the first byte of the next block. Also, the next I/O request may be either an input or an output request, thereby allowing a switch from input to output or conversely.

A CBUF request will have no effect if it occurs immediately after another buffer system request referring to the same file, i.e., if no intervening I/O request has occurred.

4.3.3.3 Release Buffer



The RBUF system service request is used to terminate processing of the mass storage file referred to by 'dsi'.

The release buffer routine will first call the clear buffer routine to terminate processing of current block. Then control of the buffer is released; if automatic buffering was used, this means that the buffer is returned to the buffer pool; otherwise, it simply implies that SYSIO "forgets" the connection between buffer and file, so that the buffer area may be used for other purposes.

An RBUF request will rewind the file.

4.3.4 Input/Output Commands

The input/output commands provided for under SYSIO are the four basic stream oriented input/output commands of SINTRAN:

- input one byte (INCH)
- output one byte (OUTCH)
- input one record (INDAT)
- output one record (DATUT)

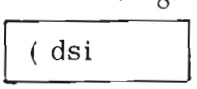
The calling sequences are modified to conform to NORD-OPS system service request standards, but the parameter lists are identical, thereby allowing the SYSIO routines to call their SINTRAN counterparts more or less direct.

It may be noted that for mass storage files, the first I/O request following the DBUF request determines the processing mode (either input or output). If a change in processing mode is wanted, a CBUF request must be executed (see Section 4.3.3.2). The same consideration does not apply to duplex unit record devices such as Teletypes, because SINTRAN provides one buffer for each transmission direction.

The input/output commands will only be reviewed in the following; a complete description is found in SINTRAN II User's Guide.

4.3.4.1 Input One Byte

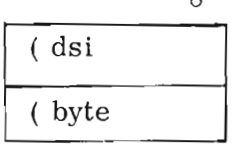
/ INCH (11_8) /



The INCH system service request returns the next byte from the mass storage file or unit record device referred to by 'dsi', in the 8 least significant bit positions of the A register. The byte is returned with no modifications, except for data from the card reader, which is assumed to be 12 bit Hollerith code, and which consequently is converted to ASCII.

4.3.4.2 Output One Byte

/ OUTCH (12_8) /



The OUTCH system service request outputs the 8 least significant bit positions of 'byte' to the mass storage file or unit record device identified by 'dsi'. The byte will be output with no modifications.

4.3.4.3 Input One Record

/ INDAT (13_8) /

(dsi
m.a.
(n.b.
(terminator

The INDAT system service request will input a string of bytes from the mass storage file or unit record device referred to by 'dsi', packing the bytes, two in each location, into the area pointed to by 'm.a.'. The operation stops when a byte equal to 'terminator' has been found and packed, in which case the A register upon return will contain the number of bytes read. Alternatively, the operation does not stop before the number of bytes equal to 'n.b.' has been input; in this case the returned value is 0.

To achieve inputting of binary data containing no sensible terminators, a 'terminator' of value greater than 255 should be specified. Then the maximum number of bytes, 'n.b.', is always read.

4.3.4.4 Output One Record

/ DATUT (14_8) /

(dsi
m.a.
(n.b.
(0

The DATUT system request outputs a string of bytes from the main storage area specified by 'm.a.' to the mass storage file or unit record device referred to by 'dsi'. The number of bytes to be transferred is defined by 'n.b.'. The bytes are assumed to be packed two-by-two in the main storage area locations.

Note: The fourth word in the parameter list must contain the address of a zero location. This is a SINTRAN requirement. The interested reader is referred to the SINTRAN II User's Guide.

4.3.5 Error Handling

All SYSIO routines return a status code in the A register, refle the status of the request.

If the most significant bit is on ($(A_{15}) = 1$), then the call was un and an error code is contained in the 15 least significant bits ((.

Otherwise, i.e., if $(A_{15}) = 0$, the request was successfully con and the A register may contain significant information, as in th of INCH.

The various error codes and their meaning are displayed in Ap

4.4 Block Oriented Input/Output System

4.4.1 Introduction

In addition to the generalized stream oriented I/O system, SYS described in the previous section. NORD-OPS provides a basic oriented input/output system for the handling of block transfers between main memory and mass storage files. This system, l BLCIO, is basic in the sense that all requests for accesses to f have to pass through it. SYSIO, for instance, uses BLCIO for filling and emptying of the intermediate block buffers.

Besides the handling of basic mass storage file I/O, BLCIO als handles magnetic tape. Presently, magnetic tape is treated mo or less like a unit record device, each physical block regarded a separate unit. Therefore, the handling of mass storage I/O a magnetic tape I/O will not always be equivalent.

4.4.2 The Addressing Scheme of BLCIO

Each block within a mass storage file is identified by a (unique) block sequence number (starting on 0, see Section 3.1). This block sequence number is the basic addressing scheme on the f

Now, for each open mass storage file, BLCIO maintains a bloc counter reflecting current position in file. Normally, accesse are sequential, simply increasing the block counter for each ac To obtain random access to the files (and, indeed, to the tapes special BLCIO requests are therefore included manipulating t block counters (see Sections 4.4.3.8, .12, and .13).

4.4.3 The Input/Output Commands

Only one system service request is available:

/ BLCIO (2) /

(function
(dsi
m.a.
(n.w.
s.a.

In this general request, however, a 'function' identifying the type of service needed is specified. A detailed description of each available 'function' is contained in the following subsections; note that 'dsi' always identifies the mass storage file (or magnetic tape unit, see Section 3.2.4) referred to.

4.4.3.1 Read a Block

If 'function' is 0, BLCIO will input 'n.w.' words to the main storage area pointed to by 'm.a.', starting with the block whose block sequence number is contained in the block counter. (This will always be the next block in sequence for magnetic tape, see Sections 4.4.3.8, .12, and .13).

If a mass storage file is referred, and the number of words to be input is greater than the block size of the file, then inputting is continued from the immediately following blocks until enough words are read.

If a magnetic tape unit is referred, however, then the input operation is never continued past the first physical block.

When the input operation is completed, the block counter is increased to contain the block sequence number of the first block not affected by the request.

4.4.3.2 Write a Block

If 'function' is 1, BLCIO will output 'n.w.' words from the main storage area pointed to by 'm.a.', starting with the block whose block sequence number is contained in the block counter.

If a mass storage file is referenced, and the number of words to be output is greater than the block size of the file, then outputting is continued to the immediately following blocks until enough words are written. Also, if the last block output is only partially filled with information, due to the 'n.w.' not being a multiple of the block size, then that block is padded with zeroes before it is written.

If magnetic tape is referenced, however, then the 'n.w.' words are output as exactly one physical block.

The block counter is updated as for input operations.

4.4.3.3 Read Test a Block

If 'function' is 2, BLCIO will check the 'n.w.' first words, starting with the block whose sequence number is contained in the block counter, for correct parity. Note that the 'm.a.' is dummy with this function.

If 'n.w.' is not a multiple of the block size of the referenced file, then the last block affected will all the same be checked in its entirety.

The block counter is updated as for input and output operations.

The read test function does not apply to magnetic tape.

4.4.3.4 Compare a Block

If 'function' is 3, BLCIO will compare 'n.w.' words from the main storage area pointed to by 'm.a.', with the contents of the referenced file, starting with the block whose sequence number is contained in the block counter.

If 'n.w.' is not a multiple of the block size of the file, then the last block is padded with zeroes before the comparison.

The block counter is updated as usual.

The compare function does not apply to magnetic tape.

4.4.3.5 Advance to EOF

The 'function' 10_8 applies only to magnetic tape. It causes the referenced tape unit to be advanced past the first encountered end-of-file mark. The 'm.a.' and 'n.w.' are dummy arguments.

4.4.3.6 Reverse to EOF

As for the previous function, the 'function' 11_8 also applies only to magnetic tape. It causes the referenced tape unit to be reversed past the first encountered end-of-file mark (rewind of current magnetic tape file). The 'm.a.' and 'n.w.' are dummy arguments.

4.4.3.7 Write EOF

The 'function' 12_8 also applies only for magnetic tape. It causes BLCIO to write a file mark on the tape unit referenced. The 'm.a.' and 'n.w.' parameters are ignored.

4.4.3.8 Rewind

If 'function' is 13_8, then the referenced mass storage file or magnetic tape unit is rewinded. For a mass storage file this simply means that its block counter is reset to 0.

The 'm.a.' and 'n.w.' parameters are ignored.

4.4.3.9 Erase

This 'function', 14_8, applies only to magnetic tape. It causes approximately 4 inches of tape to be erased (skipping bad spot) on the selected tape unit. The 'm.a.' and 'n.w.' parameters are ignored.

4.4.3.10 Backspace

This 'function', 15_8, causes the selected mass storage file or magnetic tape unit to backspace one block. For a mass storage file this simply means that its block counter is decremented by 1.

The 'm.a.' and 'n.w.' parameters are ignored.

4.4.3.11 Set Parity

This 'function', 16_8, valid only for magnetic tape, causes the parity of the referenced tape unit to be set according to the contents of the 'n.w.' parameter. A 0 in this parameter selects odd parity, and a 1 selects even parity; all other codes are invalid.

The 'm.a.' parameter is ignored.

4.4.3.12 Read Status

If 'function' is 20_8, then BLCIO will simply read the status register of the magnetic tape controller (if a magnetic tape unit is referenced), or that of the relevant mass storage controller (if a mass storage file is referenced). The status is returned as the second word of the double word pointed to by 's.a.'.

The status returned may only be regarded as OK if no error is reported (in which case the status will relate to the error - see however, Section 4.4.4 for details on error handling).

The 'm.a.' and 'n.w.' parameters are ignored.

4.4.3.13 Set Block Counter

This 'function', 21_8, is treated differently for mass storage files and magnetic tape units.

If a mass storage file is referenced, then the contents of the 'n.w.' parameter are transferred to the block counter for that file, thus breaking the normal sequence of block accesses.

For the magnetic tape units, however, no block counters are maintained; so if a tape unit is referenced, the selected unit is rewinded and then the number of blocks specified in the 'n.w.' parameter is passed over.

The 'm.a.' parameter is dummy with this function.

4.4.3.14 Increment Block Counter

This 'function', 22_8, closely resembles the Set Block Counter Function.

If a mass storage file is referenced, then the contents of the 'n.w.' parameter is added to the block counter for that file. The 'n.w.' may be both positive and negative.

If a magnetic tape unit is referenced, then the number of blocks specified in the 'n.w.' parameter is passed over, either in the normal direction (if 'n.w.' is positive) or by backspacing (if 'n.w.' is negative).

The 'm.a.' parameter is dummy with this function.

4.4.3.15 Read Block Counter

If 'function' is 23_8, then BLCIO will return the block counter for the referenced file to the second word of the double word pointed to by 's.a.'. The 'm.a.' parameter is ignored.

The Read Block Counter function does not apply for magnetic tape units.

4.4.3.16 Get Block Size

If 'function' is 24_8, BLCIO will return the block size of the referenced file to the second word of the double word pointed to by 's.a.'. The 'm.a.' parameter is ignored.

The Get Block Size function does not apply for magnetic tape units.

4.4.3.17 Immediate Return

In the multicomputer version of NORD-OPS, all BLCIO functions may be executed with immediate return. This means that the BLCIO request only initiates the transfer, and returns to the user program at once. Transfer completion may be ensured by a read status call (see Section 4.4.3.12).

The 'function' for the 'immediate return transfer' is found by adding 100_8 (64_{10}) to the normal transfer function.

Example:

'Function' 101 means 'write a block with immediate return'.

Note:

The immediate return facility only exists in the task CPU's in the multicomputer configuration.

Only one BLCIO transfer can be initiated at a time. The next BLCIO request will always serve as a 'wait for completion' call for the 'immediate return' request.

4.4.3.18 Summary of Functions

The following table contains all functions available with BLCIO and indicates briefly their applicability.

Function (in octal)	Specification	Applicable for	
		Mass Storage	Mag. tape
0	Read a Block	x	x
1	Write a Block	x	x
2	Read Test a Block	x	
3	Compare a Block	x	
10	Advance to EOF		x
11	Reverse to EOF		x
12	Write EOF		x
13	Rewind	x	x
14	Erase		x
15	Backspace	x	x
16	Set Parity		x
20	Read Status	x	x
21	Set Block Counter	x	x
22	Increment Block Counter	x	x
23	Read Block Counter	x	
24	Get Block Size	x	

4.4.4 Error Handling

All BLCIO functions return a status in the double word pointed to by the 's.a.' parameter. For all requests completing in success, both these words are zero, the exception being the read status function where the second word contains the status read! Requests terminating in error will, however, turn the most significant bit of the first word on, thus indicating that an error occurred.

The interpretation of the 31 least significant bits of the status double word, in case of an error, is contained in Appendix A.

4.5 Miscellaneous Routines

4.5.1 Get current Time

/ TIME (16) /

m.a.

The TIME routine may be used to get the internal time in basic time units (usually 20 ms). The internal time is returned as a double word in the locations pointed to by 'm.a.'.

4.5.2 Get Clock and Date

/ CLOCK (17) /

m.a.

The CLOCK routine may be used to get the current clock and date. The clock and date are returned as seven words to the area pointed to by 'm.a.'.

The seven words are:

Word 0 : Basic time unit
" 1 : Second
" 2 : Minute
" 3 : Hour
" 4 : Day
" 5 : Month
" 6 : Year

5 BRF LOADER

5.1 Introduction

The BRF loader is the processor in NORD-OPS that prepares programs for execution. The loader collects and resolves references between relocatable program units so as to produce absolute programs ready for execution. The possibility of using library files for keeping relocatable copies of commonly used subroutines is included.

5.2 Binary Relocatable Format

Object output from the FORTRAN compiler and the MAC II assembler provided with NORD-OPS (see Chapter 6) is in a format called Binary Relocatable Format or BRF.

We will not go into a detailed discussion of the concepts of relocatable versus absolute object code. This discussion is found in MAC Users' Guide, and the interested reader is referred to that publication. The basic concepts of BRF will, however, be reviewed.

5.2.1 Relocability

Programs written in BRF object code may be relocatable in the sense that they may not be predestined to lie in specific areas of main storage. Instead, sufficient control information is included so that the BRF loader may place them wherever it finds it sensible.

5.2.2 Linkability

The concept of linking allows program units assembled or compiled at separate times to be loaded and linked together into a compact (absolute) program. This is achieved by permitting references to symbols in other program units (external references) to occur, and by including control information on such symbols and their use in the object (BRF) code.

5.2.3 Library Routines

The BRF object code includes the concept of conditional linking of library routines. This is achieved by identifying the library routines as such, and then, at load time, only include those referenced, i.e., used.

5.3 Executable Format

A program ready for execution under NORD-OPS (i.e., a program that may be referenced in an internal command, see Section 2.2) must be kept in a mass storage file, conforming to a format called Executable (XQT) Format.

The XQT format has the following characteristics:

1) The first block in the file contains control information used by the establisher (which is the routine in NORD-OPS that loads and establishes tasks). This information includes:

   - NAME of program; assigned by the BRF loader upon user demand (see Section 5.4. below).
   - LOWER and UPPER LIMITs of the area in which the program must execute; this area must be contained within the "user area" in NORD-OPS. Normally, the lower limit coincides with the lower limit of the "user area".
   - START ADDRESS; address of primary entry point (see Section 4.2.2.1).
   - Some STATUS FLAGS for system use, e.g., indicating whether the program should be run in user (i.e., restricted) mode or in system (i.e., unrestricted) mode.

2) The subsequent blocks contain an (absolute) core image of the program.

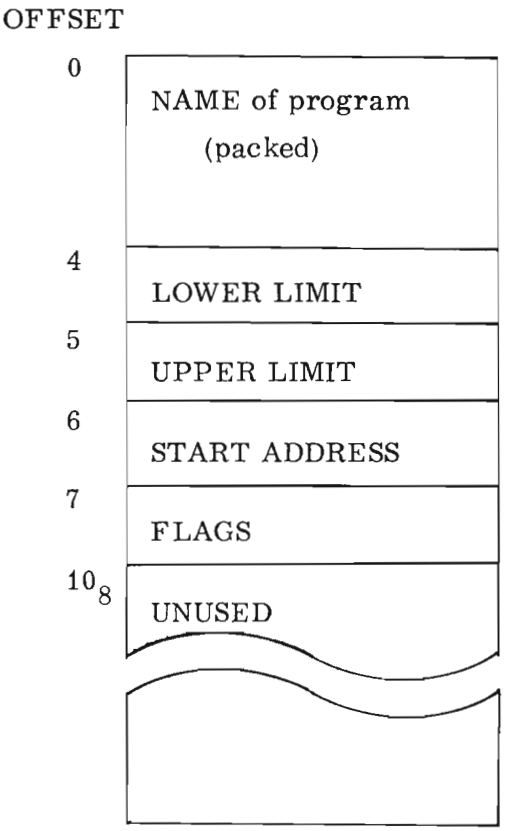


Figure: Format of first block in an XQT file.

Output from the BRF loader in NORD-OPS is in this format.

5.4 The LDR Control Command

The BRF loader is invoked by the following internal control command:

$LDR(name, odsi, ldsi, idsi_1, idsi_2, \ldots)$

Here, the parameters are interpreted as follows:

- name — is the NAME assigned to the program (see Section 5.3 above); it is alphanumeric, consisting of up to eight characters, of which at least one has to be a letter;
- odsi — is the data set identifier of the output file (in XQT format);

- ldsi — is the data set identifier of the file (or unit record device) to receive the load map, a symbolic map of the loaded program; default value for ldsi is 5 (the line printer);

- $idsi_1$, $idsi_2$,.. — is the data set identifiers of the input files (in BRF format). If no idsi's are specified, then the permanently open scratch file with dsi = 80 and the standard library file with dsi = 79 are considered the only input file.

If the 'name' parameter is omitted (together with its delimiting comma), then the 'odsi' should also be omitted (together with its delimiting comma), and the first parameter is assumed to be 'ldsi' In this case the program is loaded directly into memory, and, in due time, activated through its primary entry point.

Now, the first input file (i.e., the file identified by '$idsi_1$') is loaded. If unresolved external references still exist when the complete fiel is loaded, the second input file (i.e., the file referred to by '$idsi_2$'), if any, is loaded. This process is continued until one out of two alternatives occurs:

- Either all external references are resolved, when the loading of an input file is completed. Then the loading is terminated, although further input files may have been specified.

- Or, despite all input files have been loaded, some unresolved external references remain. Then the permanently open standard library file (dsi = 79) is loaded*. If still any unresolved external references are left, the job is aborted. Otherwise, the loading is terminated normally.

Note that, during the above procedure, a library routine is loaded only to the extent it is referenced (i.e., called) by some routine already loaded at the time the loader encounters the library routine. Thus, one has to be careful about the order in which the routines occur in the input sequence. If, for instance, routine A calls library routine B, then A must occur before B; otherwise B will not be loaded, and an unresolvable external reference will occur.

Examples of LDR cards:

```
$LDR(1,2)
```

A program is loaded from the paper tape reader; diagnostics and load map occur on the on-line Teletype. The program is activated.

---

* In the present version of the BRF loader, all input files must be specified explicitly, including the permanently open scratch file and the standard library. A version will, however, soon be available.

$LDR(MYPROG,40,5,41)

A program, given the name MYPROG, is loaded from file 41. Diagnostics and load map go to the line printer. Output is to file 40; and thus the program is not activated. The program may be activated by the internal control command $MYPROG,40.

$LDR

A program is loaded from the standard scratch file. Diagnostics and load map appear on the line printer. The program is started after having been loaded.

For further information on BRF and the BRL loader, the Binary Relocating Loader manual should be consulted.

6 AVAILABLE PROCESSORS

6.1 Introduction

In this chapter some processors available with NORD-OPS will be treated. None of them will be treated in detail; for detailed discussions the user is referenced to the appropriate manuals. Rather, their user profiles, relating to NORD-OPS, will be outlined.

6.2 MAC II Assembler

6.2.1 Invoking the Assembler

The assembler available with NORD-OPS is the standard MAC II assembler described in the MAC Users' Guide (June 1971). It will be called through the control command:

$MAC(idsi, ldsi, odsi)

where

- idsi is the data set identifier of the source stream;
- ldsi is the dsi of the list stream;
- odsi is the dsi of the object stream.

Default values are assigned to all of these parameters:

source input : standard input device

source listing : standard output device

object output : the permanently open scratch file (dsi = 80)

If no listing of the assembled program is wanted (by specifying ldsi = 0), MAC nevertheless prints error messages on the standard output device.

6.2.2 Peculiarities of the NORD-OPS Version

6.2.2.1 The $ Command

During assembly, the mass storage files or unit record devices assigned to the input/output streams of MAC may be manipulated by the $ command. However, use of this command under NORD-OPS differs slightly from the description given in the MAC Users' Guide.

First, when a file or device assignment to a spesific stream is changed, MAC normally will execute an RBUF request specifying the old data set identifier assigned to that stream. This will have the affect of rewinding the file if it is accessed again.

Second, the)LINE (or, equivalently, the @) command and the $ command without arguments are treated differently. In standard MAC,)LINE is equivalent to 1,0$. Under NORD-OPS, however,)LINE will switch source stream to standard input and list stream to the dummy device, all right. But no RBUF's will be executed; instead the old source and list stream specifications will be saved, and later, if a $ alone is encountered, then are reinstated, thus resuming the previous input/output files. Note that a $ without parameters from an input file or device other than standard input, will have no effect whatsoever!

A few examples should straighten out things:

Example 1:

> Assume source stream to be associated with standard input, currently being Teletype. Then 45$ is typed. Eventually, a)LINE command is found in file 45, switching back to Teletype. After having performed some preliminary tasks, you type $. The inputting will then be resumed from file 45, from the statement following the)LINE command.

Example 2:

> Now, assume that you type 45$ again instead of $ alone, in the previous example. Then MAC will execute an RBUF specifying the old input file (which is 45), and the input stream is then associated with the new input file (which, however, also is file 45). The effect is that file 45 is re-read from the beginning! Note that this would not have happened with input from a unit record device, e.g., paper tape reader. In that case the two examples would have yielded exactly the same result!

6.2.2.2 The)9EOF Command

The MAC assembler is terminated when an)9EOF command is found. Before returning to the Job Supervisor, MAC will write an end-of-file control byte to the object stream; then RBUF is called for the dsi's associated with all three input/output streams.

6.2.2.3 MAC as a Debugging Aid

It should be pointed out that the MAC assembler available with NORD-OPS is indeed capable of assembling non BRF code directly into main memory, subject, of course, to the memory protection restrictions. Thus, even under NORD-OPS, MAC may be utilized as an "Interactive Assembly and Debugging System".

6.2.3 File MAC

A special version of MAC may be called through the command:

$MACF(idsi,ldsi,odsi,filedsi)

This version of MAC may be used to correct programs in executable format on file 'filedsi'. The control block (Section 5.3) may be examined and changed as address 0-7.

6.3 FORTRAN Compiler

6.3.1 Invoking the Compiler

The FORTRAN compiler available with NORD-OPS is the compiler described in the NORD FORTRAN IV Reference Manual (January 1973). The compiler is invoked with the control command:

$FTN(idsi, ldsi, odsi, ladr)

where

- idsi — is the data set identifier of the input device (or file);
- ldsi — is the dsi of the list device (or file);
- odsi — is the dsi of the object file (or device);
- ladr — is an octal address (optional). If this parameter is specified, each statement in the listing (if any) of the compiled program will be preceded by its corresponding memory address, normalized by 'ladr'.

Either all or none of the data set identifiers must be specified. If omitted, they are assigned default values:

source input : card reader (dsi = 4)

source listing : line printer (dsi = 5)

object output : the permanently open scratch file (dsi = 80).

If no listing of the program is wanted (by specifying ldsi = 0), the FORTRAN compiler nevertheless prints the error messages on the line printer.

6.3.2 Peculiarities of the NORD-OPS Version

6.3.2.1 Device Unit Numbers

The device unit numbers, used in FORTRAN I/O statements are, under NORD-OPS, precisely the data set identifiers defined in Section 3.2.4. Thus, e.g., device number 5 represents the line printer.

6.3.2.2 The PAUSE and STOP Statements

The STOP statement is, under NORD-OPS, implemented as JBSUP system service requests, thus terminating the executing FORTRAN program. A program executing a PAUSE statement may restart in the statement following the PAUSE statement. Also, the numbers (if any) specified in the PAUSE and STOP statements will appear on the line printer.

6.3.2.3 The EOF Statement

The compiler terminates its execution when an EOF statement is found; thus the source deck must always be terminated with such a statement.

6.3.3 Block Oriented Input/Output

The input/output statements of NORD FORTRAN is a superset of the I/O statements of standard FORTRAN IV specifications. Nevertheless, they include only provisions for sequential I/O.

To make random oriented mass storage input/output available to FORTRAN programmers, the NORD-OPS BLCIO routine may be called from a FORTRAN program

BLCIO is called by a FORTRAN CALL statement

CALL BLCIO (<function>, <device unit no.> , < array element>, <no. of words >, <status>)

where the parameters exactly correspond to the parameters of the BLCIO system service request (see Section 4.4.3). Therefore, a detailed description of the parameters will not be given. Suffice it to say that <function>, <device unit no.> and <no. of words> must be integers or integer variables, < array element > is the first element of the array identifying the main storage area affected by I/O transfer requests, and <status> is a two item integer array, receiving the status of the call (see Section 4.4.4).

6.4 QED Editor

The conversational editor provided with NORD-OPS is the so-called QED editor as described in the QED manual.

Invoking the QED editor is very simple; it is performed by the control command:

$QED

When ready, it will type an asterisk ('*') on the standard output device and expect commands to arrive from the standard input device.

For further information, the QED manual should be consulted. Here it is sufficient to mention that default device assignments are:

- input from paper tape reader (dsi = 2);
- output to paper tape punch (dsi = 3);
- command input/output from standard input/output.

These assignments may be changed by a special command (MODE) to the editor.

6.5 The NORD-5 FORTRAN System

6.5.1 The FORTRAN Compiler

The compiler is invoked with the control command

$N5FTN(idsi, ldsi, odsi, ladr)

where the parameters have the same meaning as for the NORD-1 FORTRAN compiler (Section 6.3.1).

Restrictions in the language compared with the NORD-1 version:

- Complex and Hollerith types are not implemented.
- For each common block, only one item may be mentioned in the EQUIVALENCE statements.

6.5.2 The FORTRAN Run-time System

6.5.2.1 PAUSE and STOP Statements

PAUSE or STOP will be printed, followed by the number in the statement. For STOP, the program will be terminated. For PAUSE, the program will continue.

6.5.2.2 Run-time Errors

A run-time error will be of the form:

RUN 5 ERR <error number> <octal address>

The following error numbers may occur:

1: Computed GOTO with index = 0

2: I/O error in REWIND, BACKSPACE or ENDFILE statements

3: Error in READ/WRITE statements (system error)

4: More than 4 levels of implied DO-loops in READ/WRITE statements.

The error numbers from the FORTRAN I/O formatting routines correspond with the numbers of the NORD-1 version.

The FORTRAN library has the following error numbers:

100 Error in AXA.

AXA. was called with negative only.
Result set to zero.

Overflow in AXA.
Result set to 1.0E99.

101 Error in AXI.

Base equal to zero and exponent negative.
Result set to 1.0E99.

102 Error in ATAN2

Both arguments equal to 0.0.
Result set to 0.0.

103 Error in COSH

Argument greater than 2^{14}.
Result set to 1.0E99.

104 Error in COS

Argument greater than 2^{16}.
Result set to 0.0.

105 Error in EXP

Argument greater than 2^{16} ln2.
Result set to 1.0E99.

106 Error in IXI

Overflow in result.
Result set to 32767.

107 Error in ALOG, ALOG1, ALOG2

Argument less or equal to 0.
Result set to -1.0E99.

108 Error in SINH

Argument greater than 2^{14}.
Result set to SIGN(X) * 1.0E99.

109 Error in SIN

Argument greater than 2^{16}.
Result set to zero.

110 Error in SQRT

Argument less than zero.
Result set to zero.

6.5.3 Call of Assembly-coded Subroutines from FORTRAN Programs

An assembly-coded subroutine may be called in the same way as FORTRAN subroutines and functions.

Example of call:

```
DIMENSION IR(5), AR(7)
.
.
.
CALL ASSUB(I, R, IR, AR)
.
.
```

The call will be compiled equivalent to:

```
EXT ASSUB
RTJ BPAR,*+1,,,1
GCN ASSUB                    SUBROUTINE ENTRYPOINT
ACN I, 0                     PAR 1
ACN R, 0                     PAR 2
ACN IR, INDEX                PAR 3
ACN AR, INDEX, INDEX         PAR 4
                             RETURN HERE
```

The address to the array will point to "element 0", one step ahead of the first array element. Notice the double indexing of real arrays.

In the subroutine the actual parameters can be accessed by indirect addressing.

The example:

```
LDR AC2,1,BPAR,,1
```

will get the first parameter.

For accessing the array element AR(5) the following will do:

```
SETA INDEX,5
LDF AC2,4,BPAR,,1
```

On return from the subroutine, the registers 1 - 5 must have the same values as on entry. In addition, the register BPAR should be loaded from the calling program. The exit sequence may look like this:

```
RAD   LINK,BPAR,0      SAVE RETURN ADDRESS
LDR   BPAR,2,BVAR      GET VALUE FROM CALLING PROGRAM
RTJ   0,5,LINK         RETURN TO AFTER LAST PARAMETER
                       ADDRESS
```

Any function value is supposed to be in the register ACC. The registers mentioned have the following numbers:

```
BVAR  EQU  5           BASE FOR VARIABLES
BPAR  EQU  6           BASE FOR PARAMETERS
LINK  EQU  8           RETURN
ACC   EQU 12           MAIN ACCUMULATOR
INDEX EQU 14           ARRAY INDEX
```

6.5.4 NORD-OPS Services in the NORD-5

In the NORD-5, most of the system service routines may be called.

The call sequence resembles that of SINTRAN.

```
      SETA ADR,BPARA          POINTS TO PARAMETER LIST
      RTJ  LINK, service routine
      .                       RETURN
      .
      .
BPARA ACN  PAR1               ADDRESS TO THE ACTUAL
                              PARAMETERS
      ACN  PAR2
      .
      .
      .
ADR   EQU 11                  REGISTER USED
ACC   EQU 12                  MAIN ACCUMULATOR
```

A possible function value will be returned in register ACC.

Functions included:

BLCIO.
INDAT.
DATUT.
INCH.
OUTCH.
EXIT. (JBSUP)
ABORT. (The abort code is in register ACC)
COMPR.
ESTAB.

6.5.5 The NORD-5 Linking Loader

The loader accepts programs in a binary relocatable format, BRF, conceptually comparable to that of NORD-1. The result will be put in a file in XQT-format (executable format).

Control command:

$N5LDR(name, bindsi, listdsi, loader commands)

or

$N5LDR(listdsi, loader commands)

In the first case the file 'bindsi' will be used for the XQT-format, with the name 'name', ready for execution. In the second case the program will be loaded onto a scratch file and it will be executed at once. The dsi 'listdsi' is for list output if so specified in the loader commands.

Loader commands:

Through the 'loader commands' the loading can be controlled in detail. A command consists of a letter, followed by 0, 1 or 2 octal or symbolic parameters. If a decimal number occurs with no letter ahead of it, it will be interpreted as a new BRF input dsi. The old dsi will then be rewound.

The loader commands:

- A - Automatic load, until EOF byte in the BRF format.
- C - Change name. The name of a defined or undefined symbol will be changed.
  Parameter 1: Old name.
  Parameter 2: New name.
- D - Define symbol.
  Parameter 1: Symbol name.
  Parameter 2: Symbol value.

K - Kill symbol.
Parameter 1: Name of defined symbol to be killed.

L - Set list mode.
Disassembled contents will be printed.

M - Manual load, until an END byte in the BRF format.

N - Non list mode, supress listing.

O - Set origin.
Parameter 1: New value of program base and current loading address.

P - Print memory, disassembled.
Parameter 1: Lower limit.
Parameter 2: Upper limit.

R - Reset loader.

U - Print undefined symbols and free area.

W - Print defined symbols and free area.

Library File:

The FORTRAN library will be loaded if the library file (79) is specified as BRF input and an A-command is found. The library file consists of two parts: The run-time system and the FORTRAN library. When the loader is started, it will immediately load the run-time system.

6.6 The NORD-5 Assembler

The NORD-5 assembler accepts program in symbolic format and converts it to BRF code. The assembly needs two passes. During the first pass the source program is copied over to a scratch file while processing. The second pass takes its input from the scratch file giving list output and BRF output.

Control command:

$N5ASM(nolist, error list, bin out, idsi, ldsi, odsi, sdsi)

nolist = 1 : no listing

error list = 1 : listing of errors only

bin out = 1 : binary output

idsi : input dsi for pass 1

ldsi : listing output dsi

odsi : BRF output dsi

sdsi : scratch file for intermediate storage

See the manual: Assembler for NORD-5.

7 ACCOUNT SYSTEM

7.1 Account File

The buffered mode version of NORD-OPS maintains an account file. The account file contains information of all jobs run under NORD-OPS. The account file is updated each time a job is terminated. The information of one job in the account file is called an account record. Each 256 words block of the file contains 16 account records. The account records are chronologically ordered from block 1 of the account file. Block zero contains control information:

Word 0 - Number of account records in the account file.

Word 1 - Maximum number of account records allowed in the account file. Error message is given on the operator's console when this number is exceeded, and no more account information is saved.

Word 2 - Number of account records allowed in the account file before a warning message on the console is given. Account information is still saved. This number should be a little less than the maximum number. The message is intended as a warning that the number of account records in the file is approaching the maximum number.

Word 3 - 255 - Miscellaneous.

7.2 Account Record

The account record contains 16 words with information of one job. Each job terminated will create one account record at the end of the account file, if the maximum number of records is not exceeded. The format of the account record is:

Word 0 - 3 - Account number specified in the $JOB command (see Section 2.1.1) (8 ASCII characters).

Word 4 - 7 - Name specified in the $JOB command (8 ASCII characters).

Word 8 - 10 - Date, time on and time off, packed as 48 bits field in the following way:

Bits 47 - 43 - Year
Bits 42 - 39 - Month
Bits 38 - 34 - Day
} Date (on)

Bits 33 - 29	- Hours	Time on
Bits 28 - 23	- Minutes	
Bits 22 - 17	- Seconds	
Bits 16 - 12	- Hours	Time off
Bits 11 - 6	- Minutes	
Bits 5 - 0	- Seconds	

Word 11 - Seconds used.

Word 12 - Number of standard input lines.

Word 13 - Number of standard output lines.

Words 14-15 - For future extensions.

7.3 The Account Utility Handler

The account file can be handled by a special service program called the Account Utility Handler. The account file can be listed or dumped, and the control words can be changed.

Control command:

$ACC(dump,list,reset,desired,max)

where

dump - dsi of an open file to receive the dump. This can be any kind of file. The contents of the account file will be moved to the dump file with no editing or any other change.

If 'dump' 0 or the dump field is left empty, there will be no dump.

list - dsi of unit record device to receive an edited list of the account file. This device should normally be the line printer.

If 'list' 0 or list is omitted, no list is made.

reset - reset flag - if 'reset' 0, then the account file will be restarted after the dump is made. It is also possible to reset the file without making dump or list.

The account file is not reset if 'reset' is omitted or zero.

desired - Desired number of accounts. By use of this parameter the user specifies the number of accounts he wants in the account file before he gets the warning that the account file is running full. 'Desired' should be given as a decimal number.

max - Maximum number of accounts permitted on the file. When this limit is reached, no more accounting will take place, until the account file is reset. The jobs will still be terminated, but the request for accounting will be ignored. 'Max' should be given as a decimal number.

8 OPERATOR COMMUNICATION

The operator communication in NORD-OPS is an extension of the SINTRAN II operator communication. One or more two-way devices (Teletypes or displays) may serve as operator consoles. A complete set or a subset of the SINTRAN II operator commands is always available to the NORD-OPS operator. In addition, there is a set of special NORD-OPS commands which are documented in the following sections.

8.1 Messages to the Operator

During job execution three types of messages may appear on the operator's console:

- error messages
- system logs and warnings
- commands

8.1.1 Error Messages

An error message from the system will appear on the console in the following format:

hh.mm.ss RUN ERR nn xx yy

where

hh.mm.ss	-	error time in hours, minutes and seconds
nn	-	error number
xx,yy	-	additional error parameters

All messages of this kind are documented in the SINTRAN II User's Guide, and are not repeated here.

8.1.2 System Logs and Warnings

A system log or warning will appear on the console in the following format:

hh.mm.ss <job name> <message>

where

hh.mm.ss	-	current time
<job name>	-	name specified on the job command of the message originator
<message>	-	message

The message may be one of the messages listed below, a message given by a user in a $MSG command (see Section 2.2.1), or a system error message.

8.1.2.1 Begin Message

Each time a new job execution begins, the following message is printed:

hh.mm.ss <job name> B

This message only appears in the buffered mode version of NORD-OPS.

8.1.2.2 Terminate Message

Each time a job is terminated, the following message is printed:

hh.mm.ss <job name> T

8.1.3 Commands

A command to the operator is given in the following format:

hh.mm.ss <job name> <message>
:

where

hh.mm.ss - current time

<job name> - name of command originator

<message> - command to the operator

The operator is supposed to act according to the message, and then answers with one single character after the ':'. Three characters are recognized by the system:

G - Go, continue job.

A - Abort, terminate current job.

W - Wait, stop execution of current job. The job may be restarted by a @ GO command (see Section 8.2.3).

The command to the operator is either a user command given in a $MSG command (see Section 2.2.1) or a system command, for instance from the file system.

8.2 Messages from the Operator

The operator may give a message to the system at any time by means of the SINTRAN II Operator Communication. The operator will get access to the system by typing a '@'. The system will respond with an '*' and the operator may then type the message as a character string terminated by a carriage return. The messages may be any of the command specified in the SINTRAN II User's Guide, or one of the four NORD-OPS commands listed below:

8.2.1 Start NORD-OPS

NORD-OPS must be initialized or restarted after a $EXIT command with the following command:

@NOOPS.

When ready, the system will write a ready message.

8.2.2 Delete Job

A job execution may be interrupted, and further execution of that job prohibited by the following command:

@DELET

The system will respond with a ':' on the next line. After the ':' the job name terminated by a carriage return, must be specified.

A SYSTEM ERROR 145 will appear on the job output.

8.2.3 Restart Job

A job may be brought out of a waiting state (see Section 8.1.3) by the following command:

@GO

The job name must be specified as in the previous command.

8.2.4 Call the File System

The file system may be reached by the following command:

@FILE

The system will respond with a ':' on the next line. After the ':' a file command (see Chapter 3) must be specified. The command must begin with a '(' and terminate with a ')' and a carriage return.

Example:

```
@FILE
:(EN,DK,10,1)
```

Enter disc pack 10 on spindle 1.

Error messages from the file system will appear in the following format:

```
FILE ERROR xxx
```

where xxx is the error number (see Appendix A).

8.3 Fatal Errors

Under special conditions, the following message may appear on the console:

```
FATAL NORD-OPS ERROR!
JOB TABLE:
<job table dump>
:
```

The operator is supposed to answer with one of three characters:

- S - Stop the system.
- C - Clear current job and start with next.
- O - Transfer current job to the output queue (only used in connection with buffered mode).

If a fatal error occurs, qualified personnel should be contacted.

APPENDIX A

ERROR CODES AND MESSAGES

A.1 Input/Output Error Codes

A.1.1 The Status Words of BLCIO

The status words returned by BLCIO may be interpreted as follows:

Bit No.	Mnemonic	Interpretation
15	ER	Error flag; if set check rest of status words for error specifications.
14	BF	Busy flag; if set processing of this command is still active.
13	LE	Logical error flag; set if error is in BLCIO request or if EOF or EOT is set.
12	PE	Physical error flag; if set error is of physical nature like parity error etc.
11	CE	Channel error flag; set only in conjunction with PE, indicating a channel error.
10	-	Unused bit.
9	EOF	End of file; on mass storage, an attempt was made to read a non-existent block, on mag. tape, a file mark was encountered.
8	EOT	End of tape/allocated file; on mass storage, an attempt was made to read or write* beyond allocated part of file; on mag. tape, a tape mark was encountered.
7	EOD	End of device; an attempt was made to read or write beyond physical end of device (mass storage only).
6-0	EC	Error code; code describing logical errors; set only in conjunction with LE! A.1.2 contains the possible error codes and their meaning.
The second word is set only in conjunction with PE (see, however, Section 4.4.3.12); if set it contains the hardware status as received from the relevant controller.		

* Error only when automatic expansion is suppressed (see Section 3.2.5.6).

A.1.2 Error Codes

This table contains the error codes as received in bits 14-0 of the A register from SYSIO, or in bits 6-0 of the first status word from BLCIO.

Error code in octal	Interpretation
1	No more tracks available
2	Too many users
3	User already exists
4	No such user
5	No more room in UIB
6	File already exists
7	Illegal object type
10	No such file
11	Object index out of range
12	No more space in OBT
13	No more OFT entries
14	No access
15	Already open for write
16	Bad file number
17	No such track
20	Bad track number
21	Track already exists
22	No such page
23	Fatal error
24	Transfer error
25	Bad device type
40	'Dsi' not defined
41	No more buffer space
42*	Output after input without an intervening CBUF or RBUF (in FORTRAN: REWIND)

CONT.

CONT.

Error code in octal	Interpretation
43*	Input after output without an intervening CBUF or RBUF (REWIND)
44*	End of written part of file
45	Attempt to access restricted file
46	Illegal BLCIO function
47	Internal error

The codes marked with an asterisk may be received from SYSIO only.

A.2 Abort Messages

If a task (and thus, normally, a job) is aborted by a system routine, the job supervisor will print a message of the type

SYSTEM ERROR 'number'

on the standard output device (see Section 4.2.2.3). The 'system error numbers' may be interpreted as displayed in the following table.

System error number in octal	Originator and Interpretation
1-57	An I/O error occurred in a system routine; 'number' corresponds to one of the error codes in table A.1.2.
60-137	NORD FILE SYSTEM errors
60	Missing parameter
61	Parameter too long
62	Illegal parameter type
63	Illegal function parameter
64	Device type not in system
65	Device No. not in system
66	Illegal allocate mode
67	Illegal expand mode
70	Illegal password

CONT.

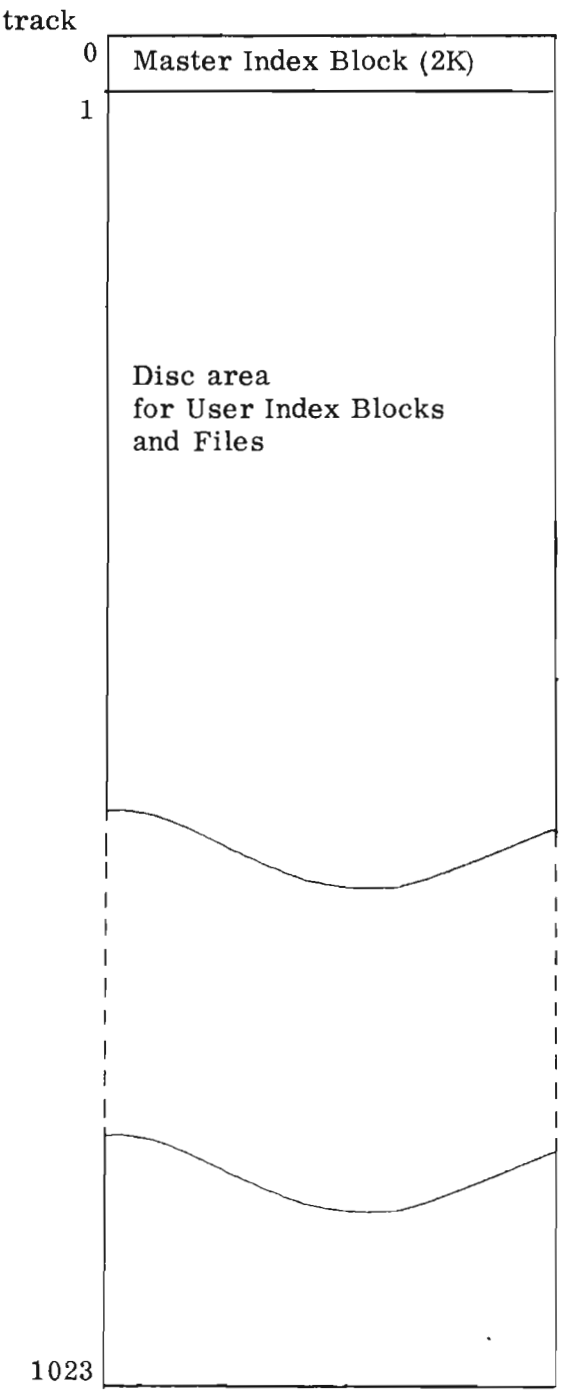
CONT.

System error number in octal	Originator and Interpretation
71	Illegal release code
72	Illegal block specifications
73	Dsi already used
74	Try to reequip unit record device
75	Illegal usage mode
77	File previously closed, save impossible
100	Illegal call to enter device
101	Device type not recognized
102	Illegal destination code
103	Illegal device No. or destination
104	File(s) opened for specified device
105	Device already entered
106	Attempt to enter wrong device
107	Illegal parity
110	Wrong density
111	Attempt to close STIP/STOP or scratch file
112	Illegal scratch file device
113	Too big blocksize
114	Close previously performed on scratch file
115	File is closed
116	Wrong device number in COPY
117	Wrong volume on specified unit
140-157	CONTROL PROGRAM errors
140	Syntax error in control command
141	Error in control command parameters
142	Wrong task name on internal control command
143	Memory protect violation
144	Illegal system service request code
145	Maximum time exceeded, or job aborted by operator
146	Maximum lines exceeded
147	ESTAB error
150	Too many nested ESTAB calls

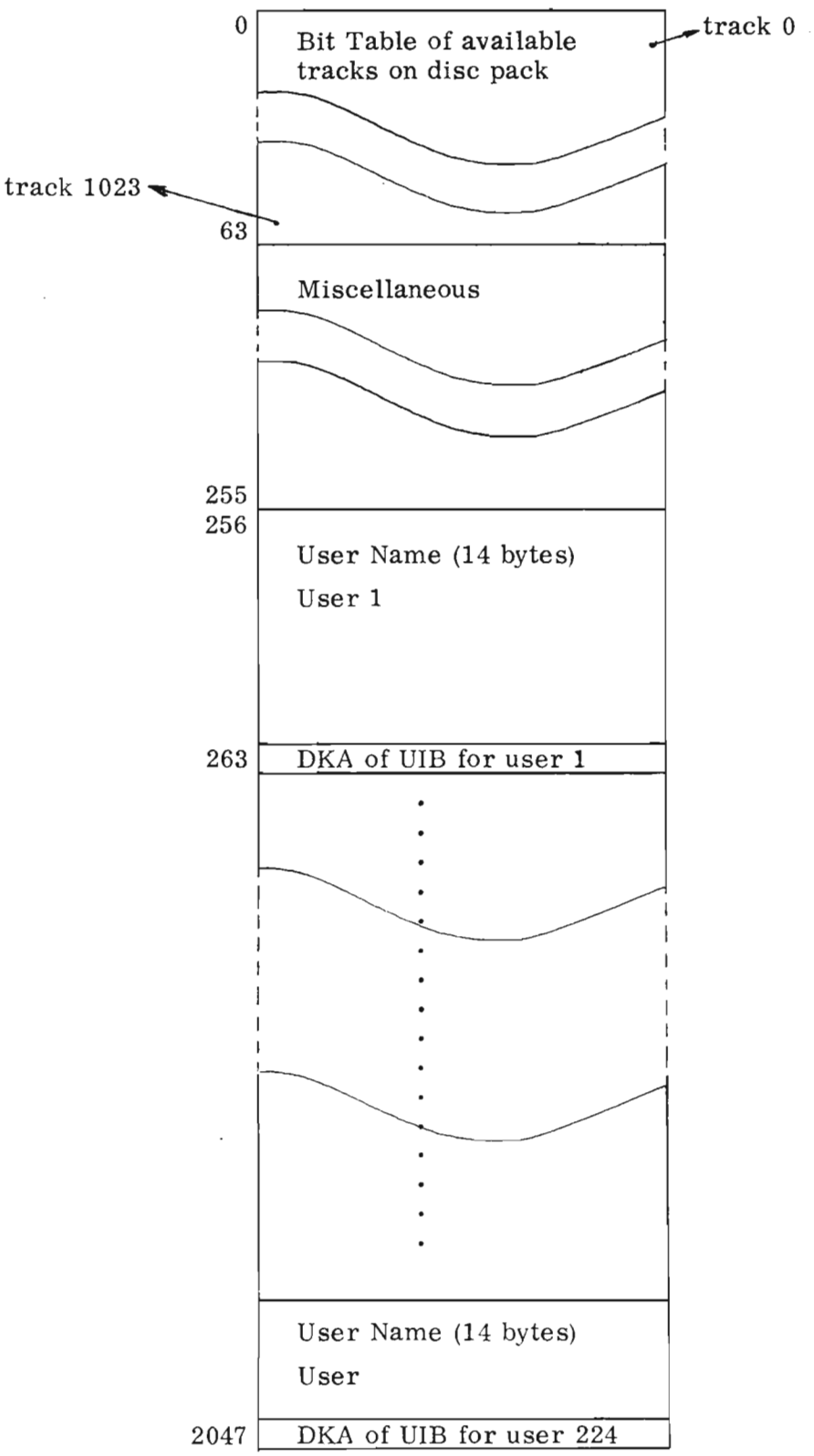
APPENDIX B

FILE SYSTEM TABLES

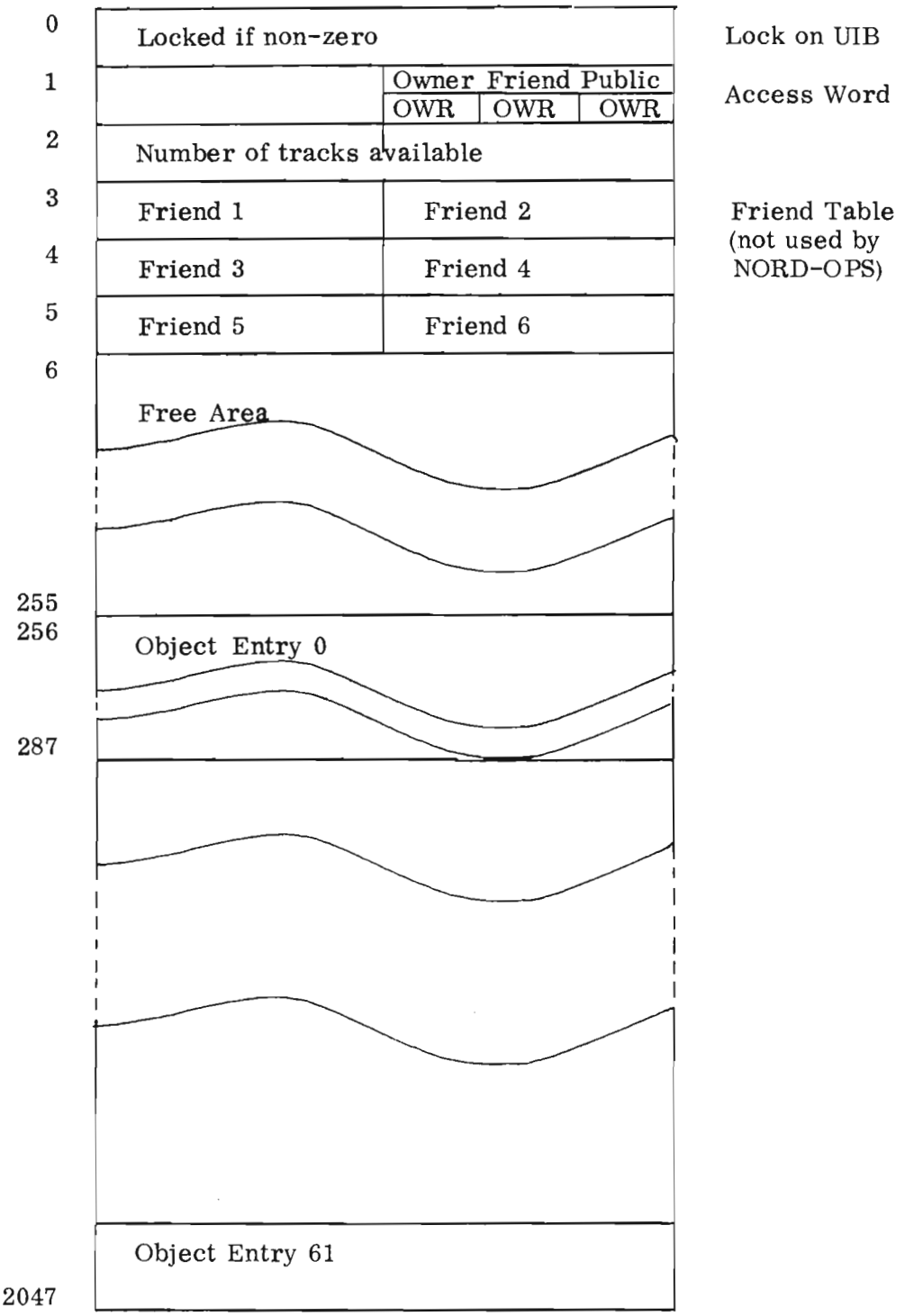
Disc Layout



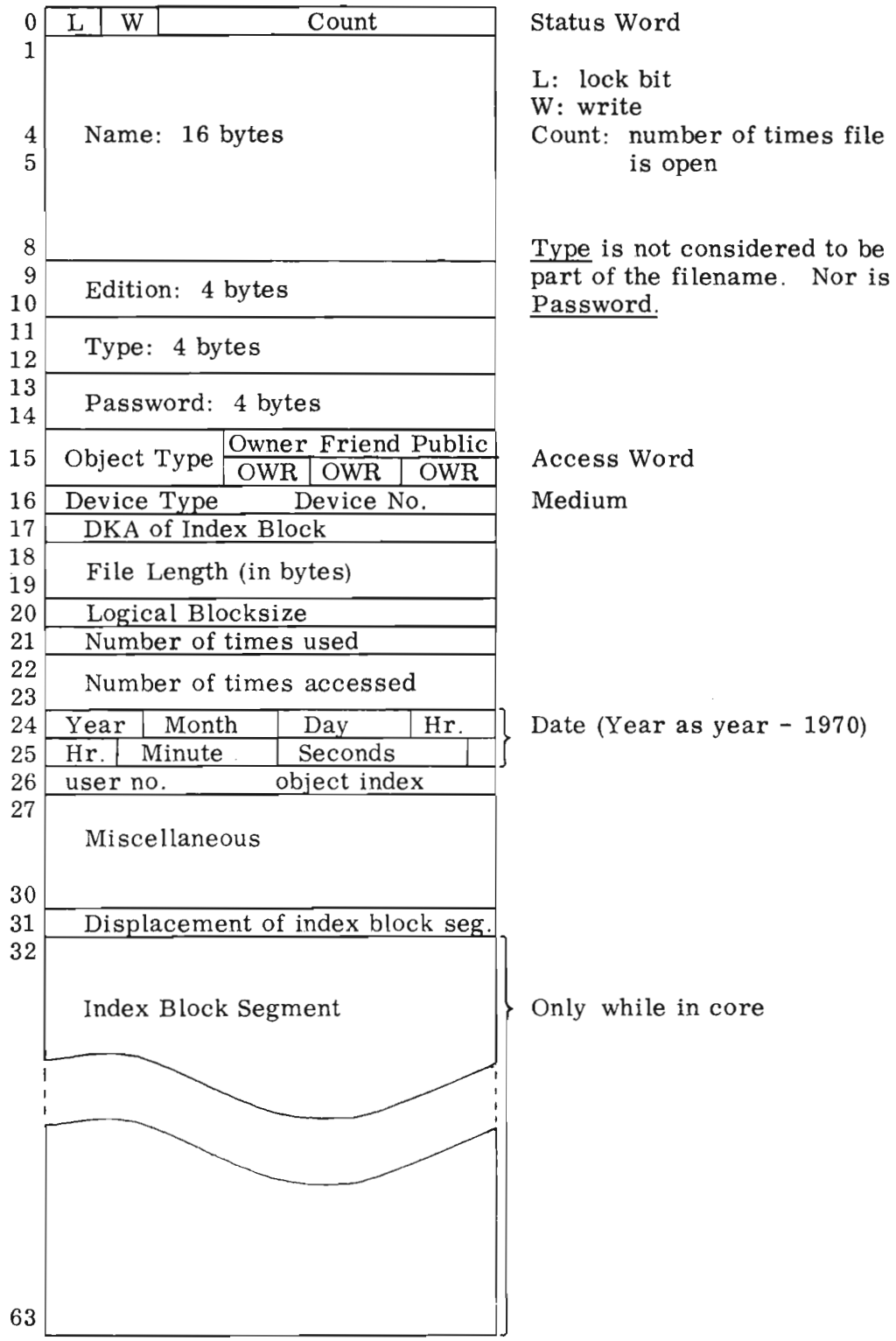
Master Index Block



User Index Block



Object Entry



APPENDIX C

SYSTEM SERVICE REQUEST CODES

Routines available for Users' Programs (via 501)

Request code in octal	Name	Purpose	Parameters	Further documentation
2	BLCIO	Block input/output	Function, dsi, memory address, number of words, status	Section 4.4
5	DBUF	Define buffer	Dsi, memory address	Section 4.3.3.1
6	CBUF	Clear buffer	Dsi	Section 4.3.3.2
7	RBUF	Release buffer	Dsi	Section 4.3.3.3
11	INCH	Input character	Dsi	Section 4.3.4.1
12	OUTCH	Output character	Dsi, character	Section 4.3.4.2
13	INDAT	Input record	Dsi, memory address, number of bytes terminator	Section 4.3.4.3
14	DATUT	Output record	Dsi, memory address, number of bytes, 0	Section 4.3.4.4
16	TIME1	Get current time	Memory address	Section 4.5.1
17	CLCK1	Get clock and date	Memory address	Section 4.5.2
20	JBSUP	Normal exit from task		Section 4.2.2.2
21	CTASK	Call subtask without return	Memory address	Section 4.2.3.2
22	ABORE	Abnormal exit from task	Error code	Section 4.2.2.3
23	ESTAB	Call subtask with return	Memory address	Section 4.2.3.1

Additional Routines available for System Programs (via 502)

Request code in octal	Name	Purpose	Parameters	Further documentation
24	LRT	Start RT-program in connected CPU	RT-description	NORD-OPS Progr. Doc.
25	LABR	Abort RT-program in connected CPU	RT-description	- " -
26	LDSC	Disconnect RT-program in connected CPU	RT-description	- " -
27	NROUT	NORD-OPS version of ROUT	Dummy, logical unit, number of items, RT-program, dummy, variable list	- " -
30	N1BLC	NORD-1 BLCIO	As BLCIO	- " -
31	N5BLC	NORD-5 BLCIO	As BLCIO	- " -
32	ABORT	Terminate RT-program	RT-description	SINTRAN II Users Guide
33	DSCNT	Disconnect RT-program	RT-description	- " -
34	CLOCK	Get clock and date	Memory address	- " -
35	INTV	Connect RT-program to time interval	RT-description, time, time unit	- " -
36	CONCT	Connect RT-program	RT-description	- " -
37	PUSH	Reserve stack	Number of locations	- " -
40	POP	Release stack		- " -
41	RT	Start RT-program	RT-description	- " -
42	RTEXT	End of RT-program		- " -
43	INIT	Initialize input	Dsi, RT-description, max number, terminator	- " -
44	MCALL	Call subroutine	Address, core load	- " -
45	MEXIT	Return from subroutine		- " -
46	INHIB	Inhibit RT-program	Priority	- " -
47	FREE	Release RT-program		- " -
50	ABSTR	Drum transfer	Function, RT-description, memory address, number of words, block address or unit	- " -
51	DKTR1	Disc transfer 1	- " -	- " -
52	DKTR2	Disc transfer 2	- " -	- " -

Request code in octal	Name	Purpose	Parameters	Further documentation
53	MTTR1	Mag. tape transfer 1	Function, RT-description, memory address, number of words, block address or unit	SINTRAN II Users Guide
54	MTTR2	Mag. tape transfer 2	- " -	- " -
55	PLOT	Plotter transfer	- " -	- " -
56	CCMTR	Core/core transfer	- " -	- " -
57	IRSET	Reset I/O buffer	Dsi	- " -
60	SETSY	Set system mode		NORD-OPS Prog. Doc.
61	SETUS	Set user mode		- " -
62	MJOB	Move job table	Job table, queue	- " -
63	COMPR	Command processor	String address	- " -
64	TERM	Terminate job		- " -
65	RTTES	Test RT-program		- " -
66	RWAIT	Restart waiting jobs		- " -
67	EST	Establish new task	String address, string address	- " -
70	OPMES	Give operator message	String address, number of words	- " -
72	ENTER	Enter job	Input dsi, output dsi, command address	- " -
73	CRTRU	Create track	File number, track number, number of tracks, command address	- " -
74	FILE	File system	Command address	- " -
75	BPAGE	Transfer page	File number & function, extra pages, page number, memory address	- " -
76	GNXTR	Get next track	File number, track number	- " -
77	RBLCK	Read block size	File number	- " -
100	SFLEN	Set file length	File number, length	- " -
101	RFLEN	Read file length	File number	- " -

APPENDIX D

CONTROL COMMANDS

Command	Type	Description
$JOB(accon,name,priority,maxsec,maxlines)	E	Section 2.1.1
$MODE(bufferflag,stip,stop)	E	Section 2.1.3
$EXIT	E	Section 2.1.4
$MSG(message,flag)	I	Section 2.2.1
$BIN	I	Section 2.2.2
$RBIN	I	Section 2.2.2
$FILE(file parameters)	I	Chapter 3
$MAC(idsi,ldsi,odsi)	P,I	Section 6.2
$MACF(idsi,ldsi,odsi,filedsi)	P,I	Section 6.2.3
$FTN(idsi,ldsi,odsi,ladr)	P,I	Section 6.3
$LDR(name,odsi,ldsi,$idsi_1$,$idsi_2$...)	P,I	Section 5.4
$QED	P,I	Section 6.4
$N5FTN(idsi,ldsi,odsi,ladr)	P,I	Section 6.5.1
$N5LDR(name,bindsi,listdsi,loader commands)	P,I	Section 6.5.5
$N5ASM(nolist,errorlist,binout,idsi,ldsi,odsi,sdsi)	P,I	Section 6.6
$ACC(dump,list,reset,desired,max)	I	Section 7.3

E - External control command

I - Internal control command

P - Processor

This list contains the most common control commands. A NORD-OPS system may contain a subset or an extension of this list.

APPENDIX E

OPERATOR COMMANDS

The answer after a ':' is in most cases one of the following 6 characters:

- G - Go on with current job
- A - Abort current job
- W - Set current job in waiting condition
- S - Stop the system
- C - Clear current job
- O - Transfer current job to the output queue.

The most used commands after a '*' are:

Command	Parameters	Purpose	Further documentation
DELET	job name*	Delete current job	Section 8.2.2
GO	job name*	Restart job	Section 8.2.3
FILF	parameter list*	Call file system	Section 8.2.4
NOOPS		Start NORD-OPS	Section 8.2.1
PIN	logical unit	Start I/O device	SINTRAN II User's Guide
PINC	logical unit	Clear buffer and start I/O device	- " -
STOP		Stop the computer	- " -
UPDAT	minute, hour, day, month, year	Update internal calendar	- " -

Further commands may be found in SINTRAN II User's Guide.

---

* on next line, after the ':'.

A/S NORSK DATA-ELEKTRONIKK
Økernveien 145, Oslo 5 - Tlf. 21 73 71

COMMENT AND EVALUATION SHEET

Publication No. ND-60.026.02 NORD-OPS Reference Manual
August 1973

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 _____

_____

_____

A/S NORSK DATA-ELEKTRONIKK
Økernveien 145, Oslo 5 - Telefon (02) 21 73 71