STRUCTURE OF iRMX 86™ NAMED FILE VOLUMES

An Intel Technical Specification

Manual Number: 143308-001

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

A 292/181/IK IP

PREFACE

This document describes the structure of an iRMX 86 volume that contains named files. Those users who wish to examine named file volumes or create their own formatting utility programs can use this informtion.

READER LEVEL

This technical specification is intended for system programmers who have had experience in reading and writing actual volume information. It does not attempt to teach the reader these functions.

RELATED PUBLICATIONS

The following manuals provide additional background and reference information about the iRMX 86 Operating System.

Manual	Number
iRMX 86 Nucleus, Terminal Handler, and Debugger Reference Manual	9803122
iRMX 86 I/O System and Loader Reference Manual	9803123
iRMX 86 System Programmer's Reference Manual	142721
Guide to Writing Device Drivers for the iRMX 86 I/O System	142926
iRMX 86 Configuration Guide for ISIS-II Users	9803126

CONTENTS

FIGURES

CHAPTER 1. INTRODUCTION

Each iRMX 86 named file volume contains ISO (International Organization for Standardization) label information as well as iRMX 86 label information and files. Figure 1-1 illustrates the general structure of a named file volume.

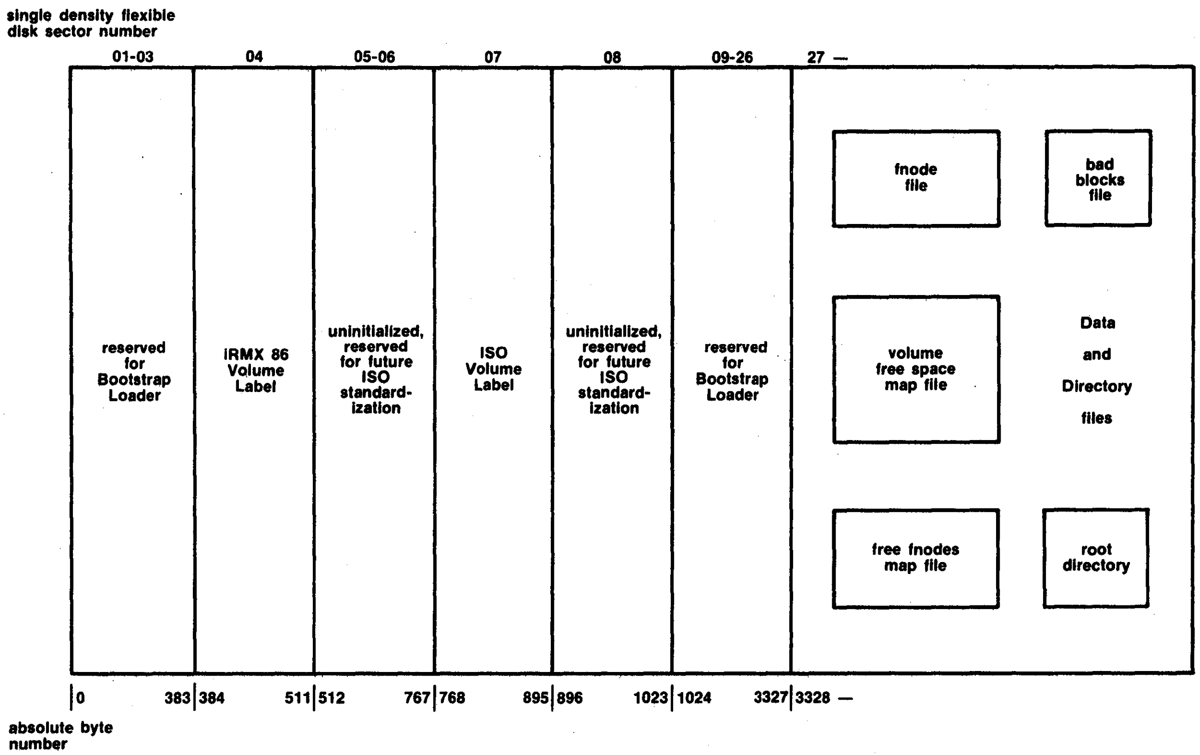

Figure 1-1. General Structure of Named File Volumes

This specification discusses the structure in more detail. It includes information concerning the following:

- ISO Volume Label
- iRMX 86 Volume Label
- fnode file
- volume free space map file
- free fnodes map file
- bad blocks file
- root directory

It also discusses the structure of directory files and the concepts of long and short files.

The blocks in Figure 1-1 that are reserved for the Bootstrap Loader are not discussed. To include these blocks on a new volume that you are formatting, you should copy them from an already formatted volume.

NOTE

The following chapters of this specification refer to a data type called DWORD. DWORD must be declared literally as POINTER. This results in a 32-bit variable for the PLM/86 models COMPACT, MEDIUM, and LARGE.

CHAPTER 2. VOLUME LABELS

This chapter describes the structure of the volume labels that must be present on a named file volume. These labels are the ISO volume label and the iRMX 86 volume label.

ISO VOLUME LABEL

The ISO (International Organization for Standardization) volume label is recorded in absolute byte positions 768 through 895 of the volume (for example, sector 07 of a single density flexible diskette). The structure of this volume label is as follows:

```
DECLARE
     ISO$VOL$LABEL  STRUCTURE(
          LABEL$ID(3)         BYTE,
          RESERVED$A          BYTE,
          VOL$NAME(6)         BYTE,
          VOL$STRUC           BYTE,
          RESERVED$B(60)      BYTE,
          REC$SIDE            BYTE,
          RESERVED$C(4)       BYTE,
          ILEAVE(2)           BYTE,
          RESERVED$D          BYTE,
          ISO$VERSION         BYTE,
          RESERVED$E(48)      BYTE);
```

Where:

LABEL$ID(3)	Label identifier. For named file volumes, this field contains the ASCII characters "VOL".
RESERVED$A	Reserved field containing the ASCII character "1".
VOL$NAME(6)	Volume name. This field can contain up to six printable ASCII characters, left justified and space filled. A value of all spaces implies that the volume name is recorded in the iRMX 86 Volume Label (absolute byte positions 384-393).
VOL$STRUC	For named file volumes, this field contains the ASCII character "N", indicating that this volume has a non-ISO file structure.
RESERVED$B(60)	This is a reserved field containing 60 bytes of ASCII spaces.

REC$SIDE	For named file volumes, this field contains the ASCII character "1" to indicate that only one side of the volume is to be recorded.
RESERVED$C(4)	This is a reserved field containing four bytes of ASCII spaces.
ILEAVE(2)	Two ASCII digits indicating the interleave factor for the volume, in decimal. ASCII digits consist of the numbers 0 through 9. When formatting named file volumes, you should set this field to the same interleave factor that you use when physically formatting the volume.
RESERVED$D	This is a reserved field containing an ASCII space.
ISO$VERSION	For named file volumes, this field contains the ASCII character "1", which indicates ISO version number one.
RESERVED$D(48)	This is a reserved field containing 48 ASCII spaces.

iRMX 86 VOLUME LABEL

The iRMX 86 Volume Label is recorded in absolute byte positions 384 through 511 of the volume (sector 04 of a single density flexible diskette). The structure of this volume label is as follows:

```
DECLARE
    RMX$VOLUME$INFORMATION  STRUCTURE(
        VOL$NAME(10)        BYTE,
        FILL                BYTE,
        FILE$DRIVER         BYTE,
        VOL$GRAN            WORD,
        VOL$SIZE            DWORD,
        MAX$FNODE           WORD,
        FNODE$START         DWORD,
        FNODE$SIZE          WORD,
        ROOT$FNODE          WORD);
```

Where:

VOL$NAME(10)	Volume name in printable ASCII characters, left justified and space filled.
FILL	Reserved field which is set to zero.
FILE$DRIVER	Number of the file driver used with this volume. For named file volumes, this field is set to four.

VOL$GRAN	Volume granularity, specified in bytes. This value must be a multiple of the device granularity. It sets the size of a logical device block, also called a volume block.
VOL$SIZE	Size of the entire volume, in bytes.
MAX$FNODE	Number of fnodes in the fnode file. Refer to the next section for a description of fnodes.
FNODE$START	A 32-bit value which represents the number of the first byte in the fnode file (byte 0 is the first byte of the volume).
FNODE$SIZE	Size of an fnode, in bytes.
ROOT$FNODE	Number of the fnode describing the root directory. Refer to the next section for further information.

The remainder of the Volume Label (bytes 412 through 511) is reserved and must be set to zero.

CHAPTER 3. INITIAL FILES

Any mechanism that formats iRMX 86 named volumes must place five files on the volume during the format process. These five files are the fnode file, the volume free space map file, the free fnodes map file, the bad blocks file, and the root directory. The first of these files, the fnode file, contains information about all of the files on the volume. The general structure of the fnode file is discussed first. Then all of the files are discussed in terms of their fnode entries and their functions.

FNODE FILE

A data structure called a file descriptor node (or fnode) describes each file in a named file volume. All the fnodes for the entire volume are grouped together in a file called the fnode file. When the I/O System accesses a file on a named volume, it examines the iRMX 86 Volume Label (described in the previous section) to determine the location of the fnode file, and then examines the appropriate fnode to determine the actual location of the file.

When a volume is formatted, the fnode file contains six allocated fnodes. These fnodes represent the fnode file, the volume free space map file, the free fnodes map file, the bad blocks file, the root directory, and one other file. Later sections of this chapter describe these files. The size of the fnode file is determined by the number of fnodes that it contains. The number of fnodes in the fnode file also determines the number of files that can be created on the volume.

NOTE

> When formatting a volume, you may be able to improve performance by placing the fnode file in the middle of the volume. By doing this, you reduce the average latency by 50%. For applications that have heavy file access, this may be desirable. However, the fnode file must start on a volume block boundary.

The structure of an individual fnode in a named file volume is as follows:

```
DECLARE
     FNODE      STRUCTURE(
          FLAGS          WORD,
          TYPE           BYTE,
          GRAN           BYTE,
          OWNER          WORD,
          CR$TIME        DWORD,
          ACCESS$TIME    DWORD,
          MOD$TIME       DWORD,
          TOTAL$SIZE     DWORD,
          TOTAL$BLKS     DWORD,
          PTR(8)         STRUCTURE(
               NUM$BLOCKS     WORD,
               BLK$PTR(3)     BYTE),
          THIS$SIZE      DWORD,
          RESERVED$A     WORD,
          RESERVED$B     WORD,
          ID$COUNT       WORD,
          ACCESSOR(3)    STRUCTURE(
               ACCESS         BYTE,
               ID             WORD),
          PARENT         WORD,
          AUX(*)         BYTE);
```

Where:

FLAGS — A WORD which defines a set of attributes for the file. The individual bits in this word indicate the following attributes (bit 0 is the rightmost bit):

Bit	Meaning
0	Allocation status. If set to one, this fnode describes an actual file. If set to zero, this fnode is available for allocation. When formatting a volume, this bit is set to one in the six allocated fnodes. In other fnodes, it is set to zero.
1	Long or short file attribute. This bit describes how the PTR fields of the fnode are interpreted. If set to zero, indicating a short file, the PTR fields identify the actual data blocks of the file. If set to one, indicating a long file, the PTR fields identify indirect blocks. Indirect blocks are described later in this section. When formatting a volume, this bit is always set to zero, since the initial files on the volume are short files.

Bit	Meaning
2	Reserved bit which is always set to one.
3-4	Reserved bits which are always set to zero.
5	Modification attribute. Whenever a file is modified, this bit is set to one. Initially, when a volume is formatted, this bit is set to zero in each fnode.
6	Deletion attribute. This bit is set to one to indicate that the file is a temporary file or that the file is going to be deleted (the deletion may be postponed because additional connections exist to the file). Initially, when the volume is formatted, this bit is set to zero in each fnode.
7-15	Reserved bits which are always set to zero.

TYPE

Type of file. The following are acceptable types:

Mnemonic	Value	Type
FT$FNODE	0	fnode file
FT$VOLMAP	1	volume free space map
FT$FNODEMAP	2	free fnodes map
FT$ACCOUNT	3	space accounting file
FT$BADBLOCK	4	bad device blocks file
FT$DIR	6	directory file
FT$DATA	8	data file

During system operation, only the I/O System can access file types other than FT$DATA and FT$DIR. These file types are discussed later in this section.

GRAN

File granularity, specified in multiples of the volume granularity. The default value is 1. For the files initially present on the volume (fnode file, volume free space map file, free fnodes map file, bad blocks file, root directory), this value can be set to any multiple of the volume granularity.

OWNER — User ID of the owner of the file. For the files initially present on the volume, this parameter is important only for the root directory. For the root directory, this parameter should specify the user WORLD (FFFFH). The I/O System does not examine this parameter for the other files (fnode file, volume free space map file, free fnodes map file, bad blocks file) and so a value of zero can be specified.

CR$TIME — Time and date that the file was created, expressed as a 32-bit value. This value indicates the number of seconds since a fixed, user-determined point in time. By convention, this point in time is 12:00 A.M., January 1, 1978. For the files initially present on the volume, this parameter is important only for the root directory. A zero can be specified for the other files (fnode file, volume free space map file, free fnodes map file, bad blocks file).

ACCESS$TIME — Time and date of the last file access (read or write), expressed as a 32-bit value. For the files initially present on the volume, this parameter is important only for the root directory.

MOD$TIME — Time and date of the last file modification, expressed as a 32-bit value. For the files initially present on the volume, this parameter is important only for the root directory.

TOTAL$SIZE — Total size, in bytes, of the actual data in the file.

TOTAL$BLKS — Total number of volume blocks used by this file, including indirect block overhead. A volume block is a block of data whose size is the same as the volume granularity. All memory in the volume is divided into volume blocks, which are numbered sequentially, starting with the block containing the smallest addresses (block 0). Indirect blocks are discussed later in this section.

PTR(8) — These structures locate the data blocks of the file. These data blocks may be scattered throughout the volume, but together they make up a complete file. If the file is a short file (bit 1 of the FLAGS field is set to zero), each PTR structure identifies an actual data block. In this case, the fields of the PTR structure contain the following:

NUM$BLOCKS — Number of volume blocks in the data block.

BLK$PTR(3) — A 24-bit value specifying the number of the first volume block in the data block. Volume blocks are numbered sequentially, starting with the block with the smallest address (block 0). The bytes in the BLK$PTR array range from least significant (BLK$PTR(0)) to most significant (BLK$PTR(2)).

If the file is a long file (bit 1 of the FLAGS field is set to one), each PTR structure identifies an indirect block, which in turn identifies the data blocks of the file. In this case, the fields of the PTR structure contain the following:

NUM$BLOCKS — Number of volume blocks pointed to by the indirect block.

BLK$PTR(3) — A 24-bit volume block number of the indirect block.

Indirect blocks are discussed later in this section.

THIS$SIZE — Size, in bytes, of the total data space allocated to the file. This figure does not include space used for indirect blocks, but it does include any data space allocated to the file, regardless of whether the file fills that allocated space.

RESERVED$A — Reserved field which is set to zero.

RESERVED$B — Reserved field which is set to zero.

ID$COUNT — Number of access-ID pairs declared in the ACCESSOR structure.

ACCESSOR(3) — This structure contains the access-ID pairs which define the access rights for the users of the file. By convention, when a file is created, the owning user's ID is inserted in ACCESSOR(0), along with the code for the access rights. The fields of the ACCESSOR structure contain the following:

ACCESS — Encoded access rights for the file. The settings of the individual bits in this field grant (if set to one) or deny (if set to zero) permission for the corresponding operation. Bit 0 is the rightmost bit.

Bit	Data File Operation	Directory Operation
0	delete	delete
1	read	display
2	append	add entry
3	update	change entry
4-7	reserved (must be 0)	

ID — ID of the user who gains the corresponding access permission.

PARENT — Fnode number of directory file which lists this file. For files initially present on the volume, this parameter is important only for the root directory. For the root directory, this parameter should specify the number of the root directory's own fnode. For other files (fnode file, volume free space map file, free fnodes map file, bad blocks file) the I/O System does not examine this field.

AUX(*) — Auxiliary bytes associated with the file. The named file driver does not interpret this field, but the user can access it by making GET$EXTENSION$DATA and SET$EXTENSION$DATA system calls (refer to the iRMX 86 SYSTEM PROGRAMMER'S REFERENCE MANUAL). The size of this field is determined by the size of the fnode, which is specified in the iRMX 86 Volume Label. The Files Utility allocates three bytes for this field by default. If you format your own volume, you can make this field as large as you wish; however, a larger AUX field implies slower file access.

Certain fnodes designate special files that appear on the volume. The following sections discuss these fnodes and the associated files.

FNODE 0 (FNODE FILE)

The first fnode structure in the fnode file describes the fnode file itself. This file contains all the fnode structures for the entire volume. It must reside in contiguous locations in the volume. Fields of fnode 0 must be set as follows:

- The bits in the FLAGS field are set to the following (bit 0 is the rightmost bit):

Bit	Value	Description
0	1	Allocated file
1	0	Short file
2	1	Primary fnode
3-4	0	Reserved bits
5	0	Initial status is unmodified
6	0	File will not be deleted
7-15	0	Reserved bits

- The TYPE field is set to FT$FNODE.
- The GRAN field is set to 1.
- The OWNER field is set to 0.
- The CR$TIME, ACCESS$TIME, and MOD$TIME fields are set to 0.
- Since the iRMX 86 Volume Label specifies the size of an individual fnode structure and the number of fnodes in the fnode file, the value specified in the TOTAL$SIZE field of fnode 0 must equal the product of the values in the FNODE$SIZE and MAX$FNODE fields of the iRMX 86 Volume Label.
- The TOTAL$BLOCKS field specifies enough volume blocks to account for the memory listed in the TOTAL$SIZE field. The product of the value in the TOTAL$BLOCKS field and the volume granularity equals the value of the THIS$SIZE field, since the fnode file is a short file.
- Since the fnode file must reside in contiguous locations in the volume, only one PTR structure describes the location of the file. The value in the NUM$BLOCKS field of that PTR structure equals the value in the TOTAL$BLOCKS field. The BLK$PTR field indicates the number of the first block of the fnode file.
- The ID$COUNT field is set to zero, indicating that no users can access the file.

FNODE 1 (VOLUME FREE SPACE MAP FILE)

The second fnode, fnode 1, describes the volume free space map file. The TYPE field for fnode 1 is set to FT$VOLMAP to designate the file as such.

The volume free space map file keeps track of all the space on the volume. It is a bit map of the volume, in which each bit represents one volume block (a block of space whose size is the same as the volume granularity). If a bit in the map is set to one, the corresponding volume block is free to be allocated to any file. If a bit in the map is set to zero, the corresponding volume block is already allocated to a file. The bits of the map correspond to volume blocks such that bit n of byte m represents volume block (8 * m) + n.

When the volume is formatted, the volume free space map file indicates that the first 3328 bytes of the volume (the label and bootstrap information) plus any files initially placed on the volume (fnode file, volume free space map file, free fnodes map file, bad blocks file) are allocated.

FNODE 2 (FREE FNODES MAP FILE)

The third fnode, fnode 2, describes the free fnodes map file. The TYPE field of fnode 2 is set to FT$FNODEMAP to designate the file as such.

The free fnodes map file keeps track of all the fnodes in the fnodes file. It is a bit map in which each bit represents an fnode. If a bit in the map is set to one, the corresponding fnode is not in use and does not represent an actual file. If a bit in the map is set to zero, the corresponding fnode already describes an existing file. The bits in the map correspond to fnodes such that bit n of byte m represents fnode number (8 * m) + n.

When the volume is formatted, the free fnodes map file indicates that fnodes 0, 1, 2, 3, and 4 are in use. If other files are initially placed on the volume, the free fnodes map file must be set to indicate this as well.

FNODE 4 (BAD BLOCKS FILE)

The fifth fnode, fnode 4, contains all the bad blocks on the volume. The TYPE field of fnode 4 is set to FT$BADBLOCK to indicate this.

If there are any unusable blocks on a volume, this fnode must be initialized to describe a file which consists of all such bad blocks. If there are no bad blocks on the volume, the fnode must still be set up as allocated, and of the indicated type, but it should not assign any actual space for the file.

ROOT DIRECTORY

The root directory is a special directory file. It is the root of the named file heirarchy for the volume. The iRMX 86 Volume Label specifies the fnode number of the root directory. The root directory is its own parent. That is, the PARENT field of its fnode specifies its own fnode number.

The root directory (and all directory files) associates file names with fnode numbers. It consists of a number of entries that have the following structure:

```
DECLARE
     DIR$ENTRY        STRUCTURE(
          FNODE            WORD,
          COMPONENT(14)    BYTE);
```

Where:

FNODE	Fnode number of a file listed in the directory.
COMPONENT(14)	A string of ASCII characters that is the final component of the path name identifying the file. This string is left justified and null padded to 14 characters.

When a file is deleted, its fnode number in the directory entry is set to zero.

OTHER FNODES

When a volume is formatted, one other fnode is set up, representing a file of type FT$ACCOUNT, The fnode is set up as allocated, and of the indicated type, but it does not assign any actual space for the file.

When formatting a volume, no other fnodes in the fnode file represent actual files. The remaining fnodes must have bit zero (allocation status) set to zero.

CHAPTER 4. LONG AND SHORT FILES

A file on a volume is not necessarily one contiguous string of bytes. In many cases, it consists of several contiguous blocks of data scattered throughout the volume. The fnode for the file indicates the locations and sizes of these blocks in one of two ways, as short files or as long files.

SHORT FILES

If the file consists of eight or less distinct blocks of data, its fnode can specify it as a short file. The fnode for a short file has bit 1 of the FLAGS field set to zero. This indicates to the I/O System that the PTR structures of the fnode identify the actual data blocks that make up the file. Figure 4-1 illustrates an fnode for a short file. Decimal numbers are used in the figure for clarity.

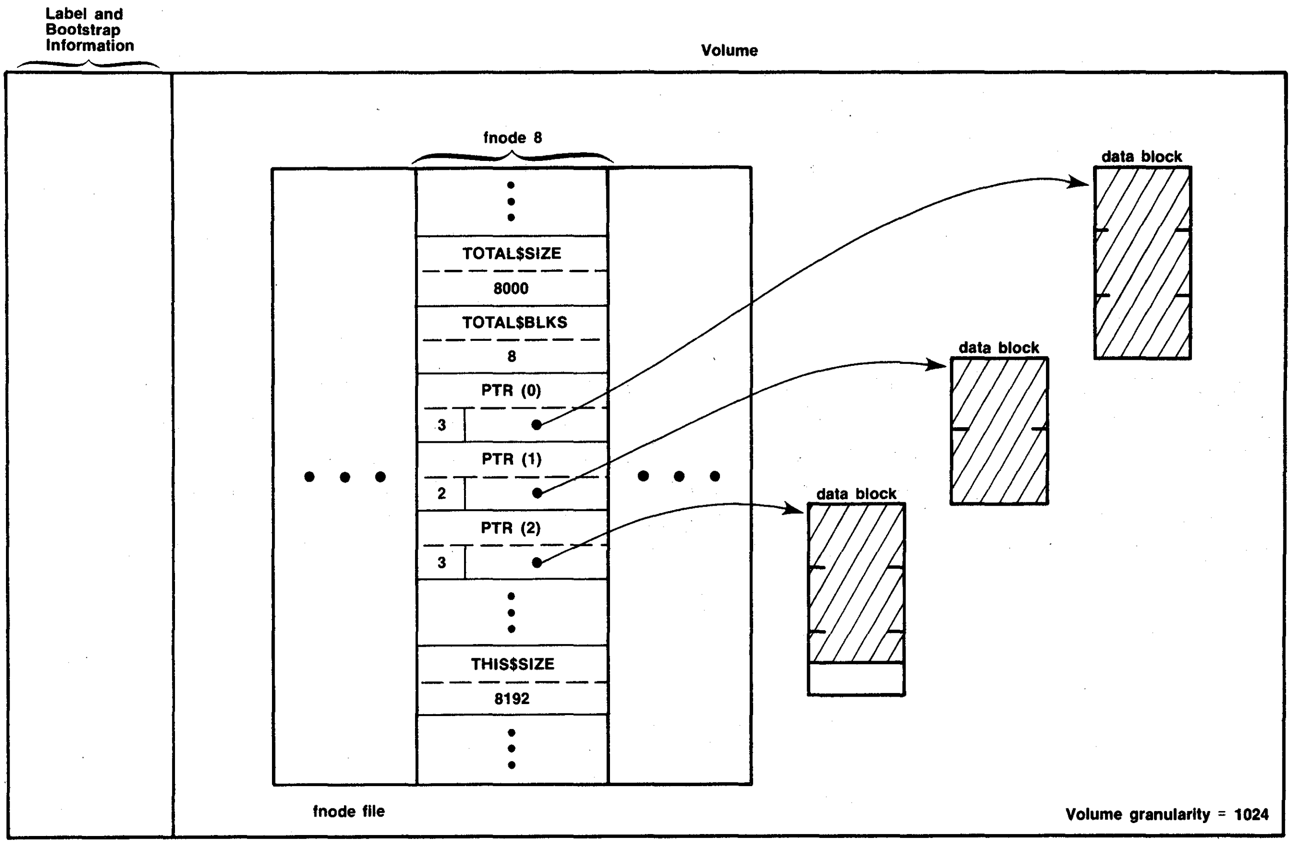

Figure 4-1. Short File Fnode

As you can see from Figure 4-1, fnode 8 identifies the short file. The file consists of three distinct data blocks. Three PTR structures give the locations of the data blocks. The NUM$BLOCKS field of each PTR structure gives the length of the data block (in volume blocks) and the BLK$PTR field points to the first volume block of the data block.

The other fields shown in Figure 4-1 include TOTAL$BLKS, THIS$SIZE, and TOTAL$SIZE. The TOTAL$BLKS field specifies the number of volume blocks allocated to the file, which in this case is nine. This equals the sum of NUM$BLOCKS values (3 + 2 + 3), since short files use all allocated space as data space.

The THIS$SIZE field specifies the number of bytes of data space allocated to the file. This is the sum of the NUM$BLOCKS values (3 + 2 + 3) multiplied by the volume granularity (1024) and equals 8192.

The TOTAL$SIZE field specifies the number of bytes of data space that the file occupies. This is designated in Figure 4-1 by the shaded area. As you can see, the file does not occupy all the space allocated for it, and so the TOTAL$SIZE value (8000) is not as large as the THIS$SIZE value.

LONG FILES

If the file consists of more than eight distinct blocks of data, its fnode must specify it as a long file. The fnode for a long file has bit 1 of the FLAGS field set to one. This tells the I/O System that the PTR structures of the fnode identify indirect blocks. The indirect blocks identify the actual data blocks that make up the file.

An indirect block consists of a number of indirect pointers, which are structures similar to the PTR structures. However, an indirect block can contain more than eight structures and thus can point to more than eight data blocks. The structure of each indirect pointer is as follows:

```
DECLARE
     IND$PTR    STRUCTURE(
          NBLOCKS   BYTE,
          BLK$PTR   BLOCK$NUM);
```

Where:

NBLOCKS	Number of volume blocks in the data block.
BLK$PTR	A 24-bit volume block number of first volume block in the data block. Volume blocks are numbered sequentially throughout the volume, starting with the block with the smallest address (block 0).

Figure 4-2 illustrates an fnode for a long file. Decimal numbers are used in the figure for clarity.

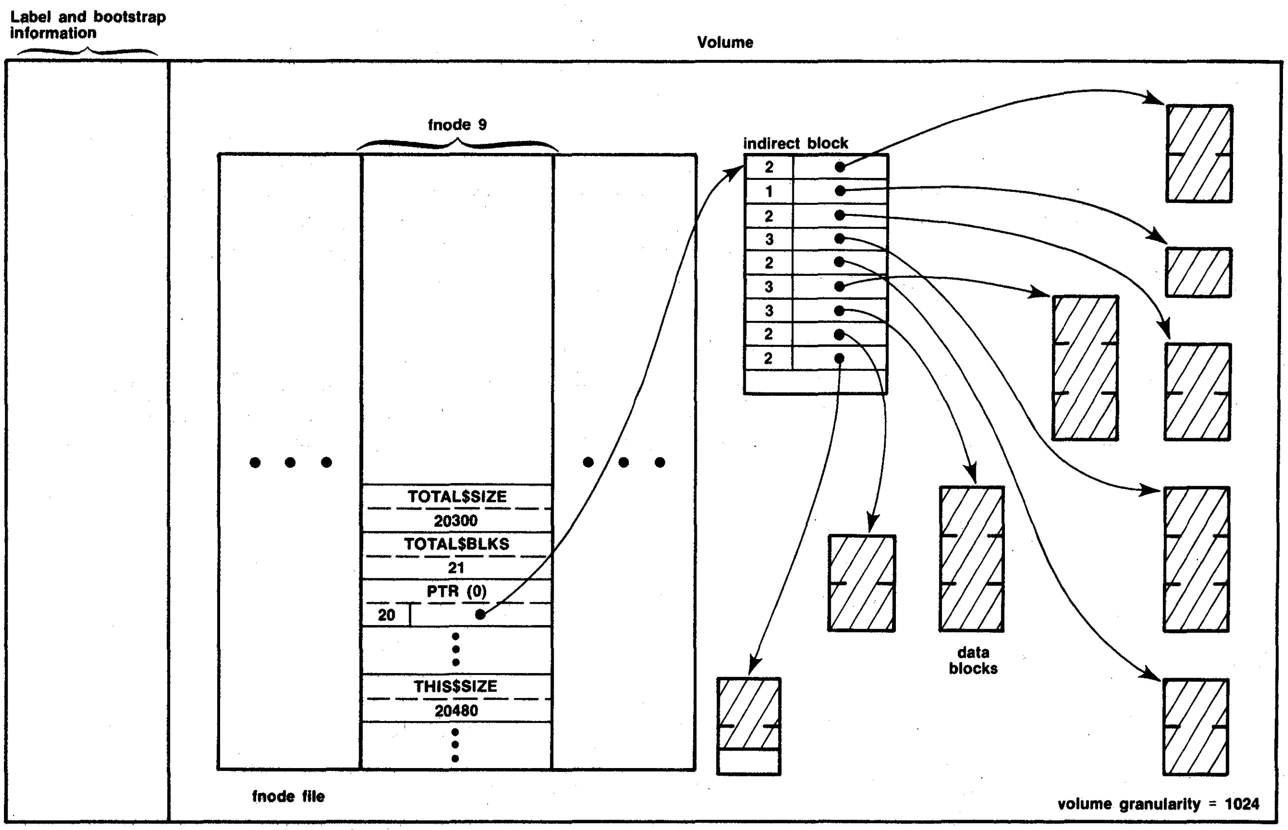

Figure 4-2. Long File Fnode

As you can see from Figure 4-2, fnode 9 identifies the long file. The actual file consists of nine distinct data blocks. One PTR structure and an indirect block give the locations of the data blocks. The NUM$BLOCKS field of the PTR structure contains the number of volume blocks pointed to by the indirect block. The BLK$PTR field points to the first volume block of the indirect block.

In the indirect block, each NBLOCKS field gives the length of an individual data block and each BLK$PTR field points to the first volume block of a data block.

Figure 4-2 also lists the TOTAL$BLKS, THIS$SIZE, and TOTAL$SIZE values, which are more complex than for a short file. The TOTAL$BLKS field specifies the number of volume blocks allocated to the file, which in this case is 21. Twenty of the volume blocks are used for actual data storage and one of the blocks is used for the indirect block.

The THIS$SIZE field specifies the number of bytes of data space allocated to the file, and does not include the size of the indirect block. This size is equal to the NUM$BLOCKS value (20) or the sum of NBLOCKS values in the indirect block (2 + 1 + 2 + 3 + 2 + 3 + 3 + 2 + 2 = 20) multiplied by the volume granularity (1024) and equals 20480.

The TOTAL$SIZE field specifies the number of bytes of data space that the file currently occupies. This is designated in Figure 4-2 by the shaded areas. As you can see, the file does not occupy all the space allocated for it, and so the TOTAL$SIZE value (20300) is not as large as the THIS$SIZE value.

CHAPTER 5. EXAMPLE VOLUME

This chapter lists the labels, fnode file, volume free space map file, free fnode map file, and root directory of a single density diskette which has been formatted by using Files Utility FORMAT command with default parameters. Refer to the iRMX 86 INSTALLATION GUIDE FOR ISIS-II USERS for further information about the Files Utility. This volume also contains one additional file whose fnode is shown.

ISO VOLUME LABEL

The following lists the individual fields of the ISO Label. Each two-digit number represents one byte, and thus one ASCII character. This label begins with byte number 768 of the diskette.

field	value (hex)	ASCII equivalent
LABEL$ID(3)	56 4F 4C	VOL
LABEL$NO	31	1
VOL$NAME(6)	20 20 20 20 20 20	(spaces)
VOL$STRUC	4E	N
RESERVED$A(60)	20 (60 times)	(spaces)
REC$SIDE	31	1
RESERVED$B(4)	20 (four times)	(spaces)
ILEAVE(2)	31 30	10
RESERVED$C	20	(space)
ISO$VERSION	31	1
RESERVED$D(48)	20 (48 times)	(spaces)

iRMX 86 VOLUME LABEL

The following lists the individual fields of the iRMX 86 Volume Label. This label begins with byte 384 of the diskette. Following this listing, the individual fields are shown.

field	value	ASCII or decimal equivalent
VOL$NAME(10)	45 58 41 4D 50 4C 45 00 00 00	EXAMPLE
FILE$DRIVER	04	4
VOL$GRAN	0080	128
VOL$SIZE	E900 0003	256256
MAX$FNODE	0064	100
FNODE$START	0D00 0000	3328
FNODE$SIZE	005A	90
ROOT$FNODE	0005	5

FNODE FILE

The following lists the individual fields of the fnodes in the fnode file. Included are fnodes for the fnode file, the free space map file, the free fnodes map file, the accounting file, the bad blocks file, the root directory, and the example file. The fnode file begins at byte number 3328 decimal (0D00H) of the diskette, as shown in the iRMX 86 Volume Label.

FNODE 0 (FNODE FILE)

field	value (hex)	decimal equivalent
FLAGS	0005	
TYPE	00	0 (FT$FNODE)
GRAN	01	1
OWNER	0000	0000
CR$TIME	0000 0000	0
ACCESS$TIME	0000 0000	0
MOD$TIME	0000 0000	0

EXAMPLE VOLUME

field	value (hex)	decimal equivalent
TOTAL$SIZE	2328 0000	9000
TOTAL$BLKS	0047 0000	71
PTR(0)		
NUM$BLOCKS	0047	71
BLK$PTR(0) - BLK$PTR(2)	1A 00 00	26
PTR(1) - PTR(7)		
NUM$BLOCKS	0000	0
BLK$PTR(0) - BLK$PTR(2)	00 00 00	0
THIS$SIZE	2380 0000	9088
RESERVED$A	0000	0
RESERVED$B	0000	0
ID$COUNT	0000	0
ACCESSOR(0) - ACCESSOR(2)		
ACCESS	FF	
ID	0000	
PARENT	0000	
AUX(*)	00 00 00	

FNODE 1 (FREE SPACE MAP)

field	value (hex)	decimal equivalent
FLAGS	0005	
TYPE	01	1 (FT$VOLMAP)
GRAN	01	1
OWNER	0000	0000
CR$TIME	0000 0000	0

field	value (hex)	decimal equivalent
ACCESS$TIME	0000 0000	0
MOD$TIME	0000 0000	0
TOTAL$SIZE	00FB 0000	251
TOTAL$BLKS	0002 0000	2
PTR(0)		
NUM$BLOCKS	0002	71
BLK$PTR(0) - BLK$PTR(2)	61 00 00	97
PTR(1) - PTR(7)		
NUM$BLOCKS	0000	0
BLK$PTR(0) - BLK$PTR(2)	00 00 00	0
THIS$SIZE	0100 0000	256
RESERVED$A	0000	0
RESERVED$B	0000	0
ID$COUNT	0000	0
ACCESSOR(0) - ACCESSOR(2)		
ACCESS	FF	
ID	0000	
PARENT	0000	
AUX(*)	00 00 00	

FNODE 2 (FREE FNODE MAP)

field	value (hex)	decimal equivalent
FLAGS	0005	
TYPE	02	1 (FT$FNODEMAP)
GRAN	01	1

EXAMPLE VOLUME

field	value (hex)	decimal equivalent
OWNER	0000	0000
CR$TIME	0000 0000	0
ACCESS$TIME	0000 0000	0
MOD$TIME	0000 0000	0
TOTAL$SIZE	000D 0000	13
TOTAL$BLKS	0001 0000	1
PTR(0)		
NUM$BLOCKS	0001	1
BLK$PTR(0) - BLK$PTR(2)	63 00 00	99
PTR(1) - PTR(7)		
NUM$BLOCKS	0000	0
BLK$PTR(0) - BLK$PTR(2)	00 00 00	0
THIS$SIZE	0080 0000	128
RESERVED$A	0000	0
RESERVED$B	0000	0
ID$COUNT	0000	0
ACCESSOR(0) - ACCESSOR(2)		
ACCESS	FF	
ID	0000	
PARENT	0000	
AUX(*)	00 00 00	

FNODE 3 (ACCOUNTING FILE)

No space for this file is allocated on the volume. However, its fnode must appear in the fnode file.

field	value (hex)	decimal equivalent
FLAGS	0005	
TYPE	03	3 (FT$ACCOUNT)
GRAN	01	1
OWNER	0000	0000
CR$TIME	0000 0000	0
ACCESS$TIME	0000 0000	0
MOD$TIME	0000 0000	0
TOTAL$SIZE	0000 0000	0
TOTAL$BLKS	0000 0000	0
PTR(0)-PTR(7)		
NUM$BLOCKS	0000	0
BLK$PTR(0) - BLK$PTR(2)	00 00 00	0
THIS$SIZE	0000 0000	0
RESERVED$A	0000	0
RESERVED$B	0000	0
ID$COUNT	0000	0
ACCESSOR(0) - ACCESSOR(2)		
ACCESS	FF	
ID	0000	
PARENT	0000	
AUX(*)	00 00 00	

FNODE 4 (BAD BLOCKS FILE)

No space for this file is allocated on the volume. However, its fnode must appear in the fnode file.

field	value (hex)	decimal equivalent
FLAGS	0005	
TYPE	04	3 (FT$BADBLOCK)
GRAN	01	1
OWNER	0000	0000
CR$TIME	0000 0000	0
ACCESS$TIME	0000 0000	0
MOD$TIME	0000 0000	0
TOTAL$SIZE	0000 0000	0
TOTAL$BLKS	0000 0000	0
PTR(0)-PTR(7)		
NUM$BLOCKS	0000	0
BLK$PTR(0) - BLK$PTR(2)	00 00 00	0
THIS$SIZE	0000 0000	0
RESERVED$A	0000	0
RESERVED$B	0000	0
ID$COUNT	0000	0
ACCESSOR(0) - ACCESSOR(2)		
ACCESS	FF	
ID	0000	
PARENT	0000	
AUX(*)	00 00 00	

FNODE 5 (ROOT DIRECTORY)

field	value (hex)	decimal equivalent
FLAGS	0025	
TYPE	06	1 (FT$DIR)
GRAN	01	1
OWNER	FFFF	(WORLD)
CR$TIME	0000 0000	0
ACCESS$TIME	0000 0000	0
MOD$TIME	0000 0000	0
TOTAL$SIZE	0010 0000	16
TOTAL$BLKS	0001 0000	1
PTR(0)		
NUM$BLOCKS	0001	1
BLK$PTR(0) - BLK$PTR(2)	70 00 00	112
PTR(1) - PTR(7)		
NUM$BLOCKS	0000	0
BLK$PTR(0) - BLK$PTR(2)	00 00 00	0
THIS$SIZE	0080 0000	128
RESERVED$A	0000	0
RESERVED$B	0000	0
ID$COUNT	0001	1
ACCESSOR(0)		
ACCESS	FF	
ID	FFFF	(WORLD)

EXAMPLE VOLUME

field	value (hex)	decimal equivalent
ACCESSOR(1) - ACCESSOR(2)		
ACCESS	FF	
ID	0000	
PARENT	0005	
AUX(*)	00 00 00	

FNODE 6 (EXAMPLE FILE)

field	value	decimal equivalent
FLAGS	0025	
TYPE	08	8 (FT$DATA)
GRAN	01	1
OWNER	FFFF	(WORLD)
CR$TIME	0000 0000	0
ACCESS$TIME	0000 0000	0
MOD$TIME	0000 0000	0
TOTAL$SIZE	01F4 0000	500
TOTAL$BLKS	0004 0000	4
PTR(0)		
NUM$BLOCKS	0004	4
BLK$PTR(0) - BLK$PTR(2)	80 00 00	128
PTR(1) - PTR(7)		
NUM$BLOCKS	0000	0
BLK$PTR(0) - BLK$PTR(2)	00 00 00	0
THIS$SIZE	0200 0000	512
RESERVED$A	0000	0
RESERVED$B	0000	0

field	value	decimal equivalent
ID$COUNT	0001	1
ACCESSOR(0)		
ACCESS	0F	
ID	FFFF	(WORLD)
ACCESSOR(1) - ACCESSOR(2)		
ACCESS	00	
ID	0000	
PARENT	0005	
AUX(*)	00 00 00	

FREE SPACE MAP FILE

The following is a listing of the free space map file. This file starts at byte 12416 of the volume (volume block 61H).

byte								
12416	0000	0000	0000	0000	0000	0000	FFF0	FFFE
12432	FFF0	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12448	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12464	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12480	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12496	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12512	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12528	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12544	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12560	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12576	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12592	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12608	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12624	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12640	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
12656	FFFF	FFFF	FFFF	FFFF	FFFF	0003	0000	0000

FREE FNODES MAP FILE

The following is a listing of the free fnodes map file. This file starts at byte 12672 of the volume (volume block 63H).

byte								
12672	FF80	FFFF	FFFF	FFFF	FFFF	FFFF	000F	0000
12688	0000	0000	0000	0000	0000	0000	0000	0000
12704	0000	0000	0000	0000	0000	0000	0000	0000
12720	0000	0000	0000	0000	0000	0000	0000	0000
12736	0000	0000	0000	0000	0000	0000	0000	0000
12752	0000	0000	0000	0000	0000	0000	0000	0000
12768	0000	0000	0000	0000	0000	0000	0000	0000
12784	0000	0000	0000	0000	0000	0000	0000	0000

ROOT DIRECTORY

The following is a listing of the root directory. This file starts at byte 14336 of the volume (volume block 70H).

byte																
14336	06	00	45	58	41	4D	50	4C	45	2E	46	49	4C	45	00	00
14352	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5
14368	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5
14384	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5
14400	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5
14416	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5
14432	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5
14448	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5	E5

REQUEST FOR READER'S COMMENTS

ntel Corporation attempts to provide documents that meet the needs of all Intel product users. This form lets ou participate directly in the documentation process.

lease restrict your comments to the usability, accuracy, readability, organization, and completeness of this ocument.

. Please specify by page any errors you found in this manual.

____

____

____

____

____

. Does the document cover the information you expected or required? Please make suggestions for improvement.

____

____

____

____

____

. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

____

____

____

____

____

____

. Did you have any difficulty understanding descriptions or wording? Where?

____

____

____

____

. Please rate this document on a scale of 1 to 10 with 10 being the best rating. ____________

AME ______________________________ DATE ____________

ITLE ____

OMPANY NAME/DEPARTMENT ______________________________

DDRESS ____

ITY ______________________________ STATE ________ ZIP CODE ____________

ease check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS . . .

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 79 BEAVERTON, OR

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
3585 S.W. 198th
Aloha, Oregon 97005

O.M.S. Technical Publications

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, CA 95051 (408) 987-8080

Printed in U.S.A.