SPERRY RAND

UNIVAC

418-III

REAL-TIME SYSTEM

FILE CONTROL ROUTINE

PROGRAMMERS REFERENCE

UP-7687

This manual is published by the Univac Division of Sperry Rand Corporation in loose leaf format. This format provides a rapid and complete means of keeping recipients apprised of UNIVAC® Systems developments. The information presented herein may not reflect the current status of the programming effort. For the current status of the programming, contact your local Univac Representative.

The Univac Division will issue updating packages, utilizing primarily a page-for-page or unit replacement technique. Such issuance will provide notification of software changes and refinements. The Univac Division reserves the right to make such additions, corrections, and/or deletions as, in the judgment of the Univac Division, are required by the development of its Systems.

PRINTED IN U.S.A.

CONTENTS

FIGURES

TABLE

1. INTRODUCTION

1.1. PURPOSE

The UNIVAC 418-III Real-Time Operating System File Control Routine provides preprogrammed methods of handling files on magnetic tape and mass storage subsystems to facilitate the writing of user programs involving files. Mass storage is defined as UNIVAC FASTRAND, FH-432, FH-880, and FH-1782 subsystems.

The programmer is relieved by the file control routine of the tasks of label handling, tape swapping, and record extraction. Therefore, complete user programming effort is on the actual processing of records.

2. GENERAL DESCRIPTION

2.1. FILE TYPES

The file control routine provides for the handling of three types of file structures: sequential, direct access, and search.

2.1.1. Sequential Files

Sequential files are written on mass storage or tape so that the records are placed physically in the order in which they are given. Since the location of each record is implied, the user need not provide any locating information.

2.1.2. Direct Access Files

Direct access files are used only on mass storage. Each record is written or retrieved according to user direction. For each record access the user must provide the logical record number. Each record is treated as an entity and its length must be an even number of words. The file is essentially a series of record slots which the user controls by private algorithm. The records do not have to be placed sequentially in the slots.

2.1.3. Search Files

Search files are used only on mass storage. Each record contains a unique key of a specified number of words. The user obtains a specific record by providing a key and requesting a search.

2.2. OPERATION OF FILE CONTROL

User control is obtained through the use of macro instructions which are tabulated according to their applicability.

MACRO	APPLICABLE TO	SEQUENTIAL	DIRECT ACCESS	SEARCH
OPEN$	FILE	YES	YES	YES
CLOSE$	FILE	YES	YES	YES
END$	FILE	YES	YES	YES
GET$	INPUT RECORD	YES	YES	NO
PUT$	OUTPUT RECORD	YES	YES	NO
RLSE$	BLOCK	YES	NO	NO
RETN$	FILE	YES	NO	NO
FORCE$	TAPE REEL	YES	NO	NO
SEEK$	RECORD	NO	NO	YES
NSERT$	RECORD	NO	NO	YES
DLETE$	RECORD	NO	NO	YES
UPDAT$	RECORD	NO	NO	YES
ADV$	RECORD	NO	NO	YES
XTEND$	RECORD	NO	NO	YES

Several of the macro instructions are not used frequently. The coding for these instructions is resident in the drum library and is brought in as overlays when required.

Other macro instructions are referenced continually within a user program. These macro instructions are provided as subroutines which are taken from the library and appended to the user program.

The macro instructions provide the user with a variety of services, including the following:

- Blocking and deblocking of records from multirecord blocks;
- Handling of fixed-or variable-length records;
- Handling records in move mode or in locate mode;
- Overlapping sequential record processing and input/output through use of double buffers;
- Writing and checking of file labels to prevent the use of incorrect reels or the overwriting of a file, the retention time of which has not expired;
- Writing and retrieving fixed-length records from specified locations of random files;

- Locating and retrieving fixed-length records from search files by use of keys provided by the user;
- Maintenance of search files by insertion of new records or deletion of existing records.

Directions must be provided to the system to call on the services of file control. The user must be acquainted with methods, requirements, and options in the following areas:

- Macro instructions which must be coded into programs to request the various functions;
- File description table which must be present in the user program;
- Requirements for work areas or user-provided subroutines;
- Labeling procedures provided;
- Data block and record formats;
- Handling of errors;
- Selection of search file parameters.

These subjects are discussed in detail in the following sections of the manual.

3. MACRO INSTRUCTIONS

3.1. MAGNETIC TAPE MACRO INSTRUCTIONS

The file control ''macro instructions'' are written as assembler procedure references. This PROC reference calls upon a library of systems procedures supplied with the Real-Time Operating System. Contained in this procedure library is a complete repertoire of file control functions. For the sake of consistency in terminology with other Univac systems and with competitive systems, the term ''macro'' has been adapted instead of PROC reference and is used throughout this manual. The operand of the macro instruction contains symbolic labels representing address values or other instruction parameters. The operation portion of the macro instruction contains the mnemonic representation of one of the functions to be performed by the file control system.

Each macro instruction causes a set of instructions to be generated. The number of instructions required depends on the function. Each macro instruction references the applicable file description table (see Section 4). These macro instructions are described in the following paragraphs.

3.1.1. OPEN

Before any file can be used, the file must be initialized for processing by use of the macro instruction OPEN$. The OPEN$ macro instruction is written:

LABEL	OPERATION	OPERAND
	OPEN$	FILEA, FILEB, FILEC, FILEn

The file names appearing in the operand must be the same as the labels used to reference the corresponding file description tables.

When an OPEN$ macro instruction is encountered during the execution of the object program, the following operations occur for the first reel of a tape file named in the operand:

(1) Variables pertaining to the file are inserted in a control area which is a section of the file description table.

(2) A check is made to determine if a tape unit has been assigned for the file.

(3) A check is made to determine if the file is already open. If it is, an error code is generated (see Appendix E).

(4) The tape is or is not rewound according to a code in the file description table.

(5) The tape label is processed as follows:

Input Files – The OPEN$ macro instruction reads and checks the tape header label for correct file (see Section 5). If the expiration date indicates the file has expired, the operator is advised by the console message "EXPIRED FILE ON SERVO NN". The operator must type in one of the following answers:

ANSWER	RESULTANT ACTION
Y	Override of the expiration date
N	Correct tape is mounted; label is read and checked
A	Job is aborted

If the FDT specifies a multifile tape, the tape is scanned to locate the appropriate file.

Output Files – For files employing the volume label option the OPEN$ macro instruction checks the retention code (see Section 4) of the mounted reel and writes a header tape label if the retention code indicates that the tape may be used for writing. For files not employing the volume label option the OPEN$ macro writes the tape label without checking retention code.

If the expiration date indicates the file has not expired, the operator is advised by the console message "UNEXPIRED FILE ON SERVO NN". The operator must type in one of the following answers:

ANSWER	RESULTANT ACTION
N	Correct tape has been mounted; label is read and checked
A	Job is aborted

End of Reel – During the running ot a program, similar operations are performed for subsequent reels of a multireel file after an end of reel condition has been detected. An end of reel condition is detected by reading a tape mark and an end of reel trailer label for input files, or sensing a reflective spot on tape for output files. The operations that occur when an end of reel condition is detected depend on whether the file is an input or an output file. These operations are:

- Input Files

  (a) The tape is rewound.

  (b) A check is made to determine if the next reel ot the file and its tape unit, if appropriate, are available to the program.

  When the next reel of tape is to be mounted on the tape unit, the operator is advised by the console typeout to "MOUNT REEL NR NN OF FILE XXXXXX ON SERVO NR XX". The operator must type in one of the following responses:

ANSWER	RESULTANT ACTION
Y	Next reel of tape has been mounted
N	No more reels of tape to the file
E	Next reel of tape is not available

(c) If the answer is Y, tape label of the next reel is processed as described for the first reel of an input file (see 3.1.1). If the answer is N, end of file condition is simulated. If the answer is E, control is transferred to user's error routine.

- Output Files

(a) A tape mark is written following the last record on the output tape.

(b) The end of reel trailer label is written on the output tape. Two tape marks are written following the trailer tape label.

(c) The tape is rewound.

(d) The tape label of the next reel is processed as described for the first reel of an output file (see 3.1.1).

An OPEN$ macro instruction must be executed to initialize the files before processing data. The OPEN$ macro instruction is also used when reopening a file.

3.1.2. CLOSE

When a file is completed, the file is closed by giving the macro instruction CLOSE$. The CLOSE$ macro instruction should be written:

LABEL	OPERATION	OPERAND
1	10 20	30 40
CLOSE$	FILEA,FILEB,FILEn	

The file names appearing in the operand must be the same as the labels used to reference the corresponding file description table.

When a CLOSE$ macro instruction is encountered during the execution of the object program, the following operations occur for the last reel of a tape file named in the operand. The operations depend on whether the file is an input or an output file.

- Input Files

(a) The tape is or is not rewound according to a code in the file description table.

(b) The file status word in the file description table is set to "not open".

- Output Files

  (a) Any records which remain in the output areas are written on the output tape.

  (b) A tape mark is written following the last output record.

  (c) The end of file trailer label is written on the output tape and two tape marks are written following the trailer tape label.

  (d) The tape is or is not rewound according to the code in the file description table.

  (e) The file status word in the file description table is set to "not open".

The CLOSE$ macro instruction may be used at the end of a program and whenever it is necessary during the execution of a program to CLOSE$ before a reopen.

3.1.3. RETN (Return)

To return control to the OPEN$ or CLOSE$ macros when user participation in the processing of labels is specified, the macro instruction RETN$ is used. The macro instruction should be written:

LABEL	OPERATION	OPERAND
RETN$	FILEA	

The operand is the name of the file that the programmer has applied to the file description table.

This instruction is used exclusively in conjunction with the OPEN$ and CLOSE$ instructions and *must* be issued at the completion of user participation in label processing.

3.1.4. END

Upon completion of a job *all* files may be closed by issuing the macro instruction END$. The END$ macro instruction should be written:

LABEL	OPERATION	OPERAND
END$		

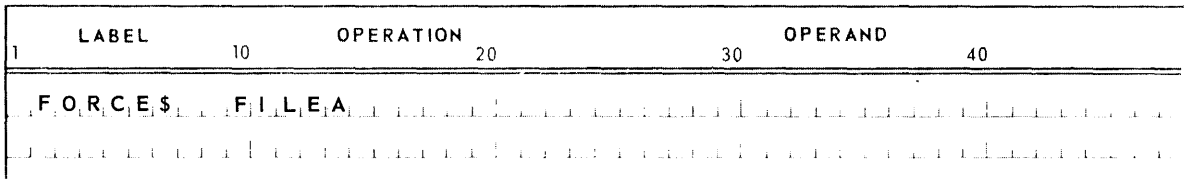
When an END$ macro instruction is encountered during the execution of the object program, it performs the same operations as the CLOSE$ macro instruction for *all* files which are still open.

By using the END$ macro instruction in place of the final CLOSE$, programming is simplified in that the programmer need not reference the files to be closed.

NOTE: All files must be closed by use of the END$ or CLOSE$ macro instruction prior to the initiation of end of job.

3.1.5. FORCE

The FORCE$ macro instruction may be used by the programmer to terminate action on the current reel of a file. The FORCE$ macro instruction should be written:

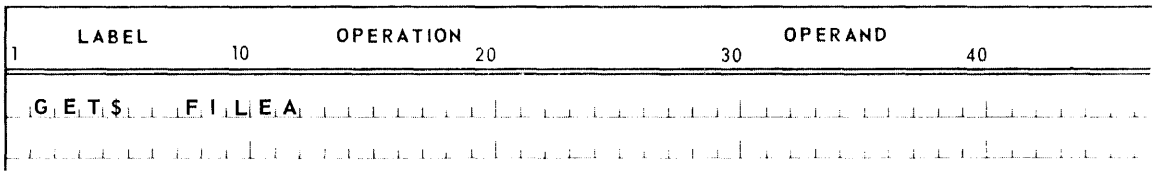


The file names appearing in the operand must be the same as the labels used to reference the corresponding file description table.

When a FORCE$ macro instruction is encountered during the execution of the object program, the effect is to terminate operations on the current reel, and proceed to the next sequential reel. The action taken is the same as described for end of reel for multireel files (see 3.1.1) except that there is no trailer label processing for input files.

3.1.6. GET

To locate a record to be processed, the macro instruction GET$ is used. The macro instruction should be written:

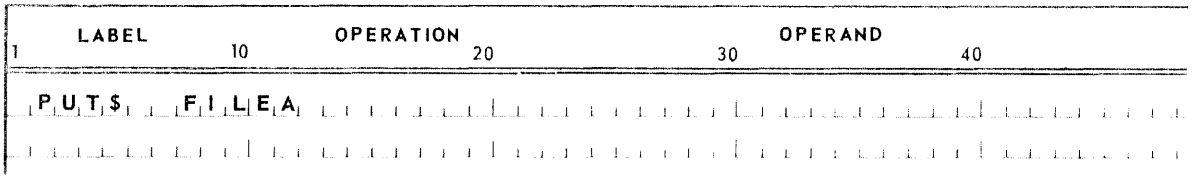


The file name appearing in the operand must be the same as the label used to reference the corresponding file description table. The instruction is applicable to input accessing and its function is to locate one record of the file for use by the worker program. However, the final action depends on the user selection of move or locate mode as given in the file description table (see Section 4).

In move mode, the record is moved to the user workspace. In locate mode, the address of the record is placed in the first word of the workspace.

3.1.7. PUT

To cause a record to be included in an output file, the macro instruction PUT$ is used. The macro instruction should be written:

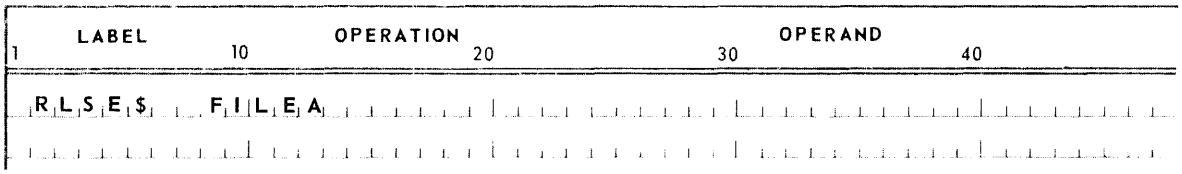


The file name appearing in the operand must be the same as the label used to reference the corresponding file description table.

The instruction is applicable to output processing and its function is to move one record from the user program workspace to the next open position of the output buffer.

3.1.8. RLSE (Release)

To begin using a new record block, the macro instruction RLSE$ can be used. The macro instruction should be written:



The file name appearing in the operand must be the same as the label used to reference the corresponding file description table.

The instruction may be used on input or output files. If the file referenced in the operand is an input file, no record is taken from the current block after the RLSE$ macro instruction is executed. The next GET$ macro instruction which refers to the file obtains the first record in the next block of records. When used with input files, RLSE$ allows the programmer to bypass the records remaining in a block of records.

If the file named in the operand of a RLSE$ macro instruction is an output file, no more records are placed in the current output block.

3.2. MACRO INSTRUCTIONS FOR MASS STORAGE SUBSYSTEMS

The file control system provides functions for the manipulation of three different file types on mass storage. The types of files are:

- Sequential Files
- Direct Access Files
- Searchable Files

The function repertoire for the handling of sequential and direct access files is quite similar to that of the sequential tape files, while the repertoire of macros for the searchable files provides for considerably more flexibility to the problem programmer. A detailed definition is given in the following paragraphs.

3.2.1. Macro Instructions for Sequential Files For Mass Storage

For sequential files, the macro instructions are used in the same manner as for sequential tape files, with the exception of the FORCE$ and RLSE$ instructions. The FORCE$ instruction is not applicable to mass storage files and will not produce any results if used. If the RLSE$ is issued for a *fixed record* output file, it is the user's responsibility to provide his own end of block sentinel to enable recognition of EOB on the subsequent input of the file. The sequential file opened for INOUT (input/output) is a special case. This mode permits the updating of existing records of the file. It does not permit the adding, deleting, or changing sizes of records. The user can rewrite a record by issuing a PUT$ instruction, which applies to the record obtained by the previous GET$. If the move mode is being used, the record in the workspace is moved back to its former location in the buffer.

3.2.2. Macro Instructions for Direct Access Files For Mass Storage

The GET$ and PUT$ instructions have special meaning for direct access files. The user issues these instructions with the logical record number in AU-AL. The program then "gets" the individual record from the file to the workspace or "puts" the record into the file from the workspace. The FORCE$ and RLSE$ are not applicable to direct access files.

3.2.3. Macro Instructions for Searchable Files For Mass Storage

For searchable files, the OPEN$, CLOSE$, and END$ instructions are used in the manner described for tape files. The remaining functions for this type of file form a separate repertoire. The following definitions of terms and layout charts are presented for the purpose of clarifying the explanation of the search file macro instructions.

- KEY

  A field from 1 to 63 words in length, occupying the initial words of a record, upon which the sequence of the file is based and by which a specific record is referenced.

- MASTER BLOCK

  A table which is located at the beginning of the mass storage file. The table is loaded into storage at the address specified in word 10 of the search file description table (see Figure 4–2). The master block (highest level of index) is shown in Figure 3–1.

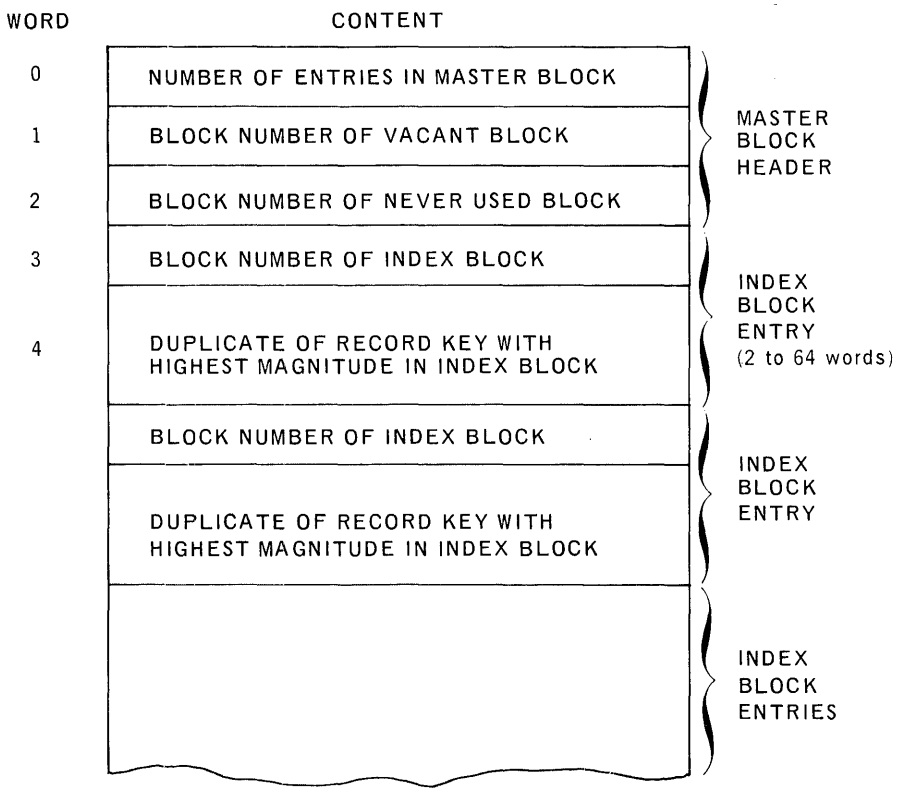


Figure 3–1. Master Block

■ INDEX BLOCK

A block of data maintained in the file that is the second level of index reference; the format is shown in Figure 3–2.

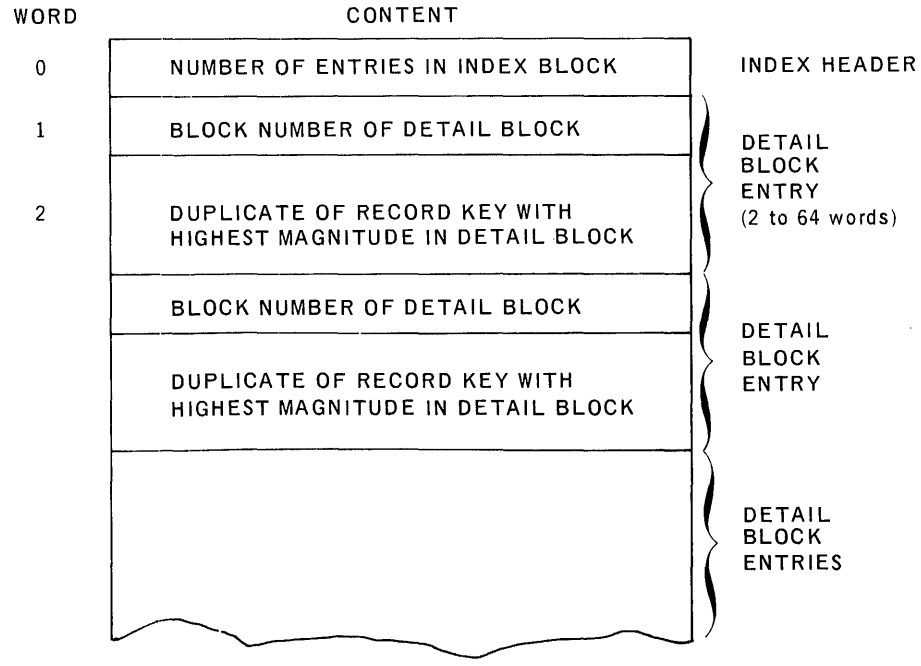


Figure 3–2. Index Block

■ DETAIL BLOCK

A block of data maintained in the file that contains the data records of the file; the format is shown in Figure 3–3.

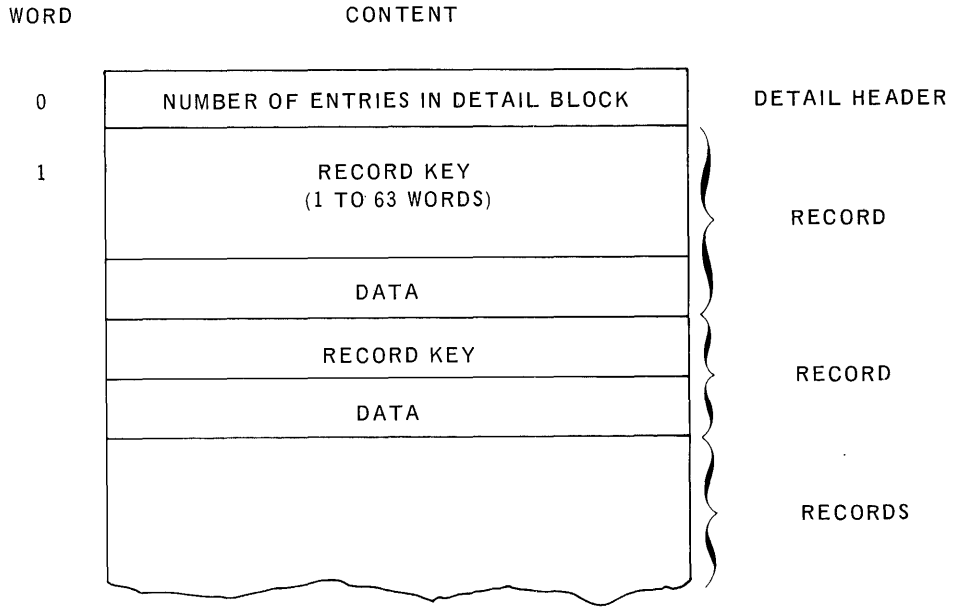


Figure 3–3. Detail Block

■ SECTION

The grouping of a master block entry, the index block referenced by the master block entry, and the detail blocks referenced by the index block. A section is shown in Figure 3–4.

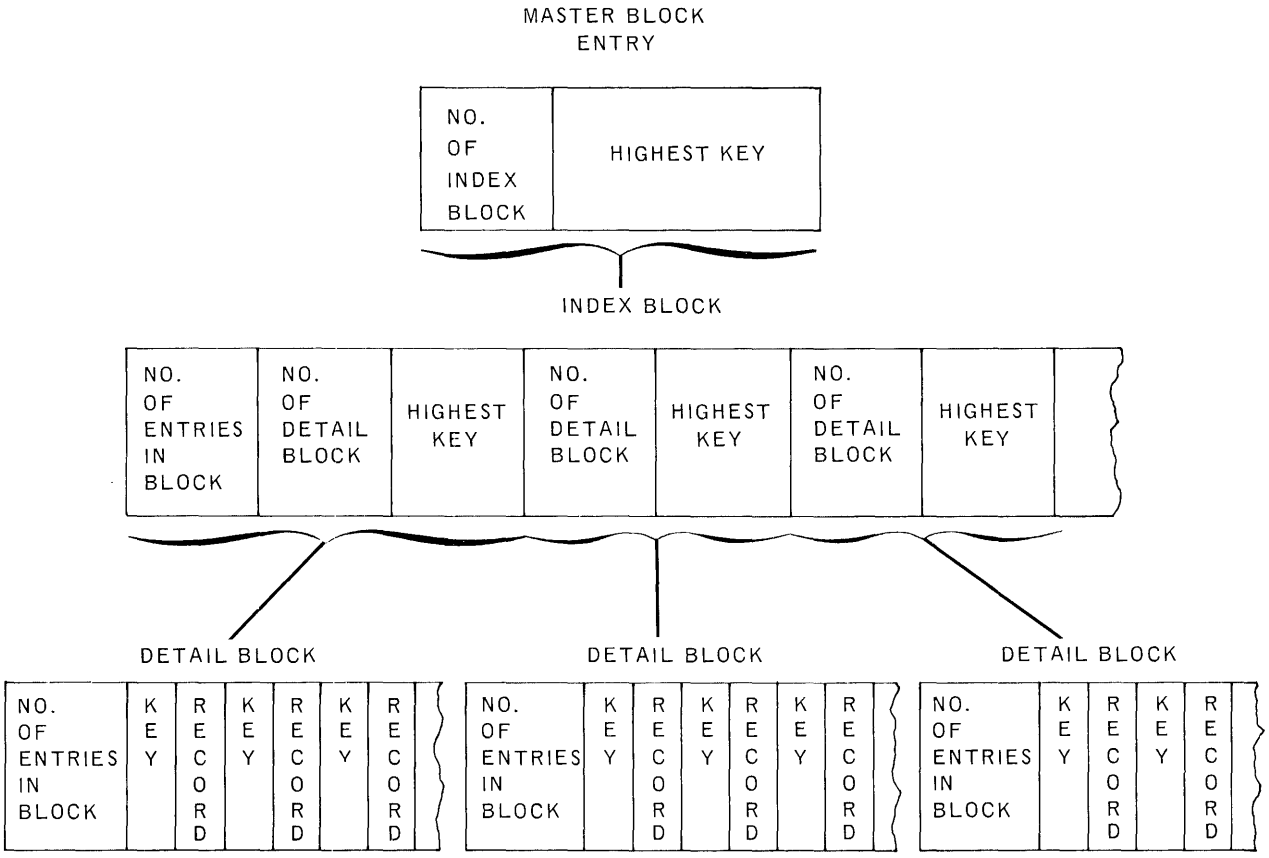


Figure 3–4. Index Sequential Control Section

3.2.3.1. SEEK

To request that a specific record be placed in working storage, the key of the desired record must be placed in the initial words of the workspace identified in word 6 of the file description table (FDT). The user then initiates the SEEK$ instruction as follows:

1 LABEL	10 OPERATION	20	30 OPERAND	40
SEEK$	FILEA			

The file name appearing in the operand must be the same as the label used to reference the corresponding file description table.

The SEEK$ instruction searches the referenced file starting with the detail block currently in the buffer. If the desired record is found in the buffer, the record is moved to the workspace and exit is made with AL = 0, which indicates a find.

If the record is not found in the buffer, the write flag is checked to determine if the buffer has been altered since read time. If it has been altered it is written on the file and the flag is reset.

A lookup is then performed on the master block to determine the specific index block which encompasses the reference key. This index block is accessed and a table lookup performed to determine the specific detail block encompassing the reference key. The detail block is then accessed and a table lookup performed to locate the key. If the key is found, the record is moved to the workspace specified in word 6 of the FDT and AL is set to 0. If the key is not found, AL is set to 1.

Minimum access time occurs when the record is found in the detail block currently in the buffer because no mass storage operations occur.

Two mass storage operations are needed for the seek when the record is not in the buffer and the buffer has not been altered. Maximum access time is three mass storage operations.

NOTE: The probability of achieving minimum access is improved as the number of records per block is increased.

3.2.3.2. ADV (Advance)

To request the next sequential record (with reference to the last record accessed) be placed in working storage, the user initiates the ADV$ instruction:

LABEL	OPERATION	OPERAND
1 10	20 30	40
ADV$	FILEA	

The file name appearing in the operand must be the same as the label used to reference the corresponding file description table.

If ADV$ is the first file control instruction to be executed on a file, the first record of that file is moved to the workspace. If the record moved to the workspace is the end of file record (key of all binary 1's) the advance call address is placed in AL and control is transferred to the address specified in word 7 of the FDT.

3.2.3.3. NSERT (Insert)

To add a record to the file the user must place the record in the workspace specified in word 6 of the FDT. The user then initiates the NSERT$ instruction:

```
LABEL          OPERATION          OPERAND
1         10        20        30        40
NSERT$    FILEA
```

The file name appearing in the operand must be the same as the label used to reference the corresponding file description table.

The NSERT$ instruction searches the file in the manner described in the SEEK$ instruction. If the key is found, it denotes a duplicate record which is not allowed; AL is set to 1. If the key is not found the record is inserted in the detail block in its proper place and AL is set to 0. If the detail block is filled to capacity, the block is split and written as two detail blocks and the appropriate index block is updated to reflect the proper number of detail blocks. If the index block is filled to capacity, it is split and written as two index blocks and the master block is updated. When a record is inserted into the last available record position of the last available block, the user is advised by error code 070001 and control is transferred to the address +6 specified in word 8 of the FDT. If an insertion is attempted and there are no additional blocks available, the user is advised by error code 070002 and control is transferred to the address +6 specified in word 8 of the FDT. (See Appendix E).

3.2.3.4. DLETE (Delete)

To delete a record from the file the key of the record to be deleted must be placed in the initial words of the workspace identified in word 6 of the FDT. The user then initiates the DLETE$ instruction:

```
LABEL          OPERATION          OPERAND
1         10        20        30        40
DLETE$    FILEA
```

The file name appearing in the operand must be the same as the label used to reference the corresponding file description table.

The delete instruction searches the file in the manner described in the SEEK$ instruction. If the key is not found, a 1 is placed in AL. If the key is found the record is deleted from the detail block by overlaying it with the trailing records of the block; AL is set to 0. If the detail block becomes vacant the block number in word 1 of the master block is placed in word 0 of the detail block. The detail block is written on the file and the block number of the detail block is placed in word 1 of the master block. The index block is updated similarly to include updating of the master block if the index block becomes vacant.

3.2.3.5. UPDAT (Update)

To request that an updated record be written into the file, the updated record must be placed in the workspace specified by word 6 of the FDT. The user then initiates the UPDAT$ instruction:

```
      LABEL          OPERATION          OPERAND
1          10         20         30         40
UPDAT$     FILEA
```

The file name appearing in the operand must be the same as the label used to reference the corresponding file description table.

The update instruction searches the file in the manner described in the SEEK$ instruction. If the key of the record to be replaced (updated) is found, the record is moved to the buffer and the write flag is set; a 0 is placed in AL. If the key is not found, AL is set to 1.

3.2.3.6. XTEND (Extend)

To initially create a file or add a volume of records to the end of a file the records must be placed in the workspace specified by word 6 of the FDT. The user then initiates the XTEND$ instruction:

```
      LABEL          OPERATION          OPERAND
1          10         20         30         40
XTEND$     FILEA
```

The file name appearing in the operand must be the same as the label used to reference the corresponding file description table.

The extend instruction moves the record from the workspace to the buffer. When the buffer contains the number of records specified by the algorithm (WPB–1)/WPR – SPACE (see 4.1.1), the buffer is written on the file as a detail block. The appropriate index block is accessed, an entry is generated for the new detail block, and the index block is written. When the index block contains the number of entries proportionate to the fill factor (same number of unused entries as specified in SPACE, word 16, FDT), an entry is generated for the master block. When records are extended out of sequence the record is not added to the file, AL is set to 5.

File limits (no available blocks) are handled as described in the NSERT$ instruction.

NOTE: If XTEND$ was the last macro initiated a dummy SEEK$ *must* be initiated prior to issuance of CLOSE$ or END$.

3.3. SELECTION OF SEARCH FILE PARAMETERS

The search file is designed to provide the means for rapid search for a specific record in a large mass storage file.

The file structure is indexed sequentially as shown in Figure 3–4. Consequently, the user has the capability of proceeding sequentially through the records as well as the capability for requesting specific records at random.

The methods for searching and for file maintenance are inherent in the file control routine and are dependent on a specific type of file structure. However, the selection of sizes for blocks, records, and keys is the province of the user. The user specifies these parameters to suit the particular data set and must provide reserved areas of suitable size for handling purposes in his program. The locations of the areas must be given in the file description table. The required areas are:

- Record workspace, equal to the size of one record;
- Block buffer, equal to the size of one block plus one record;
- Master block area, equal to 3 + nx where n is the maximum number of section control keys needed, and x is the number of words of the key + 1.

The proper selection of file parameters depends on the number of records, size of records, size of keys, and the amount of space available for the block buffer. To make a suitable selection, the user must perform some calculations. The following example illustrates a file to be filled to 3/4 capacity initially, allowing 1/4 expansion.

IF: Block size (B) = 1792 words

Record size (R) = 50 words

Key size (K) = 5 words

THEN:

Average number of records per detail block = $3/4 \times \frac{B-1}{R} = 27$

Average number of keys per index block = $3/4 \times \frac{B-1}{K+1} = 223$

Average number of records per section = 27 X 223 = 6021

Average number of blocks (detail blocks + one index block) used per section = 223 + 1 = 224

Using these figures, each section can accommodate roughly 6000 records. If the file is expected to have 60,000 records, this will require 10 section keys, requiring 63 words of reserve in the master block.

4. FILE DESCRIPTION TABLE

4.1. STANDARD FILE DESCRIPTION TABLE

Each file to be processed must have a file description table (FDT) in the user program. The same FDT will serve in the case where a file is opened for output, closed, and then opened for input.

A map of the FDT is shown in Figure 4–1. The FDT map consists of 64 computer words, 17 of which are specified by the user. Any fields represented by the symbol Z are to contain zero. These represent areas that file control needs for its activities and may not be used by the programmer. Each field shown includes the number of bits in that field. The user defines the contents of the file description table by the FDT$ macro call. This macro instruction provides for a variable list of parameters in the operand file. See Appendix I. (A separate explanation of the FDT application to search files is given in 4.1.1.)

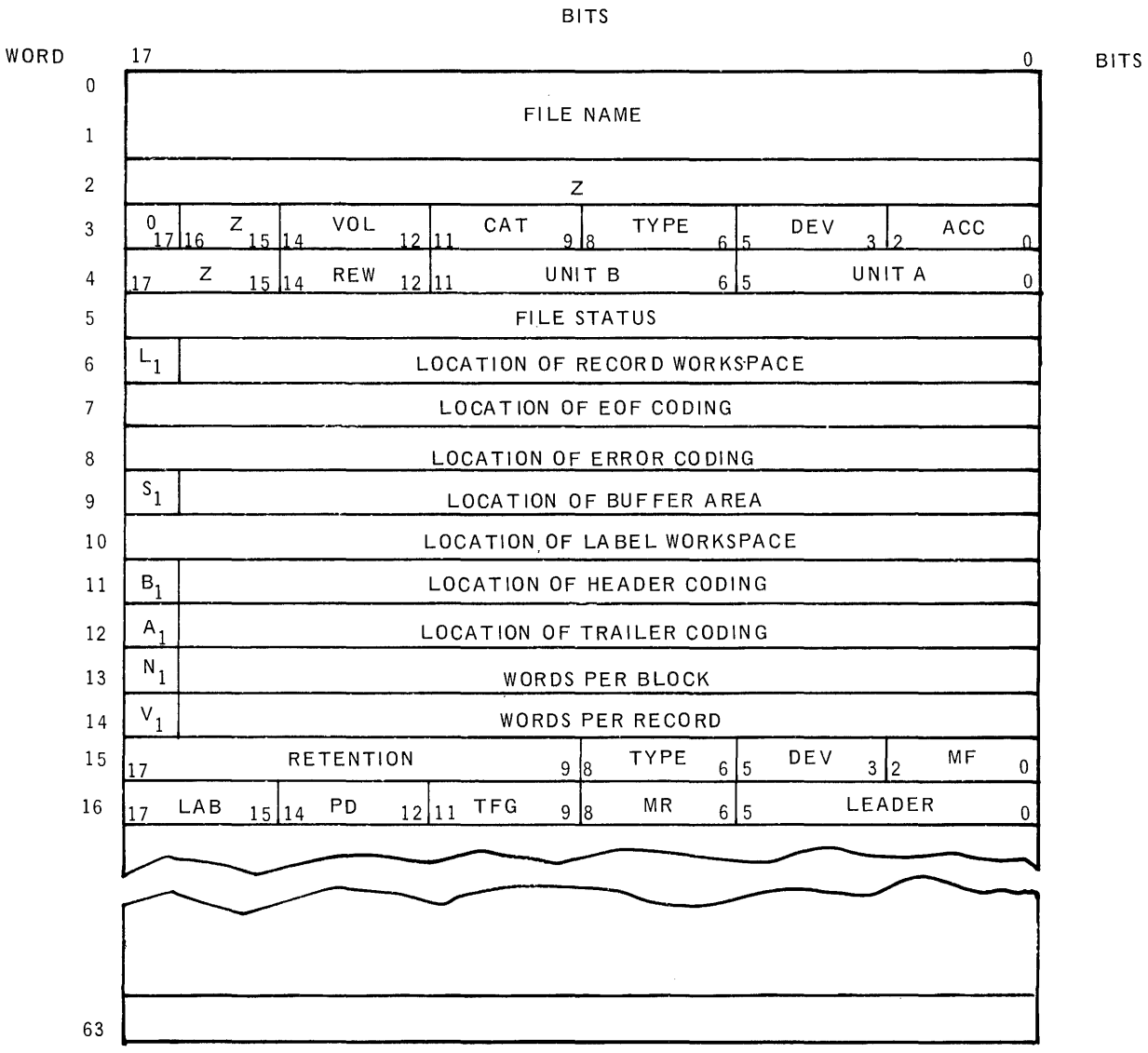


Figure 4–1. File Description Table

- Word 0

  Contains the three leftmost characters of a six-character file name.

- Word 1

  Contains the three rightmost characters of a six-character file name.

- Word 3

  - Field 0 indicates if an input file is optional.

    0 Not Optional
    1 Optional

  - Field VOL indicates if volume labels are present.

    0 Not Present
    1 Present

  - Field CAT indicates if the file has been cataloged into the file directory by use of the CAT control option.

    0 Catalogued
    1 Not Catalogued

  - Field TYPE designates the file structure.

    0 Sequential
    1 Direct Access
    2 Search

  - Field DEV designates the peripheral device.

    0 FASTRAND Subsystem
    1 Magnetic Tape
    2 FH Drum

  - Field ACC specifies the access mode.

    0 Random
    1 Input
    2 Output
    3 Input/Output

NOTE: The TYPE, DEV, and ACC fields provide means for selecting many combinations of accessing. Several of these combinations would produce contradictory specifications and may not be used.

An example of a valid code is:

TYPE	DEV	ACC
000	001	001
0	1	1
(SEQUENTIAL)	(TAPE)	(INPUT)

Table 4–1 shows the combinations that are permitted, and gives the correct octal code.

	SEQUENTIAL TAPE	SEQUENTIAL FASTRAND / DRUM	DIRECT ACCESS FASTRAND / DRUM	SEARCH FASTRAND / DRUM
RANDOM			100 / 120	200 / 220
INPUT	011	001 / 021		
OUTPUT	012	002 / 022		
INPUT/OUTPUT		003 / 023		

Table 4–1. Valid Accessing Codes

- Word 4
  - Field REW combines information on initial and final rewinding.

	INITIAL	FINAL
0	rewind	rewind/interlock
1	rewind	rewind
2	rewind	none
4	none	rewind/interlock
5	none	rewind
6	none	none

  - Field Unit B is the unit number assigned to the secondary tape unit, when swapping of tape units is desired. Octal 77 in this field indicates that no secondary tape unit is desired.

- Field Unit A is the logical unit number assigned to the primary tape unit. Within any one program, all tape unit numbers must be different. For mass storage files, this field contains the location counter for the file. Unit numbers and/or location counters from zero through fifteen (decimal) are permitted.

■ Word 5

FILE STATUS indicates to file control whether the file is open or not. An initial value of negative zero must be provided.

■ Word 6

- Field L indicates in which mode an input file is to be handled.

  0 Move mode

  1 Locate mode

- Contains the address of the workspace used for PUT/GET operations (see 4.1.2.1).

■ Word 7

Contains the address of user coding to which file control transfers control upon finding end of file during input. At entry to this coding, the lower accumulator contains the address of the macro instruction where the user entered file control (see 4.1.2.2).

■ Word 8

Contains the address of user coding to be entered in the event that file control detects logic or hardware error. At entry to this coding, the lower accumulator contains the address of the macro instruction where the user entered file control (see Appendix E and 4.1.2.3).

■ Word 9

- Field S indicates the buffering desired.

  0 One buffer

  1 Two buffers

- Contains the address of the buffer area reserved by the user (see 4.1.2.4).

■ Word 10

Contains the address of a special workspace used for participation in the labeling process or this word must be binary zero (see 4.1.2.5).

■ Word 11

Contains the location of the user coding for header label processing (see Section 5). When control is transferred to user's coding AL contains the address of the FDT. If field B is zero, the coding will be addressed after file control processing, if one, before file control processing.

- Word 12

  Contains the location of the user coding for participation in trailer label processing. If field A is zero, the coding will be addressed after file control processing, if one, before file control processing.

- Word 13

  - Field N specifies block number present or absent.

    0 Absent

    1 Present

  - Contains the total number of words comprising one block. The figure given must include the leader words and the data words. If double buffering is used, the buffer area must be at least twice the block size. For mass storage files, this must be an even number of words.

- Word 14

  - Field V specifies fixed or variable-length records.

    0 Fixed

    1 Variable

  - Words per record contain the number of words per record for fixed-length records (field V is zero). Words per record is zero if field V is one.

- Word 15

  - Field RETENTION is a binary number indicating the number of days after the creation date that a file must be protected against being overwritten (not applicable to input file).

  - Fields TYPE and DEV are as described in word 3.

  - Field MF specifies multiple files on a reel of tape (0 = NO) (1 = YES)

- Word 16

  - Field LAB is used to specify the label options and the block serial number options.

    0 Labels standard

    1 Labels nonstandard

    2 Labels omitted

  - Field PD specifies parity and density for tape (not applicable to mass storage files).

	Parity	Density
0	odd	manual setting
1	odd	200 frames/inch
2	odd	556 frames/inch
3	odd	800 frames/inch
4	even	manual setting
5	even	200 frames/inch
6	even	556 frames/inch
7	even	800 frames/inch

- Field TFG specifies translation, format, and gain for tape (not applicable to mass storage files).

	Tape to Internal and Internal to Tape Code Translation	Format	Gain
0	none	7 track	normal
1	none	7 track	low
2	none	9 track	normal
3	none	9 track	low
4	translate	7 track	normal
5	translate	7 track	low

- Field MR indicates single or multireel file (not applicable to mass storage files).

  0 single reel

  1 multireel

- Field LEADER specifies the number of block leader words desired. If block serial numbers are present, there must be at least one leader word specified.

4.1.1. File Description Table for Search Files

For search files, the same file description table is used, but certain fields are not used and others are redefined. A map of this is shown in Figure 4–2.

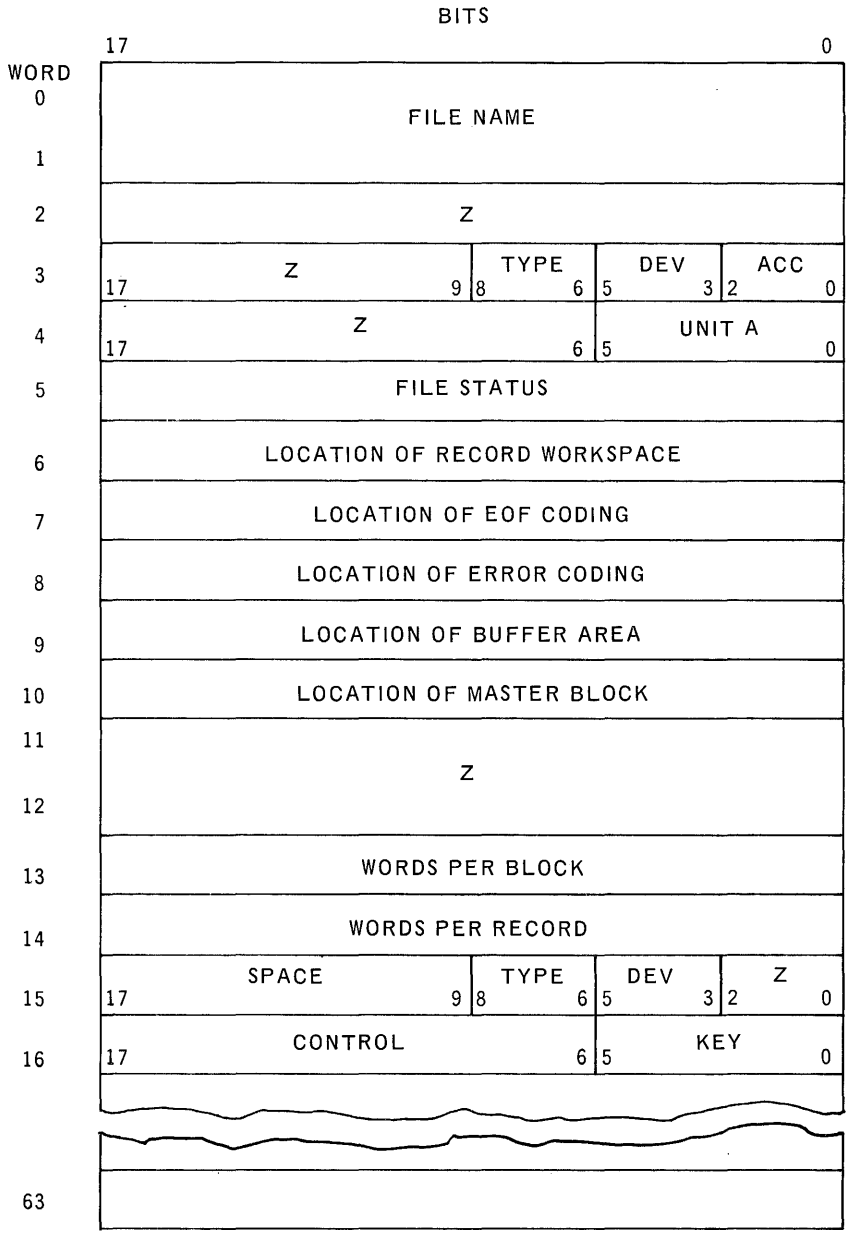


Figure 4–2. File Description Table for Search Files

- Word 0

  Contains the three leftmost characters of a six-character file name.

- Word 1

  Contains the three rightmost characters of a six-character file name.

- Word 2

  Contains 0.

- Word 3

  –Field TYPE must contain 2
  –Field DEV must contain 0 or 2
  –Field ACC must contain 2

- Word 4

  Unit A contains the location counter for the file.

- Word 5

  FILE STATUS must contain negative zero. Indicates if file is open or closed.

- Word 6

  Contains the address of the workspace. This area must be at least one record in size.

- Word 7

  Contains the address of the user's end of file coding.

- Word 8

  Contains the address of the user's error coding, the first six words of which are used by file control (see Appendix E).

- Word 9

  Contains the address of the buffer area. This area must be at least block size +1 record.

- Word 10

  Contains the location of the area for the master block. The size of this area must be at least the number of words of key plus one, times the number of section control keys, plus three, and must be an even number of words.

- Word 11 and 12

  Contains 0.

- Word 13

  Contains the number of words per block. This must be an even number of words.

- Word 14

  Contains the number of words per record.

- Word 15

  –Field SPACE contains a factor which is used during extend operations. The number given represents the number of record spaces that the user wants left vacant at detail and index block ends. Leaving such expansion space permits limited insertions without causing block splits.

  –Fields TYPE and DEV are the same as in word 3.

- Word 16

  –Control contains the number of sections specified for the file (4095 is the maximum permitted), see Figure 3–4.

  –KEY contains the number of words in the key (63 is the maximum permitted). If this number exceeds 24, the reserve starting at word 17 must be extended correspondingly.

- Word 17 – 63

  Must be reserved but are for the use of file control.

4.1.2. Work Areas and User Routines

The file description table provides space for specifying several areas of storage and user coding. For ease of use, the following amplification is furnished.

4.1.2.1. Record Workspace

The record workspace is the area that is used for transfer of records between the program and I/O buffers. It must be large enough to accommodate the largest record that the file will contain. In the case of input access, operating exclusively in locate mode (see 3.1.5), the workspace can consist of a single word.

For direct access, the record workspace is used directly as the input/output buffer. The instructions GET$ and PUT$ are essentially read and write instructions.

For search files, the record workspace is used for transfer of records between the program and the block buffer. It is also the location where the user places the subject key before executing a SEEK$ or DLETE$ instruction.

4.1.2.2. End of File Coding

The end of file coding is needed on input only. File control returns control to the EOF coding when it detects the end of data in the file. At entry to this coding, the lower accumulator contains the address of the macro instruction by which file control was entered.

4.1.2.3. Error Coding

The error coding (see Appendix E) provides for the analyzing of conditions. If it is not provided, file control will abort the program in the event of an error.

4.1.2.4. Buffer Area

The buffer area is reserved by the user program and its address placed in word 9 of the FDT. The amount of space needed is:

- The size of one block for sequential files, using one buffer
- The size of two blocks for sequential files, using two buffers
- None, for direct access files
- The size of one block plus one record, for search files.

4.1.2.5. Label Workspace

The label workspace is needed if it is intended to participate in the processing of labels. In this case, the workspace must be large enough to accommodate one label (28 words).

The user may specify participation during either input or output but only when labels are specified to be present. The file control overlays which prepare or validate labels test for the user specifications. When found, the program places the label in the label workspace and schedules the user label subroutine. The subroutine must terminate with the macro RETN$.

The label workspace does not apply to search files where word 10 has a different function. It is used with search files to contain the location of the master block which is held in storage to reduce the search time. The space needed is three + nx: where n is the maximum number of section control keys the file will require and x is the number of words of key plus one.

4.1.2.6. Header Coding

This coding is not used unless word 10 has been used for the location of a label workspace. If so, word 11 of the FDT gives the location of the user coding for use with header labels at the beginning of reels.

4.1.2.7. Trailer Coding

This coding is not used unless word 10 has been used for the location of a label workspace. If so, word 12 gives the location of the user coding for use with trailer labels at the end of reels.

5. LABELING PROCEDURES

5.1. STANDARD LABELS

The standard labels are described for tape and mass storage in the following paragraphs. A label decision table (see Table 5–1) is presented at the end of this section.

5.1.1. Magnetic Tape Files

When a file is written on the tape, the old header label is replaced by the header label appropriate to the new file. Following the header label there are: one tape mark, data blocks, and one tape mark. The format of the header label is as follows:

FIELD NUMBER	WORD NR	CONTENTS	DESCRIPTION	EXAMPLE
1	0	HDR	LABEL IDENTIFIER	HDR
2	1–2	XXXXXX	FILE NAME	INVTRY
3	3–4	XXXXXX	SERIAL NUMBER	327
4	5	XXX	REEL NUMBER	1
5	6	XXXX	CREATION DATE	
6	7	XXXX	EXPIRATION DATE	
7	8–9	XXXXX	ACCOUNT NUMBER	123
8	10–27	BLANK		

- Field 1

  Identifies the label as a file header.

- Field 2

  Contains the file name. This name must agree with the name specified in the first two words of the FDT.

- Field 3

  Contains the serial number of the tape. In a multireel file, all header labels will contain this same serial number. This field will be blank unless volume labels are specified and they contain a serial number.

- Field 4

  Contains the reel sequence number.

- Field 5

  Contains the creation date as a four-digit decimal coded julian data (YDDD).

- Field 6

  Contains the date after which this file may be overwritten in the same format as creation date.

- Field 7

  Contains the account number which may be used to limit access to the file.

- Field 8

  Not used by the system; this space is available for the entering of data. To do so, a workspace and a subroutine to be used must be specified in the file description table.

Following the tape mark after the data blocks may be either of the following combinations:

- End of reel label plus two tape marks
- End of file label plus two tape marks

The format for the end of reel label is:

FIELD NUMBER	WORD NR	CONTENTS	DESCRIPTION	EXAMPLE
1	0	EOV	LABEL IDENTIFIER	EOV
2	1	XXXXXX	BLOCK COUNT	2715
3	2	XXXXXX	RECORD COUNT	27147
4	3–27	BLANK		

- Field 1

  Identifies the label as an end of reel.

- Field 2

  The number, in binary, of data blocks of this file on this reel.

- Field 3

  The number, in binary, of records of this file on this reel.

- Field 4

  Not used by the system.

NOTE: The end of file label format is the same as the end of reel label format except that Field 1 contains EOF.

5.1.1.1. Volume Labels

As an optional feature, for use with tape files, volume labels are provided. The format of the volume label is as follows:

FIELD NUMBER	WORD NR	CONTENTS	DESCRIPTION
1	0	VOL	VOLUME LABEL IDENTIFIER
2	1–2	BLANK	
3	3–4	XXXXXX	TAPE SERIAL NUMBER *
4	5–27	BLANK	

*TAPE SERIAL NUMBER CONVENTION IS THE USER'S PREROGATIVE.

One purpose of this option is to act as an additional safeguard to preclude overwriting a tape that is still valid. This is accomplished by requiring the OPEN macro to read two blocks to check the HDR label expiration date prior to writing of any kind if volume labels have been specified (word 3 of FDT). The volume label is retained on the file. To employ the use of the volume label option the tape must have a volume and header label present prior to creation of the file. This can be done by utilizing the system routine VOLAB$.

5.1.2. Mass Storage File Labels

There are no provisions for standard labels involving mass storage files.

5.2. NONSTANDARD LABELS

Nonstandard magnetic tapes and Mass Storage labels are defined in the following paragraphs.

5.2.1. Magnetic Tape Files

For input files the header label block is read, moved to user's label workspace, and control transferred to user's coding for header label processing. For output files the standard trailer label is generated and moved to user's label workspace. Control is transferred to user's coding for trailer label processing. When control is returned to file control by way of the RETN$ macro, the nonstandard label is written directly from user's label workspace. Nonstandard labels are effectively 28 words in length since any label words in excess of 28 are ignored.

5.2.2. Mass Storage File Labels

There are no provisions for nonstandaid labels involving mass storage files.

CONDITIONS	Labels Omitted	X																	
	Standard Labels		X	X	X	X	X	X	X	X	X								
	Non-Standard Labels											X	X	X	X				
	User Participation			X		X			X	X					X				
	Volume Labels Specified & Present				X	X													
	Volume Labels Not Specified but Present							X	X										
	Volume Labels Specified but Not Present						X												
	No User Label Workspace Specified									X				X					
											X	X							
RESULTING ACTION	Header Label Block Moved to User Label Workspace			X		X			X						X				
	Header Label Block Read or Written		X	X	X	X		X	X	X		X	X	X*	X				
	No Header Label Block Read or Written	X																	
	Tape Mark Following Header Label Read or Written		X	X	X	X		X	X	X		X	X	X	X				
	Backspace If no Tape Mark Following Header Label											X	X	X	X				
	Volume Label Read				X	X													
	Skip Volume Label							X	X										
	Generate Standard Trailer Label		X	X	X	X		X	X	X					X				
	Standard Trailer Label Read or Written		X	X	X	X		X	X	X									
	Standard Trailer Label Moved to User Label Workspace			X		X			X						X				
	Write Standard Label from User Workspace														X				
	No Trailer Label Read or Written	X										X	X	X					
	Control Transferred to User Error Routine						X				X								

*Pertains to Input File Only.

Table 5–1. Decision Table for Label Processing

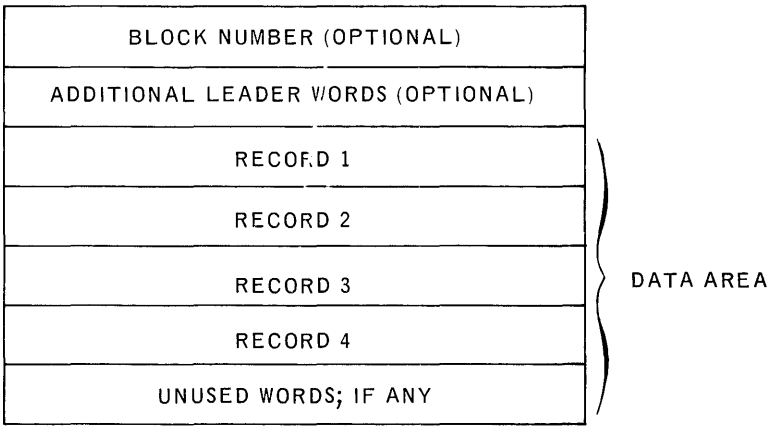
APPENDIX A. DATA BLOCK FORMATS

A.1. SEQUENTIAL FILE

Data block size may be selected to suit the needs of a particular data set. However, all data blocks for a particular file are required to be the same size. For mass storage files, this size must be an even number of words.

A.1.1. Fixed-Length Records

Where fixed-length records are specified, the data block has the following format:

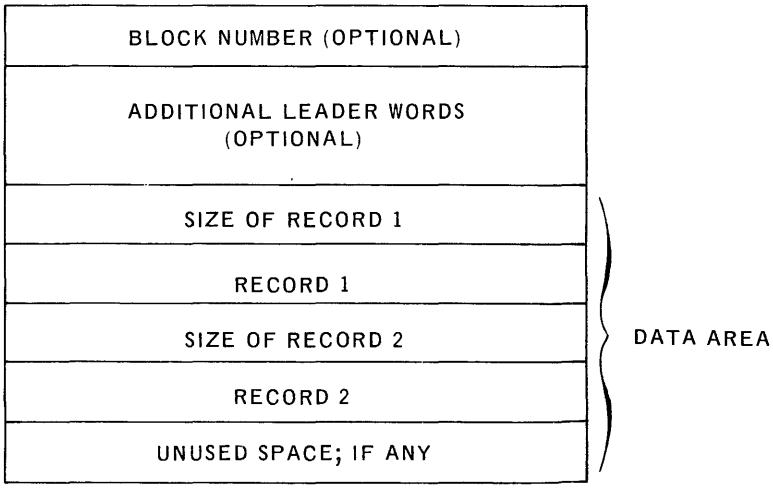


On output, the program places as many complete records as the data area will hold.

On input, the program retrieves records as long as the remaining data area is sufficient to contain a complete record.

A.1.2. Variable-Length Records

For variable-length records, the number of records per block may vary. The first word of each record states the number of words of the record. The first word is considered to be a part of the record. The data block format is as follows:



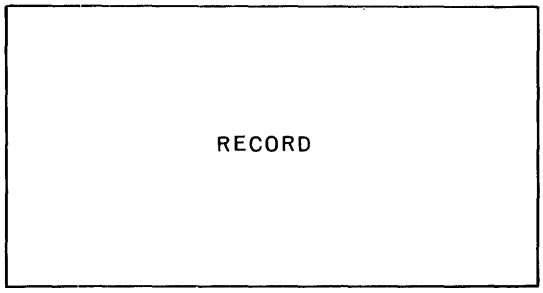
On output, the program places records in the buffer until it finds a record that will not fit in the remaining space of the data area. At that time, it closes out the block by placing a word of binary zero after the last good record, unless the block is exactly full.

On input, the program reads records until the word of binary zero is found, or the end of block is reached.

The user must ensure that no single record exceeds the data area of the block.

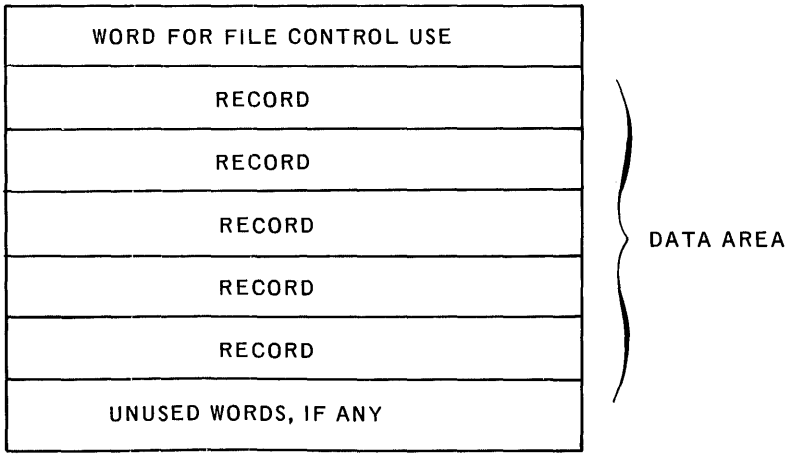
A.2. DIRECT ACCESS FILE

For mass storage direct access files, each record is an entire block and must be an even number of words. All blocks are the same size and there are no block numbers or leader words handled by the program. Any such features are within the record and are known only to the user. Therefore, the block format for direct access files is:



A.3. SEARCH FILE

In a search file, the blocks must be an even number of words. The first word of the block is reserved for file control use. The block format is:

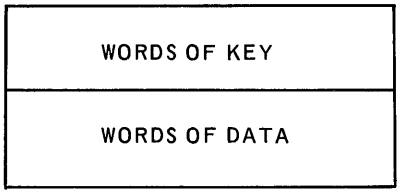


APPENDIX B. RECORD FORMATS

For sequential files, records may have a fixed or variable number of words. If the records are fixed, the entire record is available for data. If the records are variable, the first word of the record must contain, in binary, the size of the record; thus, this word is not available for data. The size word is counted as one of the words of the record.

For direct access files, the records must be in fixed length and an even number of words.

For search files, the records must be in fixed length and an even number of words. The key used for search must be an integer multiple of words. The key must occupy the initial words of the record.



APPENDIX C. FILE REGISTRATION

There must be a permanent registration of all mass storage file control files. This is accomplished by the use of the assign (@ASG) and the catalog (@CAT) control cards when adding new files. Decataloging a file is accomplished by using a control card (DCAT) in the same manner.

The catalog control card provides a means to make a mass storage file permanent beyond the duration of an active run. The @CAT control card is meaningful only if the necessary mass storage space for the file has been obtained previously from the allocator by means of the @ASG control card.

The user may initiate a run for the sole purpose of cataloging, or he may catalog following a data processing job. In the latter case he may prevent the intended cataloging by aborting the run before the @CAT control card comes up for processing. This may be desirable if the data for the file is rejected as erroneous or unreadable.

Following is a sequence of cards for a cataloging only run:

```
@RUN      SALES,950020
@ASG      42 = 50, 44 = 200
@CAT      42,BOSTON,A00123,S,400,100,0,32
@CAT      44,INVENT,B13570,S,264,44,0,32
```

The logical unit numbers 40 through 55 indicate FASTRAND, the file names are BOSTON and INVENT respectively; the file name is followed by the account number; "S" means sequential files, and the remaining parameters indicate words per block, words per record, and block header words.

Another sample run which shows cataloging during a run is as follows:

```
@RUN      SALES,950020
@ASG      42 = 50
@XQT,A    CREATE
@CAT      42,BOSTON,A00123,S,400,100,0,32
@XQT      EDIT
@FIN
```

APPENDIX D. FILE ASSIGNMENT

In order to obtain the use of the necessary tape drives, the requirements of the Real-Time Operating System must be followed. These provide for dynamic requests, or requests submitted in the control stream. Either method may be selected, but the assignment must take place before the OPEN$ macro instruction is reached in the running of the program.

In order to obtain the use of mass storage files, an @ASG control card must be provided and the assignments made before the file control OPEN$ macro instruction is reached. The executive also provides aids to permit concurrently operating programs to use the same file. For more detail see Appendix F.

The following sample @ASG control card shows assignment of a permanent FASTRAND file to a run.

```
@RUN      SALES,982123
@ASG      42 = BOSTON, 43 = INVENT
@XQT,A    UPDATE
@FIN
```

A sample @ASG control card for tape assignment is as follows:

```
@ASG      3 = DETAIL,4,5 = MAST, 6 = ERROR
```

APPENDIX E. ERROR HANDLING

During the running of a program, file control may detect errors of various types. At the time an error is discovered, file control tests the FDT of the file being referenced for the presence of an error location. If none is provided, the program will be aborted.

If there is an error address in the FDT, file control provides diagnostic information and then schedules the error address to receive control. The user may provide separate error routines or one routine for all files. The information supplied by file control enables the user to determine which file occasioned the error.

In constructing the error routine, the programmer must leave the first six words blank. File control will place the diagnostic information in these words, and then schedule ERROR + 6 as the Place to Go. The contents of the diagnostic table are as follows:

WORD		
1	ERROR TYPE	ERROR CODE
2	HANDLER REPORT	
3	FDT ADDRESS	
4	ADDRESS OF MACRO CALL	
5	REEL NUMBER	
6	BLOCK NUMBER	

The following list is a tabulation of error codes with an explanation of the type of error encountered. The codes are shown as the octal content of the first word of the error routine.

CODE	EXPLANATION
010001	The I/O handlers have reported an unrecoverable hardware error. (The handler report word may be analyzed to determine the exact nature of the error.)
020002	There has been a failure to assign a tape drive to the subject tape file.
020003	File control does not have a file directory record of the subject file and the presence of such record is mandatory; or the FDT does not agree with directory block produced by @CAT.
020005	A macro instruction has been executed to reference a file that is not open.
020006	A file that is already open has been referenced with another OPEN$ macro instruction.
020007	File control has detected incorrect information in the file description table.
020010	A macro instruction has been executed that is inappropriate to the file type or access.
020011	File control detects, at the end of a tape input reel, that a tape mark is not followed by either an end of file or end of reel label; or file control detects that a standard header label is not followed by a tape mark.
020012	File control detects that a variable record submitted is too large to be handled in the data block.
070001	The search file has just been filled to capacity. Any further inserts will not be processed if they require the splitting of a block.
070002	There is no room in the search file to insert this record.

APPENDIX F. PROGRAMMER CONSIDERATIONS

1. The greater the blocking factor the more efficient 418-III File Control operates. This fact is particularly significant in a volatile file.

2. Though it is not required, sorting of input will measurably increase the efficiency of the system when employing search or direct access files.

3. For efficiency, keys in search files should be reduced to minimum size (binary as opposed to XS-3) if they exceed one word.

4. The creation of a search file should be through use of the XTEND$ instruction and should be an independent program if availability of core storage is a factor.

5. If a record is out of sequence when creating or extending a search file it may be properly positioned in the file by issuing an NSERT$ instruction. This procedure is very time consuming and should be held to a minimum.

6. The logical unit (+40 for FASTRAND or +20 for drum) specified in the FDT (word 4) must also be used in the @ASG control card.

7. An error flag of 070002 can be either exceeding the limits of the file or exceeding the number of sections specified in the FDT (applies to search files only).

8. Block size for search files should be a multiple of record size + 1 because the first word of each block is used by file control.

9. For optional files that have no facilities assigned – file control will simulate end of file if that file is accessed by a GET$. There is no resultant action when a PUT$ is issued under the same conditions.

10. The error 070003 (no end of file record) occurs when records are to be extended to the end of an existing search file and file control cannot find the end of file record.

11. If the user does not want to use buffers he must specify the same address in both word 6 and word 9 of the FDT (insufficient core storage).

12. If a file is referenced that has not been opened and the file is optional, control is transferred to user's EOF coding.

13. It is recommended that the number of words per block for magnetic tape files be modules 4 (divisible by 4) to ensure compatibility with all tape units.

14. It is the responsibility of the user to detect end of file and end of reel conditions for magnetic tape input files that have labels omitted or nonstandard labels specified, and issue FORCE$, if appropriate.

15. The locate mode is a valid option for input files only.

APPENDIX G. MACRO CODE GENERATION

MACRO CALLING SEQUENCE		CODE GENERATED	
OPEN$	FILEA	0773000	
		+FILEA	
CLOSE$	FILEA	0773100	
		+FILEA	
END$		0773101	
GET$	FILEA	LB	(FILEA)
		SLJI	(G$ET)
PUT$	FILEA	LB	(FILEA)
		SLJI	(P$UT)
RLSE$	FILEA	LB	(FILEA)
		SLJI	(R$LSE)
FORCE$	FILEA	0773300	
		+FILEA	
RETN$	FILEA	LB	(FILEA)
		SLJI	(R$ETN)
SEEK$	FILEA	LB	(FILEA)
		SLJI	(S$EEK)
ADV$	FILEA	LB	(FILEA)
		SLJI	(A$DV)
NSERT$	FILEA	LB	(FILEA)
		SLJI	(N$SERT)
DLETE$	FILEA	LB	(FILEA)
		SLJI	(D$LETE)
UPDAT$	FILEA	LB	(FILEA)
		SLJI	(U$PDAT)
XTEND$	FILEA	LB	(FILEA)
		SLJI	(X$TEND)

APPENDIX H. FILE SHARING

In a multiprogramming environment there are occasions when two or more programs that are running concurrently may be accessing the same file on mass storage. Depending on file usage, this may or may not create problems of conflicting accesses. In order to assist the programmers in the solution of such problems, the following considerations should be adhered to.

- Each program expecting to share a file must provide a " lock list" in which it will hang up "Keep Out" signs when it needs exclusive use of some part of the file.
- Each program must examine the lock lists in the other programs and must avoid trespassing on the locked areas.
- The conventions for lock list format are left to the discretion of the installation programmers. Obviously, all parties must honor the conventions so that a workable system results.
- In order to examine areas in other programs, it is necessary to know where these areas are. For this purpose, the executive establishes and maintains share tables as required. (If a file is used only by one program, there is no reason to employ the sharing procedures. Also, there is the possibility that the installation will arrange so that conflicting programs are never in the job stream at the same time.)
- A program expecting to share a file requests the executive to enter the location of its lock list in a share table. It provides to the executive the location of the applicable lock list and a six-character file identification.
- The executive finds or creates this share table and posts the lock list location in it. (Provisions are made for seven such postings.) The executive then returns to the program with the location of the *share* table.
- The participating program is now in a position to examine the share table, and all lock lists stemming from it.
- The participating programs are expected to test for conflict before setting a lock on an area and to wait when a conflict is found.
- In order to release an area from lock, the program has only to cancel the lock in its own list.

The services provided by the executive system are as follows. The first two are of concern to the user.

- A dispatcher request to post the location of a lock list in a share table.
- A dispatcher request to remove a lock list from a share table.
- The necessary action, at program termination, of removing from all share tables any lock list locations of the terminating program.

The first two services are described in detail in the following text.

- GET SHARE TABLE

Calling Sequence: GETST$ addr,'id_1','id_2'

where: 'id_1','id_2' are the file identifiers (file name).

addr is the lock list address.

Code Generated: 0772300

+addr

+id_1

+id_2

Description: The GETST$ dispatcher request is the means for sharing data between programs; it also controls access to shared files.

Each request identifies a lock list by supplying a six-character identifier and an address which indicates where the lock list is maintained. A maximum of seven of these lock lists can be maintained for any one list identifier.

The executive maintains a table of the supplied addresses for each lock identifier and supplies the program making an entry in the table with the start address of the seven-word table. This control table is maintained in protected memory so that each program may only examine it.

Upon return from the GETST$ call AL contains the control table address if a successful entry was made. If the table is full, or no entry can be made for some other reason, a status of zero (0) is returned in AL.

Return is made to call+4 as a PTG.

- RELEASE SHARE TABLE

Calling Sequence: RELST$ addr,'id_1','id_2'

where: 'id_1', 'id_2' are the file identifiers (file name).

addr is the lock list address.

Code Generated: 0772301

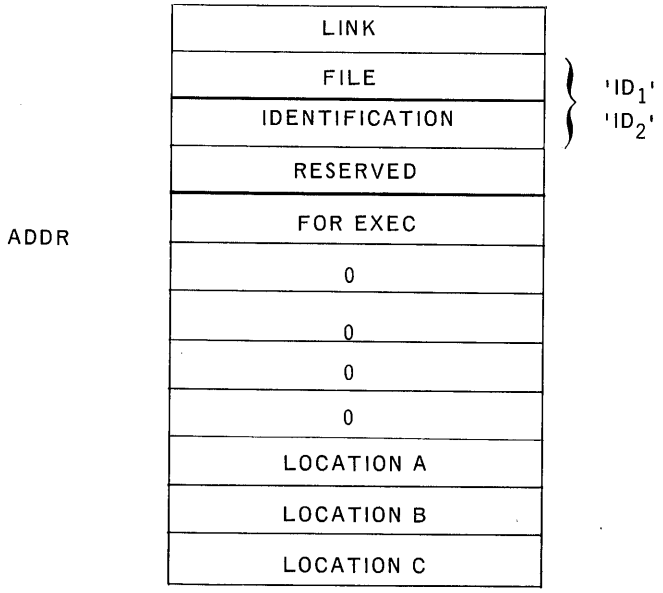
+addr

+id_1

+id_2

Description: In order to remove lock lists from the share table maintained by executive, the RELST$ dispatcher request is supplied. After the address is found in the table identified by id_1, id_2, it is removed and if it is the last entry, the control table is destroyed.

The share table is composed of twelve words with the final seven containing the lock list locations. This is shown in the diagram below:

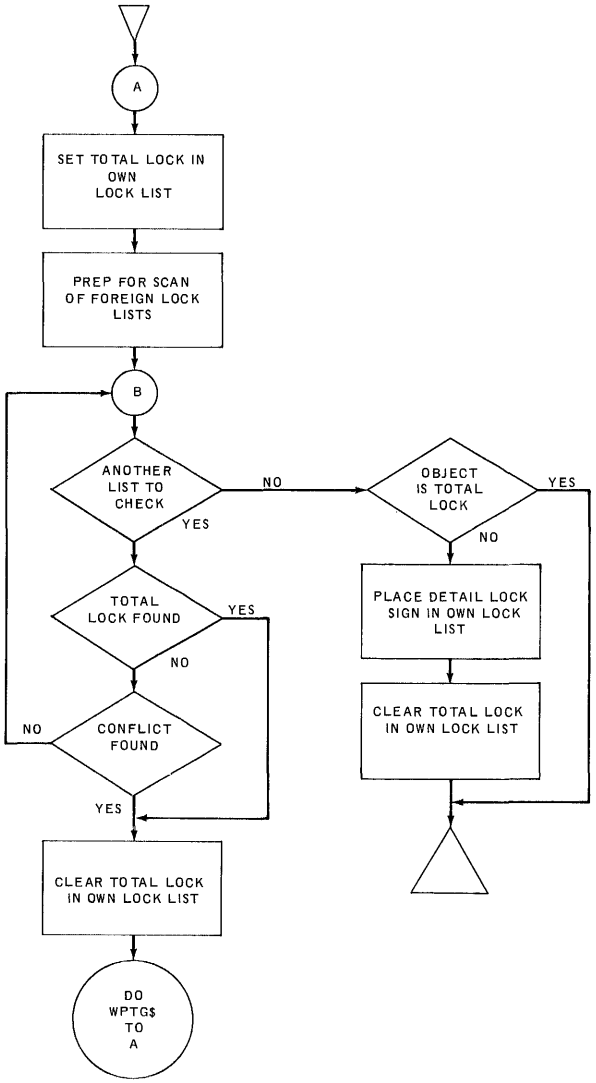


The user is given the location of the first of the seven words which may contain lock list locations. Zero within these words means there is no entry to be considered. The table is filled from the bottom up because this makes the coding of a scanning subroutine somewhat easier.

The features described previously provide the means by which the installation programmers can develop file sharing that is suitable for the specific files and circumstances encountered. The executive overhead is virtually nil and the additional overhead for locking and unlocking may be held to a minimum by careful construction of the scanning subroutine.

If the need for file sharing arises, it is expected that the installation will establish conventions for the lock list formats and will program standard subroutines for use in all sharing programs. Regarding such subroutines, the following suggestions are offered.

- Depending on the structure and handling of the subject file, it may be desirable to lock by record, by block, or by track.
- It may be desirable, at times, to lock an entire file during a rearrangement period.
- Provisions should be made for the situation that one program is interrupted while scanning lock lists, and another program gains control and begins to scan the same lists.
- The two points above can be covered together if the first packet of the lock list is designated as a total lock indicator. Using this plan, a scan routine could operate according to the following flowchart.



NOTE: The wait function need not be included in the subroutine if it appears better to exit with a reject signal and give the main program the chance to do other things before re-trying the scan.

APPENDIX I. FILE DESCRIPTION TABLE GENERATION PROC

To facilitate the orderly generation of a file description table, the PROC FDT$ is available. The expressions below correspond to the file description table entries defined in Section 4.

Calling sequence for sequential and direct access files:

FDT$ Type, 'NNN', 'NNN', O, VOL, CAT, DEV, ACC, REW, UNITB, UNITA, L, WORKSPACE, EOF, ERROR, S, BUFFER, LABEL WORKSPACE, B, HEADER, A, TRAILER, N, WPB, V, WPR, RET, MF, LAB, PD, TFG, MR, LDR

Calling sequence for search files:

FDT$ Type, 'NNN', 'NNN', DEV, ACC, UNITA, WORKSPACE, EOF, ERROR, BUFFER, MASTER BLOCK, WPB, WPR, SPACE, CONTROL, KEY

where: 'NNN', 'NNN' is the file name.

UNIVAC
418-III

FILE CONTROL
ROUTINE
UP-7687

UNIVAC SYSTEMS PROGRAMMING LIBRARY SERVICES

RELEASE

UNIVAC 418-III Real-Time System Library Memo 7 announces the release and availability of "UNIVAC 418-III Real-Time System File Control Routine Programmers Reference," UP-7687, covers and 49 pages. This is a Standard Library Item (SLI).

This document describes the functions and uses of the UNIVAC 418-III Real-Time System File Control Routine.

A detailed description is given for the MACRO instructions used for the magnetic tape and mass storage subsystems. Information is also presented to aid in the formatting of file structure.

Distribution of UP-7687 has been made as indicated below. Additional copies may be requisitioned from Holyoke, Massachusetts via a Sales Help Requisition through your local Univac Representative.

MANAGER
Documentation and Library Services

TO LISTS:

ATTACHMENTS: UP-7687 plus Library Memo 7 to Lists 10U, 217, 630, 692 and S.P.L.S. Lists 57 and 58.

THIS SHEET IS: UNIVAC 418-III Real-Time System Library Memo 7.

DATE: August 8, 1969

UP-7687