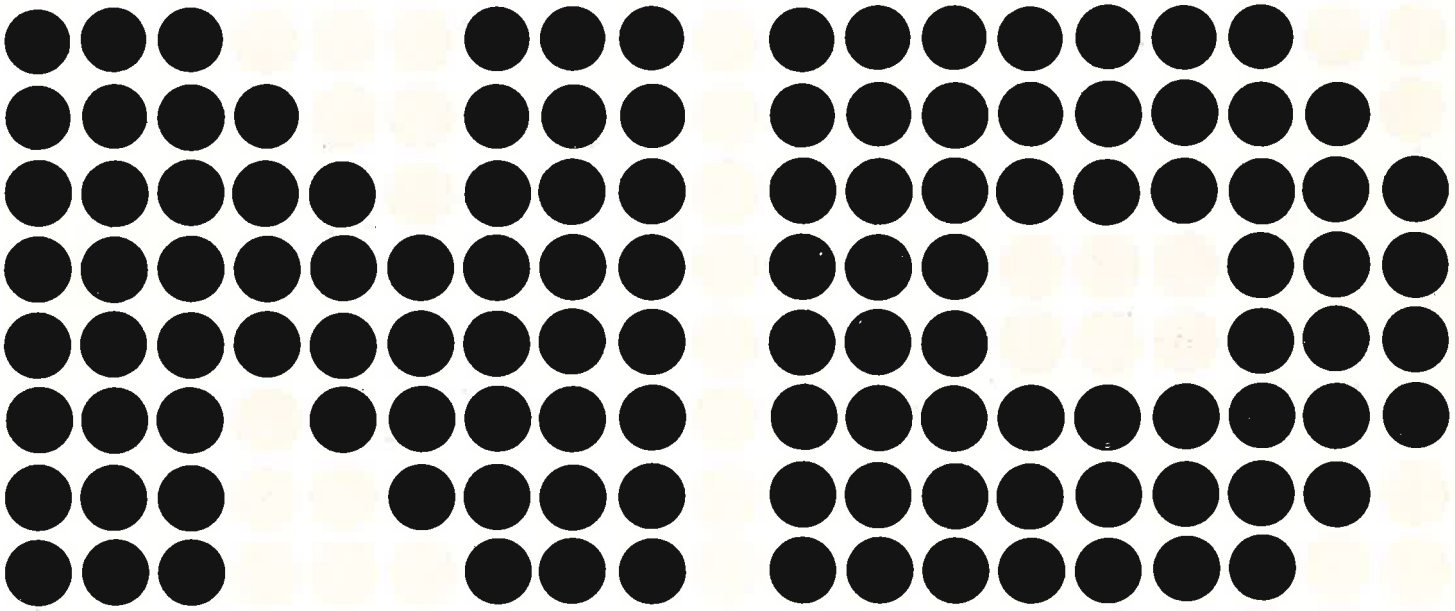
NORD File System
System Documentation

NORSK DATA A.S



NORD File System
System Documentation

NOTICE

PRINTING RECORD

Printing	Notes
10/79	Original Printing
01/80	Second Edition

NORD FILE SYSTEM – System Documentation
Publication No. ND-60.122.02

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

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

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

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

These forms, together with all types of inquiry and requests for documentation should be sent to:

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

PREFACE

The Product

This manual gives a detailed description of the NORD File System operations and design, as implemented under SINTRAN III, version 79.07.15A.

The Reader

The manual is addressed to system programmers working with support and development functions.

Prerequisite Knowledge

The reader of this manual is supposed to possess a general knowledge of file operations from the user's viewpoint. He/she should also have a broad knowledge of SINTRAN III design, and should know segment handling and background procesor operations particularly well. Recommended manuals supplying this knowledge are:

SINTRAN III User's Guide (ND-60.050)
SINTRAN III System Documentation (ND-60.062)

The Manual

This manual is part of the course material for a related course, but it may also be used for self-studies or as reference material for maintenance and development purposes.

TABLE OF CONTENTS

+ + +

Section:		*Page:*
1	INTRODUCTION	1—1
2	PHYSICAL AND LOGICAL STRUCTURE OF THE FILE MEDIA	2—1
2.1	Physical Layout of Disks	2—1
2.2	Physical Layout of Floppy Disks	2—4
2.3	Logical Structure (Directory) on Disks and Floppy Disks	2—4
2.4	Directory Entry	2—5
2.5	User File	2—7
2.6	Object File	2—9
2.7	Bit File	2—12
2.8	Data Files	2—13
2.9	Peripherals	2—16
3	FILE SYSTEM DESIGN	3—1
3.1	Memory Organization	3—1
3.2	System Disk Organization	3—2
3.3	Interface to Other Parts of SINTRAN III	3—3
3.3.1	File System Monitor Call Handling	3—4
3.3.2	File System Command Handling	3—6
3.4	Data Structures	3—7
3.4.1	Memory Map of Data Structures	3—7
3.4.2	Name Table	3—8
3.4.3	Directory Table	3—10
3.4.4	User File Buffer	3—13
3.4.5	Object File Buffer	3—15
3.4.6	Bit File Buffers	3—17
3.4.7	System Segment	3—18
3.4.8	Open File Tables	3—21
3.4.9	Device Buffers	3—24
3.4.10	File System Stack	3—26
3.4.11	File System Error Handling	3—33
3.5	File System Commands	3—35
3.5.1	Parameter Collection	3—35
3.5.2	Create Directory	3—36
3.5.3	Enter Directory	3—38
3.5.4	Create User	3—40
3.5.5	Delete User	3—41
3.5.6	Create File	3—42
3.5.7	Delete File	3—43
3.5.8	Open File	3—44

Section: *Page:*

3.6 File Handling Monitor calls ..... 3—46
3.6.1 RFILE/WFILE ..... 3—46
3.6.2 Input Byte ..... 3—47

Appendix:

A A GUIDE TO THE FILE SYSTEM LISTING ..... A—1

1 INTRODUCTION

The NORD File System is an integrated part of the operating system SINTRAN III. Its function is to offer organized structures for storing and retrieving data. The user of SINTRAN III may operate on data through commands and monitor calls. When a file system command or monitor call is used, SINTRAN III will invoke the corresponding routine in the file system.

The file system gives the user simplified functions for accessing data on various file media. These functions are based on logical structures (directories). The physical organization, storing and retrieving of data is taken care of by appropriate calls to the I/O system from the file system. Figure 1.1 illustrates the file system's place in SINTRAN III.

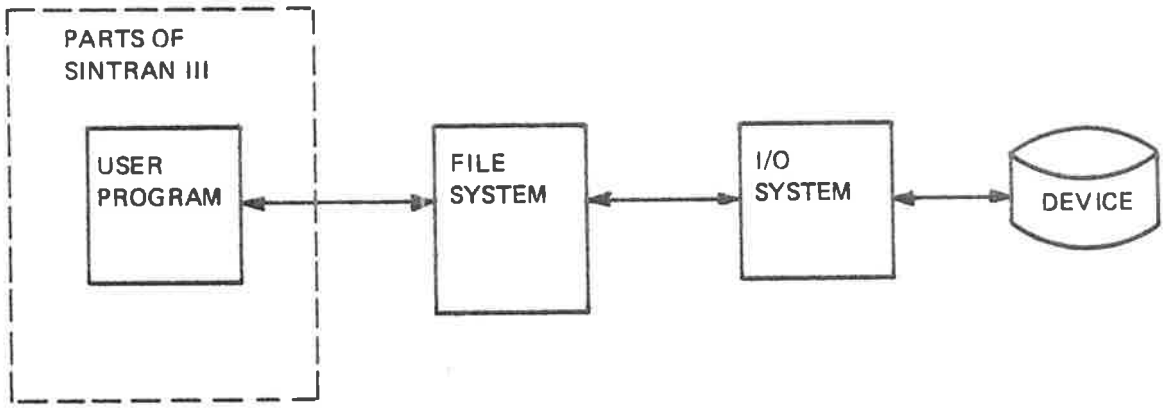


Figure 1.1: File System's Function

The file system uses a set of internal tables and buffers holding information on the item (device, user, directory, etc.) being processed. Through reentrant routines and systematic lock techniques, several users may simultaneously use the file sytem.

2 PHYSICAL AND LOGICAL STRUCTURE OF FILE MEDIA

2.1 *PHYSICAL LAYOUT OF DISKS*

The file system supports a number of different disk types with different physical layout. The general structures, however, are common to all of them and will be discussed first.

All disk packs consist of one or more platters, providing a number of recordable surfaces.

Figure 2.1: Disk Pack

For a given disk, some of the surfaces are used for alignment purposes, while the rest are available for data. The available surfaces are numbered from 0 and upwards. The numbering method is disk dependent.

Each surface has a number of concentric circles, called tracks. The number of tracks is disk dependent, varying between 400 and 823 for our disk types. The tracks are numbered from 0 and upwards, starting at the outer track.

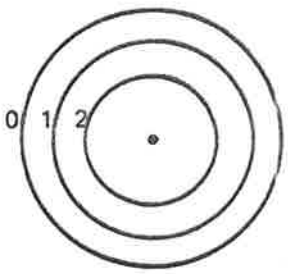


Figure 2.2: Surface with Tracks

Each surface has a track number 0, a track number 1, etc. All tracks of a given number are referred to as a cylinder. Thus, we may speak of cylinder 0 being track 0 on all surfaces.

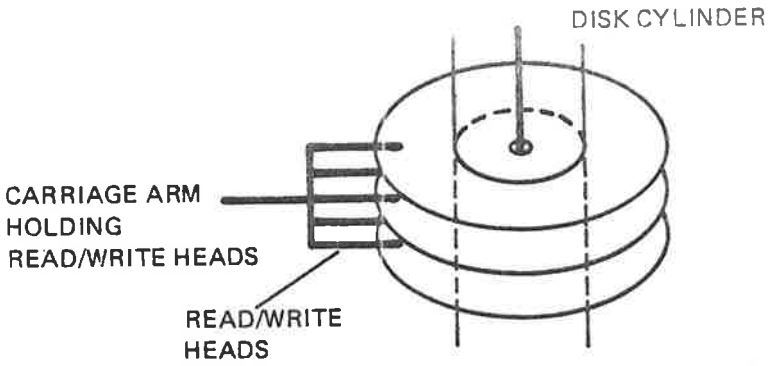


Figure 2.3: Disk Cylinder

Physical disk addresses are organized by disk cylinders, i.e., the lowest disk addresses are in cylinder 0, the next in cylinder 1, etc. This reduces carriage arm movements when accessing data at subsequent disk addresses. In each cylinder the lowest disk addresses are on surface 0, the next on surface 1, etc.

Each track is divided into sectors. The number of sectors per track is 16, 18 or 24 for our disk systems. The sectors are numbered from 0 and upwards.

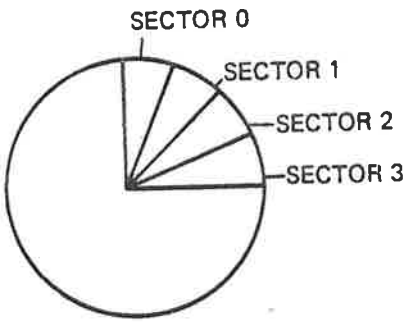


Figure 2.4: Sectors

Each sector consists of a number of 8 bit bytes. In our disk systems this number is either 256 or 1024. The number of bytes per sector is the same for all tracks on a disk. Therefore, the tracks closer to the center of the disk have a higher density than those at the edge.

The file system operates in units of pages (= 1024 words = 2048 bytes). The table below gives the physical characteristics of the disk types supported by SINTRAN III.

Disk type	*Size designation*	*Exact capacity in bytes*	*No. of surfaces*	*No. of tracks/ surface*	*No. of sectors/ track*	*No. of bytes/ sector*	*No. of pages/ cylinder*	*Total capacity in pages*
HAWK	5MB	5,591,040	2/pack	405	24	256	6	2,430
SMD	33MB	32,768,000	5	400	16	1024	40	16,000
SMD	66MB	65,536,000	5	800	16	1024	40	32,000
SMD	37MB	37,969,920	5	412	18	1024	45	18,540
SMD	75MB	75,847,680	5	823	18	1024	45	37,035
CMD	30MB	30,339,072	2	823	18	1024	18	14,814
CMD	60MB	60,678,144	4	823	18	1024	36	29,628
CMD	90MB	91,017,216	6	823	18	1024	54	44,442
SMD	288MB	288,221,184	19	823	18	1024	171	140,733

Figure 2.5: Physical Characteristics of Various Disk Types

The numbering of the surfaces is illustrated in Figure 2.6.

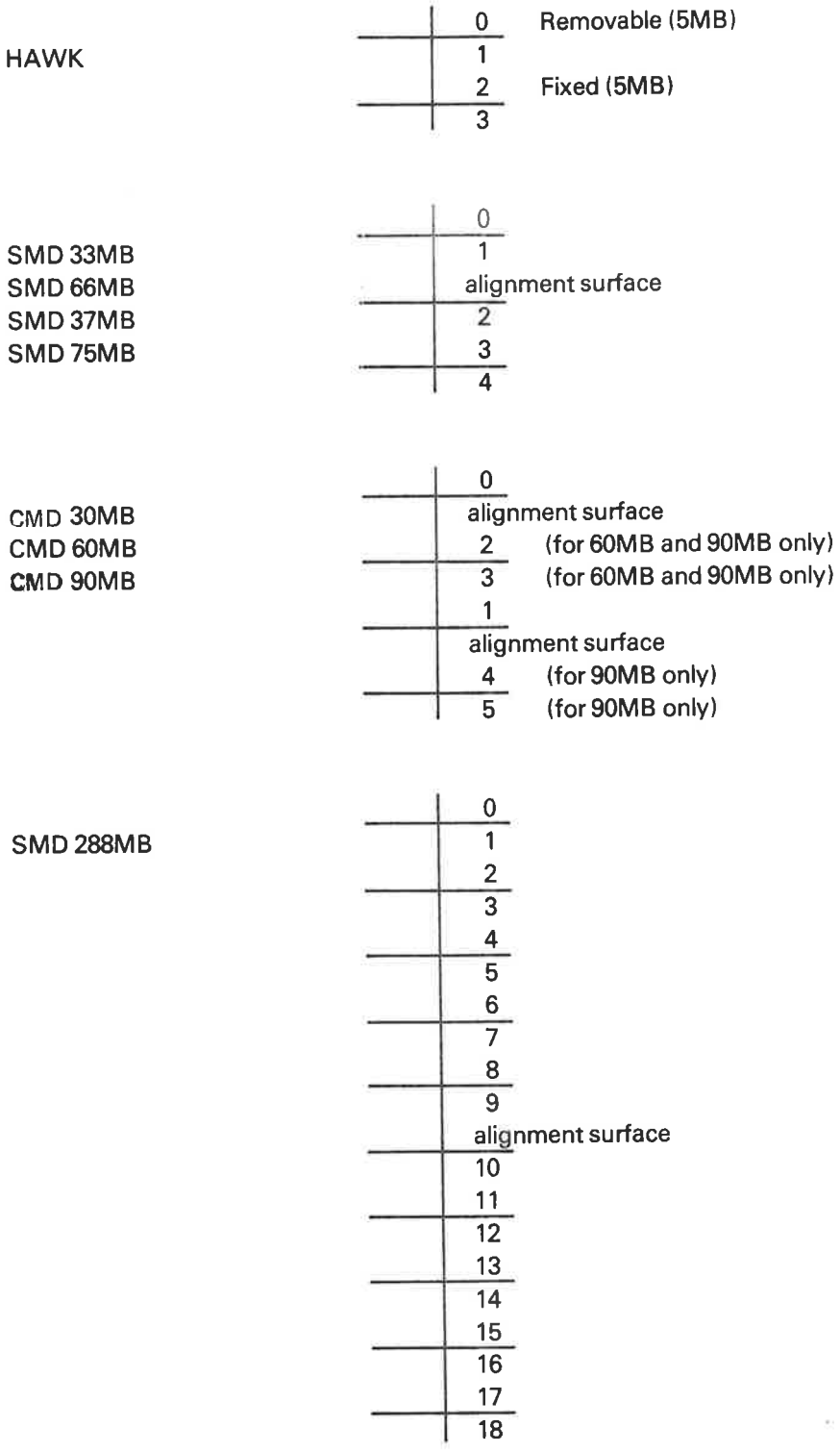


Figure 2.6: Numbering of Disk Pack Surfaces

2.2 *PHYSICAL LAYOUT OF FLOPPY DISKS*

Floppy disks have the same general physical structure as disks (see Section 2.1). The file system supports only one type of floppy disks, with the following physical characteristics.

Exact capacity in bytes	— 315,392
No. of surfaces	— 1
No. of tracks	— 77
No. of sectors/track	— 8
No. of bytes/sector	— 512
Total capacity in pages	— 154

2.3 *LOGICAL STRUCTURE (DIRECTORY) ON DISKS AND FLOPPY DISKS*

All mass storage devices use a page (1K words) as the logical storage unit. From the file system, devices are addressed using page numbers, and file transfers are performed in units of 1 page.

A directory on a disk or floppy disk is logically organized as follows:

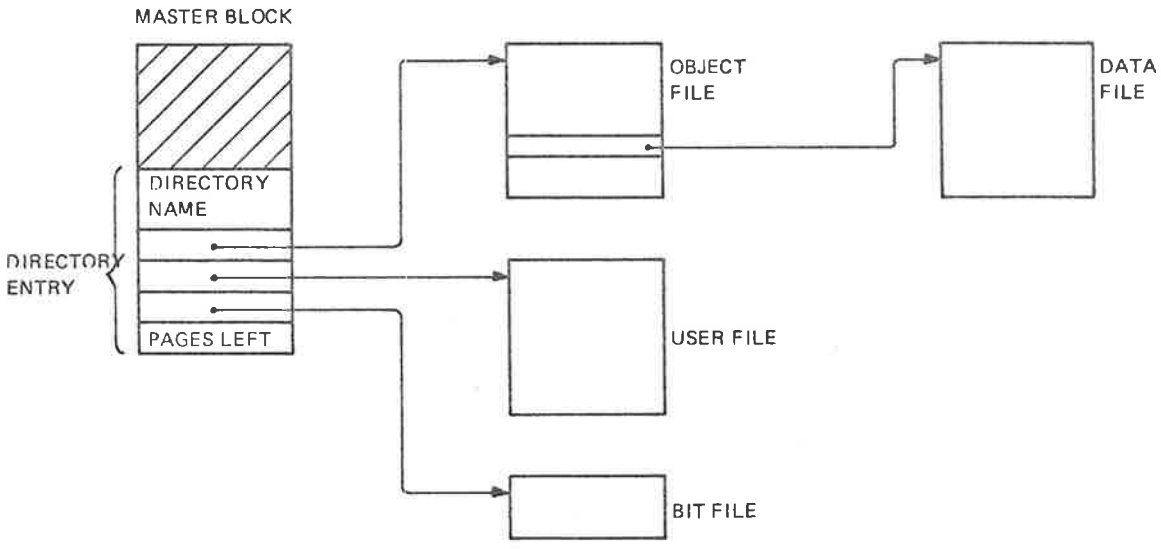


Figure 2.7: Directory

The master block is a 1K block located at the lowest address on the medium, i.e., page 0. The first part (the shaded region) of the master block may contain a bootstrap program to load SINTRAN. The remaining part of the master block (address 1760_8 - 1777_8) holds a directory entry.

2.4 *DIRECTORY ENTRY*

The directory entry contains the directory name, pointers to the object file, user file, and bit file, and the number of unreserved pages on the directory.

The layout of the directory entry is as follows:

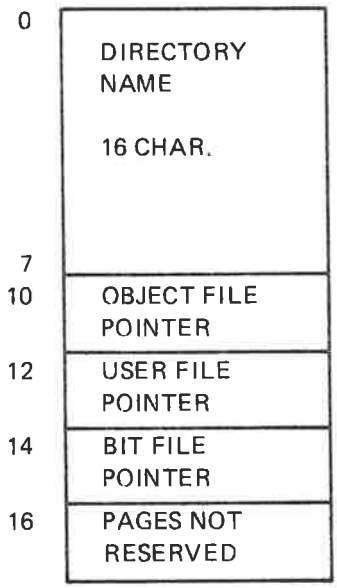


Figure 2.8: Directory Entry

All pointers on file media are double word pointers. The two most significant bits are used to indicate subindexing and indexing as illustrated below.

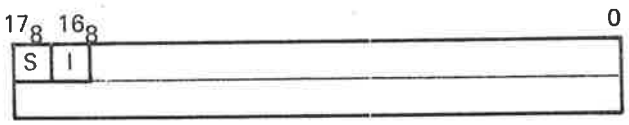


Figure 2.9: Pointer Layout

S — subindexing
I — indexing

The following combinations apply:

S:	I:	
0	0	no indexing is used
0	1	indexing is used
1	0	subindexing is used
1	1	error in file structure (this should not occur)

Examples of indexed and subindexed structures follow in the discussion on the user file, object file and data files.

2.5 *USER FILE*

The *user file* contains information on all the users of the medium. Each medium may have 256 users. Each user has a 32 word entry in the user file.

The user file is organized as an indexed file, i.e., the user file pointer in the directory entry points to an index block. The index block contains up to 8 double word pointers to user file pages. This structure is illustrated in Figure 2.10.

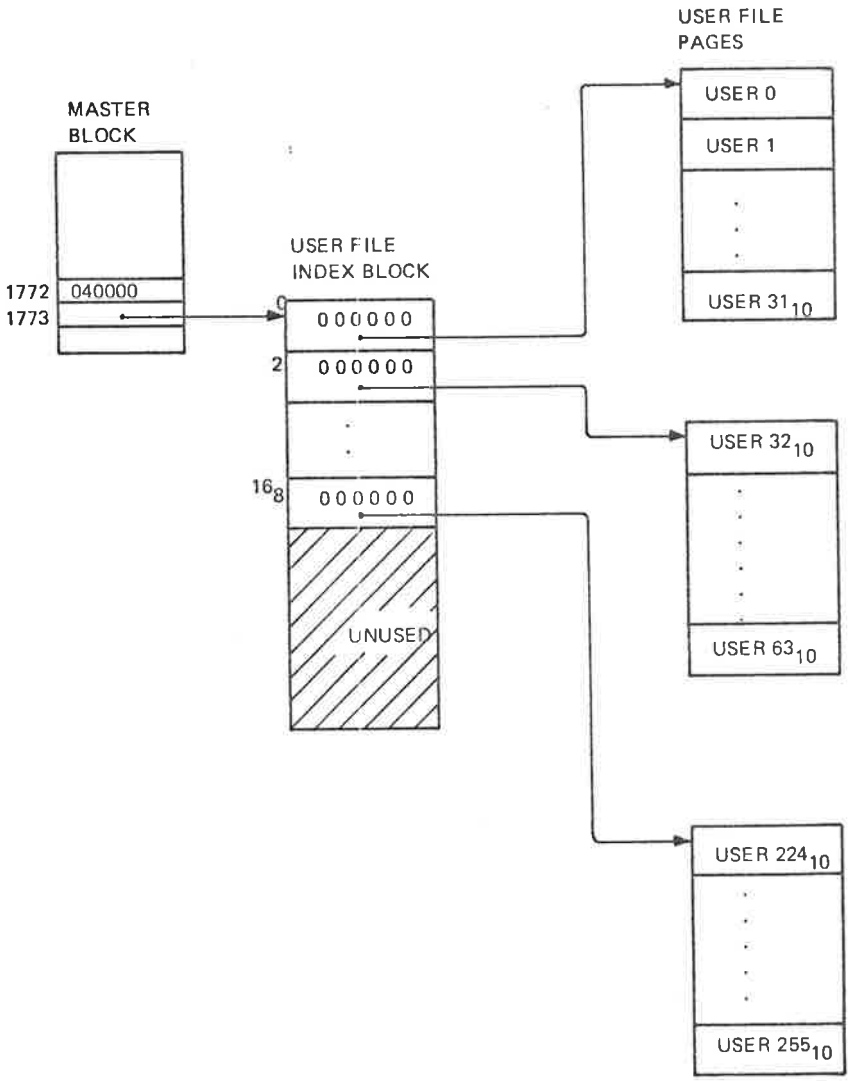


Figure 2.10: User File

Location 1772_8 in the master block has bit 16_8 set to indicate that indexing is used.

The layout of a user file entry is illustrated in Figure 2.11.

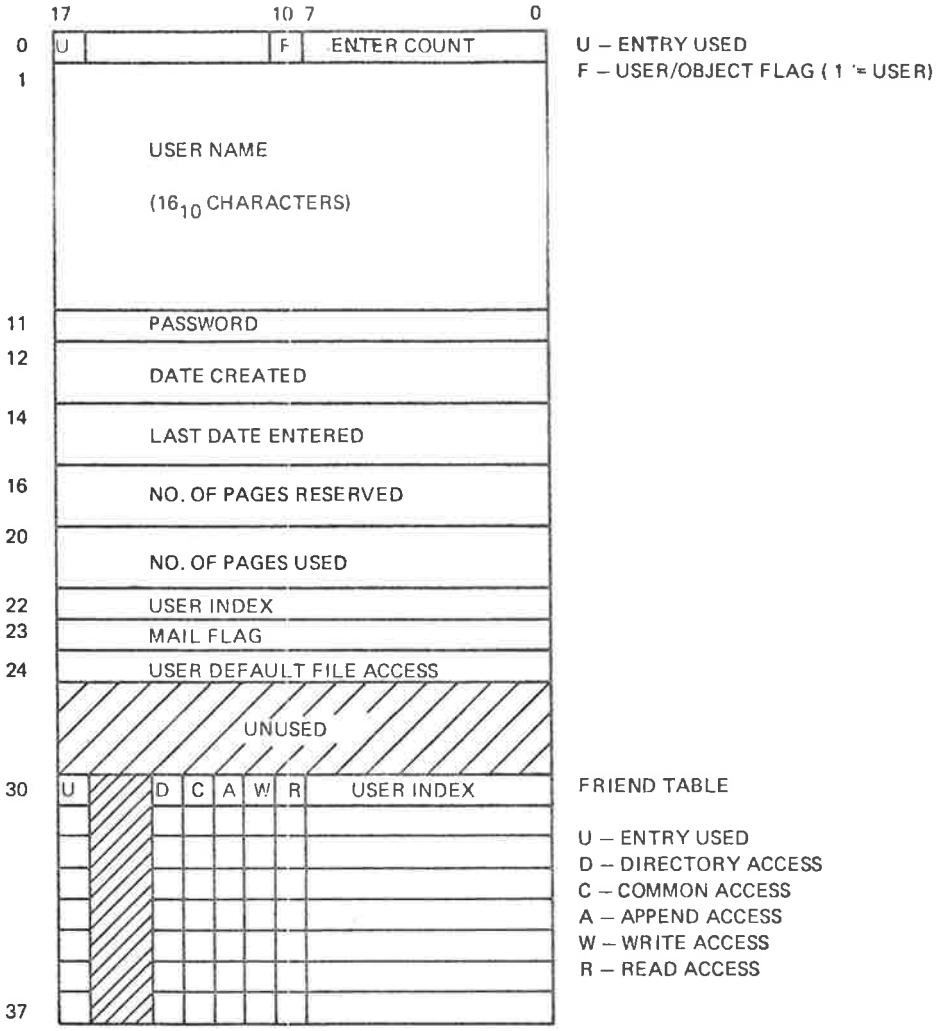


Figure 2.11: User File Entry

All dates in the file system are represented in double word elements with the following layout:

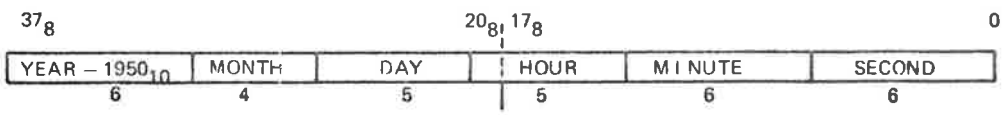


Figure 2.12: Date Layout

2.6 *OBJECT FILE*

The *object file* contains information on all users' files on a medium. Each medium may have 256_{10} files for each user. Each file has a 32_{10} word entry in the object file. The first 256_{10} entries are reserved for user 0, the next 256_{10} entries are reserved for user 1, etc. I.e., each user has maximum 8 pages of entries.

The object file is organized as an indexed or subindexed file. If the highest user index is less than 64_{10}, the object file is indexed, with the structure illustrated in Figure 2.13.

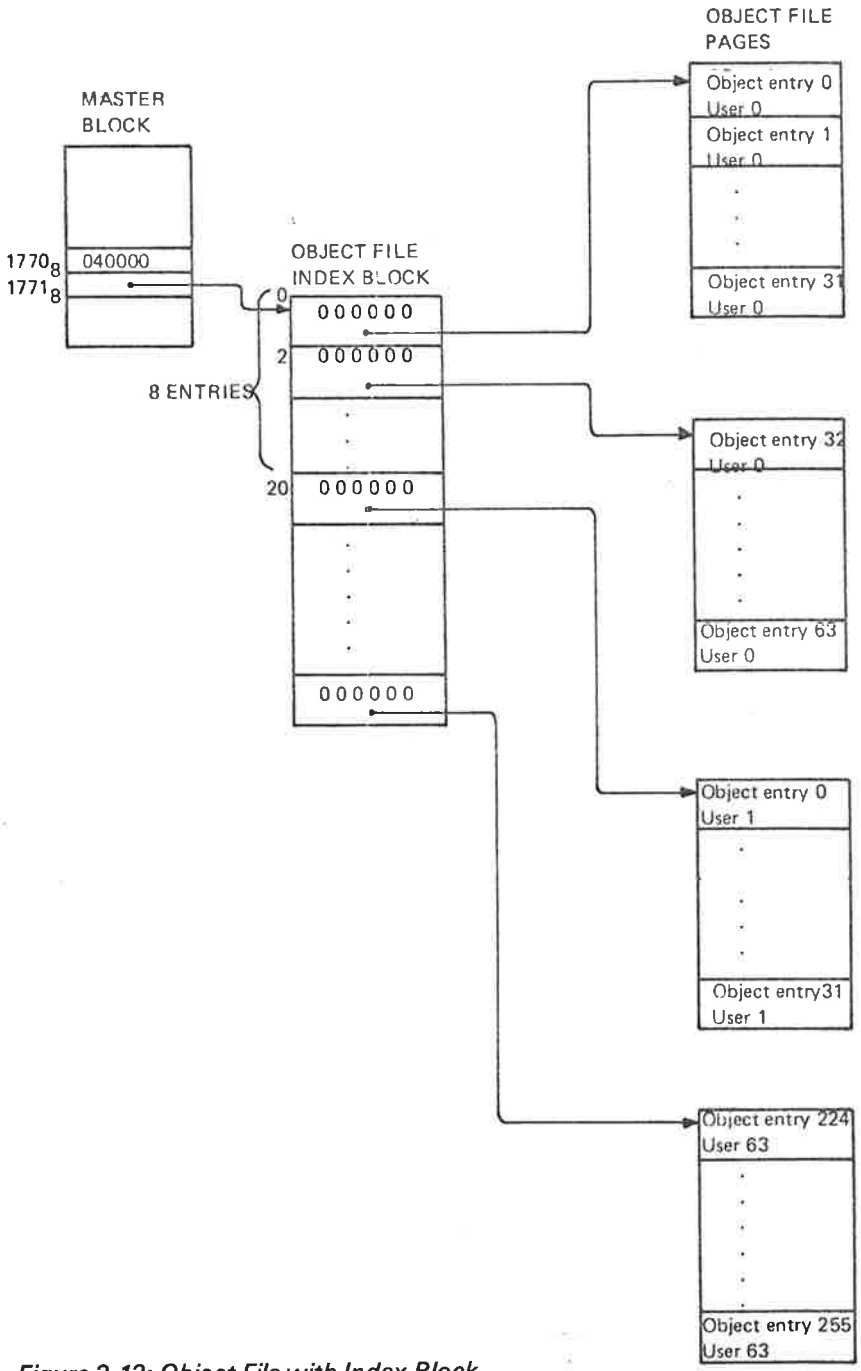


Figure 2.13: Object File with Index Block

Bit 16_8 in location 1770_8 in the master block is set to indicate indexing.

If a directory contains a user with user index exceeding 63_{10}, the object file must be subindexed. (The file system will automatically establish a subindexed structure when user 64_{10} is created.) The subindexed structure is illustrated in Figure 2.14.

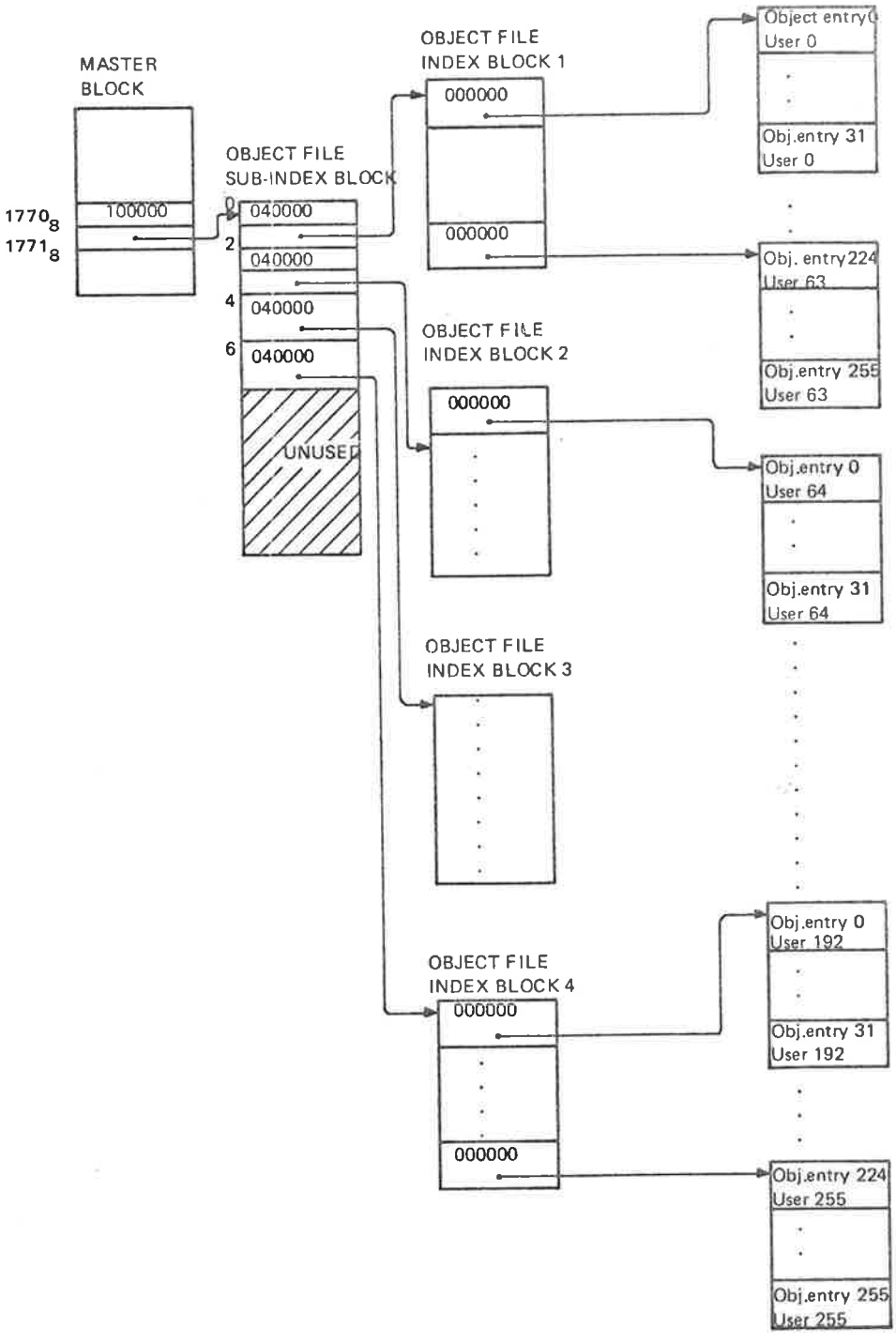


Figure 2.14: Object File with Subindex Block

Bit 17_8 in location 1770_8 in the master block is set to indicate subindexing. The layout of an object file entry is illustrated in Figure 2.15.

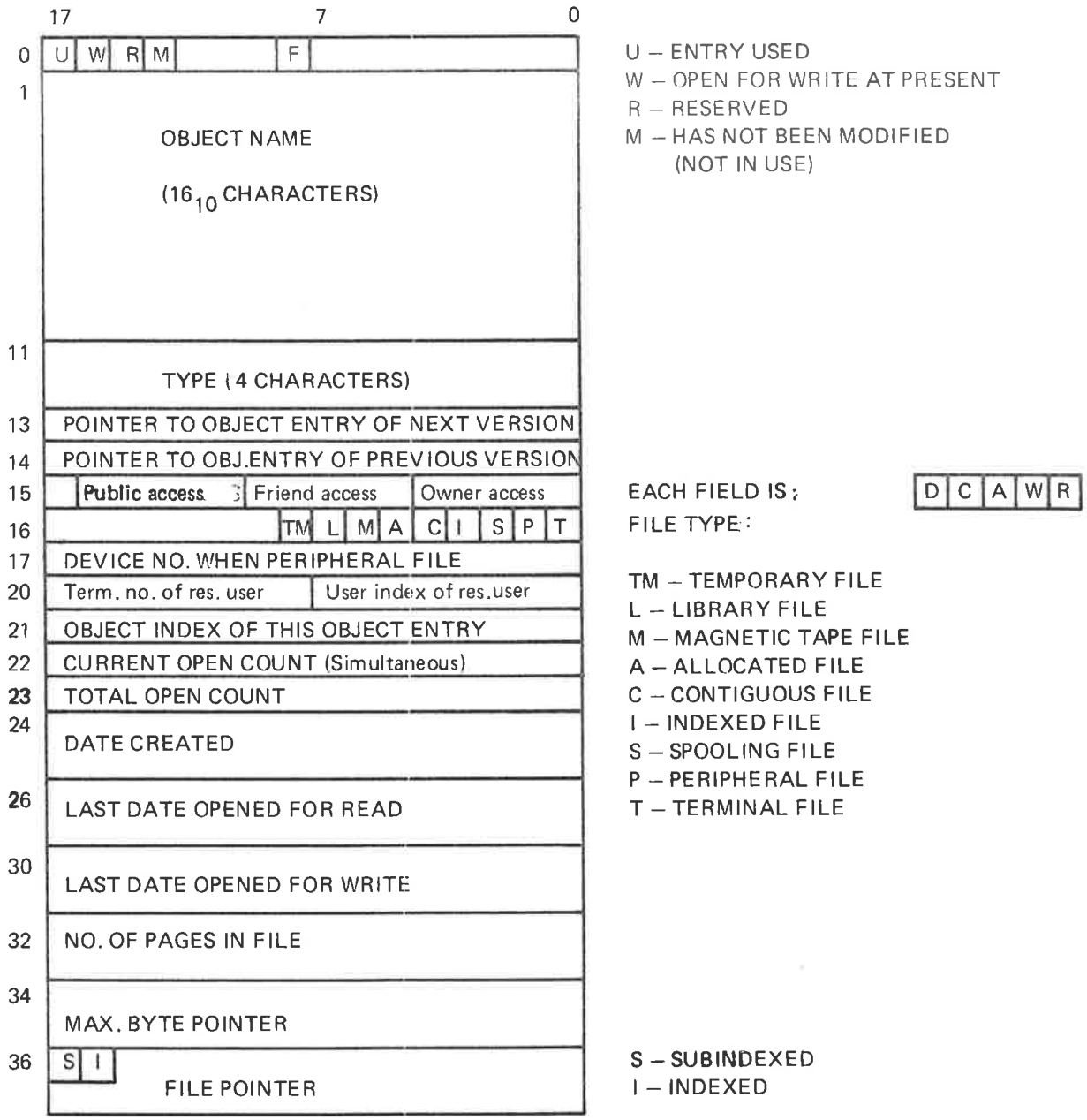


Figure 2.15: Object File Entry

2.7 *BIT FILE*

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

The bit file is a contiguous file. Its size depends on the file medium as listed below:

File Medium:	Bit File Size (in pages):
Floppy disk	1
Disks:	
5MB	1
30MB	2 (1 per unit)
33MB	1
37MB	2
60MB	4 (1 per unit)
66MB	2
75MB	3
90MB	6 (1 per unit)
288MB	9

The bit in the bit file corresponding to a given page is found as follows:

> Suppose bits 17_8 - 0 contain a page number. Then, bits 17_8 - 4 give word number in bit file and bits 3 - 0 give bit number in the word, counted from right to left. This is illustrated in Figure 2.16.

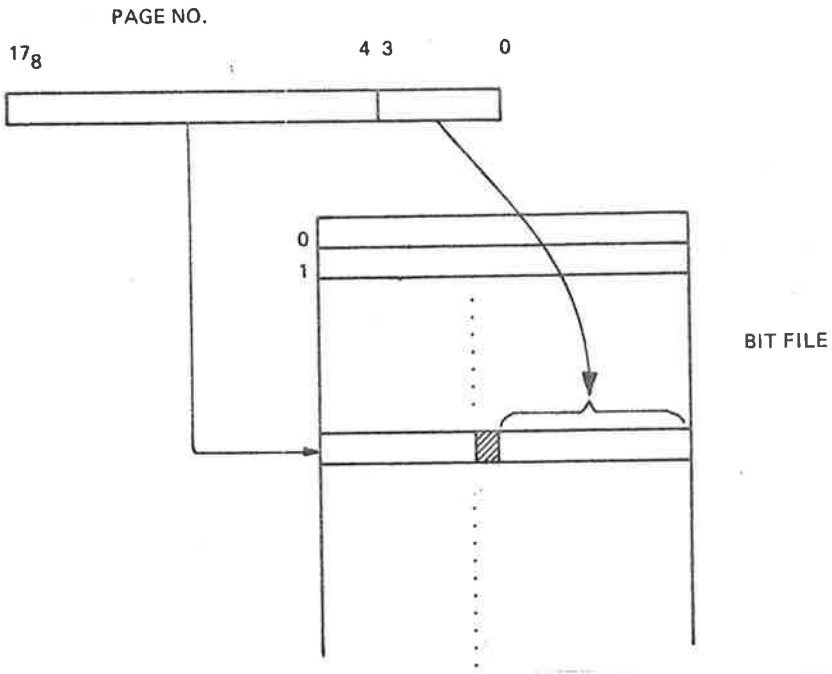


Figure 2.16: Correspondence between Page Number and Bit File Element

2.8 *DATA FILES*

Data files contain user data. All data files have a corresponding object entry. If a data file has any pages, the file pointer of the object entry will point to these pages (see Section 2.2).

A data file is either indexed or contiguous.

A contiguous file has all its pages located in a contiguous area on the file medium, as illustrated in Figure 2.17.

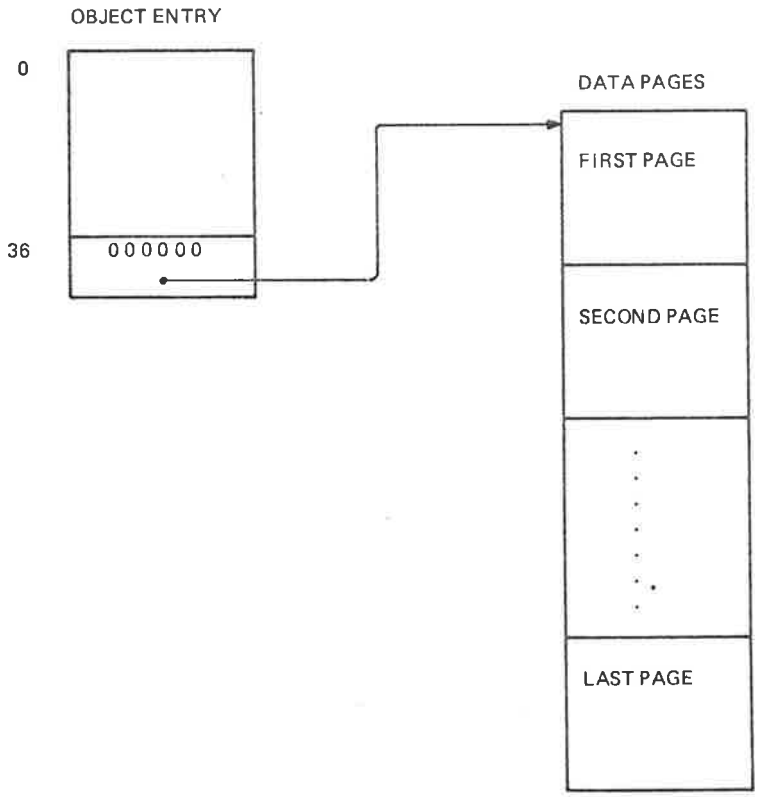


Figure 2.17: Contiguous File

An indexed file has its pages arbitrarily located on the file medium. Each page is referenced by a pointer in an index block. If the file has less than 513 data pages, the structure is as illustrated in Figure 2.18.

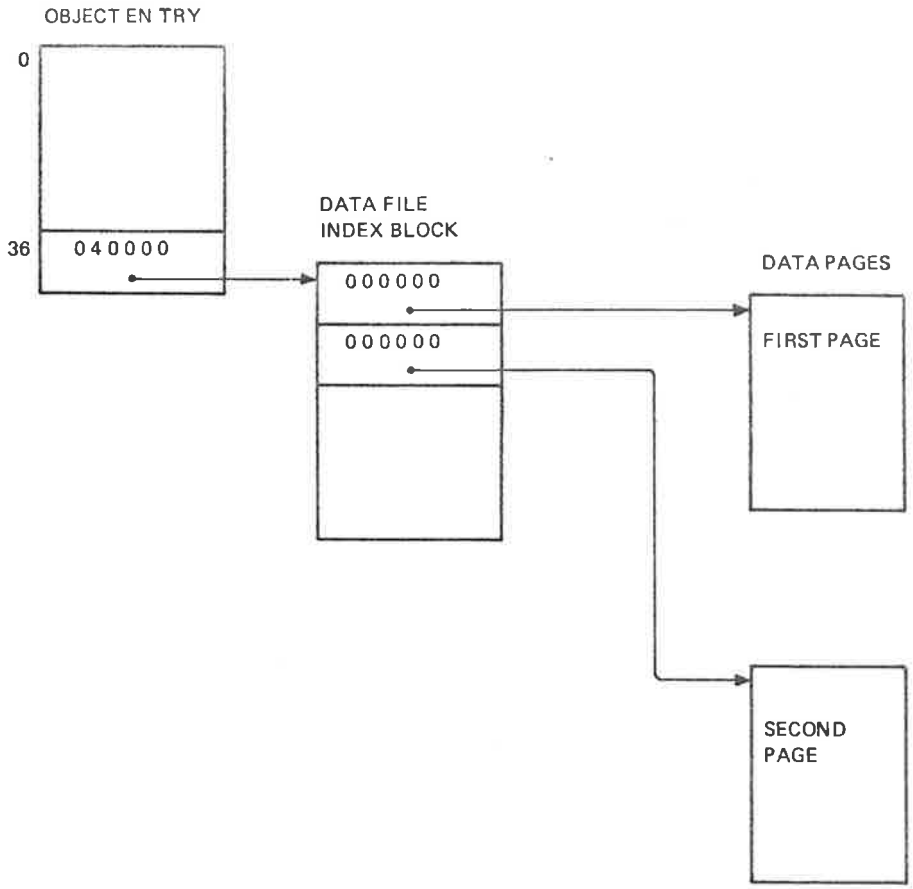


Figure 2.18: Indexed File with less than 513 Data Pages

If an indexed file has more than 512 data pages, a subindex block is used, as illustrated in Figure 2.19.

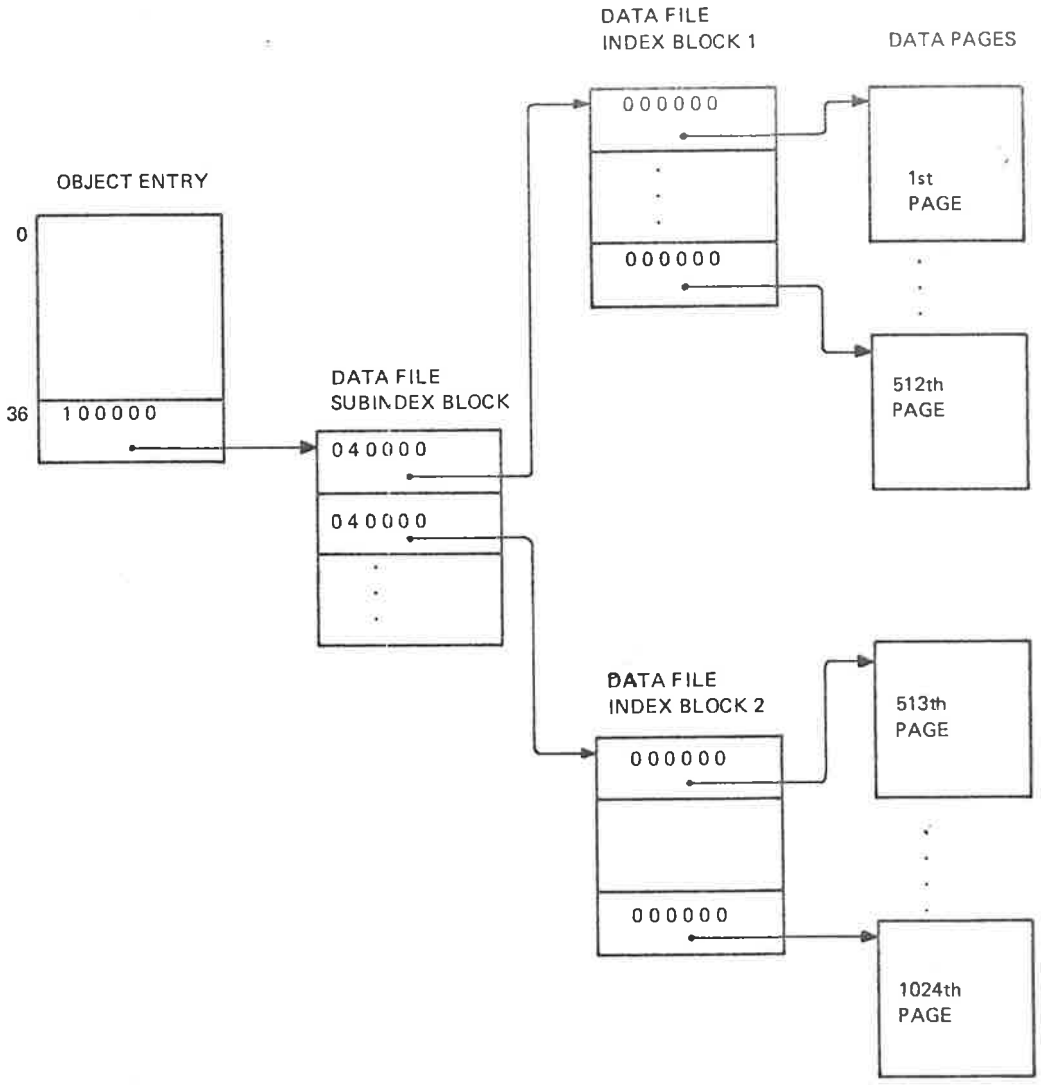


Figure 2.19: Indexed File with more than 512 Pages

The maximum number of data pages in an indexed file is:

512 x 512 pages = 262,144 pages = 512 MB

2.9 *PERIPHERALS*

The file system may support all kinds of peripherals available. Each peripheral device unit to be supported by the file system must be represented by an object entry belonging to user SYSTEM on the main directory. Such an object entry is entered with the SINTRAN commands @SET-PERIPHERAL-FILE and @SET-TERMINAL-FILE. Consequently, a file is either a mass storage file, a peripheral file or a terminal file. The object entries of these 3 types of files compared in Figure 2.20. See also Figure 2.15.

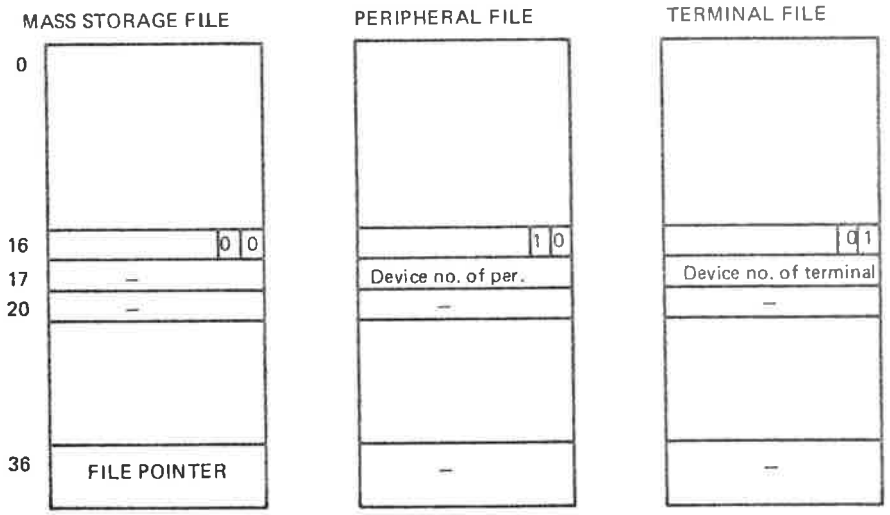


Figure 2.20: Object Entries for a Mass Storage File, a Peripheral File and a Terminal File

3 FILE SYSTEM DESIGN

3.1 *MEMORY ORGANIZATION*

The bulk of the file system is placed on a separate file system segment, segment number 6, with logical address space from 100000_8 to 173777_8 (page 40_8 through 75_8). Some parts of the file system interfacing the I/O system must be resident. (This is required by the I/O system.) Since the file system supports users of a multi-programming operating system, its services must be available to several users simultaneously. Therefore, the file system segment contains only reentrant routines. The data area and some non-reentrant routines needed by a file system user are allocated on the user's system segment (if background) or on the foreground data area in resident memory (if foreground). The system segment lies in the logical address space from 70000_8 to 77777_8 (page 34_8 through 37_8).

Figure 3.1 illustrates how the parts of the file system fit together.

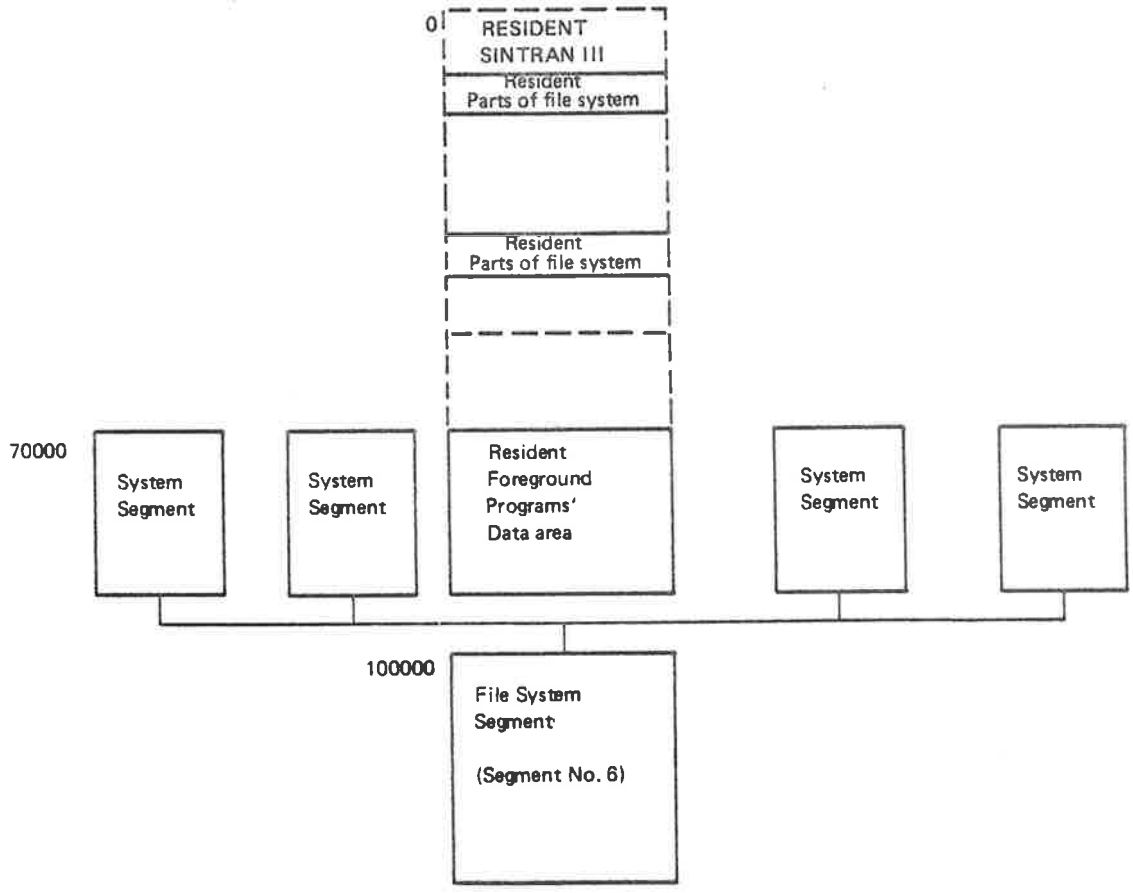


Figure 3.1: Memory Organization

3.2 *SYSTEM DISK ORGANIZATION*

On a system disk the code of the file system segment is placed in the MACM-AREA file, while the code of the system segments is placed on the SINTRAN file.

If SINTRAN III is initialized with the)HENT command in MACM the file system segment code will be moved from the MACM-AREA file to the file system segment (segment 6) on SEGFIL0. Also, the system segments code and constants will be copied from the SINTRAN file into all system segments in the system.

Figure 3.2 shows a system disk layout with the relevant parts.

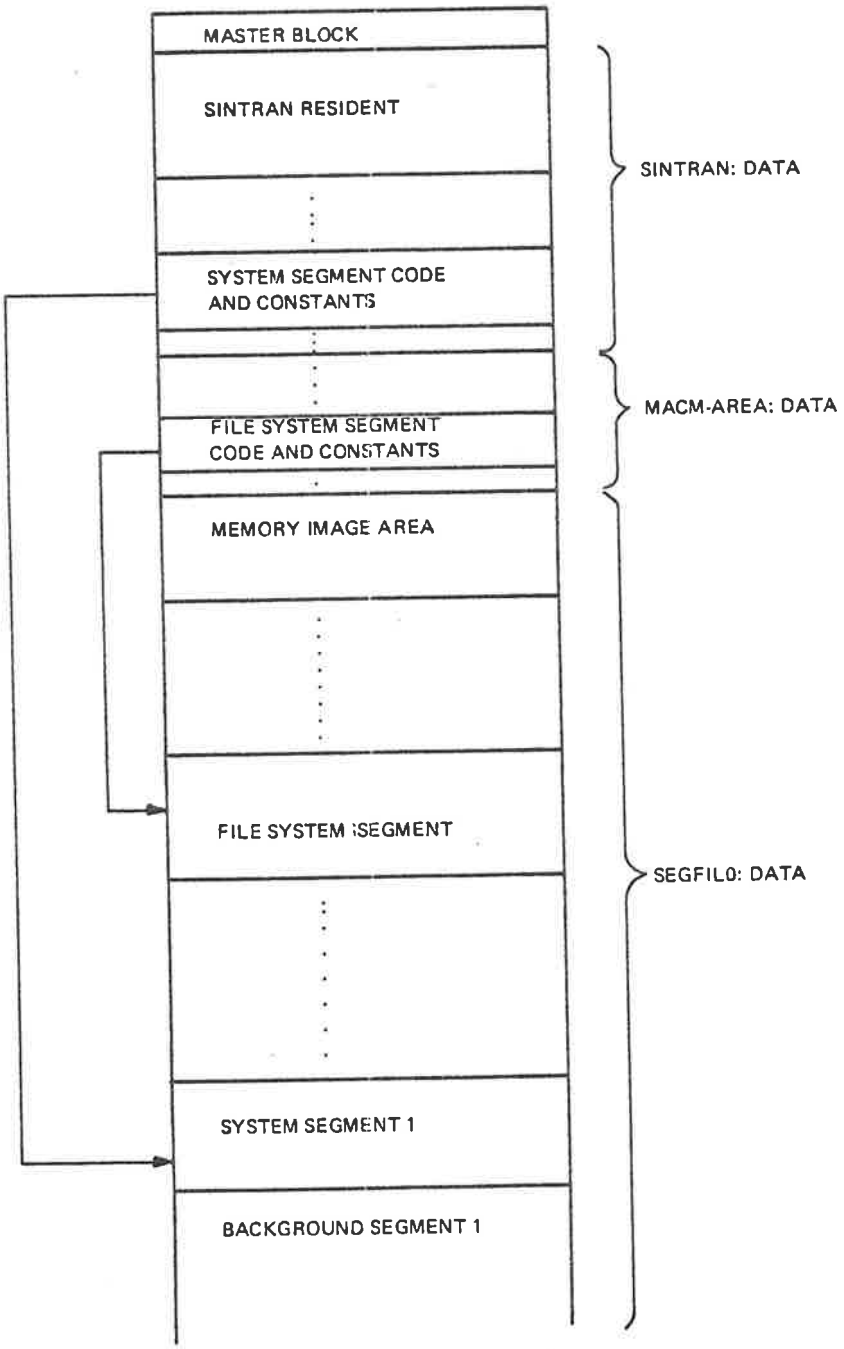


Figure 3.2: System Disk Layout

3.3 *INTERFACE TO OTHER PARTS OF SINTRAN III*

The file system is entered because:

— a monitor call involving file system functions has been executed

— a file system command has been issued

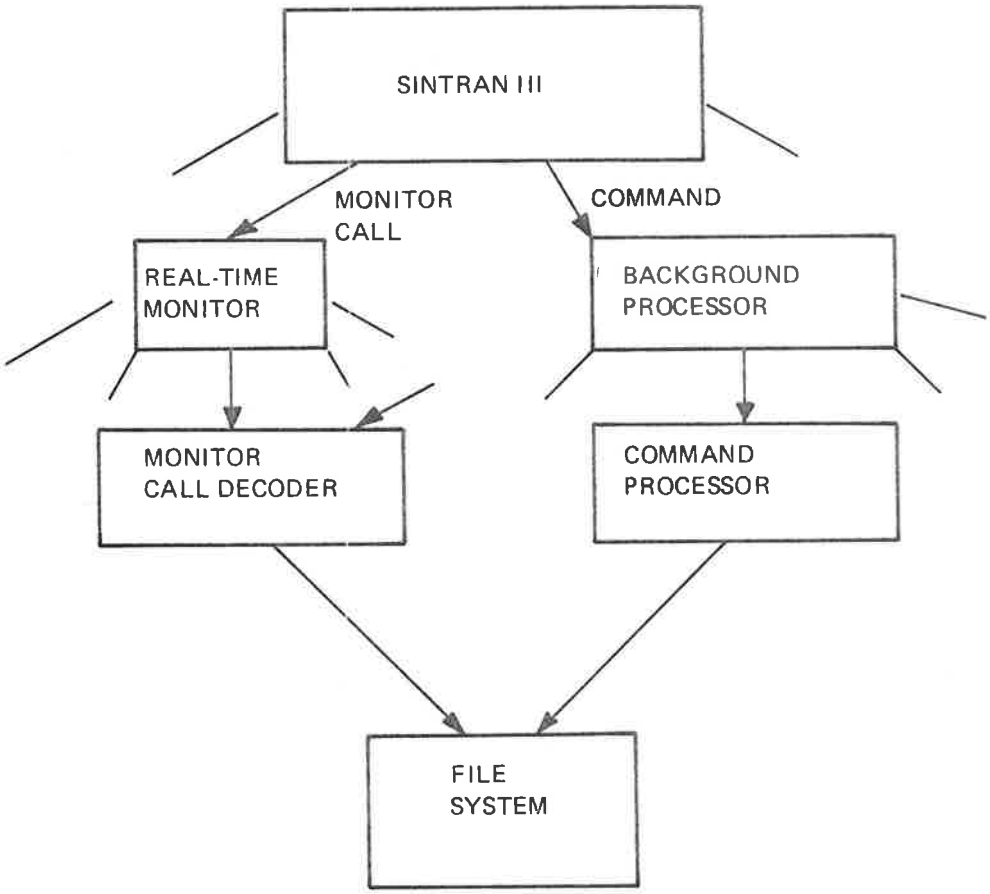


Figure 3.3: Interface to Other Parts of SINTRAN III

The two events are treated differently and will be discussed separately.

3.3.1 *File System Monitor Call Handling*

The monitor call decoder takes different actions for background and foreground programs. For background programs the routine COMENTRY on the system segments gets control. This routine calls the routine MMEXY to bring in the file system segment. Then the proper monitor call routine on the file system segment is called. Upon return to COMENTRY the background segment is brought back through a new call to MMEXY. This sequence of operations is illustrated in Figure 3.4.

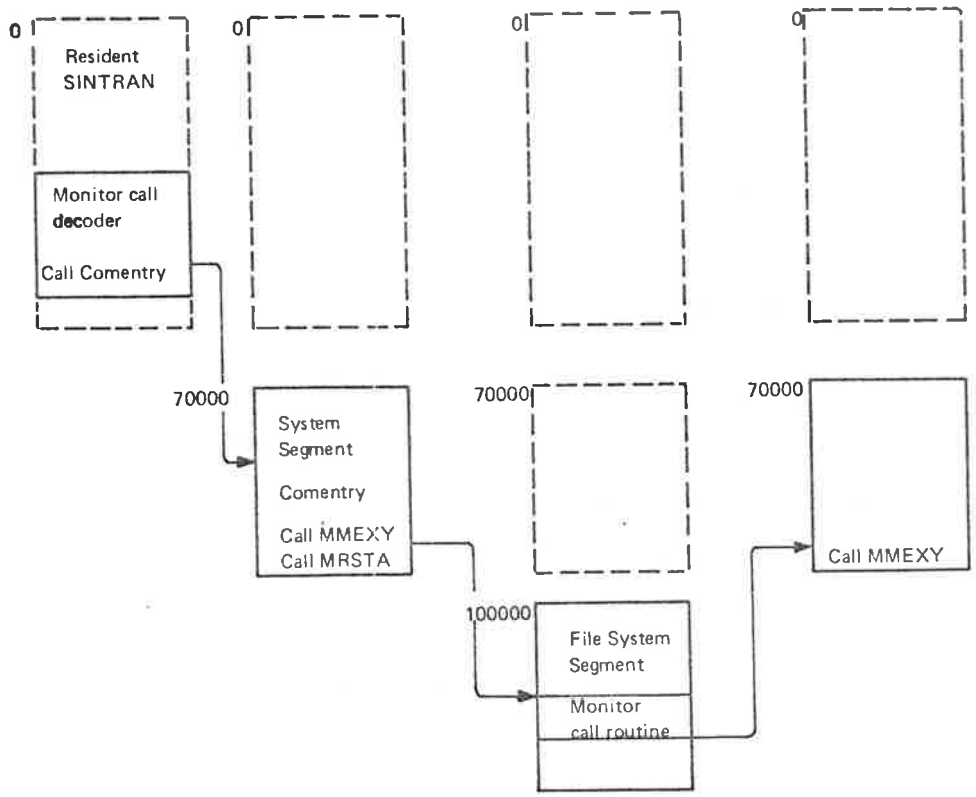


Figure 3.4: File System Monitor Call Handling from Background Programs

Note that both the resident part of SINTRAN and the system segment (with all tables and buffers) are available when executing the monitor call routine on the file system segment.

Foreground programs have no corresponding system segment. File system monitor calls are therefore adminstrated from special file transfer RT programs. There is one file transfer RT program per device type. The proper file transfer RT program is started from the monitor call decoder. The RT program has its code in SINTRAN's resident part. The code contains a call to the routine MMEXY to bring in the file system segment. Then the proper monitor call routine on the file system segment is called. Upon return the file transfer RT program terminates itself. These operations are illustrated in Figure 3.5.

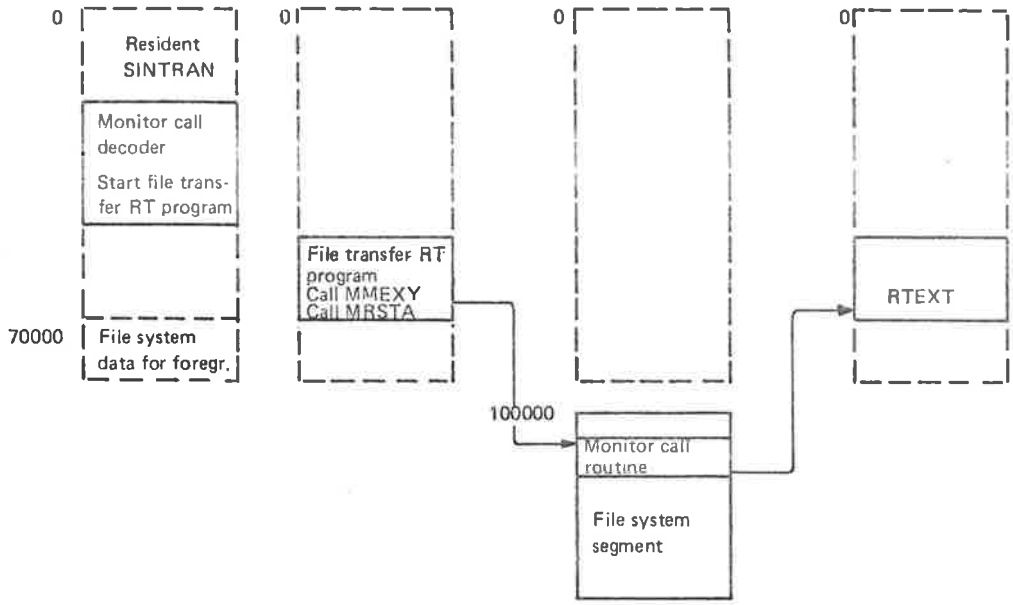


Figure 3.5: File System Monitor Call Handling from Foreground Programs

Note that the resident part of SINTRAN, including file system data for foreground programs, is available when executing the monitor call routine on the file system segment.

3.3.2 *File System Command Handling*

The command segment contains a command processor. When the command processor finds a file system command, the routine FILSYS on the system segment is called. One parameter, the address of the file system command monitor (CMMON) is transferred in the call. FILSYS exchanges segments by bringing in the file system segment on the expense of the command segment. The routine CMMON on the file system segment (parameter to FILSYS) is then called, taking care of the file system operations. Upon return to FILSYS, the command segment is brought back and control is returned to the command processor. These operations are illustrated in Figure 3.6.

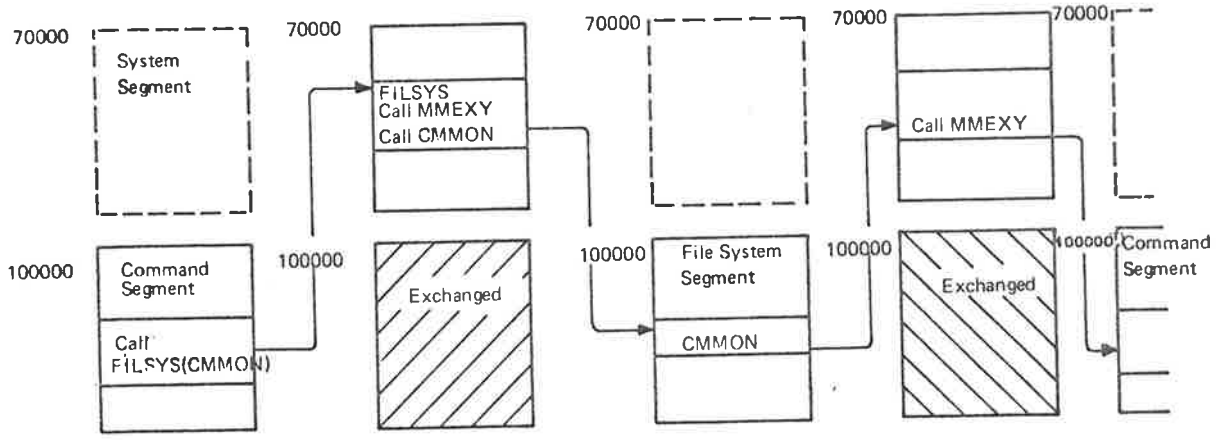


Figure 3.6: File System Command Handling

3.4 *DATA STRUCTURES*

3.4.1 *Memory Map of Data Structures*

Figure 3.7 gives an example of the placement of the most important data structures used by the file system. The addresses given are taken from a specific system and may differ somewhat from one system to another.

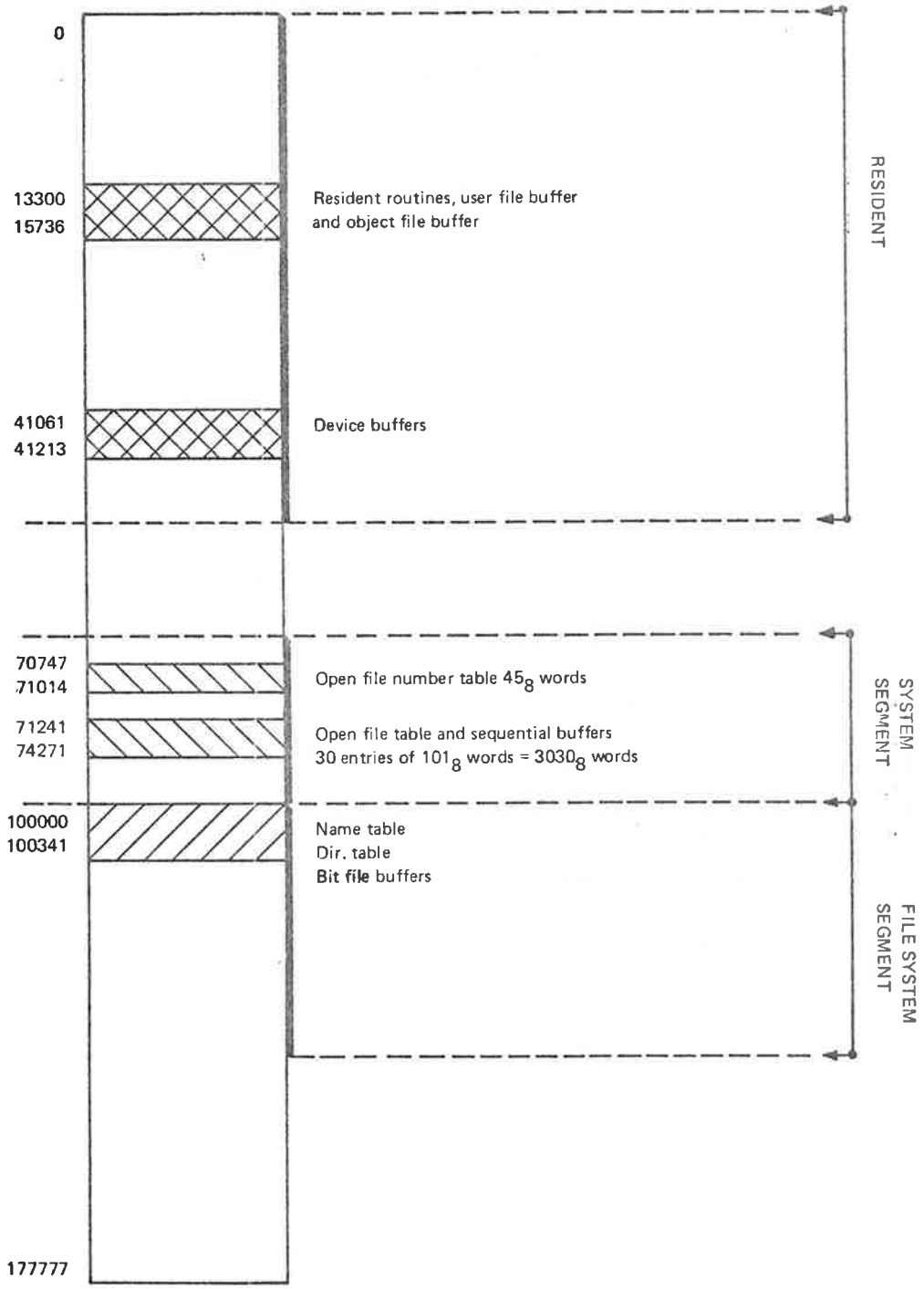


Figure 3.7: Data Structures used by the File System

3.4.2 *Name Table*

Figure 3.8 shows an example of a configuration with some mass storage devices.

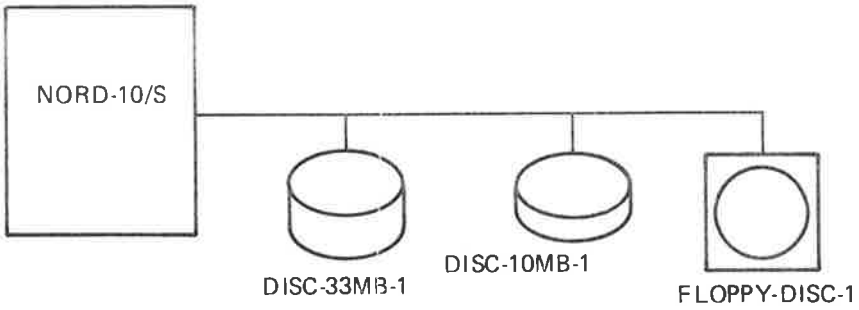


Figure 3.8: Mass Storage Devices

The name table has one entry per mass storage device type available to a given SINTRAN system. A name table entry gives some general information on the corresponding device type. Each entry also has a text string identifying the name of the device. (The name table is defined in SINTRAN III listing, part 2, Section 29.13.) The start address of the name table is 100000_8. Each name table entry consists of 16_8 words. Figure 3.9 illustrates the layout of a name table entry (as defined in the file system listing, Section 1.5). Figure 3.10 illustrates the complete name table.

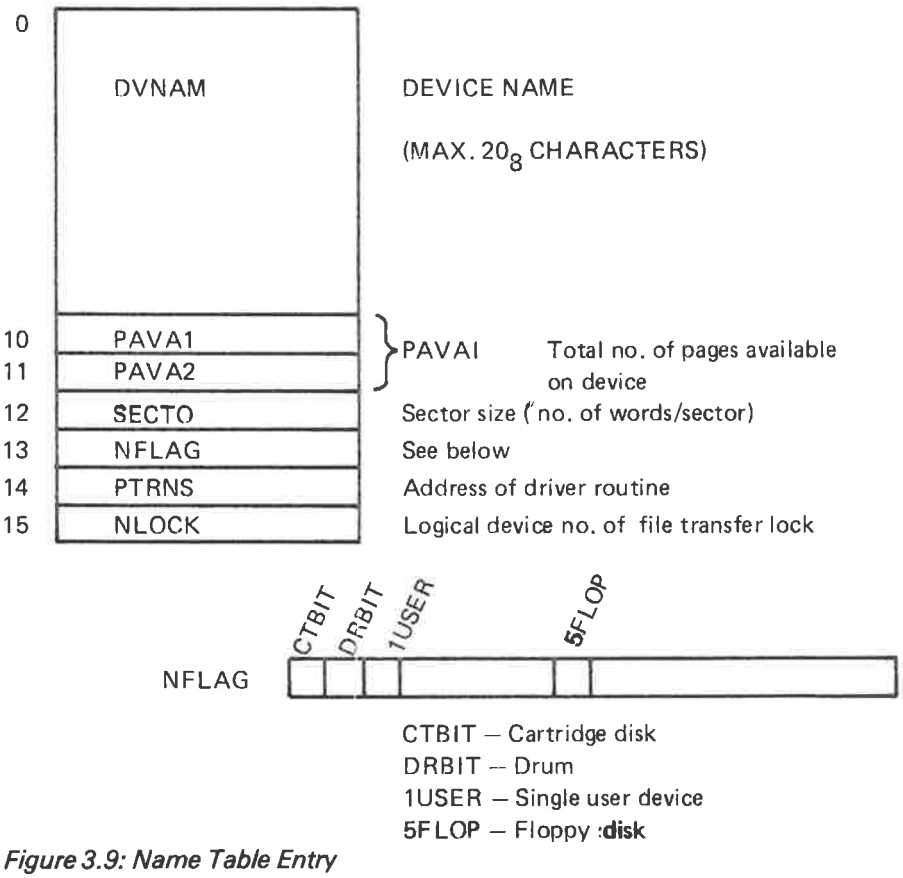


Figure 3.9: Name Table Entry

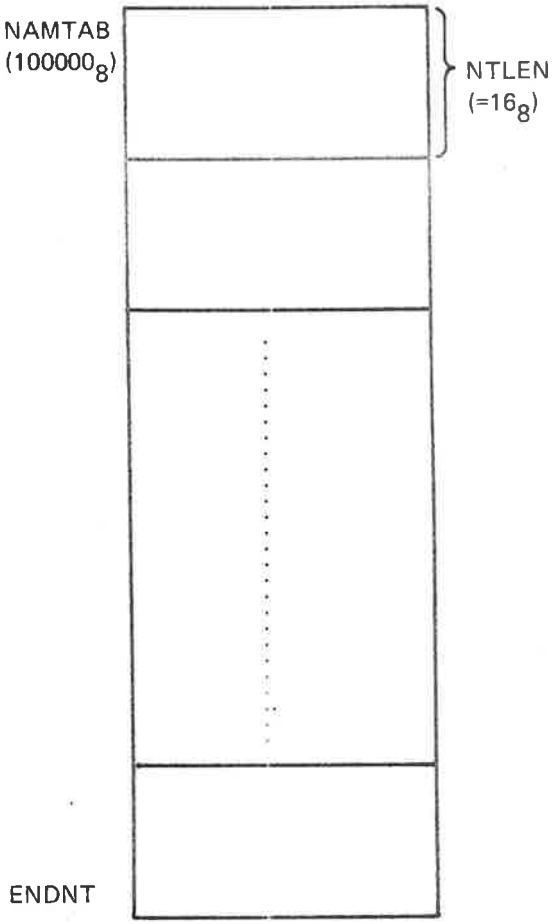


Figure 3.10: Name Table

3.4.3 *Directory Table*

The directory table has one entry per file unit in the system, i.e., there is one entry corresponding to each device being capable of holding a directory, as illustrated in Figure 3.11.

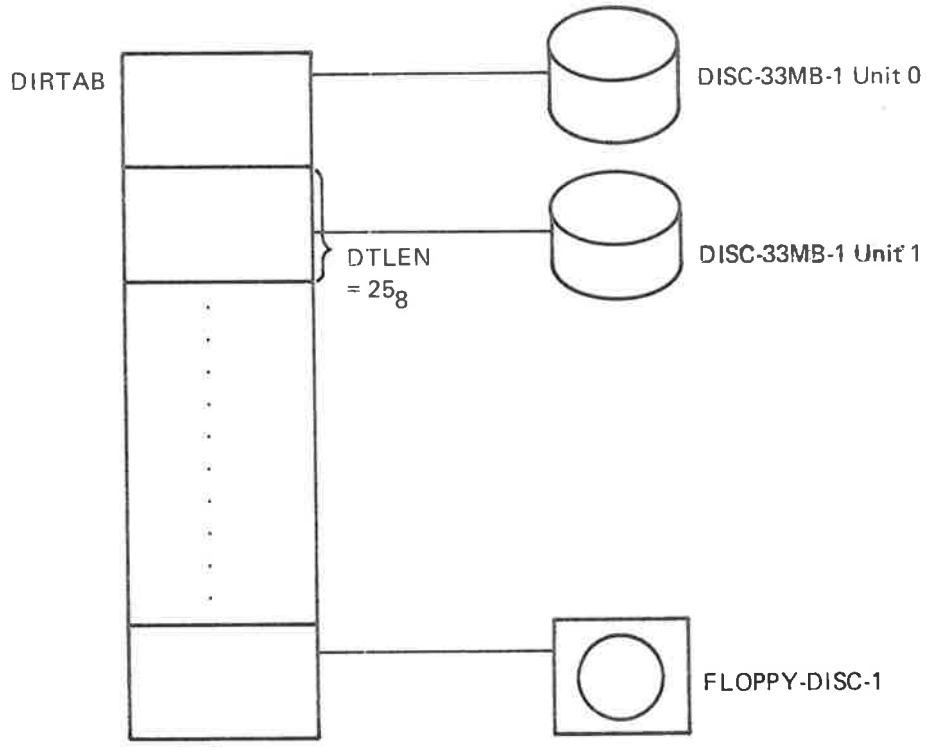
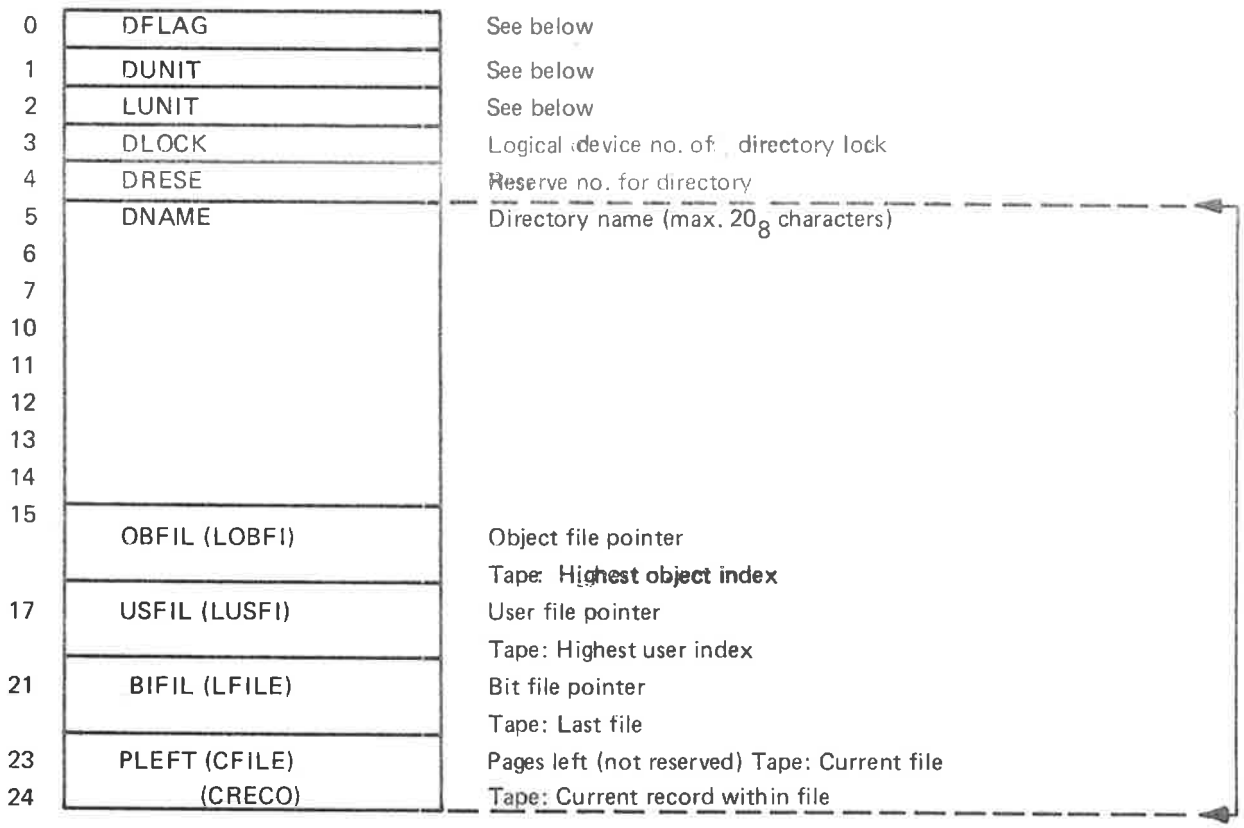
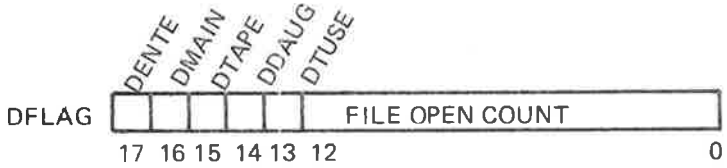


Figure 3.11: Directory Table

Some of the information in a directory table entry is fixed at system generation (defined in SINTRAN III listing, part 2, Section 19.14), while some of it changes dynamically depending on the medium currently mounted, the usage of the medium, etc. Each directory table entry has 25_8 words with layout illustrated in Figure 3.12 (as defined in the File System listing, Section 1.4).

The entries OBFIL, USFIL, BIFIL and BLEFT are double word elements. The entries LOBFI, LOSFI, LFILE and CFILE used for tapes, are single word elements, they occupy only the first word in the double word element.





DENTE — directory entered
DMAIN — main device
DTAPE — tape flag
DDAUF — default directory
DTUSE — tape used

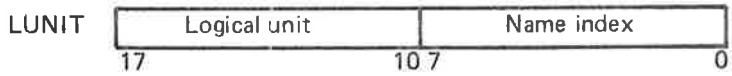


Figure 3.12: Directory Table Entry

Figure 3.13 illustrates the permanent relationship between a directory table entry and the name table.

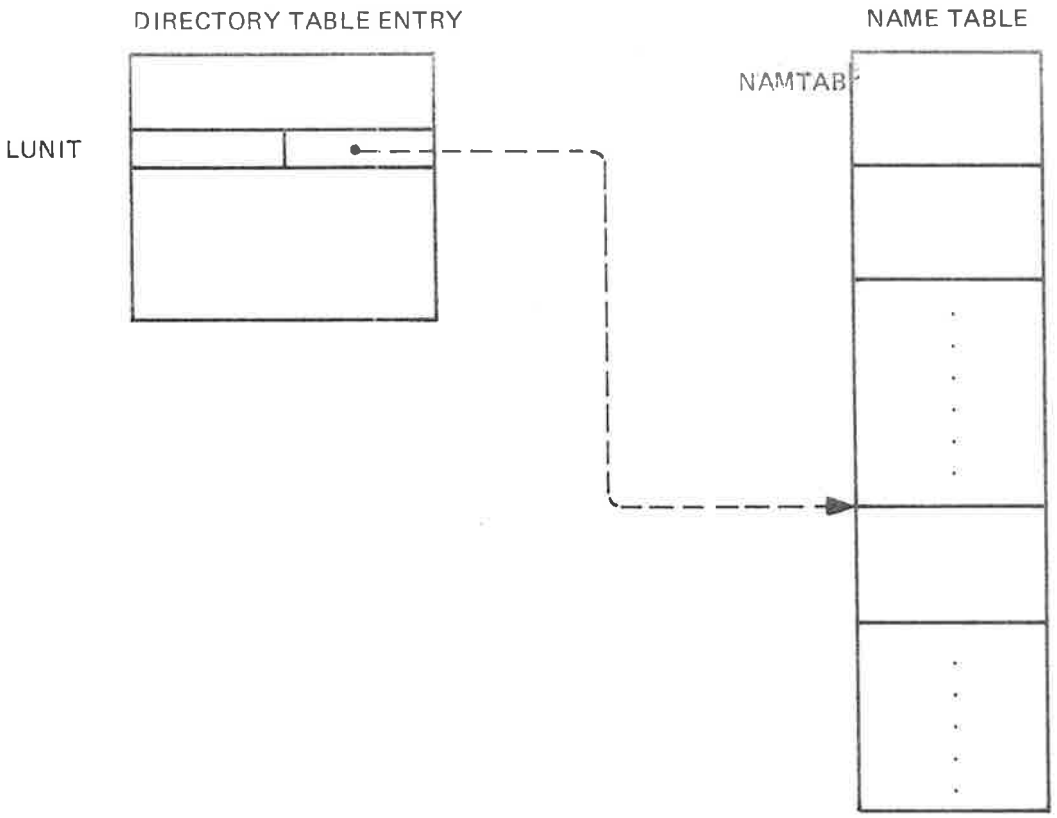


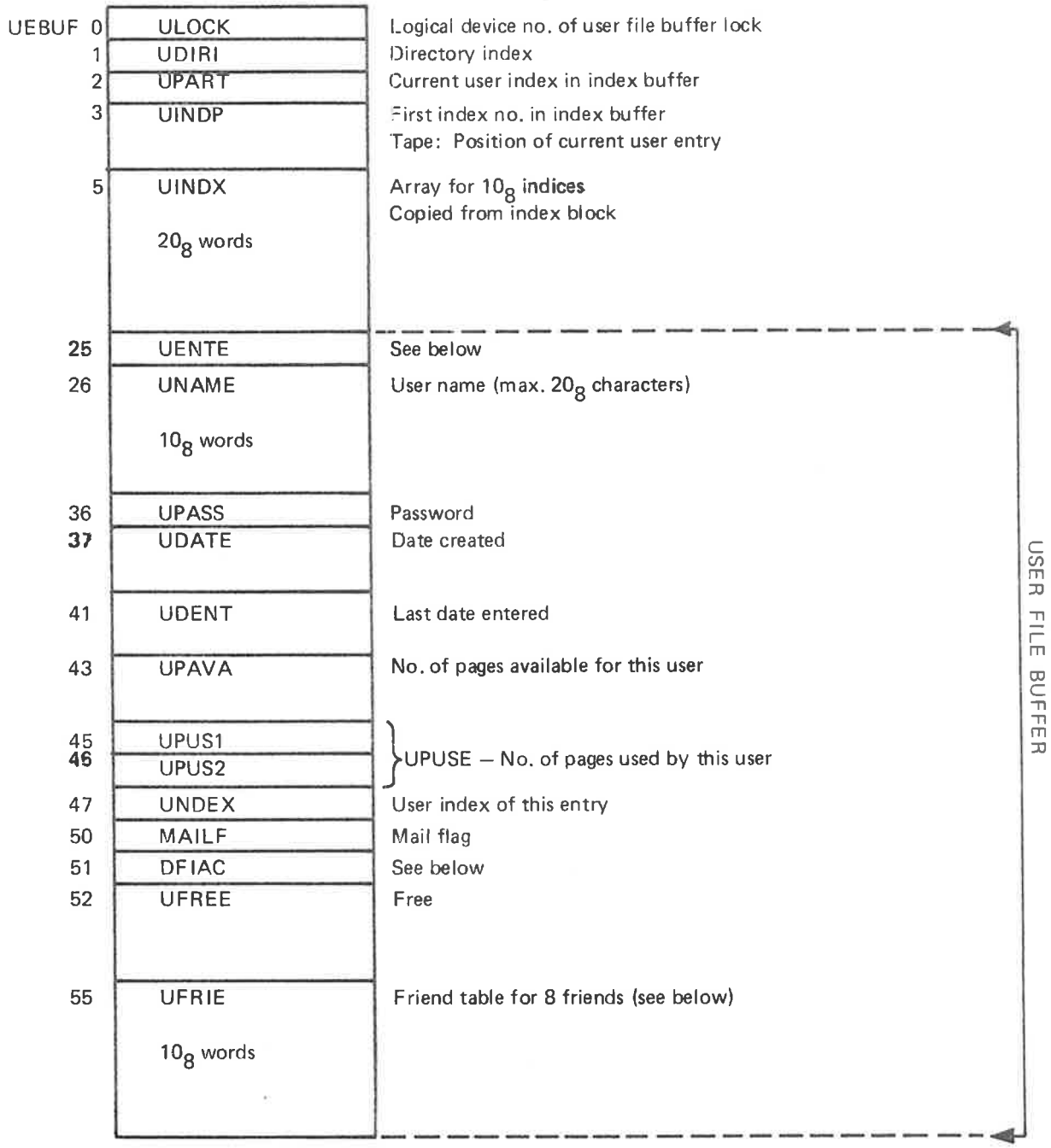
Figure 3.13: Directory Table Entry/Name Table Relationship

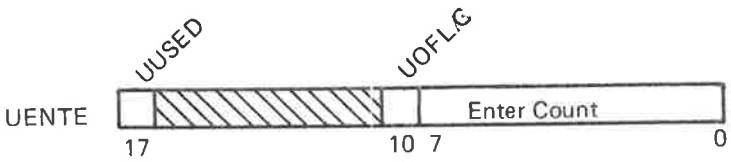
The broken line indicates an "implicit" pointer represented by an entry number used as index to locate the proper entry in the name table.

Operations in the directory table are protected to prevent simultaneous accesses to common data. All searches for a specific directory are protected through a general lock semaphore, GLDN. Once the desired directory entry is found, the directory lock semaphore, DLOCK, of this specific directory entry is reserved, before the geneal lock semaphore is released.

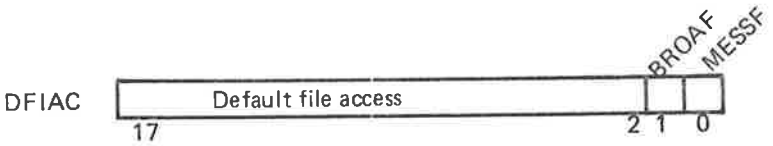
3.4.4 *User File Buffer*

The user file buffer resides in resident memory. It is preceded by a control information part related to the index block structure of the user file. The buffer area is used for one user file entry at a time. The size of the user file buffer and preceding control information is 65_8. It has the layout illustrated in Figure 3.14.



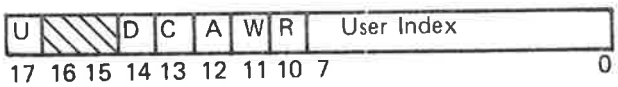


UUSED	—	entry in use flag
UOFLG	—	user object entry flag (1 = user entry)
ENTER COUNT	—	gives the number of times this user has been entered



BROAF	—	broadcast flag
MESSF	—	message flag

An entry in friend table has the following layout:



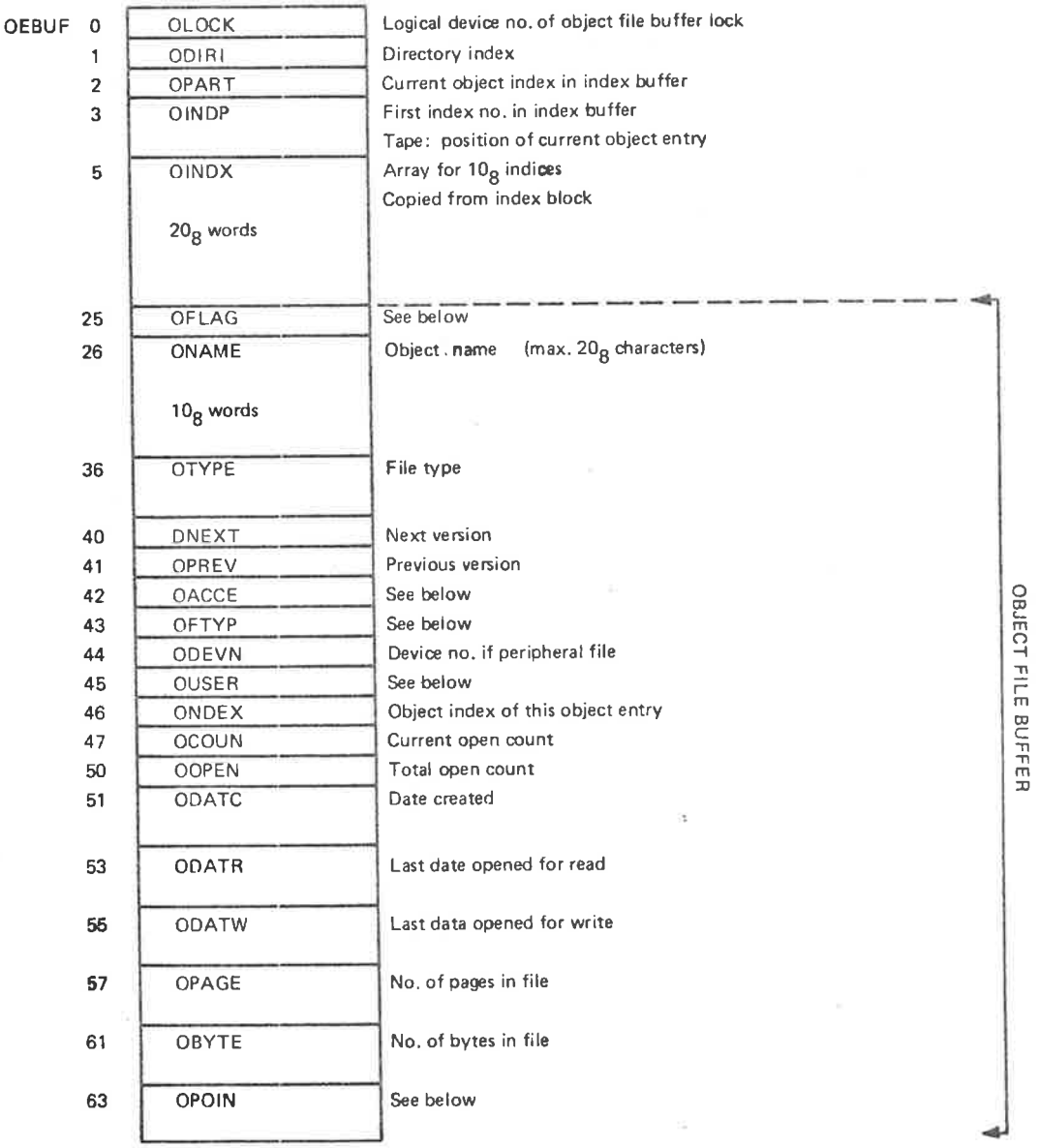
U	entry used
D	directory access
C	common access
A	append access
W	write access
R	read access

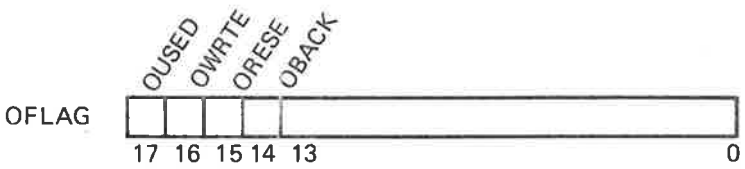
Figure 3.14: User File Buffer

Operations on a user file entry takes place when the entry resides in the user file buffer. The program operating on the buffer has reserved the user file buffer lock, ULOCK. The locations UDIRI and UPART identify the contents of the buffer. The array UINDX holds 10_8 indices (20_8 words) from the user file index block of the corresponding directory (see Figure 2.10). This, in fact, is the entire index information. Therefore, UINDP, which was supposed to identify which part of the index block that was present in UINDX, is redundant and is not used.

3.4.5 *Object File Buffer*

The object file buffer is organized in the same way as the user file buffer. It resides in resident memory. The layout is illustrated in Figure 3.15.

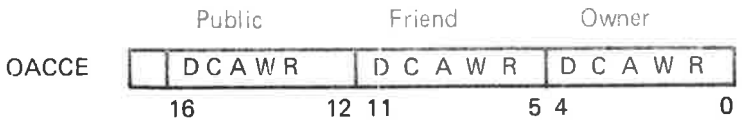




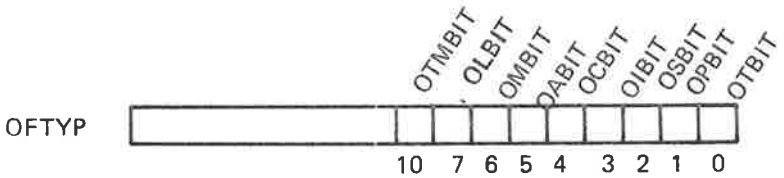
(continues)

(continued)

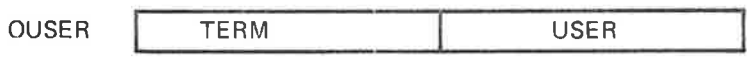
OUSED	—	entry used
OWRTE	—	opened for write at present
ORESE	—	reserved
OBACK	—	has been modified (originally aimed towards backup systems. Not in use at present)



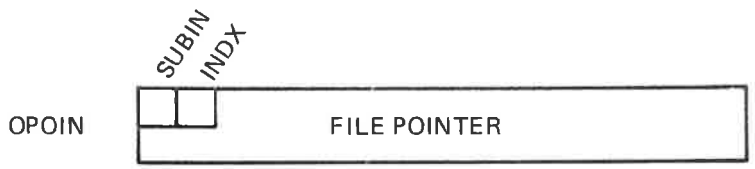
D	directory name
C	common access
A	append access
W	write access
R	read access



OTMBIT	—	temporary file
OLBIT	—	library file
OMBIT	—	magnetic tape file
OABIT	—	allocated file
OCBIT	—	contiguous file
OIBIT	—	index sequential file
OSBIT	—	spooling file
OPBIT	—	peripheral file
OTBIT	—	terminal file



TERM	—	terminal number of reserving user
USER	—	user index of reserving user



SUBIN	—	subindex pointer
INDX	—	index pointer

Figure 3.15: Object File Buffer

3.4.6 *Bit File Buffers*

There is one 20_8 words buffer for each disk or drum directory entry. The buffer will only hold the current part of the bit file. Each buffer is preceded by 3 words control information. The layout is illustrated in Figure 3.16.

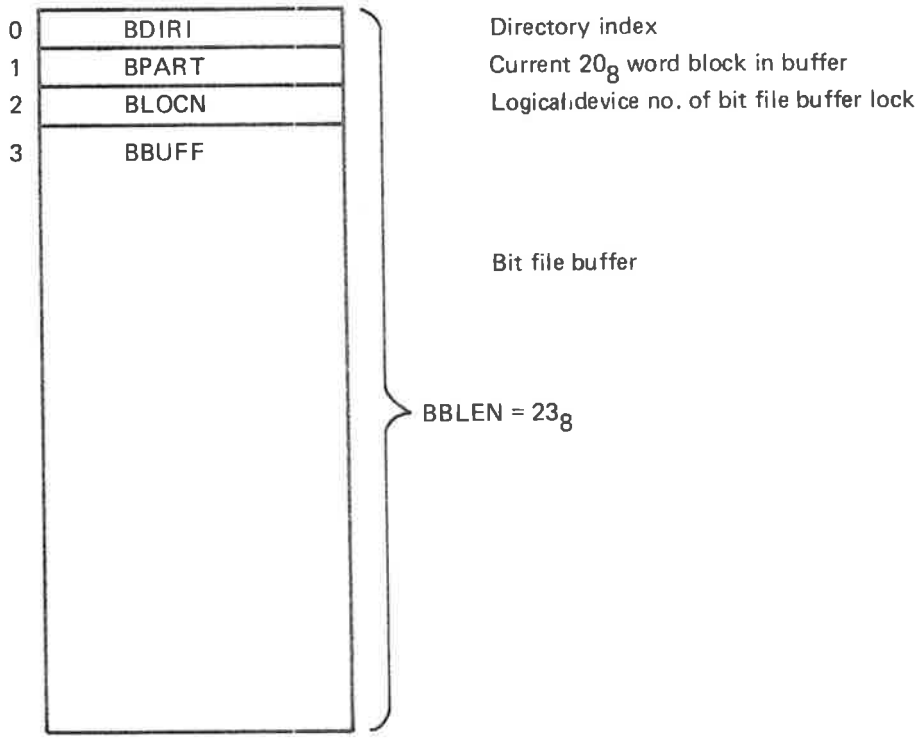


Figure 3.16: Bit File Buffer

The bit file buffers reside on the file system segment (segment 6) from location BFBUF to location ENDBF.

A bit file is split into logical blocks of 20_8 words. BDIRI and BPART identify the block being present in the buffer. Operations on the bit file are protected through reservation and release of the bit file buffer lock semaphore, BLOCN. To maintain a high degree of security, the file system attempts to keep directory structures consistent at all times. As part of this attempt, the file system will always write a bit file buffer back to the device as soon as possible whenever a change has taken place.

3.4.7 *System Segment*

The first part of every system segment is used by the file system for operations requested by the corresponding background program. Foreground programs do not have a system segment. Instead, all foreground programs share a file system data area in resident memory with the same layout as the system segments. The logical (and physical) address space of this area corresponds to the logical address space of the system segments. This has been done to allow similar operations for background and foreground programs. In the rest of this section we discuss the layout of system segments. The discussion also applies to the file system data area for foreground programs.

Each system segment has the layout illustrated in Figure 3.17.

70000	TDVN	List device no. (=1)
1	CUSER	Current user entered (= –1 initially)
2	USDI	User's default directory
3	USNO	User index in default directory
4	CRTREF	RTREF of calling program
5	OFLCK	Logical device no. of open file table lock
6	STACK 700_8 words	Stack used for data by routines in the file system
70706	ESTCK 7 words	Stack overflow area
70715	ASTCK	A & D registers saved by ENTER
70717	CSTCK	Current stack pointer (= STACK initially)
70720	SUBR SPUSH	Push routine for ENTER
70736	SUBR SPOP	Pop routine for LEAVE
70747	DV100	Max. no. of files simuntaneously opened
70750	OPTAB 40_8 words	Open file no. table Table to convert from file no. to address of corresponding file table entry. Used by routine LOGPH.
71010	OPSPO	Table for spooling entries
71014	SPOOL	Start of free list
71015	NPOOL	
71016	SDFLAG	
71017		Misc. monitor call routines for INBT and OUTBT
71145		(continues)

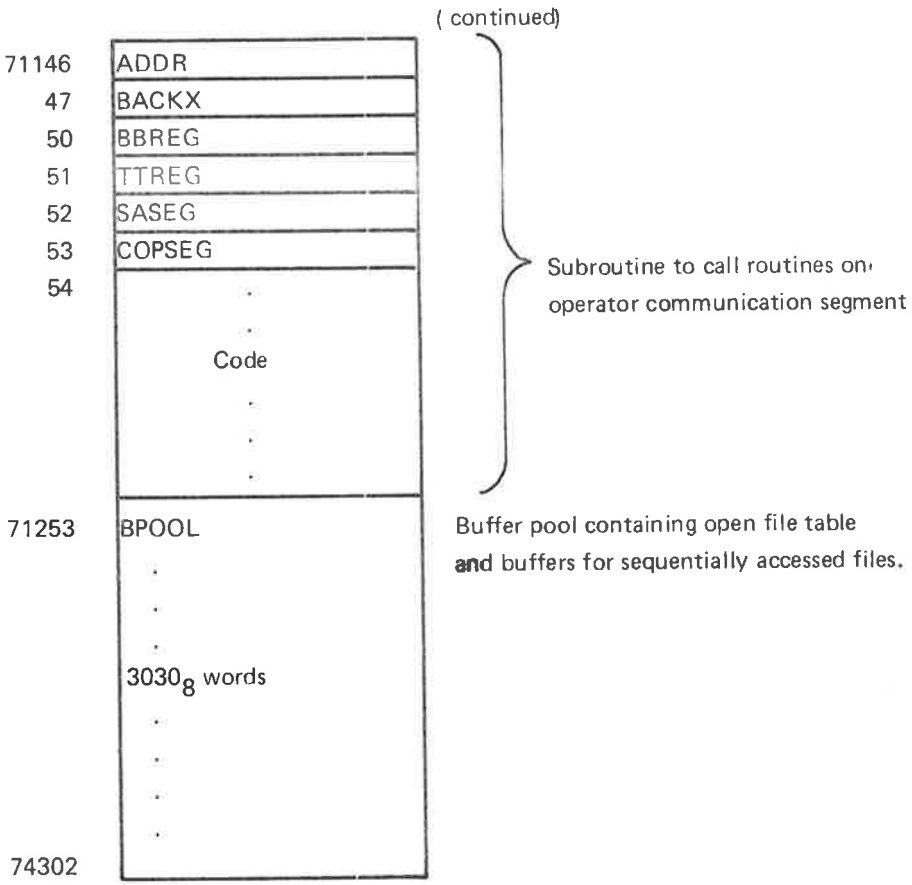


Figure 3.17: System Segment Layout

There must be at least one 64_{10} word buffer for each open file (see Section 3.4.8). These buffers are allocated from BPOOL. Each buffer is preceded by a link cell giving the layout of Figure 3.18.

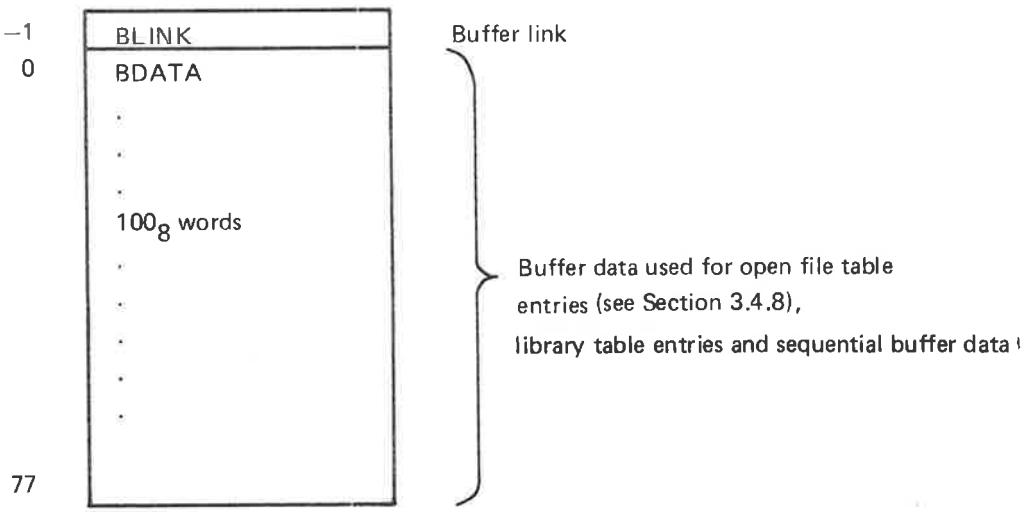


Figure 3.18: Buffer Layout

3.4.8 *Open File Tables*

Open file tables contain information on opened files. Open file table entries of background programs are allocated from the BPOOL area on the corresponding system segment (see Figure 3.17). Open file table entries of foreground programs are allocated from the resident file system data area.

Each opened file has an associated file number in the range 100_8 - 121_8. The open file number table, OPTAB, has a pair of entries for each file number. The first entry of the pair is used when a file is opened for input, while the second entry is used when a file is opened for output. Each entry contains the address of the corresponding open file table entry. OPTAB resides on the system segments (for background programs) and in the resident file system data area (for foreground programs). See Figure 3.19. The structure of OPTAB is similar to that of the logical number tables used by the I/O system. Therefore, the routine LOGPH is used for lookup in both tables.

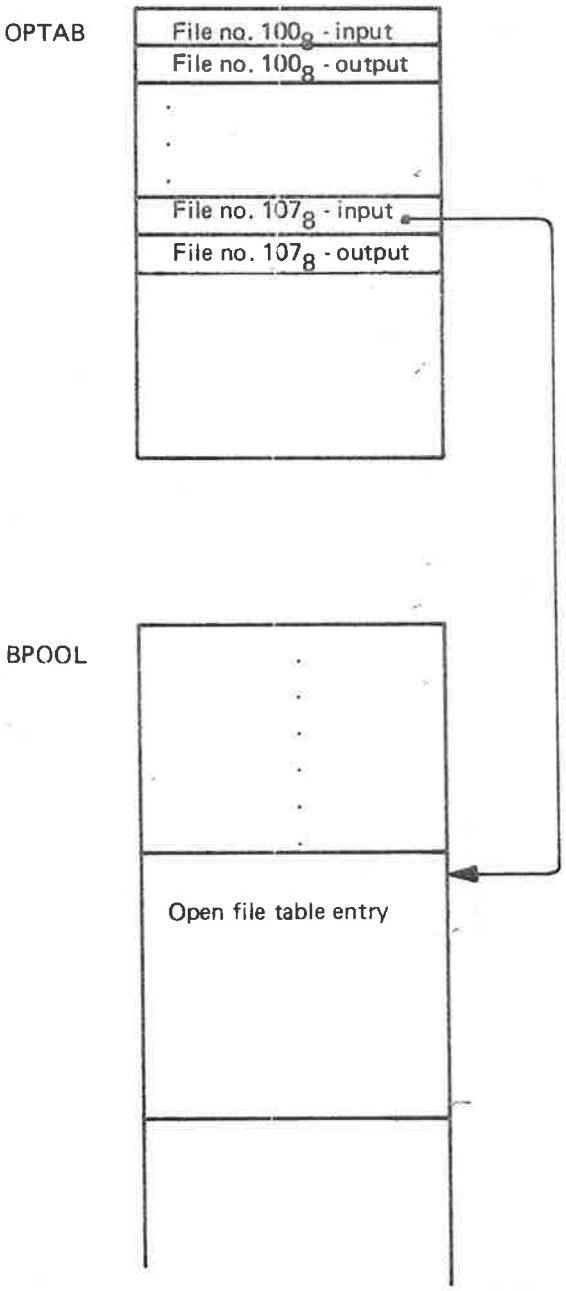
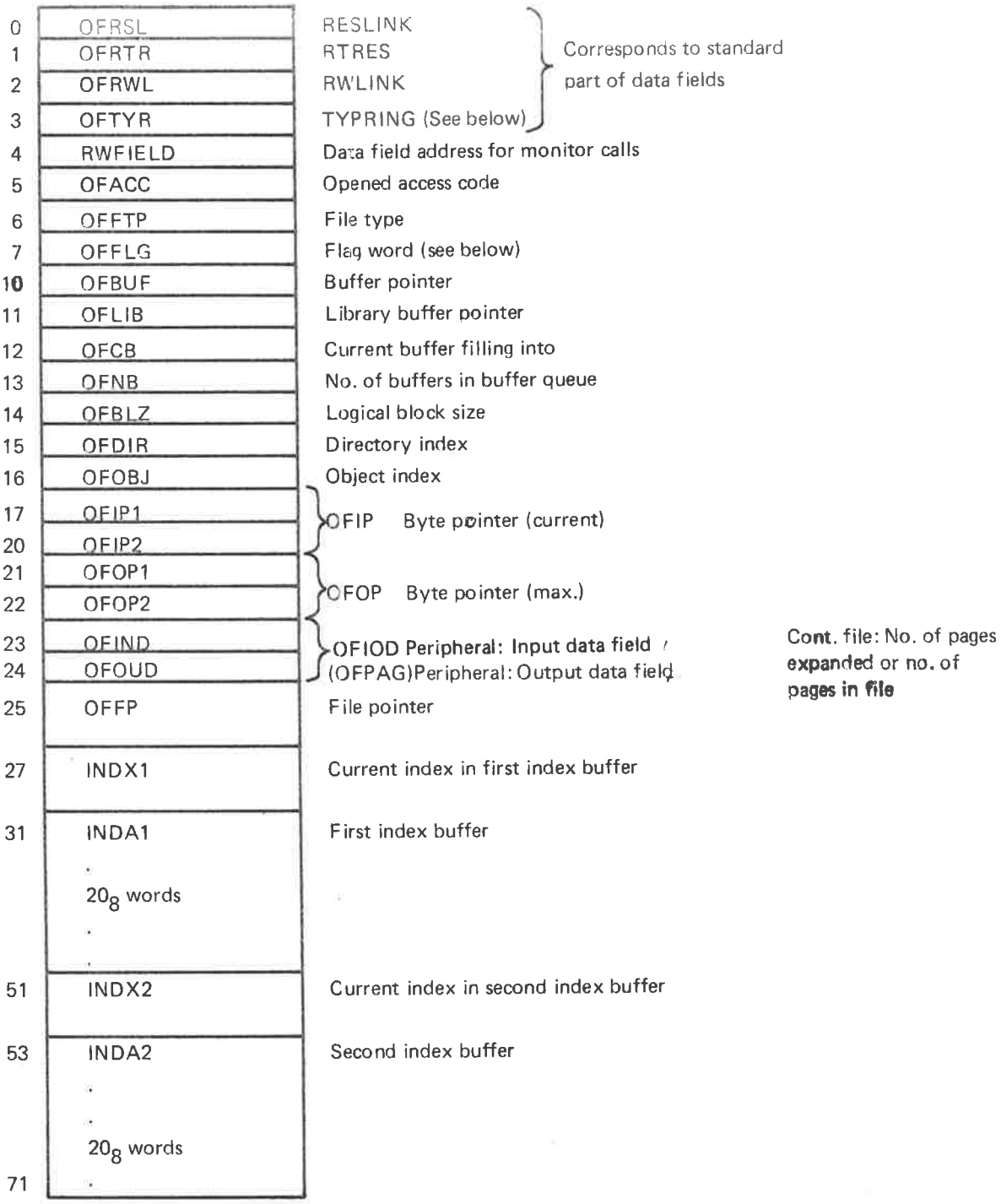


Figure 3.19: Correspondence between File Number and Open File Table Entry

The layout of an open file table entry is illustrated in Figure 3.20.

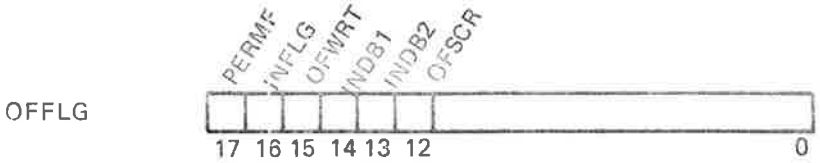
Since a file may be reserved, the first part of an open file table entry may be used to establish the entry as an element in a reservation queue. This explains the resemblance with the standard part of data fields.



(continues)

(continued)

OFIOB	—	opened for sequential access
OFRFI	—	mass storage file



PERMF	—	permanent opened file
INFLG	—	change index buffer flag 0 = first buffer last changed 1 = last buffer last changed
OFWRT	—	file opened for write
INDB1	—	write back index buffer one
INDB2	—	write back index buffer two
OFSCR	—	scratch file

Figure 3.20: Open File Table Entry

Open file table is allocated from BPOOL (declared in file system listing, Sections 1.6.3 and 1.2.8) at address 71253_8.

3.4.9 *Device Buffers*

Device buffers are used for random I/O. Each device buffer has room for one page (1K words). The minimum number of device buffers in a system is:

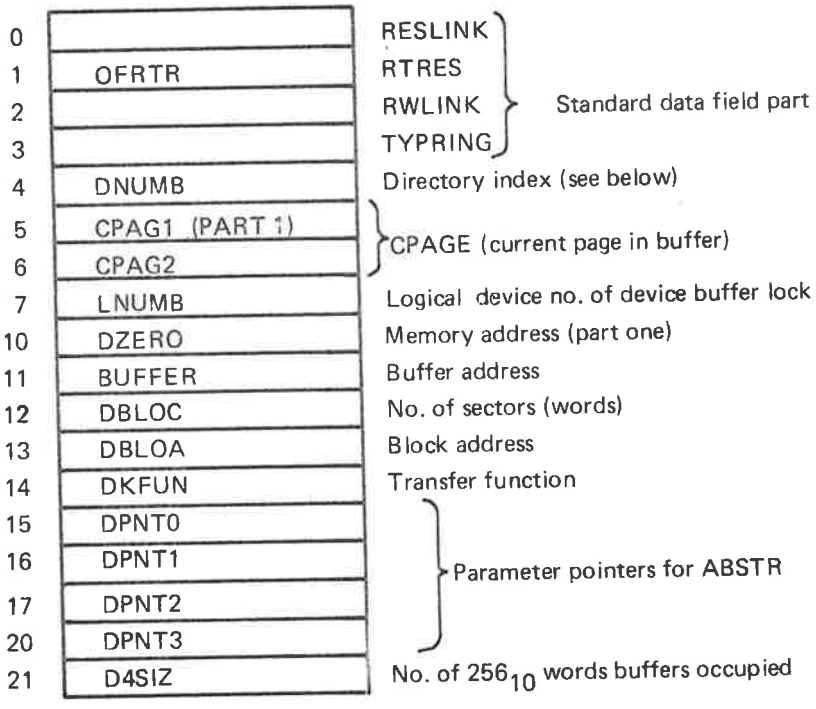
- one for each floppy unit
- one for each mag. tape unit
- one for each spooling device
- one shared among all disks

If additional device buffers are wanted, this must be specified through a SINTRAN generation parameter.

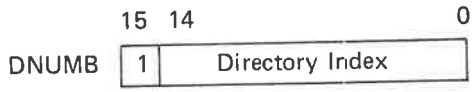
If a block oriented device is accessed sequentially, only a ¼K part of the buffer is used. This will be indicated in the device buffer header location DNUMB (see below), and applies to Versatec, mag. tape, floppy disk and spooling devices.

Each device buffer has a corresponding device buffer header. The header contains descriptive information identifying the contents of the buffer.

The layout of a device buffer header is as given in Figure 3.21.



DBLEN = 22_8



0 - 377: file system directory
400: only ¼K in use

Figure 3.21: Device Buffer Header

The device buffers resice in resident memory from DEVBU to ENDBU (declared in SINTRAN III listing, part 2, Section 29.10) .

Figure 3.22 illustrates some relations between data structures.

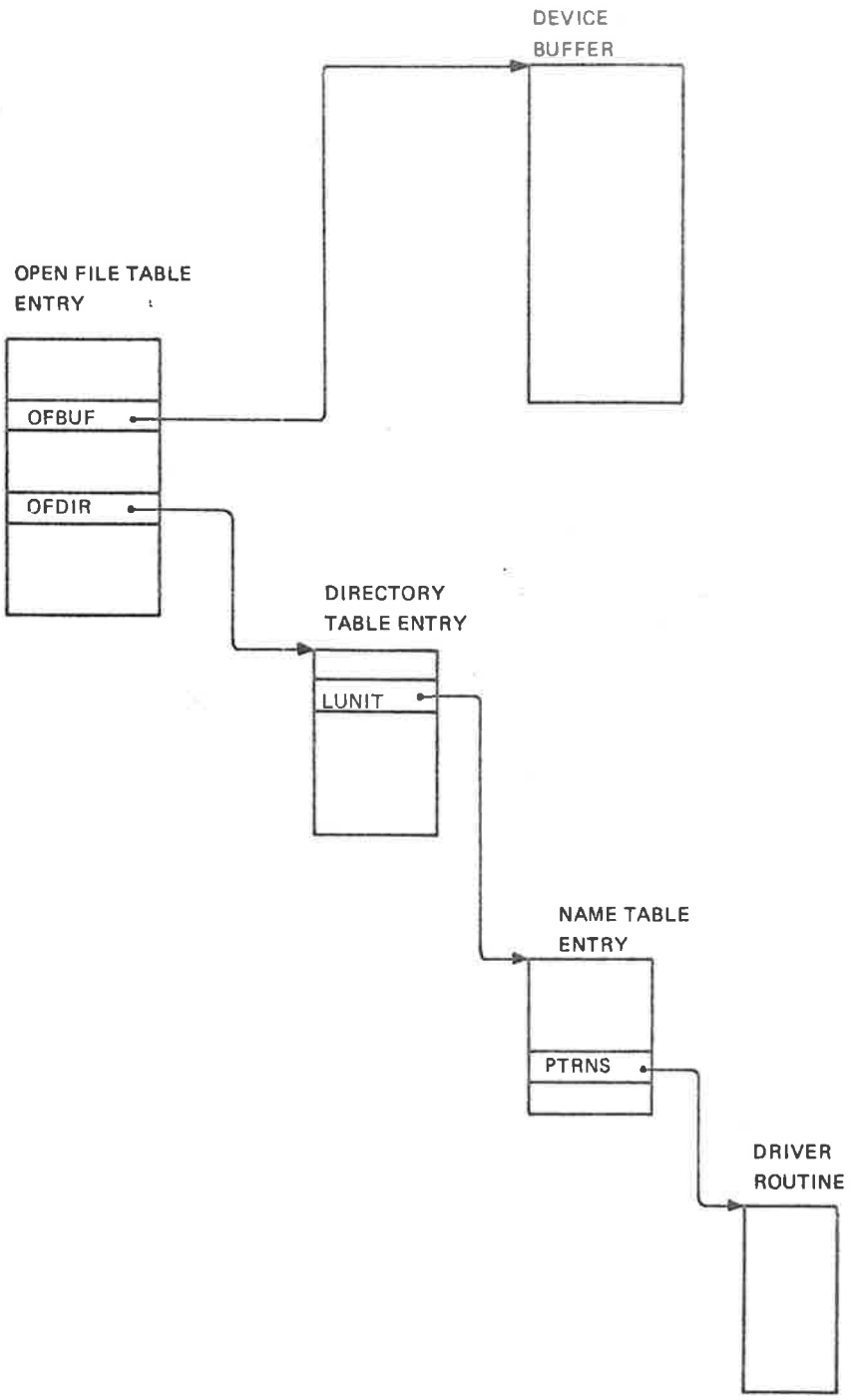


Figure 3.22: Some Relations between Data Structures

3.4.10 *File System Stack*

Each system segment has a stack (STACK) used for data by routines in the file system (see Figure 3.18). Foreground programs use a stack in resident memory. This allows several background programs and one foreground program to be inside the file system simultaneously. Figure 3.23 illustrates how the file system utilizes several stacks simultaneously.

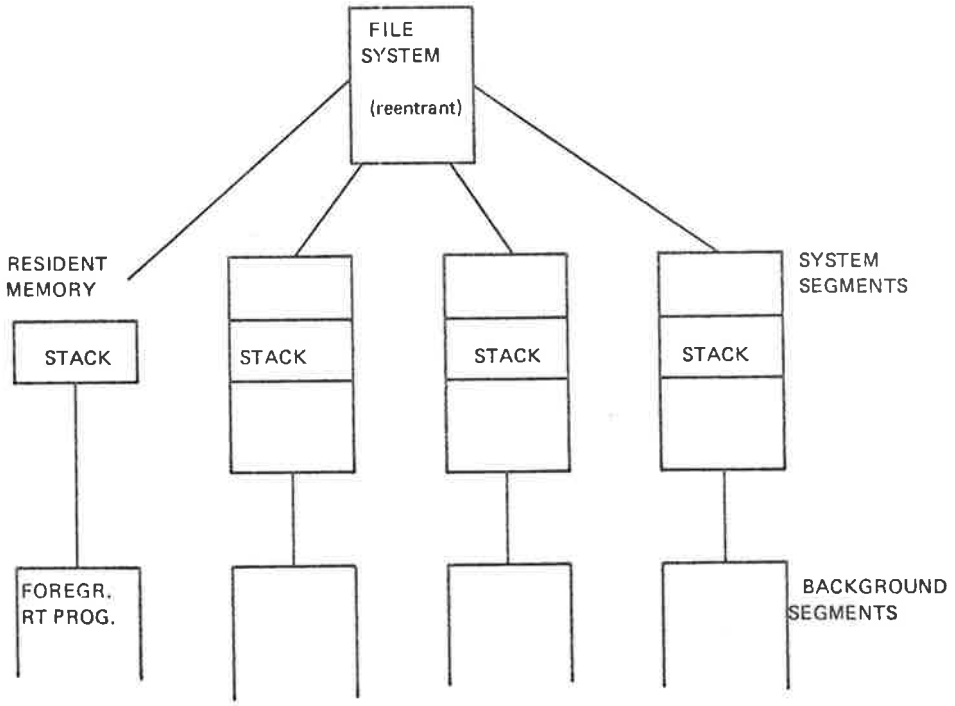


Figure 3.23: File System Stack Usage

Whenever a routine in the file system has been called, an entry in the active stack (i.e., the stack on the system segment of the active background program or the stack used for foreground RT programs) is allocated. The size of the entry varies depending on the called routine. The area is released prior to return from the routine.

The administration of the file system stack is performed by two sequences of instructions enclosing all routines. These sequences (macro expansions) are called ENTER and LEAVE, respectively. See Figure 3.24.

```
ENTER          % allocate stack entry
.
.
.              % routine code
.
.
LEAVE          % release stack entry
```

Figure 3.24: File System Routine Organization

The actual operations on the stack are performed in the routines SPUSH (allocate) and SPOP (release). These routines are called from ENTER and LEAVE, respectively. SPUSH and SPOP operate on the current stack pointer, CSTCK, which always points to the first free location in the stack. See Figure 3.25.

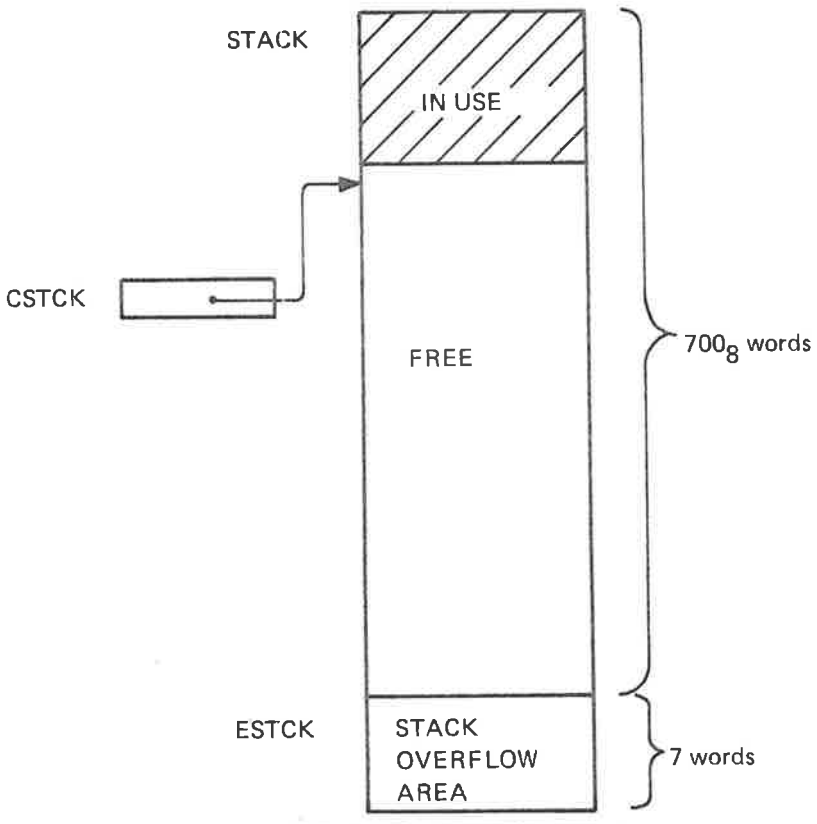


Figure 3.25: Stack Organization

The overflow area is used as stack entry for the error routine in case of stack overflow.

A stack entry consists of two parts: a 6 word register save area and a variable length data area. The layout is given in Figure 3.26.

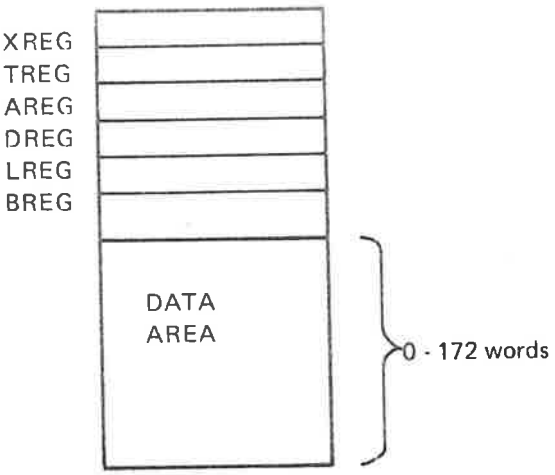


Figure 3.26: Stack Entry

ENTER and LEAVE flow charts are illustrated in Figure 3.27 and 3.28, respectively.

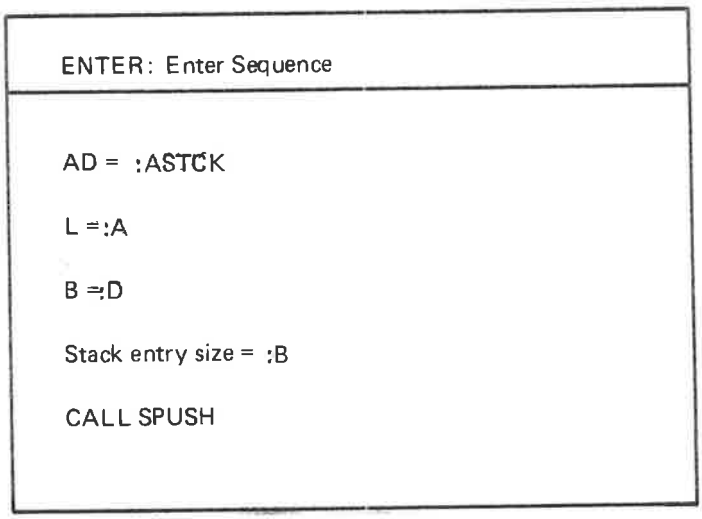


Figure 3.27: ENTER

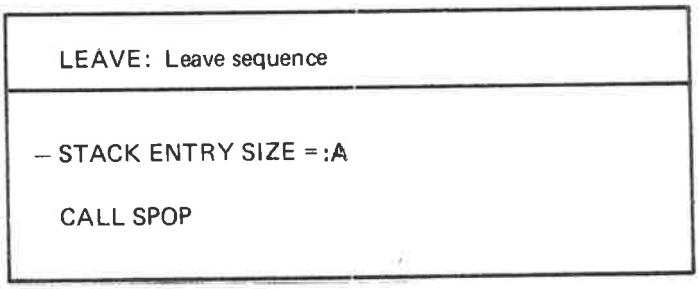


Figure 3.28: LEAVE

The routines SPUSH and SPOP, are described in flow charts in Figure 3.29 and 3.30, respectively. The state of the stack before and after ENTER is illustrated in Figure 3.31, while Figure 3.32 shows the state before and after LEAVE.

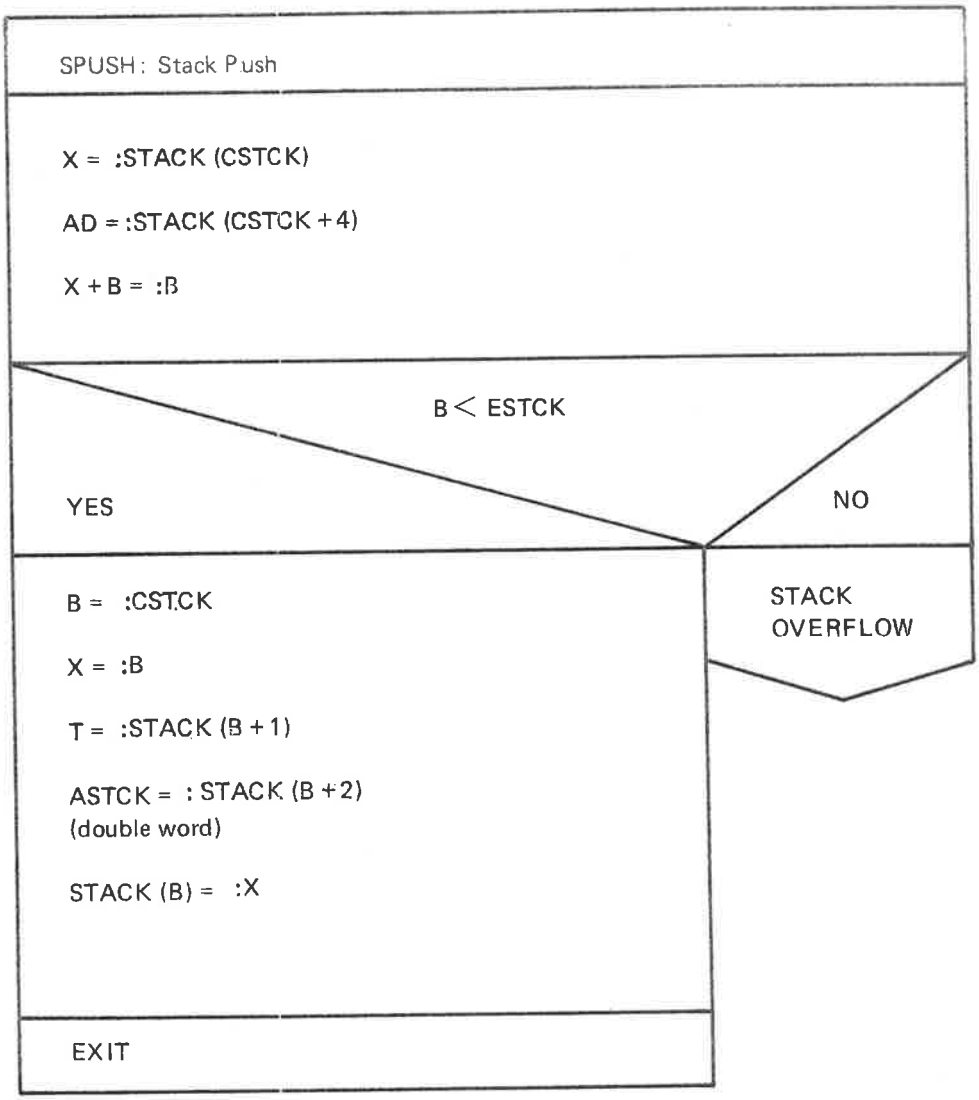


Figure 3.29: SPUSH

SPOP : Stack pop

```
CSTCK + A = :CSTCK

B = :X

STACK (CSTCK + 4) =:L
STACK (CSTCK + 5) =:B
STACK (CSTCK + 1) =:T
STACK (CSTCK + 2) = :A
STACK (CSTCK + 3) = :D
STACK (CSTCK) = :X
```

EXIT

Figure 3.30: SPOP

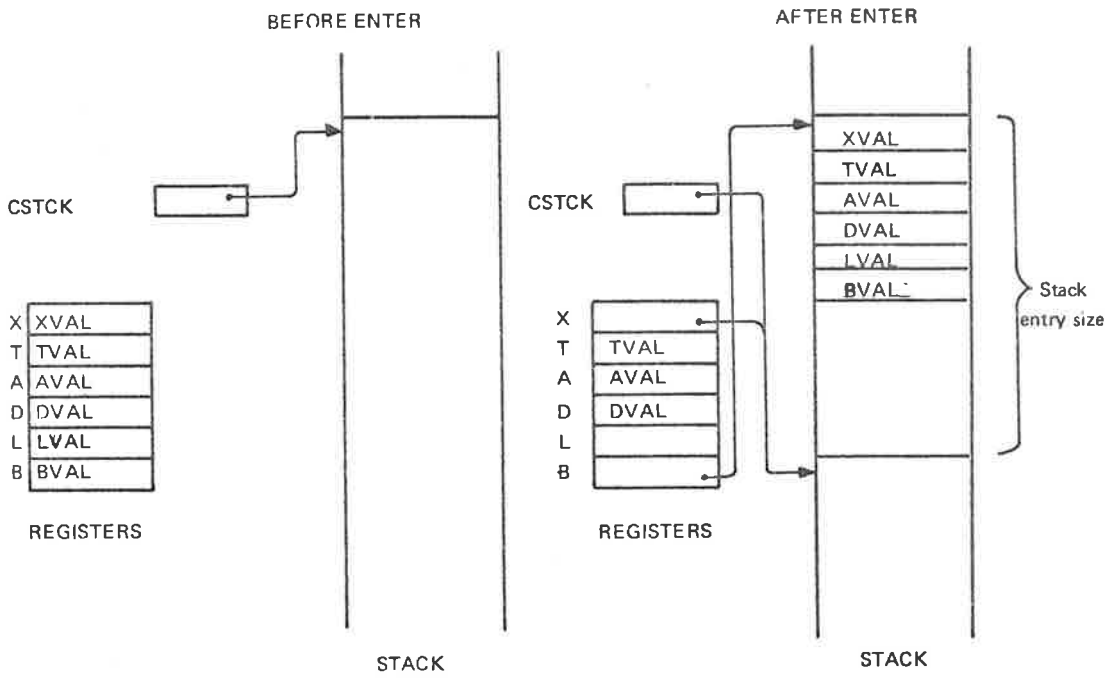


Figure 3.31: Stack State before and after ENTER

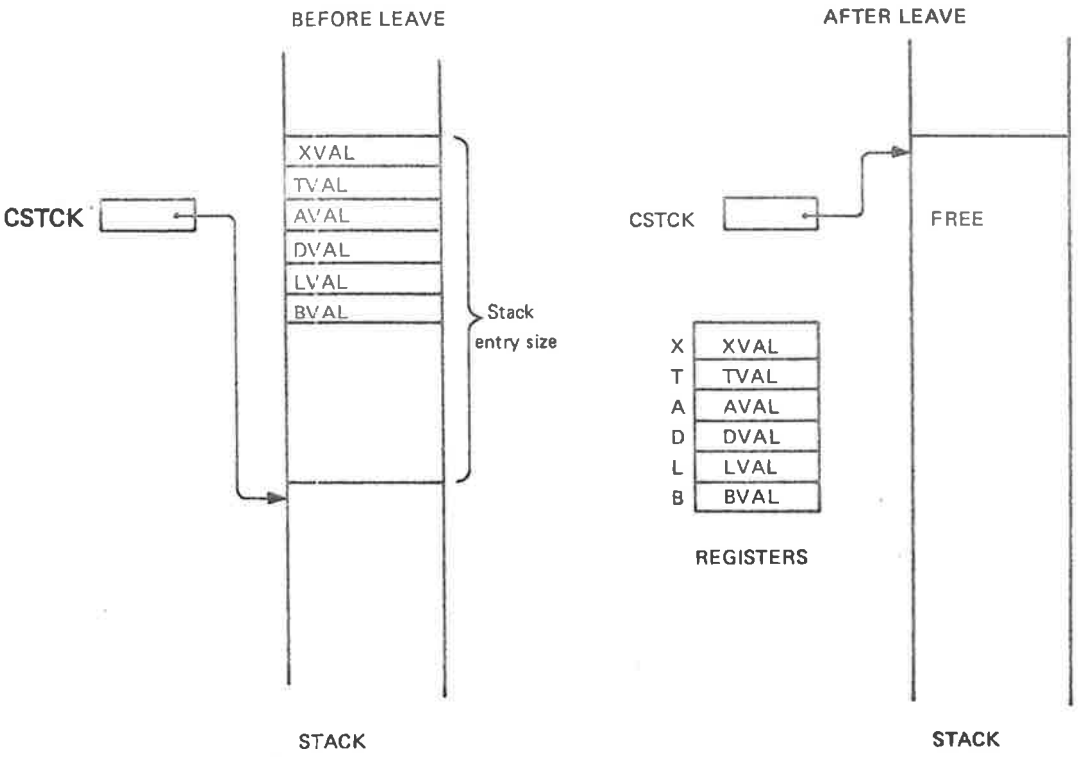


Figure 3.32: Stack State before and after LEAVE

Due to the systematic stack technique used in the file system, it is possible to read the dynamic routine call structure out of the stack.

The LREG location in a stack entry always gives the return address from the corresponding routine. The BREG location in a stack entry always gives the address of the previous stack entry. See Figure 3.33.

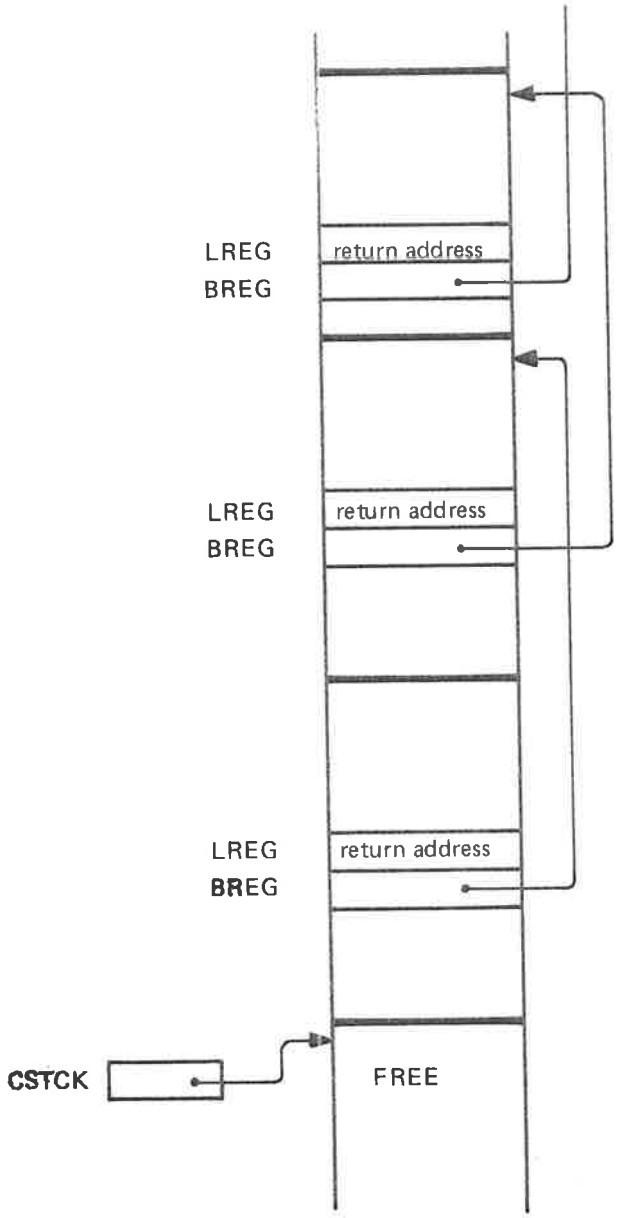


Figure 3.33: Stack Showing Dynamic Routine Call Structure

3.4.11 *File System Error Handling*

The File System is organized as a set of routines. For a given operation a certain calling sequence will be performed. With a few exceptions all routine calls have two return points: an error return and a normal return. The error return is the instruction following the call, while the normal return is the second instruction following the call, therefore referred to as the *skip return.*

Example from the FOPEN routine (9.5):

```
      .
      .
CALL FCON
GO ERET          % error return
IF ....          % normal (skip) return
      .
      .
```

The called routine (FCON in the example above) has the responsibility of returning to the proper address. The LREG location in the stack entry (see Figure 3.33) will, as a result of the SPUSH routine (part of the ENTER macro), contain the address following the call, i.e., the error return address. LREG is used by the SPOP routine (part of the LEAVE macro) when a routine wants to return to the caller. When skip return is desired, the LREG locatin must be incremented by 1 prior to execution of SPOP. Therefore, the final part of routines usually read:

```
MIN LREG         % increment LREG
      .
      .
      .
*LEAVE           % return
```

The structured error handling design described above, enables the file system to report an error situation upward in the call hierarchy. An error detected in a routine at any level can be reported all the way up to the top before it is communicated to the user. For file system commands the top of the hierarchy is represented by the routine CMMON on the system segment (see Section 3.3.2). This routine also has an error return and a normal return. The error return part will call an error routine which will communicate the error message to the user's terminal. For monitor calls the top of the hierarchy is represented by the COMENTRY routine on the system segment for background programs, and by the COMMON routine called form the file transfer RT programs (part of the I/O system) for foreground programs (see Section 3.3.1). These routines also have an error return and a normal return. In the normal return part the P register of the calling program will be incremented by 1, in the error return part it will not. Thus, the error is reported over to the calling program where a corresponding error return/normal return technique may be used to take care of file system errors. The structure of the routines CMMON and COMENTRY is shown below.

```
CMMON:

      .
      .
JPL   0, X          % Call some file system routine
CALL ERROR          % Error return
*LEAVE              % Normal return

COMENTRY:

      .
      .
CALL MRSTA          % Call some file system routine (MRSTA is a location in a
                    % working data field)

GO    ERET          % Error return
MIN ZPREG           % Normal return
      .             % (Increment P register of calling program)
```

The routine COMMON has a structure similar to that of COMENTRY.

3.5 *FILE SYSTEM COMMANDS*

3.5.1 *Parameter Collection*

When a user issues a file system command, the appropriate routine on the file system segment is entered (see Section 3.3.2). The parameters to the command will then be collected by calls to the file system routine CLPAR. CLPAR will be called once for each parameter. CLPAR will then call the routine OPCAL on the system segment. OPCAL exchanges the file system segment with the command segment and calls the routine GLPAR on the command segment. GLPAR is the general parameter collect routine used for all commands. When control is returned to OPCAL, the original (file system) segment is brought back (exchanged with the command segment) and control is then returned to the routine CLPAR.

Figure 3.34 illustrates the sequence of operations involved in parameter collecting. Note that OPCAL is a general routine taking one parameter, which is the routine to be called (GLPAR in this case).

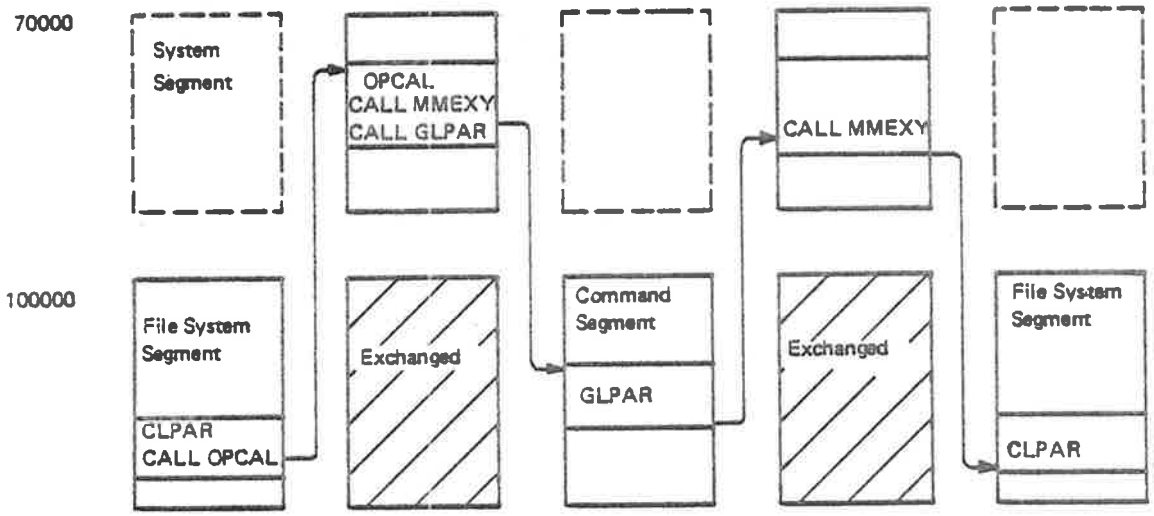
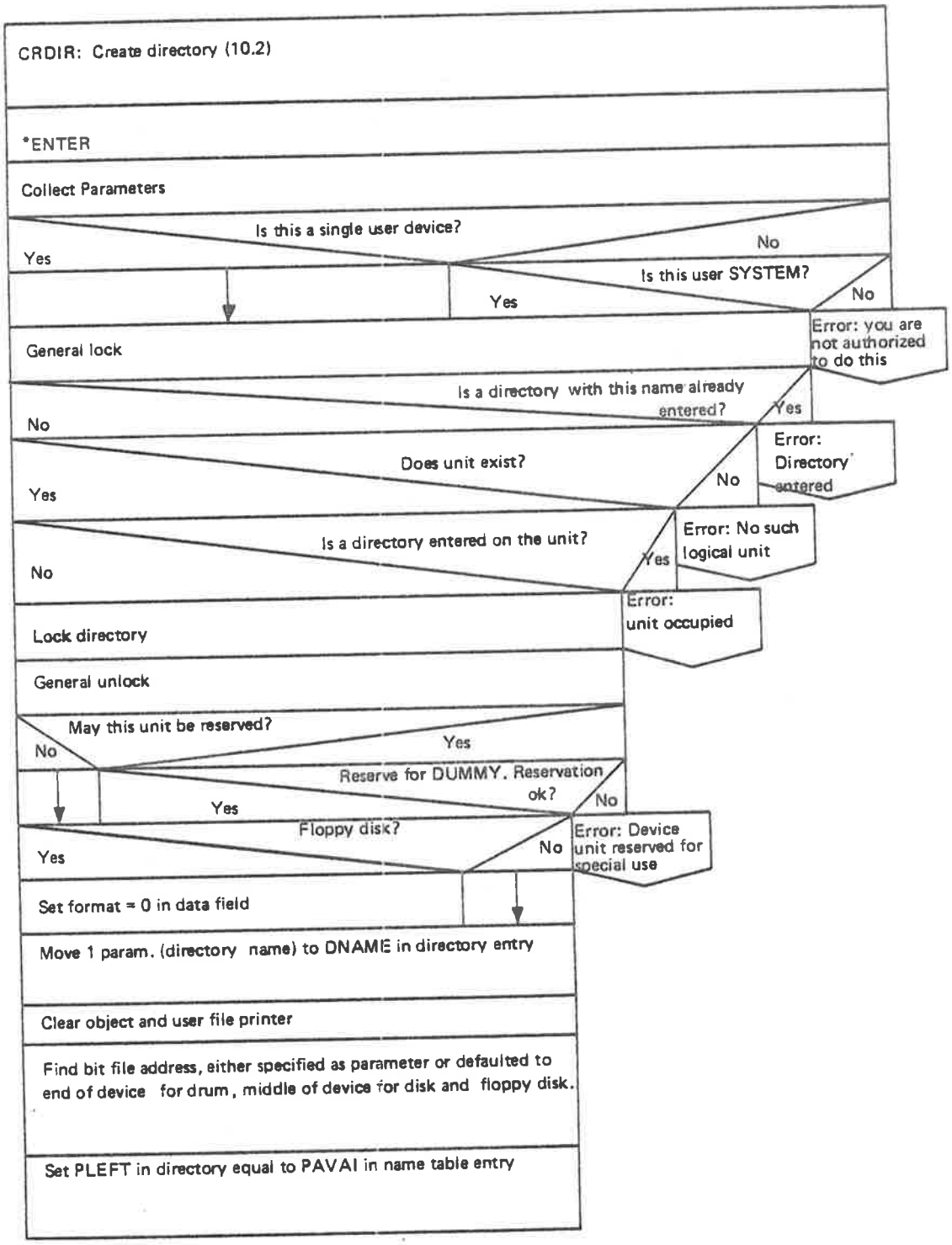


Figure 3.34: Parameter Collection

3.5.2 *Create Directory*

Below is a flow diagram of the create directory routine. (The number in parentheses refers to the section number in the file system listing.)



(continues)

(continued)

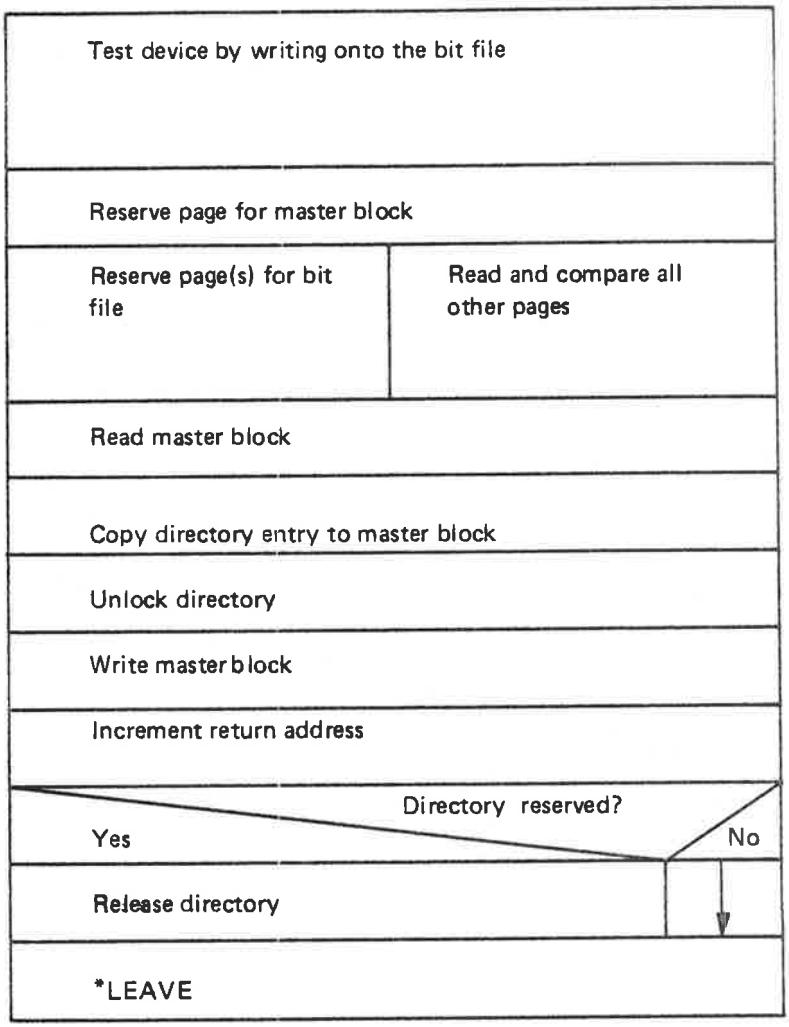
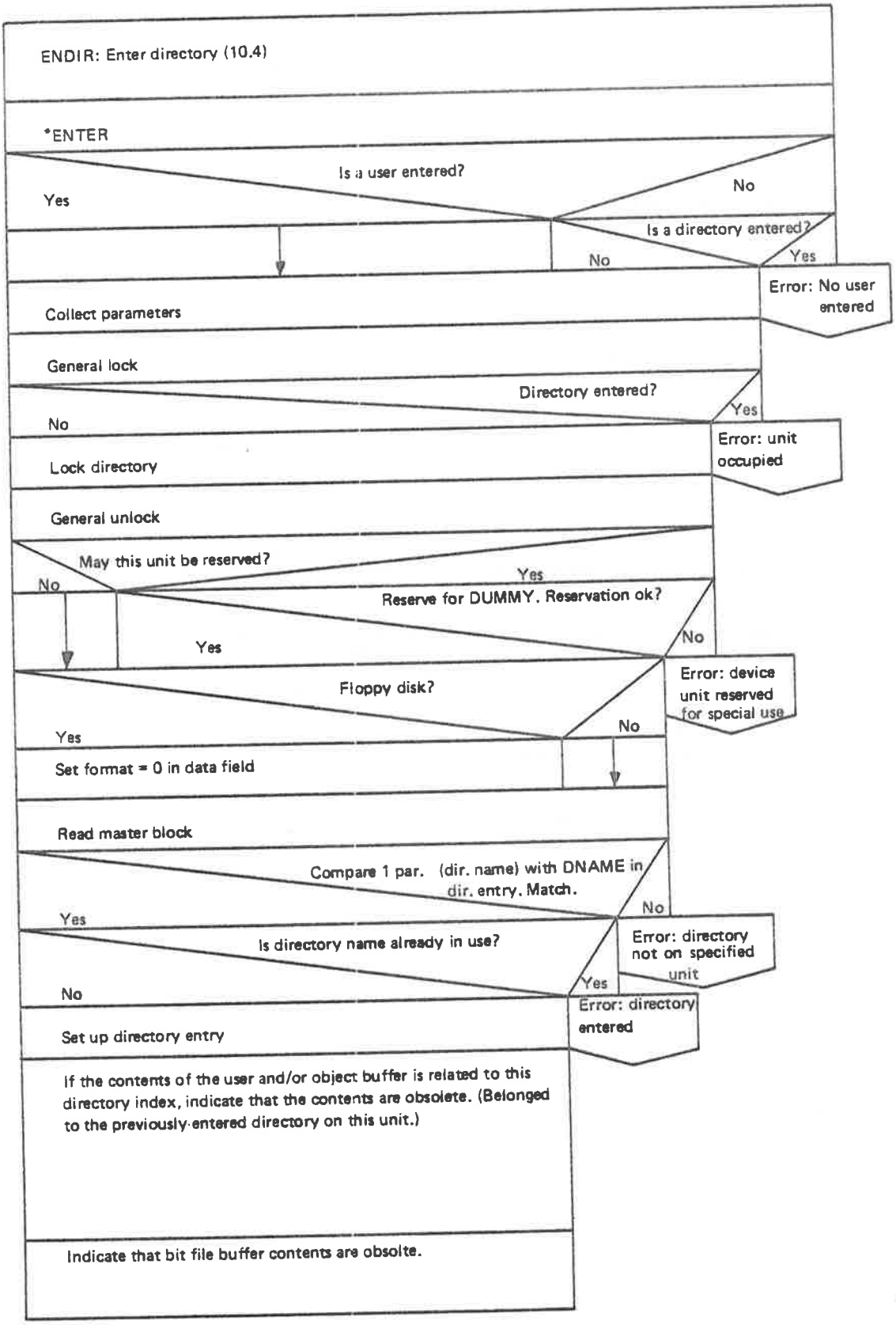


Figure 3.35: Create Directory

3.5.3 *Enter Directory*



(continues)

(continued)

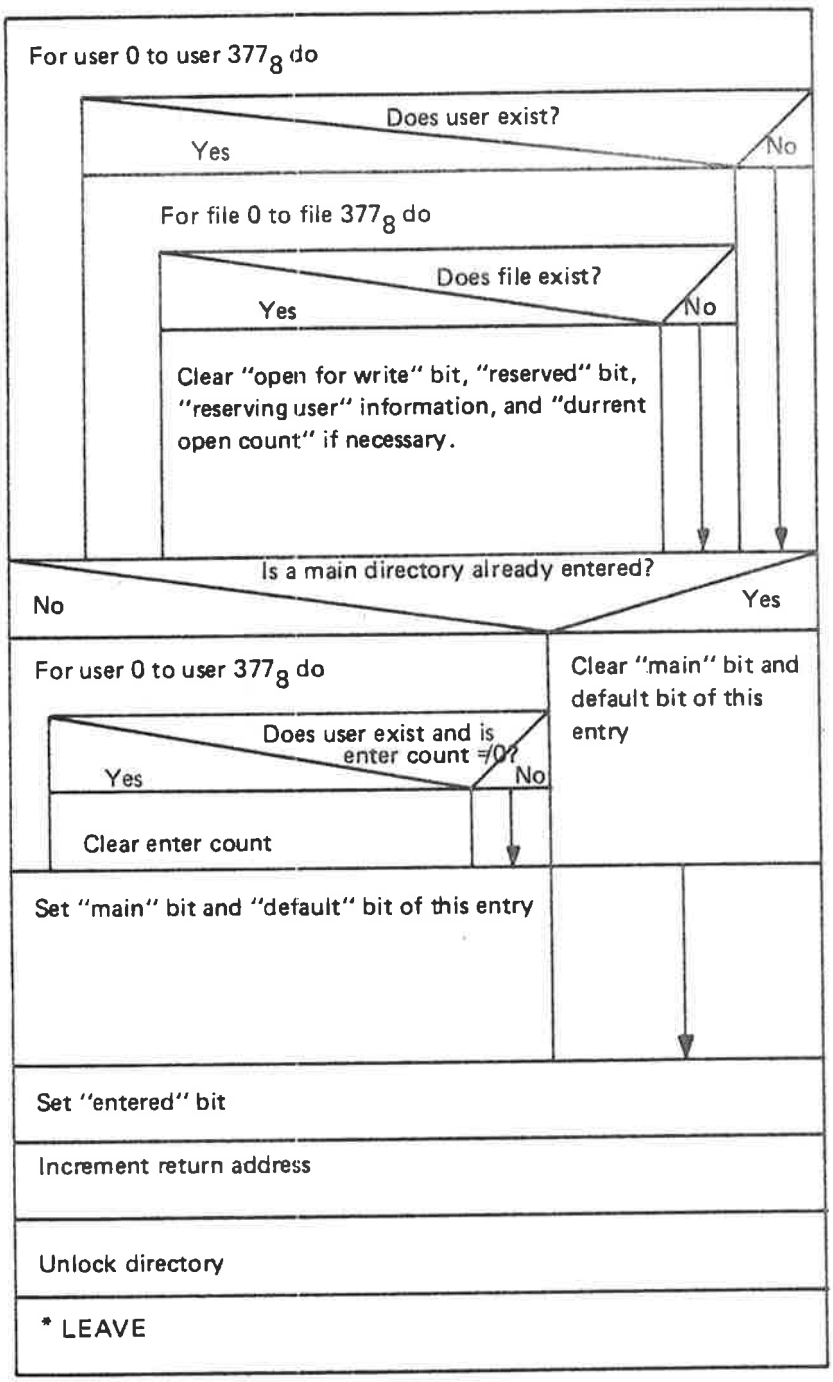


Figure 3.36: Enter Directory

3.5.4 *Create User*

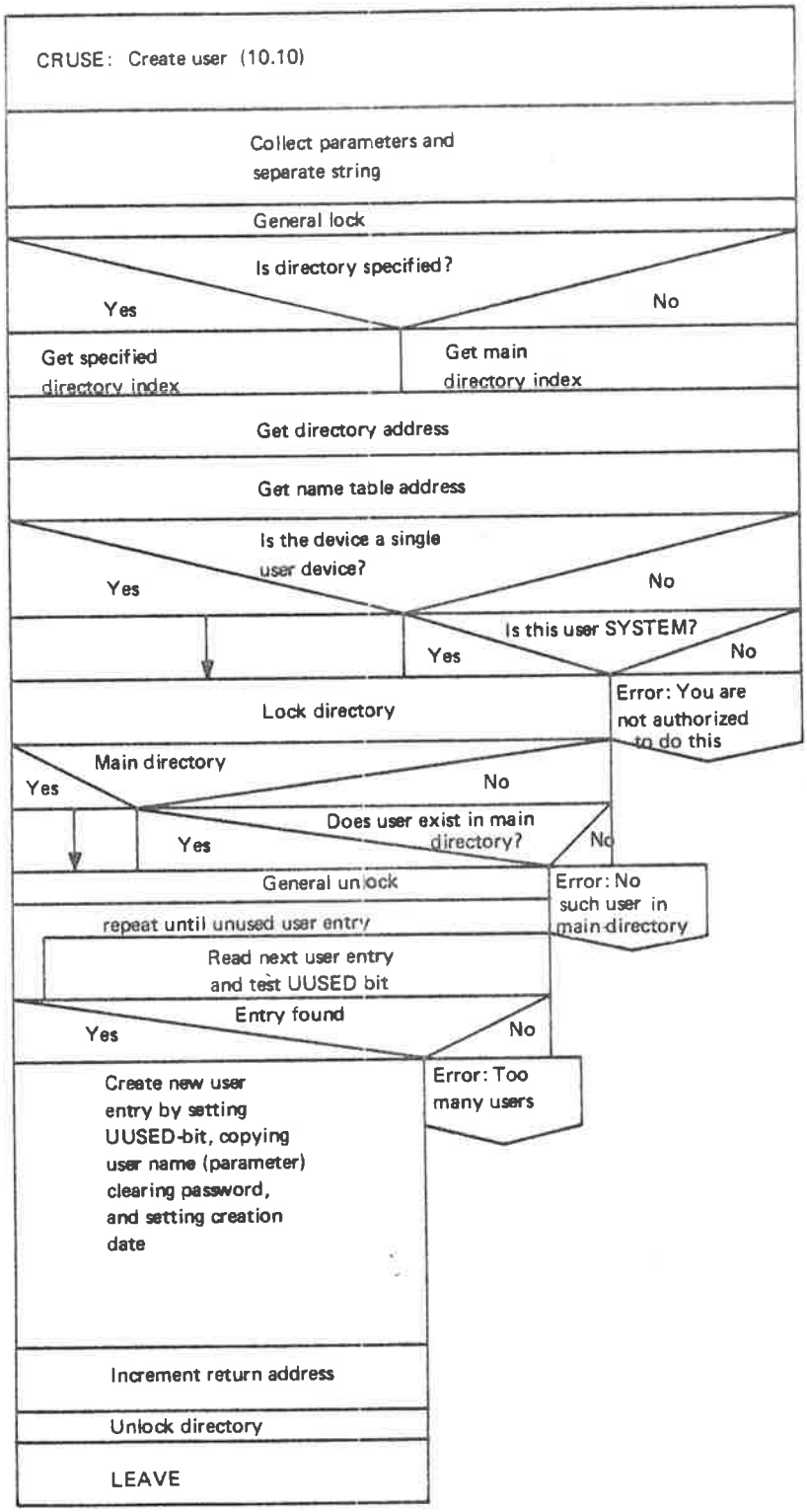


Figure 3.37: Create User

3.5.5 *Delete User*

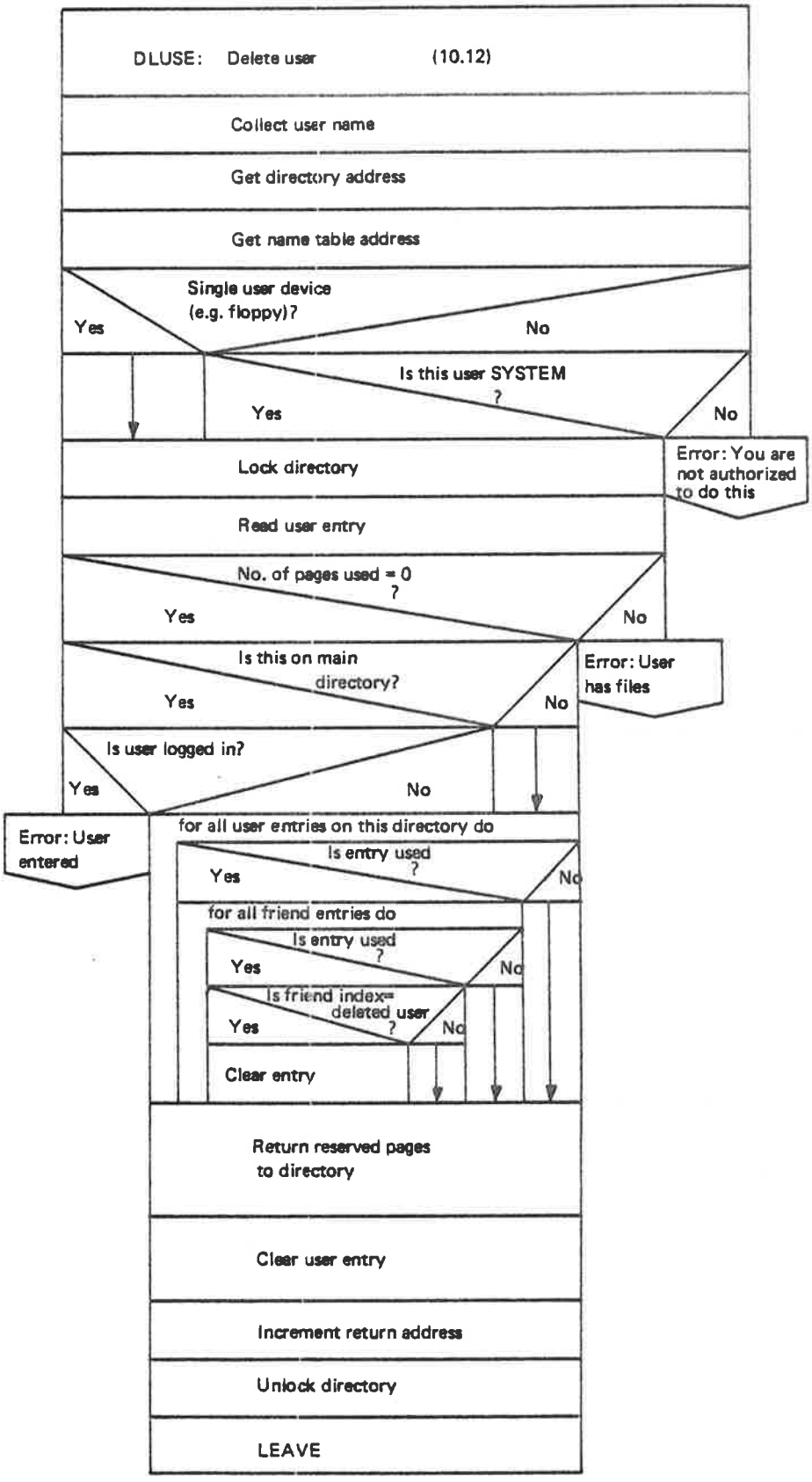


Figure 3.38: Delete User

3.5.6 *Create File*

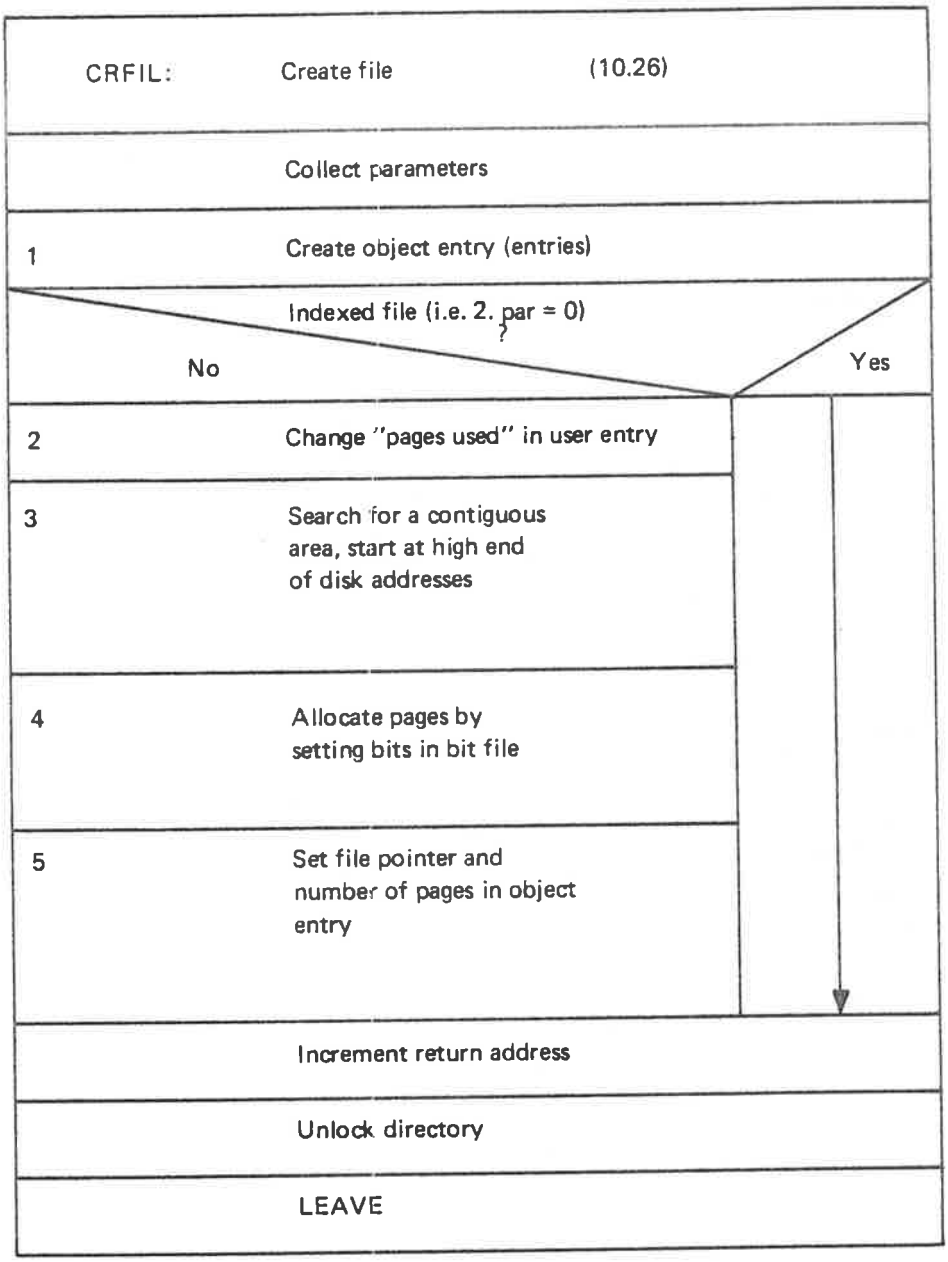


Figure 3.39: Create File

If a file is to be created in several versions or if the command CREATE-NEW-VERSIONS is issued, the actions in the boxes numbered 2, 3, 4 and 5 will be repeated for each version. Also, the box numbered 1 will create one object entry for each version.

3.5.7 *Delete File*

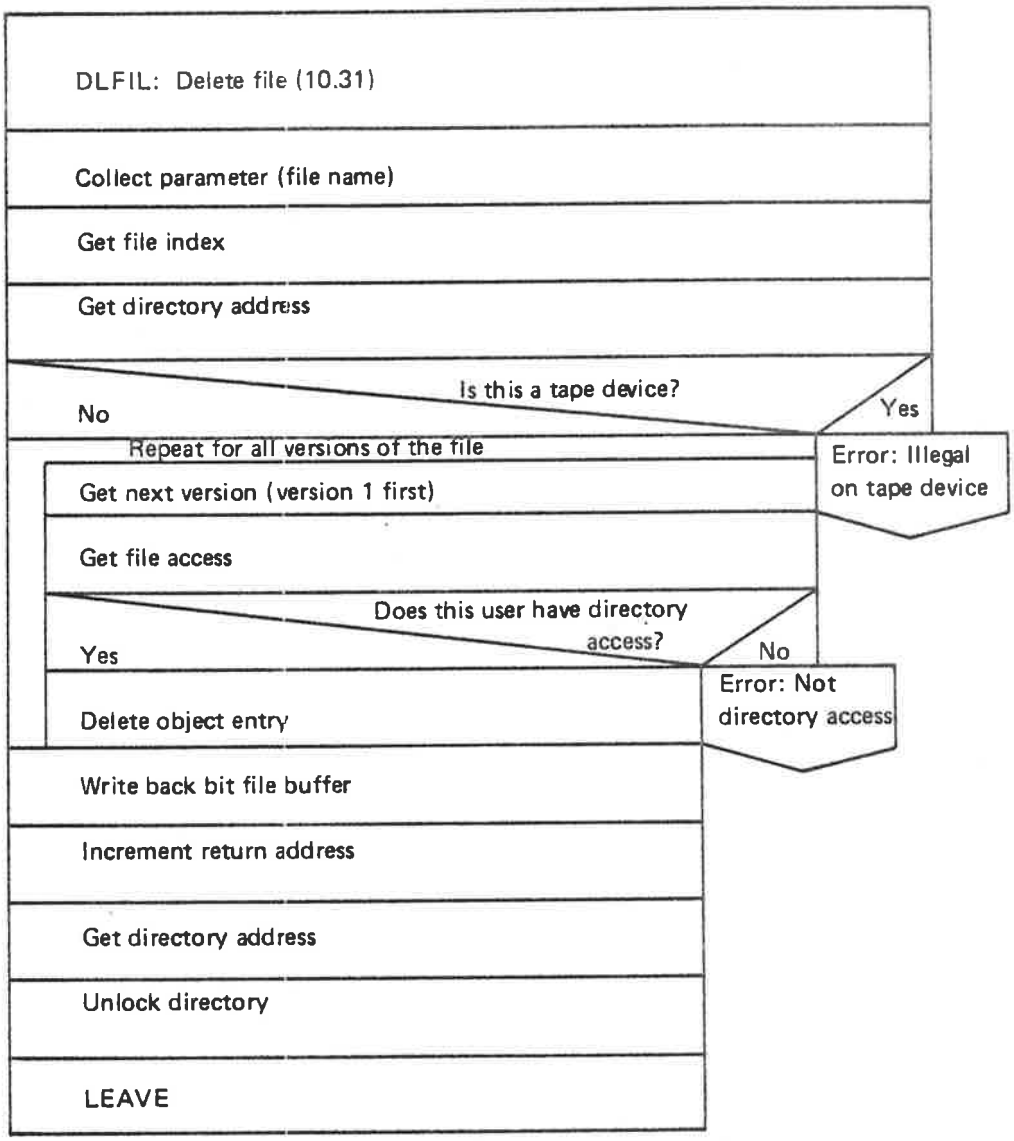


Figure 3.40: Delete File

3.5.8 *Open File*

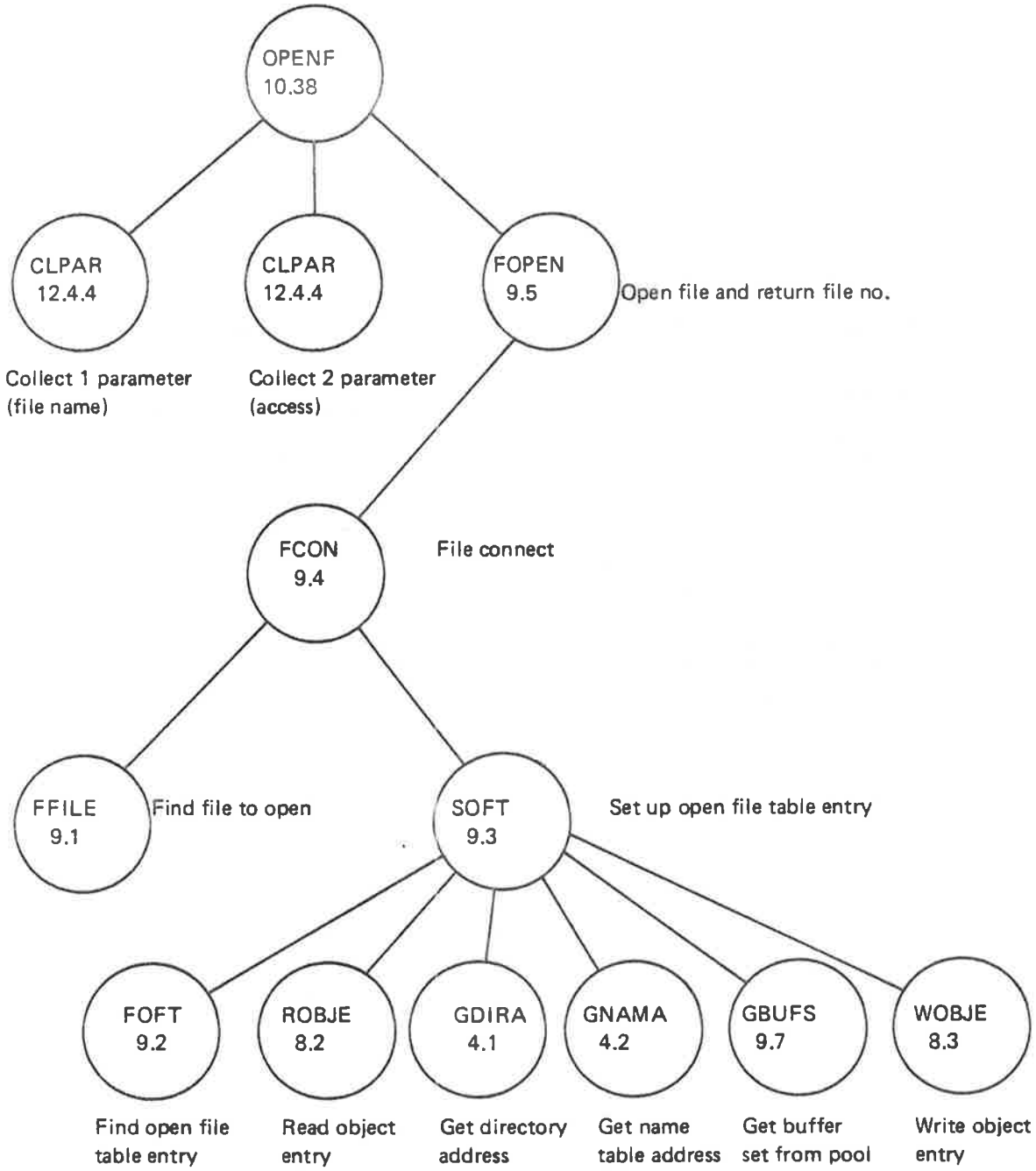


Figure 3.41: Open file command — Call Hierarchy

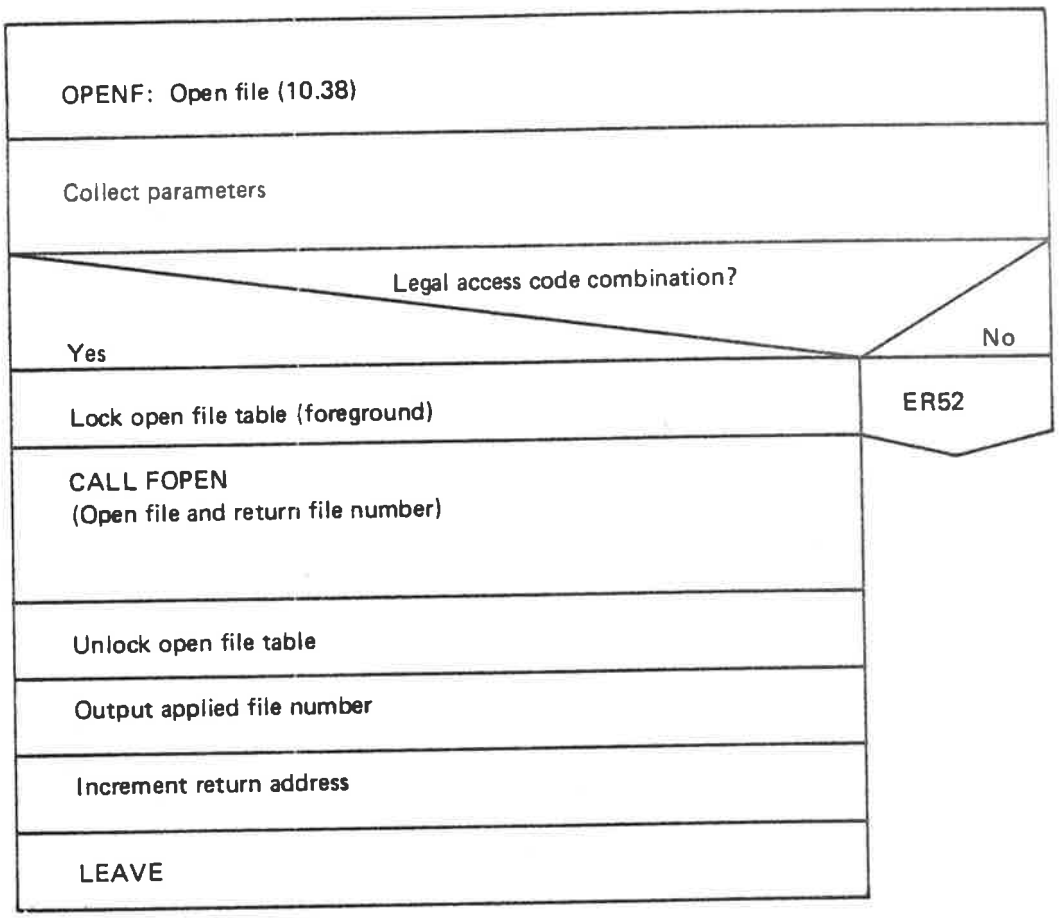


Figure 3.42: Open File Routine — Flow Diagram

The last parameter to the open file command, the access type parameter, is returned (from GLPAR) as a function value in the A register. The access type is coded in the 6 rightmost bits as follows:

All legal combinations form a set of values. The table below shows the legal combinations, the corresponding values returned in the A register, and the internal access codes used by the file system.

Parameter	*Value returned from GLPAR*	*Internal access code*
W	2	0
R	1	1
WX	22	2
RX	21	3
RW	3	4
WA	6	5
WC	12	6
RC	11	7

3.6 *FILE HANDLING MONITOR CALLS*

For a general discusssion on file system monitor call handling see Section 3.3.1.

3.6.1 *RFILE/WFILE*

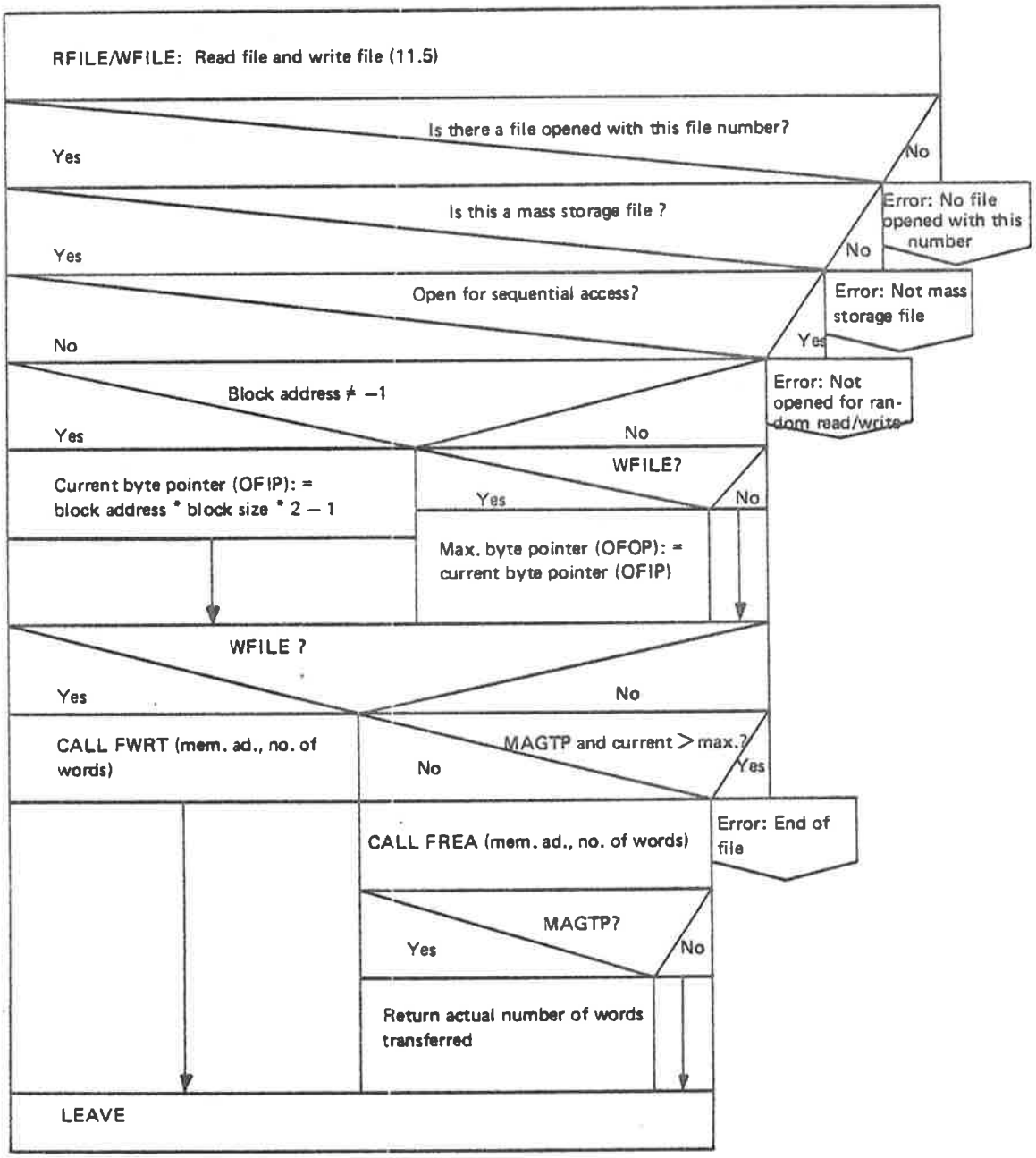


Figure 3.43: RFILE/WFILE

3.6.2 *Input Byte from File*

This routine is executed on level 4 (INBT/OUTBT level).

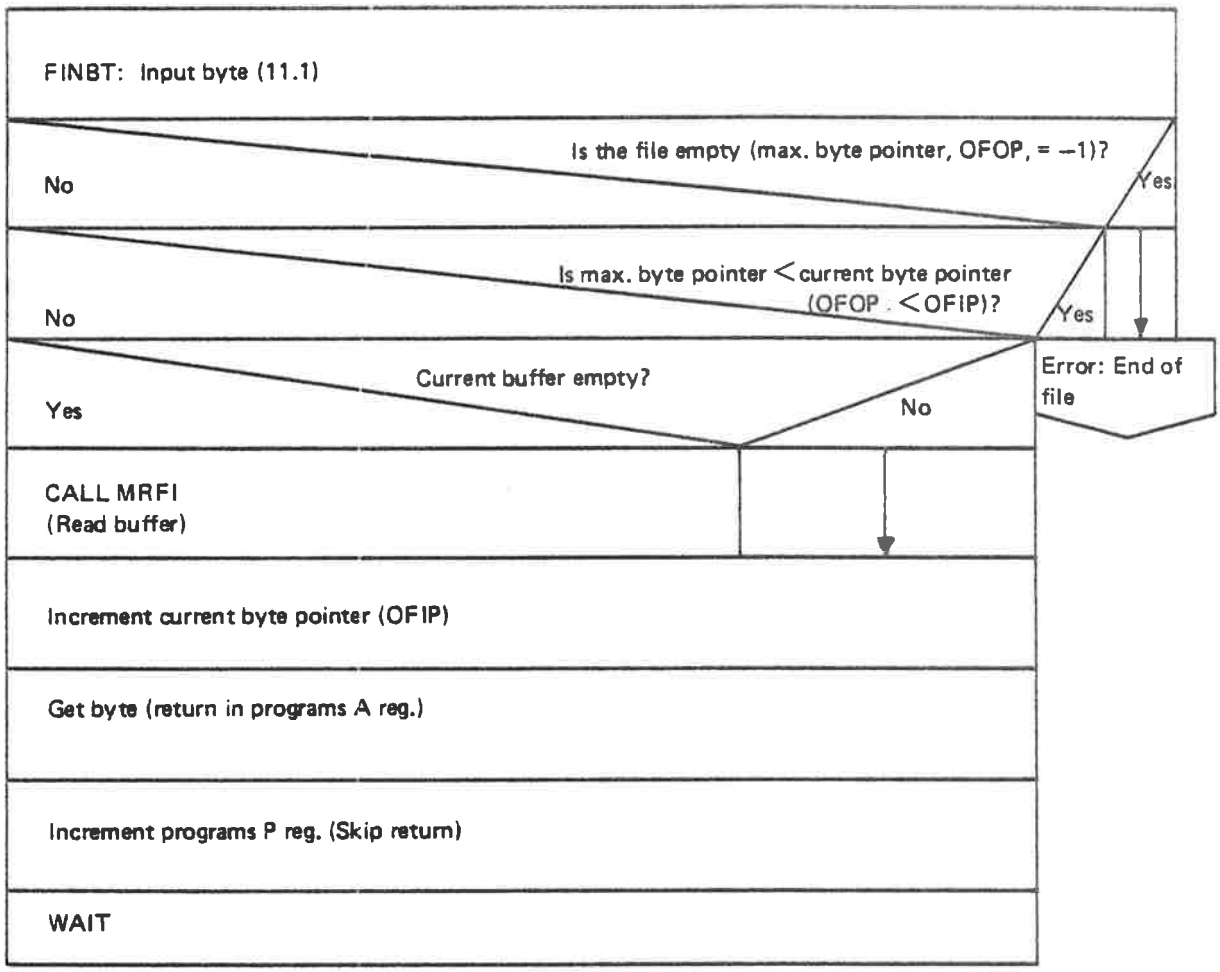


Figure 3.44: FINBT

APPENDIX A

A GUIDE TO THE FILE SYSTEM LISTING

The most frequently used routines on the file system segment are placed at its beginning in order to reduce swapping overhead.

The routines in the file system listing, however, are organized with respect to coherent operations. This will not coincide with an ordering based on memory addresses.

Table A.1 illustrates the correspondence between memory address and file system section number.

Table A.2 gives the contents of the listing ordered by chapter, while Table A.3 lists all routines ordered by section number. Table A.4 lists all routine names ordered alphabetically.

Chapter:	
1	Definitions
2	Buffer routines
3	Auxiliary routines
4	Directory routines
5	Bit file routines
6	Index block routines and spooling routines
7	User file routines
8	Object file routines
9	Open file table routines
10	Command processing
11	Monitor call processing
12	ABSTR, MAGTP, semaphore lock routines, etc.

Table A.2: Chapters of the File System Listing

Address in resident	Section in Listing
14140	1.8 - 1.9
14312	2.0
15511	3.5 - 3.7.1
15603	3.9 - 3.10
15675	4.1 - 4.2
15763	9.13 (last part)
16044	9.14 - 9.15
16263	9.16 - 9.17 (parts)
16326	11.7 (partly)
16412	11.9 (partly)
16717	11.11
16733	11.12
16764	11.14
17002	11.15
17112	12.1
17115	12.2.2 - 12.3.5
17433	12.7.4
17516	

Address on file system segment	Section in Listing
101300	2.1 - 2.9
103267	6.1 - 6.2
104416	9.11 - 9.13
106421	9.16 - 9.17
107013	11.3 - 11.14
110356	12.1
110500	9.21
110550	3.1 - 3.4
112026	3.7.2 - 3.8
112233	3.11 - 3.15
113410	4.3 - 5.8
115322	6.3 - 9.10
137102	9.18 - 9.27
141205	10.2 - 10.63
173331	12.4 - 12.6
173645	

Address on system segment	Section in Listing
70000	1.6.1
70750	1.6.2
71014	1.6.3
71017	11.1 - 11.2
71146	12.8 - 12.9
74252	

Table A.1: Memory Address -- Section Number

3.11	BDUMP	FILSEG2	DUMP BLOCK ON TERMINAL
3.11	DUMP	FILSEG2	DUMP BLOCK ON TERMINAL
3.12	CHANG	FILSEG2	CHANGE BLOCK
3.13	SEPST	FILSEG2	SEPARATE STRING
3.14	SEPPA	FILSEG2	SEPARATE FILE STRING
3.15	SEPFS	FILSEG2	SEPARATE FILE STRING IN THREE
4.01	GDIRA	RESIDENT	GET DIRECTORY ADDRESS
4.02	GNAMA	RESIDENT	GET NAME TABLE ADDRESS
4.03	GDIRI	FILSEG2	GET DIRECTORY INDEX
4.04	GNAMI	FILSEG2	GET NAME INDEX
4.05	GDIRE	FILSEG2	GET DIRECTORY INDEX
4.06	COLDE	FILSEG2	COLLECT DEVICE NAME AND UNIT
4.06	XCOLD	FILSEG2	COLLECT DEVICE NAME AND UNIT
4.07	GMAIN	FILSEG2	GET MAIN DIRECTORY INDEX
4.08	WDIRE	FILSEG2	WRITE DIRECTORY ENTRY
5.01	FBFBU	FILSEG2	FIND BIT FILE BUFFER ADDRESS
5.02	RBFBL	FILSEG2	READ BIT FILE BLOCK
5.03	WBFBL	FILSEG2	WRITE BIT FILE BLOCK
5.04	WBFBU	FILSEG2	WRITE BIT FILE BUFFER
5.05	ALBIT	FILSEG2	FIND BIT FILE ADDRESS
5.06	ALPAG	FILSEG2	ALLOCATE PAGE IN BIT FILE
5.06	RLPAG	FILSEG2	RELEASE PAGE IN BIT FILE
5.07	TPAGF	FILSEG2	TEST PAGE FREE
5.08	RSPAG	FILSEG2	RESERVE FIRST FREE PAGE
6.01	RINDX	FILSEG1	READ INDEX BLOCK
6.01A	FINDX	FILSEG1	READ INDEX BLOCK
6.02	WINDX	FILSEG1	WRITE INDEX BLOCK
6.03	STARS	FILSEG2	START SPOOLING
6.04	STSPL	FILSEG2	STOP SPOOLING
6.05	ABORS	FILSEG2	ABORT SPOOLING PRINT
6.05	STOPR	FILSEG2	STOP PRINT
6.05	STAPR	FILSEG2	START PRINT
6.05	RESTS	FILSEG2	RESTART SPOOLING PRINT
6.06	LSPOQ	FILSEG2	LIST SPOOLING QUEUE
6.07	APPES	FILSEG2	APPEND SPOOLING QUEUE
6.08	DELES	FILSEG2	DELETE SPOOLING FILE
6.09	RMSPF	FILSEG2	REMOVE FROM SPOO. QUEUE
6.09	GIVES	FILSEG2	GIVE SPOOLING PAGES
6.10	TAKES	FILSEG2	TAKE SPOOLING PAGES
6.11	SPOPL	FILSEG2	NUMBER OF SPOOL. PAGES LEFT
6.12	INPER	FILSEG2	INPUT SPOOLING PERIPHERAL
6.13	FINDQ	FILSEG2	FIND SPOOLING QUEUE
6.14	DEABB	FILSEG2	DEABBREVIATE FILE NAME
6.15	GFILN	FILSEG2	GET FILE NAME
6.16	HEAPRINT	FILSEG2	PRINT SPOOLING HEADER
6.16	TRAPRINT	FILSEG2	PRINT SPOOLING TRAILER
6.17	LOCKQ	FILSEG2	FIND NUMBER OF ELEM. IN QUEUE
6.18	UNLCQ	FILSEG2	UNLOCK QUEUE
6.19	READQ	FILSEG2	READ ONE QUEUE ELEMENT
6.20	WRITQ	FILSEG2	WRITE ONE QUEUE ELEMENT
6.21	APPEQ	FILSEG2	APPEND TO QUEUE
6.22	TAKEQ	FILSEG2	TAKE FROM SPOOLING QUEUE
6.23	INITQ	FILSEG2	INITIALIZE QUEUE
6.24	FPERIV	FILSEG2	FIND PERIPHERAL VERSION
6.25	FFILISQ	FILSEG2	FIND FILE IN SPOOL. QUEUE
6.26	MSPQENT	FILSEG2	MOVE SPOOL. QUEUE ENTRY
6.27	SNSPCOPY	FILSEG2	SET NO. OF PRINT COPIES
6.28	FWSPRINT	FILSEG2	FORWARD SPACE PRINT
6.28	BSPRINT	FILSEG2	BACKSPACE PRINT

Table A.3: Sections of the File System Listing

1.01	AUXILIARY	DECLARATIONS	SYMBOL DEFINITIONS
1.02	MACROES	DECLARATIONS	REGISTER DEFINITIONS
1.03	DEVICE BUFFERS	DECLARATIONS	
1.04	DIRECTORY TABLE	DECLARATIONS	
1.05	NAME TABLE	DECLARATIONS	
1.06.1	SPOP	SYSEG	POP SUBROUTINE STACK
1.06.1	SPUSH	SYSEG	PUSH SUBROUTINE STACK
1.06.1	SUBR. STACK	SYSEG	ENTER/LEAVE STACK
1.06.2	CONTEXT BL	DECLARATIONS	OPEN FILE TABLE
1.06.3	BUFFER POOL	SYSEG	
1.07	BIT FILE BUFFER	DECLARATIONS	
1.08	USER FILE BUFF	FILSEG2	BUFFER FOR USER ENTRY
1.09	OBJ. FILE BUFF	FILSEG2	BUFFER FOR OBJECT ENTRY
1.10	FILE RT-PROG	DECLARATIONS	
2.00	G3BUF	RESIDENT	GET MASS STORAGE BUFFER
2.00	G3IBUF	RESIDENT	GET MASS STORAGE BUFFER
2.00	G3NWT	RESIDENT	GET MASS STORAGE BUFFER
2.00	G5BUF	RESIDENT	GET MASS STORAGE BUFFER
2.00	R3BUF	RESIDENT	RELEASE MASS STORAGE BUFFER
2.00	R3IBUF	RESIDENT	RELEASE MASS STORAGE BUFFER
2.00	R5BUF	RESIDENT	RELEASE MASS STORAGE BUFFER
2.01	GDEVB	FILSEG1	GET DEVICE BUFFER
2.01A	FIDBU	FILSEG1	FIND DEVICE BUFFER HEADER
2.02	RDEVB	FILSEG1	RELEASE DEVICE BUFFER
2.03	RBLOC	FILSEG1	READ 1K FROM DEVICE
2.03A	RCBLO	FILSEG1	READ AND COMPARE 1K FROM DEVI
2.04	WBLOC	FILSEG1	WRITE 1K TO DEVICE
2.04A	WCBLO	FILSEG1	WRITE AND COMPARE 1K TO DEVIC
2.05	PTAPE	FILSEG1	POSITION TAPE
2.06	WEOT	FILSEG1	WRITE END OF TAPE
2.09	WTAPE	FILSEG1	WRITE DATA ON TAPE
3.01.1	MOCTA	FILSEG2	OUTPUT OCTAL NUMBER ON TERM.
3.01.1	OCTAL	FILSEG2	OUTPUT OCTAL NUMBER ON TERM.
3.01.2	DECIM	FILSEG2	OUTPUT DECIMAL NUMBER ON TERM.
3.01.2	MDECI	FILSEG2	OUTPUT DECIMAL NUMBER ON TERM.
3.01.3	DDECI	FILSEG2	OUTPUT DOUBLE DECIMAL NUMBER
3.01.3	MDDEC	FILSEG2	OUTPUT DOUBLE DECIMAL NUMBER
3.01.3	MTWOD	FILSEG2	OUTPUT TWO DIGITS DECIMAL
3.01.3	TWODE	FILSEG2	OUTPUT TWO DIGITS DECIMAL
3.02.1	OUTRC	FILSEG2	OUTPUT STRING ON TERMINAL
3.02.1	OUTST	FILSEG2	OUTPUT STRING ON TERMINAL
3.03.1	LDATE	FILSEG2	LIST DATE
3.03.1	MDATE	FILSEG2	LIST DATE
3.03.2	LACCW	FILSEG2	LIST ACCESS WORD
3.04	INSTR	FILSEG2	INPUT STRING
3.04	STRNG	FILSEG2	INPUT STRING
3.05	GETCH	RESIDENT	GET CHARACTER FROM STRING
3.05	PUTCH	RESIDENT	PUT CHARACTER TO STRING
3.07.1	ACOPY	RESIDENT	COPY STRING (ALT. PAGE TABLE)
3.07.1	COPYS	RESIDENT	COPY STRING
3.07.2	APPST	FILSEG2	APPEND STRING TO STRING
3.08	COMPS	FILSEG2	COMPARE STRINGS
3.09	SETBL	RESIDENT	SET BLOCK CONTENTS
3.10	COPYB	RESIDENT	COPY BLOCK

9.15	FPUT	FILSEG1	PUT BYTE ON FILE
9.16	FREA	FILSEG1/RESIDENT	FILE READ
9.16	FREA	RESIDENT/FILSEG1	FILE READ
9.17	FWRT	FILSEG1/RESIDENT	FILE WRITE
9.17	FWRT	RESIDENT/FILSEG1	FILE WRITE
9.18	RBYTE	FILSEG2	READ BYTE POINTER
9.18	RMAXB	FILSEG2	READ MAX POINTER
9.18	SBYTE	FILSEG2	SET BYTE POINTER
9.18	SMAXB	FILSEG2	SET MAX POINTER
9.19	SBLOP	FILSEG2	SET BLOCK POINTER
9.20	SDATF	FILSEG2	SET DATAFIELD RESERVED
9.21	CDATF	FILSEG2	CLEAR DATAFIELD RESERVED
9.22	OPSCR	FILSEG2	OPEN SCRATCH FILE
9.23	CPFIL	FILSEG2	COPY FILE
9.24	COLFI	FILSEG2	COLLECT FILE NAME
9.25	CLOUT	FILSEG2	CLOSE OUTPUT FILE
9.26	REMOPFI	FILSEG2	REMOTE OPEN FILE
9.27	NBAVA	FILSEG2	WAIT FOR ANSWER ON REMOTE BR.
10.02	CRDIR	FILSEG2	CREATE DIRECTORY
10.03	RNDIR	FILSEG2	RENAME DIRECTORY
10.04	ENDIR	FILSEG2	ENTER DIRECTORY
10.05	RLDIR	FILSEG2	RELEASE DIRECTORY
10.06	SDDIR	FILSEG2	SET DEFAULT DIRECTORY
10.07	DIRST	FILSEG2	DIRECTORY STATISTICS
10.07	LIDIR	FILSEG2	LIST DIRECTORIES ENTERED
10.08	DUDIR	FILSEG2	DUMP DIRECTORY ENTRY
10.09	CHDIR	FILSEG2	CHANGE DIRECTORY ENTRY
10.10	CRUSE	FILSEG2	CREATE USER
10.11	RNUSE	FILSEG2	RENAME USER
10.12	DLUSE	FILSEG2	DELETE USER
10.13	GIUSE	FILSEG2	GIVE USER SPACE
10.14	TAUSE	FILSEG2	TAKE USER SPACE
10.15	LIUSE	FILSEG2	LIST USERS
10.15	USEST	FILSEG2	USER STATISTICS
10.16	DUUSE	FILSEG2	DUMP USER ENTRY
10.17	CHUSE	FILSEG2	CHANGE USER ENTRY
10.18	ENUSE	FILSEG2	ENTER USER
10.19	RLUSE	FILSEG2	RELEASE USER
10.20	CHANP	FILSEG2	CHANGE PASSWORD
10.21	CLPAS	FILSEG2	CLEAR PASSWORD
10.22	CRFRI	FILSEG2	CREATE FRIEND
10.23	DLFRI	FILSEG2	DELETE FRIEND
10.24	SFRIA	FILSEG2	SET FRIEND ACCESS
10.25	LIFRI	FILSEG2	LIST FRIENDS
10.26	CRFIL	FILSEG2	CREATE FILE
10.26	CRNVE	FILSEG2	CREATE NEW FILE VERSION
10.27	ALFIL	FILSEG2	ALLOCATE FILE
10.27	ALNVE	FILSEG2	ALLOCATE NEW FILE VERSION
10.28	EXFIL	FILSEG2	EXPAND FILE
10.30	RNFIL	FILSEG2	RENAME FILE
10.31	DLFIL	FILSEG2	DELETE FILE
10.32	STERF	FILSEG2	SET TERMINAL FILE
10.32	STMPF	FILSEG2	SET TEMPORARY FILE
10.33	SPERF	FILSEG2	SET PERIPHERAL FILE
10.34	SFLAC	FILSEG2	SET FILE ACCESS
10.34A	SDFIA	FILSEG2	SET DEFAULT FILE ACCESS
10.35	DEUFI	FILSEG2	DELETE USERS FILES
10.35	FILST	FILSEG2	FILE STATISTICS
10.35	LIFIL	FILSEG2	LIST FILES

6.29	DSCOND	FILSEG2	DEFINE SPOOLING CONDITIONS
7.01.1	TUSSY	FILSEG2	TEST USER SYSTEM
7.01.2	TUSRT	FILSEG2	TEST USER RT
7.02	TUSEN	FILSEG2	TEST USER ENTERED
7.02A	RUSPW	FILSEG2	READ USER PASSWORD
7.03	FUSEB	FILSEG2	FIND USER ENTRY BUFFER
7.04	RUSER	FILSEG2	READ USER ENTRY
7.05	WUSER	FILSEG2	WRITE USER ENTRY
7.06	RUSEB	FILSEG2	RELEASE USER ENTRY
7.07	GUSEI	FILSEG2	GET USER INDEX
7.08	GMUSI	FILSEG2	GET MAIN USER INDEX
7.09	COLUN	FILSEG2	COLLECT USER NAME
7.10	GUSEN	FILSEG2	GET USER NAME
7.11	CUSED	FILSEG2	CHANGE USER SPACE
7.12	GDEFD	FILSEG2	GET DEFAULT DIRECTORY
7.13	GUSAC	FILSEG2	GET USER ACCESS
8.01	FOBJB	FILSEG2	FIND OBJECT ENTRY BUFFER
8.02	ROBJE	FILSEG2	READ OBJECT ENTRY
8.03	WOBJE	FILSEG2	WRITE OBJECT ENTRY
8.04	ROBJB	FILSEG2	RELEASE OBJECT ENTRY BUFFER
8.05	GOBJI	FILSEG2	GET OBJECT INDEX
8.06	SEPOB	FILSEG2	SEPARATE OBJECT NAME
8.07	GFILI	FILSEG2	GET FILE INDEX
8.08	GPREV	FILSEG2	GET PREVIOUS VERSION
8.09	GNEXV	FILSEG2	GET NEXT VERSION
8.10	COBJE	FILSEG2	CREATE OBJECT ENTRY
8.11	CHIGV	FILSEG2	CREATE NEW HIGHER VERSION
8.11	CNEWV	FILSEG2	CREATE NEW VERSION
8.12	CROBJ	FILSEG2	CREATE OBJECTS
8.14	DLOBJ	FILSEG2	DELETE OBJECT
8.15	CRNEW	FILSEG2	CREATE NEW VERSION OF FILE
8.16	GVERS	FILSEG2	GET VERSION NUMBER
8.17	GFIAC	FILSEG2	GET FILE ACCESS
8.18	GCFIL	FILSEG2	GET OR CREATE FILE
8.19	DLPAG	FILSEG2	DELETE PAGES OF FILE
8.19	DLSPA	FILSEG2	DELETE PAGES OF FILE
9.01	FFILE	FILSEG2	FIND FILE TO OPEN
9.02	FOFT	FILSEG2	FIND OPEN FILE TABLE
9.03	SOFT	FILSEG2	SET UP OPEN FILE TABLE
9.03A	OFRND	FILSEG2	OPEN FILE FOR RANDOM ACCESS
9.04	FCON	FILSEG2	FILE CONNECT
9.05	FOPEN	FILSEG2	FILE OPEN
9.06	FCLOS	FILSEG2	FILE CLOSE
9.06	XFCLOS	FILSEG2/RESIDENT	CLOSE SPOOLING FILE
9.06	XFCLOS	FILSEG2	FILE CLOSE (NO VERSION CHANGE)
9.07	GBUF	FILSEG2	GET BUFFER FROM POOL
9.07	GBUFS	FILSEG2	GET BUFFER SET FROM POOL
9.08	RBUF	FILSEG2	RETURN BUFFER TO POOL
9.09	SBLSZ	FILSEG2	SET BLOCK SIZE
9.10	SETPO	FILSEG2	SET PERMANENT OPEN
9.11.1	GPADR	FILSEG1	GET PAGE ADDRESS OF FILE
9.11.1	GPREA	FILSEG1	GET PAGE ADDRESS FOR READ
9.11.2	WBACK	FILSEG1	WRITE BACK INDEXES
9.11.3	GPAGE	FILSEG1	GET PAGE FOR FILE
9.11.4	RESSTAR	FILSEG1	RESER. SEMAPH. FOR START PROG
9.12	REBUF	FILSEG1	READ BUFFERS FROM FILE
9.13	FLYTT	RESIDENT	MOVE 100 WORDS
9.13	WRBUF	FILSEG1	WRITE BUFFERS ON FILE
9.14	FGET	FILSEG1	GET BYTE FROM FILE

11.10	ERMSG	FILSEG1	WRITE ERROR MESSAGE
11.10	QERMS	FILSEG1	WRITE ERROR MESSAGE AND STOR
11.11	MROBJ	FILSEG1/RESIDENT	READ OBJECT ENTRY
11.11	MROBJ	RESIDENT/FILSEG1	READ OBJECT ENTRY
11.12	MRUSE	FILSEG1/RESIDENT	READ USER ENTRY
11.13	MPYAT	FILSEG1	AD:=A*T
11.12	MRUSE	RESIDENT/FILSEG1	READ USER ENTRY
11.15	RSPQE	RESIDENT	READ SPOOLING QUEUE ENTRY
12.01	ELOCK	FILSEG1	ESCAPE LOCK
12.01	EULOC	FILSEG1	ESCAPE UNLOCK
12.01	FATAL	RESIDENT	FATAL ERROR
12.01	LOCK	RESIDENT	LOCK SEMAPHORE
12.01	UNLOC	RESIDENT	UNLOCK SEMAPHORE
12.02.3	WHERE	RESIDENT	WHERE IS SEMAPHORE
12.03.1	CABST	RESIDENT	CARTRIDGE DISC ABSTRANS
12.03.2	DRABS	RESIDENT	DRUM ABSTRANS
12.03.3	BABST	RESIDENT	BIG DISC ABSTRANS
12.03.4	MABST	RESIDENT	MAG TAPE ABSTRANS
12.03.5	FDABS	RESIDENT	FLOPPY DISC ABSTRANS
12.04.3	CMMON	FILSEG2	COMMAND MONITOR
12.04.4	CLPAR	FILSEG2	COLLECT PARAMETER
12.04.5	ERROR	FILSEG2	WRITE ERROR MESSAGE
12.05.1	INITF	FILSEG2	INITIATE FILE SYSTEM TABLES
12.06	GDATE	FILSEG2	GET DATE
12.07.4	SINBT	RESIDENT	INPUT BYTE TO FILE SYSTEM
12.07.4	SOUTBT	RESIDENT	OUTPUT BYTE FROM FILE SYSTEM
12.08	OPCAL	SYSEG	CALL ROUTINE ON OP.COM.SEG

Table A.4: Routines in the File System, ordered alphabetically

6.05	ABORS	FILSEG2	ABORT SPOOLING PRINT
3.07.1	ACOPY	RESIDENT	COPY STRING (ALT. PAGE TABLE)
5.05	ALBIT	FILSEG2	FIND BIT FILE ADDRESS
10.27	ALFIL	FILSEG2	ALLOCATE FILE
10.27	ALNVE	FILSEG2	ALLOCATE NEW FILE VERSION
5.06	ALPAG	FILSEG2	ALLOCATE PAGE IN BIT FILE
6.21	APPEQ	FILSEG2	APPEND TO QUEUE
6.07	APPES	FILSEG2	APPEND SPOOLING QUEUE
3.07.2	APPST	FILSEG2	APPEND STRING TO STRING
1.01	AUXILIARY	DECLARATIONS	SYMBOL DEFINITIONS
12.03.3	BABST	RESIDENT	BIG DISC ABSTRANS
3.11	BDUMP	FILSEG2	DUMP BLOCK ON TERMINAL
1.07	BIT FILE BUFFER	DECLARATIONS	
6.28	BSPRINT	FILSEG2	BACKSPACE PRINT
1.06.3	BUFFER POOL	SYSEG	
12.03.1	CABST	RESIDENT	CARTRIDGE DISC ABSTRANS
9.21	CDATF	FILSEG2	CLEAR DATAFIELD RESERVED
3.12	CHANG	FILSEG2	CHANGE BLOCK
10.20	CHANP	FILSEG2	CHANGE PASSWORD
10.55	CHBIT	FILSEG2	CHANGE BIT TABLE
10.09	CHDIR	FILSEG2	CHANGE DIRECTORY ENTRY
8.11	CHIGV	FILSEG2	CREATE NEW HIGHER VERSION
10.37	CHOBJ	FILSEG2	CHANGE OBJECT ENTRY
10.53	CHPAG	FILSEG2	CHANGE PAGE
10.17	CHUSE	FILSEG2	CHANGE USER ENTRY
11.08	CLOFI	FILSEG1	CLOSE FILE
10.40	CLOSF	FILSEG2	CLOSE FILE
9.25	CLOUT	FILSEG2	CLOSE OUTPUT FILE
12.04.4	CLPAR	FILSEG2	COLLECT PARAMETER
10.21	CLPAS	FILSEG2	CLEAR PASSWORD
10.63	CLPRY	FILSEG2	CLEAR PARITY IN TAPE LABEL
10.51	CLRTF	FILSEG2	CLOSE RT FILE
12.04.3	CMMON	FILSEG2	COMMAND MONITOR
8.11	CNEWV	FILSEG2	CREATE NEW VERSION
8.10	COBJE	FILSEG2	CREATE OBJECT ENTRY
4.06	COLDE	FILSEG2	COLLECT DEVICE NAME AND UNIT
9.24	COLFI	FILSEG2	COLLECT FILE NAME
7.09	COLUN	FILSEG2	COLLECT USER NAME
3.08	COMPS	FILSEG2	COMPARE STRINGS
10.39	CONNF	FILSEG2	CONNECT FILE
1.06.2	CONTEXT BL	DECLARATIONS	OPEN FILE TABLE
10.57	COPDI	FILSEG2	COPY DIRECTORY
10.57	COPFI	FILSEG2	COPY FILE
3.10	COPYB	RESIDENT	COPY BLOCK
3.07.1	COPYS	RESIDENT	COPY STRING
10.50	CORTF	FILSEG2	CONNECT RT FILE
9.23	CPFIL	FILSEG2	COPY FILE
10.62	CPUFIL	FILSEG2	COPY USERS FILES
10.02	CRDIR	FILSEG2	CREATE DIRECTORY
10.60	CREVOL	FILSEG2	CREATE VOLUME
10.26	CRFIL	FILSEG2	CREATE FILE
10.22	CRFRI	FILSEG2	CREATE FRIEND
8.15	CRNEW	FILSEG2	CREATE NEW VERSION OF FILE

10.26	CRNVE	FILSEG2	CREATE NEW FILE VERSION
8.12	CROBJ	FILSEG2	CREATE OBJECTS
10.10	CRUSE	FILSEG2	CREATE USER
7.11	CUSED	FILSEG2	CHANGE USER SPACE
3.01.3	DDECI	FILSEG2	OUTPUT DOUBLE DECIMAL NUMBER
6.14	DEABB	FILSEG2	DEABBREVIATE FILE NAME
3.01.2	DECIM	FILSEG2	OUTPUT DECIMAL NUMBER ON TERM.
6.08	DELES	FILSEG2	DELETE SPOOLING FILE
10.35	DEUFI	FILSEG2	DELETE USERS FILES
1.03	DEVICE BUFFERS	DECLARATIONS	
1.04	DIRECTORY TABLE	DECLARATIONS	
10.07	DIRST	FILSEG2	DIRECTORY STATISTICS
10.31	DLFIL	FILSEG2	DELETE FILE
10.23	DLFRI	FILSEG2	DELETE FRIEND
8.14	DLOBJ	FILSEG2	DELETE OBJECT
8.19	DLPAG	FILSEG2	DELETE PAGES OF FILE
8.19	DLSPA	FILSEG2	DELETE PAGES OF FILE
10.12	DLUSE	FILSEG2	DELETE USER
12.03.2	DRABS	RESIDENT	DRUM ABSTRANS
6.29	DSCOND	FILSEG2	DEFINE SPOOLING CONDITIONS
10.54	DUBIT	FILSEG2	DUMP BIT TABLE
10.08	DUDIR	FILSEG2	DUMP DIRECTORY ENTRY
3.11	DUMP	FILSEG2	DUMP BLOCK ON TERMINAL
10.36	DUOBJ	FILSEG2	DUMP OBJECT ENTRY
10.52	DUPAG	FILSEG2	DUMP PAGE
10.16	DUUSE	FILSEG2	DUMP USER ENTRY
12.01	ELOCK	FILSEG1	ESCAPE LOCK
10.04	ENDIR	FILSEG2	ENTER DIRECTORY
10.18	ENUSE	FILSEG2	ENTER USER
11.10	ERMSG	FILSEG1	WRITE ERROR MESSAGE
12.04.5	ERROR	FILSEG2	WRITE ERROR MESSAGE
12.01	EULOC	FILSEG1	ESCAPE UNLOCK
10.28	EXFIL	FILSEG2	EXPAND FILE
12.01	FATAL	RESIDENT	FATAL ERROR
5.01	FBFBU	FILSEG2	FIND BIT FILE BUFFER ADDRESS
9.06	FCLOS	FILSEG2	FILE CLOSE
9.04	FCON	FILSEG2	FILE CONNECT
12.03.5	FDABS	RESIDENT	FLOPPY DISC ABSTRANS
6.25	FFILISQ	FILSEG2	FIND FILE IN SPOOL. QUEUE
9.01	FFILE	FILSEG2	FIND FILE TO OPEN
9.14	FGET	FILSEG1	GET BYTE FROM FILE
2.01A	FIDBU	FILSEG1	FIND DEVICE BUFFER HEADER
1.10	FILE RT-PROG	DECLARATIONS	
10.35	FILST	FILSEG2	FILE STATISTICS
11.01	FINBT	SYSEG	INPUT BYTE
6.13	FINDQ	FILSEG2	FIND SPOOLING QUEUE
6.01A	FINDX	FILSEG1	READ INDEX BLOCK
9.13	FLYTT	RESIDENT	MOVE 100 WORDS
8.01	FOBJB	FILSEG2	FIND OBJECT ENTRY BUFFER
9.02	FOFT	FILSEG2	FIND OPEN FILE TABLE
9.05	FOPEN	FILSEG2	FILE OPEN
11.02	FOUTBT	SYSEG	OUTPUT BYTE
6.24	FPERIV	FILSEG2	FIND PERIPHERAL VERSION
9.15	FPUT	FILSEG1	PUT BYTE ON FILE
9.16	FREA	FILSEG1/RESIDENT	FILE READ
9.16	FREA	RESIDENT/FILSEG1	FILE READ
7.03	FUSEB	FILSEG2	FIND USER ENTRY BUFFER
9.17	FWRT	FILSEG1/RESIDENT	FILE WRITE
6.28	FWSPRINT	FILSEG2	FORWARD SPACE PRINT

9.17	FWRT	RESIDENT/FILSEG1	FILE WRITE
2.00	G3BUF	RESIDENT	GET MASS STORAGE BUFFER
2.00	G3IBUF	RESIDENT	GET MASS STORAGE BUFFER
2.00	G3NWT	RESIDENT	GET MASS STORAGE BUFFER
2.00	G5BUF	RESIDENT	GET MASS STORAGE BUFFER
9.07	GBUF	FILSEG2	GET BUFFER FROM POOL
9.07	GBUFS	FILSEG2	GET BUFFER SET FROM POOL
8.18	GCFIL	FILSEG2	GET OR CREATE FILE
12.06	GDATE	FILSEG2	GET DATE
7.12	GDEFD	FILSEG2	GET DEFAULT DIRECTORY
2.01	GDEVB	FILSEG1	GET DEVICE BUFFER
4.01	GDIRA	RESIDENT	GET DIRECTORY ADDRESS
4.05	GDIRE	FILSEG2	GET DIRECTORY INDEX
4.03	GDIRI	FILSEG2	GET DIRECTORY INDEX
3.05	GETCH	RESIDENT	GET CHARACTER FROM STRING
8.17	GFIAC	FILSEG2	GET FILE ACCESS
8.07	GFILI	FILSEG2	GET FILE INDEX
6.15	GFILN	FILSEG2	GET FILE NAME
10.13	GIUSE	FILSEG2	GIVE USER SPACE
6.09	GIVES	FILSEG2	GIVE SPOOLING PAGES
4.07	GMAIN	FILSEG2	GET MAIN DIRECTORY INDEX
7.08	GMUSI	FILSEG2	GET MAIN USER INDEX
4.02	GNAMA	RESIDENT	GET NAME TABLE ADDRESS
4.04	GNAMI	FILSEG2	GET NAME INDEX
8.09	GNEXV	FILSEG2	GET NEXT VERSION
8.05	GOBJI	FILSEG2	GET OBJECT INDEX
9.11.1	GPADR	FILSEG1	GET PAGE ADDRESS OF FILE
9.11.3	GPAGE	FILSEG1	GET PAGE FOR FILE
9.11.1	GPREA	FILSEG1	GET PAGE ADDRESS FOR READ
8.08	GPREV	FILSEG2	GET PREVIOUS VERSION
7.13	GUSAC	FILSEG2	GET USER ACCESS
7.07	GUSEI	FILSEG2	GET USER INDEX
7.10	GUSEN	FILSEG2	GET USER NAME
8.16	GVERS	FILSEG2	GET VERSION NUMBER
6.16	HEAPRINT	FILSEG2	PRINT SPOOLING HEADER
11.01	INBT	SYSEG	INPUT BYTE
12.05.1	INITF	FILSEG2	INITIATE FILE SYSTEM TABLES
6.23	INITQ	FILSEG2	INITIALIZE QUEUE
6.12	INPER	FILSEG2	INPUT SPOOLING PERIPHERAL
3.04	INSTR	FILSEG2	INPUT STRING
3.03.2	LACCW	FILSEG2	LIST ACCESS WORD
3.03.1	LDATE	FILSEG2	LIST DATE
10.07	LIDIR	FILSEG2	LIST DIRECTORIES ENTERED
10.35	LIFIL	FILSEG2	LIST FILES
10.25	LIFRI	FILSEG2	LIST FRIENDS
10.41	LIOPF	FILSEG2	LIST OPENED FILES
10.41	LIRTO	FILSEG2	LIST RT OPENED FILES
10.15	LIUSE	FILSEG2	LIST USERS
10.61	LIVOL	FILSEG2	LIST VOLUME
12.01	LOCK	RESIDENT	LOCK SEMAPHORE
6.17	LOCKQ	FILSEG2	FIND NUMBER OF ELEM. IN QUEUE
6.06	LSPOQ	FILSEG2	LIST SPOOLING QUEUE
12.03.4	MABST	RESIDENT	MAG TAPE ABSTRANS
1.02	MACROES	DECLARATIONS	REGISTER DEFINITIONS
3.03.1	MDATE	FILSEG2	LIST DATE
3.01.3	MDDEC	FILSEG2	OUTPUT DOUBLE DECIMAL NUMBER
3.01.2	MDECI	FILSEG2	OUTPUT DECIMAL NUMBER ON TERM.
3.01.1	MOCTA	FILSEG2	OUTPUT OCTAL NUMBER ON TERM.
11.13	MPYAT	FILSEG1	AD:=A*T

11.11	MROBJ	FILSEG1/RESIDENT	READ OBJECT ENTRY
11.11	MROBJ	RESIDENT/FILSEG1	READ OBJECT ENTRY
11.12	MRUSE	FILSEG1/RESIDENT	READ USER ENTRY
6.26	MSPQENT	FILSEG2	MOVE SPOOLING QUEUE ENTRY
11.12	MRUSE	RESIDENT/FILSEG1	READ USER ENTRY
3.01.3	MTWOD	FILSEG2	OUTPUT TWO DIGITS DECIMAL
1.05	NAME TABLE	DECLARATIONS	
9.27	NBAVA	FILSEG2	WAIT FOR ANSWER ON REMOTE OP.
1.09	OBJ. FILE BUFF	FILSEG2	BUFFER FOR OBJECT ENTRY
3.01.1	OCTAL	FILSEG2	OUTPUT OCTAL NUMBER ON TERM.
9.03A	OFRND	FILSEG2	OPEN FILE FOR RANDOM ACCESS
11.07	OLDOP	FILSEG1/RESIDENT	OLD OPEN FILE
11.07	OLDOP	RESIDENT/FILSEG1	OLD OPEN FILE
12.08	OPCAL	SYSEG	CALL ROUTINE ON OP.COM.SEG.
10.38	OPENF	FILSEG2	OPEN FILE
10.51	OPENS	FILSEG2	OPEN SCRATCH FILE
11.07	OPFIL	FILSEG1/RESIDENT	OPEN FILE
11.07	OPFIL	RESIDENT/FILSEG1	OPEN FILE
10.49	OPRTF	FILSEG2	OPEN RT FILE
9.22	OPSCR	FILSEG2	OPEN SCRATCH FILE
11.02	OUTBT	SYSEG	OUTPUT BYTE
3.02.1	OUTRC	FILSEG2	OUTPUT STRING ON TERMINAL
3.02.1	OUTST	FILSEG2	OUTPUT STRING ON TERMINAL
2.05	PTAPE	FILSEG1	POSITION TAPE
3.05	PUTCH	RESIDENT	PUT CHARACTER TO STRING
11.10	QERMS	FILSEG1	WRITE ERROR MESSAGE AND STOR
2.00	R3BUF	RESIDENT	RELEASE MASS STORAGE BUFFER
2.00	R3IBUF	RESIDENT	RELEASE MASS STORAGE BUFFER
2.00	R5BUF	RESIDENT	RELEASE MASS STORAGE BUFFER
5.02	RBFBL	FILSEG2	READ BIT FILE BLOCK
2.03	RBLOC	FILSEG1	READ 1K FROM DEVICE
9.08	RBUF	FILSEG2	RETURN BUFFER TO POOL
9.18	RBYTE	FILSEG2	READ BYTE POINTER
2.03A	RCBLO	FILSEG1	READ AND COMPARE 1K FROM DEVI
2.02	RDEVB	FILSEG1	RELEASE DEVICE BUFFER
11.04	RDISK	FILSEG1	READ DISK
11.09	REABT	FILSEG1	READ BYTE POINTER
6.19	READQ	FILSEG2	READ ONE QUEUE ELEMENT
9.12	REBUF	FILSEG1	READ BUFFERS FROM FILE
10.56	REGDI	FILSEG2	REGENERATE DIRECTORY
10.47	RELFI	FILSEG2	RELEASE FILE
10.58	RELTU	FILSEG2	RELEASE DEVICE UNIT
9.26	REMOPFI	FILSEG2	REMOTE OPEN FILE
10.46	RESFI	FILSEG2	RESERVE FILE
9.11.4	RESSTAR	FILSEG1	RESER. SEMAPH. FOR START PROG
6.05	RESTS	FILSEG2	RESTART SPOOLING PRINT
10.58	RESTU	FILSEG2	RESERVE DEVICE UNIT
11.05	RFILE	FILSEG1	READ FILE
6.01	RINDX	FILSEG1	READ INDEX BLOCK
10.05	RLDIR	FILSEG2	RELEASE DIRECTORY
5.06	RLPAG	FILSEG2	RELEASE PAGE IN BIT FILE
10.19	RLUSE	FILSEG2	RELEASE USER
11.09	RMAX	FILSEG1	READ MAX POINTER
9.18	RMAXB	FILSEG2	READ MAX POINTER
6.08	RMSPF	FILSEG2	REMOVE SPOOL. QUEUE ENTRY
10.03	RNDIR	FILSEG2	RENAME DIRECTORY
10.30	RNFIL	FILSEG2	RENAME FILE
10.11	RNUSE	FILSEG2	RENAME USER
8.04	ROBJB	FILSEG2	RELEASE OBJECT ENTRY BUFFER

8.02	ROBJE	FILSEG2	READ OBJECT ENTRY
11.03	RPAGE	FILSEG1	READ PAGE
5.08	RSPAG	FILSEG2	RESERVE FIRST FREE PAGE
11.15	RSPQE	RESIDENT	READ SPOOLING QUEUE ENTRY
7.06	RUSEB	FILSEG2	RELEASE USER ENTRY
7.04	RUSER	FILSEG2	READ USER ENTRY
7.02A	RUSPW	FILSEG2	READ USER PASSWORD
10.59	SAVDI	FILSEG2	SAVE DIRECTORY
11.01	SBINBT	SYSEG	INPUT BYTE
10.45	SBLOC	FILSEG2	SET BLOCK POINTER
9.19	SBLOP	FILSEG2	SET BLOCK POINTER
10.42	SBLOS	FILSEG2	SET BLOCK SIZE
9.09	SBLSZ	FILSEG2	SET BLOCK SIZE
11.02	SBOUTBT	SYSEG	OUTPUT BYTE
11.09	SBSIZ	FILSEG1	SET BLOCK SIZE
9.18	SBYTE	FILSEG2	SET BYTE POINTER
10.44	SBYTP	FILSEG2	SET BYTE POINTER
9.20	SDATF	FILSEG2	SET DATAFIELD RESERVED
10.06	SDDIR	FILSEG2	SET DEFAULT DIRECTORY
10.34A	SDFIA	FILSEG2	SET DEFAULT FILE ACCESS
3.15	SEPFS	FILSEG2	SEPARATE FILE STRING IN THREE
8.06	SEPOB	FILSEG2	SEPARATE OBJECT NAME
3.14	SEPPA	FILSEG2	SEPARATE FILE STRING
3.13	SEPST	FILSEG2	SEPARATE STRING
11.09	SETBC	FILSEG1	SET BLOCK POINTER
3.09	SETBL	RESIDENT	SET BLOCK CONTENTS
11.09	SETBY	FILSEG1	SET BYTE POINTER
9.10	SETPO	FILSEG2	SET PERMANENT OPEN
11.09	SETUP	RESIDENT	STRING DESCRIPTOR SET UP
11.09	SETW	RESIDENT	SET WRITE POINTER OF STRING
10.34	SFLAC	FILSEG2	SET FILE ACCESS
10.24	SFRIA	FILSEG2	SET FRIEND ACCESS
12.07.4	SINBT	RESIDENT	INPUT BYTE TO FILE SYSTEM
11.09	SMAX	FILSEG1	SET MAX POINTER
9.18	SMAXB	FILSEG2	SET MAX POINTER
6.27	SNSPCOPY	FILSEG2	SET NO. OF PRINT COPIES
9.03	SOFT	FILSEG2	SET UP OPEN FILE TABLE
12.07.4	SOUTBT	RESIDENT	OUTPUT BYTE FROM FILE SYSTEM
10.33	SPERF	FILSEG2	SET PERIPHERAL FILE
10.43	SPERO	FILSEG2	SET PERMANENT OPENED
1.06.1	SPOP	SYSEG	POP SUBROUTINE STACK
6.11	SPOPL	FILSEG2	NUMBER OF SPOOL. PAGES LEFT
1.06.1	SPUSH	SYSEG	PUSH SUBROUTINE STACK
6.05	STAPR	FILSEG2	START PRINT
6.03	STARS	FILSEG2	START SPOOLING
10.32	STERF	FILSEG2	SET TERMINAL FILE
10.32	STMPF	FILSEG2	SET TEMPORARY FILE
6.05	STOPR	FILSEG2	STOP PRINT
3.04	STRNG	FILSEG2	INPUT STRING
6.04	STSPL	FILSEG2	STOP SPOOLING
1.06.1	SUBR. STACK	SYSEG	ENTER/LEAVE STACK
6.22	TAKEQ	FILSEG2	TAKE FROM SPOOLING QUEUE
6.10	TAKES	FILSEG2	TAKE SPOOLING PAGES
10.14	TAUSE	FILSEG2	TAKE USER SPACE
10.56	TESDI	FILSEG2	TEST DIRECTORY
5.07	TPAGF	FILSEG2	TEST PAGE FREE
6.16	TRAPRINT	FILSEG2	PRINT SPOOLING TRAILER
7.02	TUSEN	FILSEG2	TEST USER ENTERED
7.01.2	TUSRT	FILSEG2	TEST USER RT

7.01.1	TUSSY	FILSEG2	TEST USER SYSTEM
3.01.3	TWODE	FILSEG2	OUTPUT TWO DIGITS DECIMAL
6.18	UNLCQ	FILSEG2	UNLOCK QUEUE
12.01	UNLOC	RESIDENT	UNLOCK SEMAPHORE
1.08	USER FILE BUFF	FILSEG2	BUFFER FOR USER ENTRY
10.15	USEST	FILSEG2	USER STATISTICS
9.11.2	WBACK	FILSEG1	WRITE BACK INDEXES
5.03	WBFBL	FILSEG2	WRITE BIT FILE BLOCK
5.04	WBFBU	FILSEG2	WRITE BIT FILE BUFFER
2.04	WBLOC	FILSEG1	WRITE 1K TO DEVICE
2.04A	WCBLO	FILSEG1	WRITE AND COMPARE 1K TO DEVIC
11.09	WCI	RESIDENT	WRITE BYTE TO STRING
4.08	WDIRE	FILSEG2	WRITE DIRECTORY ENTRY
11.04	WDISK	FILSEG1	WRITE DISK
2.06	WEOT	FILSEG1	WRITE END OF TAPE
11.05	WFILE	FILSEG1	WRITE FILE
10.48	WHEFI	FILSEG2	WHERE IS FILE
12.02.3	WHERE	RESIDENT	WHERE IS SEMAPHORE
6.02	WINDX	FILSEG1	WRITE INDEX BLOCK
8.03	WOBJE	FILSEG2	WRITE OBJECT ENTRY
11.03	WPAGE	FILSEG1	WRITE PAGE
9.13	WRBUF	FILSEG1	WRITE BUFFERS ON FILE
6.20	WRITQ	FILSEG2	WRITE ONE QUEUE ELEMENT
2.09	WTAPE	FILSEG1	WRITE DATA ON TAPE
7.05	WUSER	FILSEG2	WRITE USER ENTRY
4.06	XCOLD	FILSEG2	COLLECT DEVICE NAME AND UNIT
9.06	XFCLOS	FILSEG2	FILE CLOSE (NO VERSION CHANGE)

M E M O R Y A N D S E G M E N T M A P

1.01	AUXILIARY	DECLARATIONS	SYMBOL DEFINITIONS
1.02	MACROES	DECLARATIONS	REGISTER DEFINITIONS
1.03	DEVICE BUFFERS	DECLARATIONS	
1.04	DIRECTORY TABLE	DECLARATIONS	
1.05	NAME TABLE	DECLARATIONS	
1.06.2	CONTEXT BL	DECLARATIONS	OPEN FILE TABLE
1.07	BIT FILE BUFFER	DECLARATIONS	
1.10	FILE RT-PROG	DECLARATIONS	
2.01	GDEVB	FILSEG1	GET DEVICE BUFFER
2.01A	FIDBU	FILSEG1	FIND DEVICE BUFFER HEADER
2.02	RDEVB	FILSEG1	RELEASE DEVICE BUFFER
2.03	RBLOC	FILSEG1	READ 1K FROM DEVICE
2.03A	RCBLO	FILSEG1	READ AND COMPARE 1K FROM DEVI
2.04	WBLOC	FILSEG1	WRITE 1K TO DEVICE
2.04A	WCBLO	FILSEG1	WRITE AND COMPARE 1K TO DEVIC
2.05	PTAPE	FILSEG1	POSITION TAPE
2.06	WEOT	FILSEG1	WRITE END OF TAPE
2.09	WTAPE	FILSEG1	WRITE DATA ON TAPE
6.01	RINDX	FILSEG1	READ INDEX BLOCK
6.01A	FINDX	FILSEG1	READ INDEX BLOCK
6.02	WINDX	FILSEG1	WRITE INDEX BLOCK
9.11.1	GPADR	FILSEG1	GET PAGE ADDRESS OF FILE
9.11.1	GPREA	FILSEG1	GET PAGE ADDRESS FOR READ
9.11.2	WBACK	FILSEG1	WRITE BACK INDEXES
9.11.3	GPAGE	FILSEG1	GET PAGE FOR FILE
9.11.4	RESSTAR	FILSEG1	RESER. SEMAPH. FOR START PROG
9.12	REBUF	FILSEG1	READ BUFFERS FROM FILE
9.13	WRBUF	FILSEG1	WRITE BUFFERS ON FILE
9.14	FGET	FILSEG1	GET BYTE FROM FILE
9.15	FPUT	FILSEG1	PUT BYTE ON FILE
9.16	FREA	FILSEG1/RESIDENT	FILE READ
9.17	FWRT	FILSEG1/RESIDENT	FILE WRITE
11.03	RPAGE	FILSEG1	READ PAGE
11.03	WPAGE	FILSEG1	WRITE PAGE
11.04	RDISK	FILSEG1	READ DISK
11.04	WDISK	FILSEG1	WRITE DISK
11.05	RFILE	FILSEG1	READ FILE
11.05	WFILE	FILSEG1	WRITE FILE
11.07	OLDOP	FILSEG1/RESIDENT	OLD OPEN FILE
11.07	OPFIL	FILSEG1/RESIDENT	OPEN FILE
11.08	CLOFI	FILSEG1	CLOSE FILE
11.09	REABT	FILSEG1	READ BYTE POINTER
11.09	RMAX	FILSEG1	READ MAX POINTER
11.09	SBSIZ	FILSEG1	SET BLOCK SIZE
11.09	SETBC	FILSEG1	SET BLOCK POINTER
11.09	SETBY	FILSEG1	SET BYTE POINTER
11.09	SMAX	FILSEG1	SET MAX POINTER
11.10	ERMSG	FILSEG1	WRITE ERROR MESSAGE
11.10	QERMS	FILSEG1	WRITE ERROR MESSAGE AND STOP
11.11	MROBJ	FILSEG1/RESIDENT	READ OBJECT ENTRY
11.12	MRUSE	FILSEG1/RESIDENT	READ USER ENTRY
11.13	MPYAT	FILSEG1	AD:=A*T
12.01	ELOCK	FILSEG1	ESCAPE LOCK

12.01	EULOC	FILSEG1	ESCAPE UNLOCK
1.08	USER FILE BUFF	FILSEG2	BUFFER FOR USER ENTRY
1.09	OBJ. FILE BUFF	FILSEG2	BUFFER FOR OBJECT ENTRY
3.01.1	MOCTA	FILSEG2	OUTPUT OCTAL NUMBER ON TERM.
3.01.1	OCTAL	FILSEG2	OUTPUT OCTAL NUMBER ON TERM.
3.01.2	DECIM	FILSEG2	OUTPUT DECIMAL NUMBER ON TERM.
3.01.2	MDECI	FILSEG2	OUTPUT DECIMAL NUMBER ON TERM.
3.01.3	DDECI	FILSEG2	OUTPUT DOUBLE DECIMAL NUMBER
3.01.3	MDDEC	FILSEG2	OUTPUT DOUBLE DECIMAL NUMBER
3.01.3	MTWOD	FILSEG2	OUTPUT TWO DIGITS DECIMAL
3.01.3	TWODE	FILSEG2	OUTPUT TWO DIGITS DECIMAL
3.02.1	OUTRC	FILSEG2	OUTPUT STRING ON TERMINAL
3.02.1	OUTST	FILSEG2	OUTPUT STRING ON TERMINAL
3.03.1	LDATE	FILSEG2	LIST DATE
3.03.1	MDATE	FILSEG2	LIST DATE
3.03.2	LACCW	FILSEG2	LIST ACCESS WORD
3.04	INSTR	FILSEG2	INPUT STRING
3.04	STRNG	FILSEG2	INPUT STRING
3.07.2	APPST	FILSEG2	APPEND STRING TO STRING
3.08	COMPS	FILSEG2	COMPARE STRINGS
3.11	BDUMP	FILSEG2	DUMP BLOCK ON TERMINAL
3.11	DUMP	FILSEG2	DUMP BLOCK ON TERMINAL
3.12	CHANG	FILSEG2	CHANGE BLOCK
3.13	SEPST	FILSEG2	SEPARATE STRING
3.14	SEPPA	FILSEG2	SEPARATE FILE STRING
3.15	SEPFS	FILSEG2	SEPARATE FILE STRING IN THREE
4.03	GDIRI	FILSEG2	GET DIRECTORY INDEX
4.04	GNAMI	FILSEG2	GET NAME INDEX
4.05	GDIRE	FILSEG2	GET DIRECTORY INDEX
4.06	COLDE	FILSEG2	COLLECT DEVICE NAME AND UNIT
4.06	XCOLD	FILSEG2	COLLECT DEVICE NAME AND UNIT
4.07	GMAIN	FILSEG2	GET MAIN DIRECTORY INDEX
4.08	WDIRE	FILSEG2	WRITE DIRECTORY ENTRY
5.01	FBFBU	FILSEG2	FIND BIT FILE BUFFER ADDRESS
5.02	RBFBL	FILSEG2	READ BIT FILE BLOCK
5.03	WBFBL	FILSEG2	WRITE BIT FILE BLOCK
5.04	WBFBU	FILSEG2	WRITE BIT FILE BUFFER
5.05	ALBIT	FILSEG2	FIND BIT FILE ADDRESS
5.06	ALPAG	FILSEG2	ALLOCATE PAGE IN BIT FILE
5.06	RLPAG	FILSEG2	RELEASE PAGE IN BIT FILE
5.07	TPAGF	FILSEG2	TEST PAGE FREE
5.08	RSPAG	FILSEG2	RESERVE FIRST FREE PAGE
6.03	STARS	FILSEG2	START SPOOLING
6.04	STSPL	FILSEG2	STOP SPOOLING
6.05	ABORS	FILSEG2	ABORT SPOOLING PRINT
6.05	RESTS	FILSEG2	RESTART SPOOLING PRINT
6.05	STOPR	FILSEG2	STOP PRINT
6.05	STAPR	FILSEG2	START PRINT
6.06	LSPOQ	FILSEG2	LIST SPOOLING QUEUE
6.07	APPES	FILSEG2	APPEND SPOOLING QUEUE
6.08	DELES	FILSEG2	DELETE SPOOLING FILE
6.08	RMSPF	FILSEG2	REMOVE FROM SPOOL. QUEUE
6.09	GIVES	FILSEG2	GIVE SPOOLING PAGES
6.10	TAKES	FILSEG2	TAKE SPOOLING PAGES
6.11	SPOPL	FILSEG2	NUMBER OF SPOOL. PAGES LEFT
6.12	INPER	FILSEG2	INPUT SPOOLING PERIPHERAL
6.13	FINDQ	FILSEG2	FIND SPOOLING QUEUE
6.14	DEABB	FILSEG2	DEABBREVIATE FILE NAME
6.15	GFILN	FILSEG2	GET FILE NAME

6.16	HEAPRINT	FILSEG2	PRINT SPOOLING HEADER
6.16	TRAPRINT	FILSEG2	PRINT SPOOLING TRAILER
6.17	LOCKQ	FILSEG2	FIND NUMBER OF ELEM. IN QUEUE
6.18	UNLCQ	FILSEG2	UNLOCK QUEUE
6.19	READQ	FILSEG2	READ ONE QUEUE ELEMENT
6.20	WRITQ	FILSEG2	WRITE ONE QUEUE ELEMENT
6.21	APPEQ	FILSEG2	APPEND TO QUEUE
6.22	TAKEQ	FILSEG2	TAKE FROM SPOOLING QUEUE
6.23	INITQ	FILSEG2	INITIALIZE QUEUE
6.24	FPERIV	FILSEG2	FIND PERIPHERAL VERSION
6.25	FFILISQ	FILSEG2	FIND FILE IN SPOOL. QUEUE
6.26	MSPQENT	FILSEG2	MOVE SPOOL. QUEUE ENTRY
6.27	SNSPCOPY	FILSEG2	SET NO. OF PRINT COPIES
6.28	FWSPRINT	FILSEG2	FORWARD SPACE PRINT
6.28	BSPRINT	FILSEG2	BACKSPACE PRINT
6.29	DSCOND	FILSEG2	DEFINE SPOOLING CONDITIONS
7.01.1	TUSSY	FILSEG2	TEST USER SYSTEM
7.01.2	TUSRT	FILSEG2	TEST USER RT
7.02	TUSEN	FILSEG2	TEST USER ENTERED
7.02A	RUSPW	FILSEG2	READ USER PASSWORD
7.03	FUSEB	FILSEG2	FIND USER ENTRY BUFFER
7.04	RUSER	FILSEG2	READ USER ENTRY
7.05	WUSER	FILSEG2	WRITE USER ENTRY
7.06	RUSEB	FILSEG2	RELEASE USER ENTRY
7.07	GUSEI	FILSEG2	GET USER INDEX
7.08	GMUSI	FILSEG2	GET MAIN USER INDEX
7.09	COLUN	FILSEG2	COLLECT USER NAME
7.10	GUSEN	FILSEG2	GET USER NAME
7.11	CUSED	FILSEG2	CHANGE USER SPACE
7.12	GDEFD	FILSEG2	GET DEFAULT DIRECTORY
7.13	GUSAC	FILSEG2	GET USER ACCESS
8.01	FOBJB	FILSEG2	FIND OBJECT ENTRY BUFFER
8.02	ROBJE	FILSEG2	READ OBJECT ENTRY
8.03	WOBJE	FILSEG2	WRITE OBJECT ENTRY
8.04	ROBJB	FILSEG2	RELEASE OBJECT ENTRY BUFFER
8.05	GOBJI	FILSEG2	GET OBJECT INDEX
8.06	SEPOB	FILSEG2	SEPARATE OBJECT NAME
8.07	GFILI	FILSEG2	GET FILE INDEX
8.08	GPREV	FILSEG2	GET PREVIOUS VERSION
8.09	GNEXV	FILSEG2	GET NEXT VERSION
8.10	COBJE	FILSEG2	CREATE OBJECT ENTRY
8.11	CHIGV	FILSEG2	CREATE NEW HIGHER VERSION
8.11	CNEWV	FILSEG2	CREATE NEW VERSION
8.12	CROBJ	FILSEG2	CREATE OBJECTS
8.14	DLOBJ	FILSEG2	DELETE OBJECT
8.15	CRNEW	FILSEG2	CREATE NEW VERSION OF FILE
8.16	GVERS	FILSEG2	GET VERSION NUMBER
8.17	GFIAC	FILSEG2	GET FILE ACCESS
8.18	GCFIL	FILSEG2	GET OR CREATE FILE
8.19	DLPAG	FILSEG2	DELETE PAGES OF FILE
8.19	DLSPA	FILSEG2	DELETE PAGES OF FILE
9.01	FFILE	FILSEG2	FIND FILE TO OPEN
9.02	FOFT	FILSEG2	FIND OPEN FILE TABLE
9.03	SOFT	FILSEG2	SET UP OPEN FILE TABLE
9.03A	OFRND	FILSEG2	OPEN FILE FOR RANDOM ACCESS
9.04	FCON	FILSEG2	FILE CONNECT
9.05	FOPEN	FILSEG2	FILE OPEN
9.06	FCLOS	FILSEG2/RESIDENT	FILE CLOSE
9.06	XFCLOS	FILSEG2	FILE CLOSE (NO VERSION CHANGE)

9.07	GBUF	FILSEG2	GET BUFFER FROM POOL
9.07	GBUFS	FILSEG2	GET BUFFER SET FROM POOL
9.08	RBUF	FILSEG2	RETURN BUFFER TO POOL
9.09	SBLSZ	FILSEG2	SET BLOCK SIZE
9.10	SETPO	FILSEG2	SET PERMANENT OPEN
9.18	RBYTE	FILSEG2	READ BYTE POINTER
9.18	RMAXB	FILSEG2	READ MAX POINTER
9.18	SBYTE	FILSEG2	SET BYTE POINTER
9.18	SMAXB	FILSEG2	SET MAX POINTER
9.19	SBLOP	FILSEG2	SET BLOCK POINTER
9.20	SDATF	FILSEG2	SET DATAFIELD RESERVED
9.21	CDATF	FILSEG2	CLEAR DATAFIELD RESERVED
9.22	OPSCR	FILSEG2	OPEN SCRATCH FILE
9.23	CPFIL	FILSEG2	COPY FILE
9.24	COLFI	FILSEG2	COLLECT FILE NAME
9.25	CLOUT	FILSEG2	CLOSE OUTPUT FILE
9.26	REMOPFI	FILSEG2	REMOTE OPEN FILE
9.27	NBAVA	FILSEG2	WAIT FOR ANSWER ON REMOTE 0B.
10.02	CRDIR	FILSEG2	CREATE DIRECTORY
10.03	RNDIR	FILSEG2	RENAME DIRECTORY
10.04	ENDIR	FILSEG2	ENTER DIRECTORY
10.05	RLDIR	FILSEG2	RELEASE DIRECTORY
10.06	SDDIR	FILSEG2	SET DEFAULT DIRECTORY
10.07	DIRST	FILSEG2	DIRECTORY STATISTICS
10.07	LIDIR	FILSEG2	LIST DIRECTORIES ENTERED
10.08	DUDIR	FILSEG2	DUMP DIRECTORY ENTRY
10.09	CHDIR	FILSEG2	CHANGE DIRECTORY ENTRY
10.10	CRUSE	FILSEG2	CREATE USER
10.11	RNUSE	FILSEG2	RENAME USER
10.12	DLUSE	FILSEG2	DELETE USER
10.13	GIUSE	FILSEG2	GIVE USER SPACE
10.14	TAUSE	FILSEG2	TAKE USER SPACE
10.15	LIUSE	FILSEG2	LIST USERS
10.15	USEST	FILSEG2	USER STATISTICS
10.16	DUUSE	FILSEG2	DUMP USER ENTRY
10.17	CHUSE	FILSEG2	CHANGE USER ENTRY
10.18	ENUSE	FILSEG2	ENTER USER
10.19	RLUSE	FILSEG2	RELEASE USER
10.20	CHANP	FILSEG2	CHANGE PASSWORD
10.21	CLPAS	FILSEG2	CLEAR PASSWORD
10.22	CRFRI	FILSEG2	CREATE FRIEND
10.23	DLFRI	FILSEG2	DELETE FRIEND
10.24	SFRIA	FILSEG2	SET FRIEND ACCESS
10.25	LIFRI	FILSEG2	LIST FRIENDS
10.26	CRFIL	FILSEG2	CREATE FILE
10.26	CRNVE	FILSEG2	CREATE NEW FILE VERSION
10.27	ALFIL	FILSEG2	ALLOCATE FILE
10.27	ALNVE	FILSEG2	ALLOCATE NEW FILE VERSION
10.28	EXFIL	FILSEG2	EXPAND FILE
10.30	RNFIL	FILSEG2	RENAME FILE
10.31	DLFIL	FILSEG2	DELETE FILE
10.32	STERF	FILSEG2	SET TERMINAL FILE
10.32	STMPF	FILSEG2	SET TEMPORARY FILE
10.33	SPERF	FILSEG2	SET PERIPHERAL FILE
10.34	SFLAC	FILSEG2	SET FILE ACCESS
10.34A	SDFIA	FILSEG2	SET DEFAULT FILE ACCESS
10.35	DEUFI	FILSEG2	DELETE USERS FILES
10.35	FILST	FILSEG2	FILE STATISTICS
10.35	LIFIL	FILSEG2	LIST FILES

10.36	DUOBJ	FILSEG2	DUMP OBJECT ENTRY
10.37	CHOBJ	FILSEG2	CHANGE OBJECT ENTRY
10.38	OPENF	FILSEG2	OPEN FILE
10.39	CONNF	FILSEG2	CONNECT FILE
10.40	CLOSF	FILSEG2	CLOSE FILE
10.41	LIOPF	FILSEG2	LIST OPENED FILES
10.41	LIRTO	FILSEG2	LIST RT OPENED FILES
10.42	SBLOS	FILSEG2	SET BLOCK SIZE
10.43	SPERO	FILSEG2	SET PERMANENT OPENED
10.44	SBYTP	FILSEG2	SET BYTE POINTER
10.45	SBLOC	FILSEG2	SET BLOCK POINTER
10.46	RESFI	FILSEG2	RESERVE FILE
10.47	RELFI	FILSEG2	RELEASE FILE
10.48	WHEFI	FILSEG2	WHERE IS FILE
10.49	OPRTF	FILSEG2	OPEN RT FILE
10.50	CORTF	FILSEG2	CONNECT RT FILE
10.51	CLRTF	FILSEG2	CLOSE RT FILE
10.51	OPENS	FILSEG2	OPEN SCRATCH FILE
10.52	DUPAG	FILSEG2	DUMP PAGE
10.53	CHPAG	FILSEG2	CHANGE PAGE
10.54	DUBIT	FILSEG2	DUMP BIT TABLE
10.55	CHBIT	FILSEG2	CHANGE BIT TABLE
10.56	REGDI	FILSEG2	REGENERATE DIRECTORY
10.56	TESDI	FILSEG2	TEST DIRECTORY
10.57	COPDI	FILSEG2	COPY DIRECTORY
10.57	COPFI	FILSEG2	COPY FILE
10.58	RELTU	FILSEG2	RELEASE DEVICE UNIT
10.58	RESTU	FILSEG2	RESERVE DEVICE UNIT
10.59	SAVDI	FILSEG2	SAVE DIRECTORY
10.60	CREVOL	FILSEG2	CREATE VOLUME
10.61	LIVOL	FILSEG2	LIST VOLUME
10.62	CPUFIL	FILSEG2	COPY USERS FILES
10.63	CLPRY	FILSEG2	CLEAR PARITY IN TAPE LABEL
12.04.3	CMMON	FILSEG2	COMMAND MONITOR
12.04.4	CLPAR	FILSEG2	COLLECT PARAMETER
12.04.5	ERROR	FILSEG2	WRITE ERROR MESSAGE
12.05.1	INITF	FILSEG2	INITIATE FILE SYSTEM TABLES
12.06	GDATE	FILSEG2	GET DATE
2.00	G3BUF	RESIDENT	GET MASS STORAGE BUFFER
2.00	G3IBUF	RESIDENT	GET MASS STORAGE BUFFER
2.00	G3NWT	RESIDENT	GET MASS STORAGE BUFFER
2.00	G5BUF	RESIDENT	GET MASS STORAGE BUFFER
2.00	R3BUF	RESIDENT	RELEASE MASS STORAGE BUFFER
2.00	R3IBUF	RESIDENT	RELEASE MASS STORAGE BUFFER
2.00	R5BUF	RESIDENT	RELEASE MASS STORAGE BUFFER
3.05	GETCH	RESIDENT	GET CHARACTER FROM STRING
3.05	PUTCH	RESIDENT	PUT CHARACTER TO STRING
3.07.1	ACOPY	RESIDENT	COPY STRING (ALT. PAGE TABLE)
3.07.1	COPYS	RESIDENT	COPY STRING
3.09	SETBL	RESIDENT	SET BLOCK CONTENTS
3.10	COPYB	RESIDENT	COPY BLOCK
4.01	GDIRA	RESIDENT	GET DIRECTORY ADDRESS
4.02	GNAMA	RESIDENT	GET NAME TABLE ADDRESS
9.13	FLYTT	RESIDENT	MOVE 100 WORDS
9.16	FREA	RESIDENT/FILSEG1	FILE READ
9.17	FWRT	RESIDENT/FILSEG1	FILE WRITE
11.07	OLDOP	RESIDENT/FILSEG1	OLD OPEN FILE
11.07	OPFIL	RESIDENT/FILSEG1	OPEN FILE
11.09	SETUP	RESIDENT	STRING DESCRIPTOR SET UP

11.09	SETW	RESIDENT	SET WRITE POINTER OF STRING
11.09	WCI	RESIDENT	WRITE BYTE TO STRING
11.11	MROBJ	RESIDENT/FILSEG1	READ OBJECT ENTRY
11.12	MRUSE	RESIDENT/FILSEG1	READ USER ENTRY
11.15	RSPQE	RESIDENT	READ SPOOLING QUEUE ENTRY
12.01	FATAL	RESIDENT	FATAL ERROR
12.01	LOCK	RESIDENT	LOCK SEMAPHORE
12.01	UNLOC	RESIDENT	UNLOCK SEMAPHORE
12.02.3	WHERE	RESIDENT	WHERE IS SEMAPHORE
12.03.1	CABST	RESIDENT	CARTRIDGE DISC ABSTRANS
12.03.2	DRABS	RESIDENT	DRUM ABSTRANS
12.03.3	BABST	RESIDENT	BIG DISC ABSTRANS
12.03.4	MABST	RESIDENT	MAG TAPE ABSTRANS
12.03.5	FDABS	RESIDENT	FLOPPY DISC ABSTRANS
12.07.4	SINBT	RESIDENT	INPUT BYTE TO FILE SYSTEM
12.07.4	SOUTBT	RESIDENT	OUTPUT BYTE FROM FILE SYSTEM
1.06.1	SPOP	SYSEG	POP SUBROUTINE STACK
1.06.1	SPUSH	SYSEG	PUSH SUBROUTINE STACK
1.06.1	SUBR. STACK	SYSEG	ENTER/LEAVE STACK
1.06.3	BUFFER POOL	SYSEG	
11.01	FINBT	SYSEG	INPUT BYTE
11.01	INBT	SYSEG	INPUT BYTE
11.01	SBINBT	SYSEG	INPUT BYTE
11.02	FOUTBT	SYSEG	OUTPUT BYTE
11.02	OUTBT	SYSEG	OUTPUT BYTE
11.02	SBOUTBT	SYSEG	OUTPUT BYTE
12.08	OPCAL	SYSEG	CALL ROUTINE ON OP.COM.SEG

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

COMMENT AND EVALUATION SHEET

NORD FILE SYSTEM – System Documentation ND-60.122.02
January 1980

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 preaddressed form and mail it. Please be specific wherever possible.

FROM .....

.....

.....