60499300

CONTROL DATA CORPORATION

CYBER RECORD MANAGER ADVANCED ACCESS METHODS VERSION 2 REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 1
NOS/BE 1

REVISION RECORD	
REVISION	**DESCRIPTION**
A	Original release.
(3-31-78)	
Publication No. 60499300	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

or use Comment Sheet in the back of this manual

CYBER Record Manager Advanced

Manual Title Access Methods Version 2 Reference Manual Pub. No. 60499300 Rev. A

As part of Control Data's continuing quality improvement program, we invite you to complete this questionnaire so that you may have a more direct influence on the manuals you use.

Please rate this manual for each general and individual category on a scale of 1 through 5 as follows:

1 - Excellent 2 - Good 3 - Fair 4 - Poor 5 - Unacceptable

I. Writing Quality _____

- A. Technical accuracy _____
- B. Completeness _____
- C. Audience defined properly _____
- D. Readability _____
- E. Understandability _____
- F. Organization _____

II. Examples _____

- A. Quantity _____
- B. Placement _____
- C. Applicability _____
- D. Quality _____
- E. Instructiveness _____

III. Format _____

- A. Type size _____
- B. Page density _____
- C. Art work _____
- D. Legibility _____
- E. Printing/Reproduction _____

IV. Miscellaneous _____

- A. Index _____
- B. Glossary _____

V. Please provide a yes or no answer regarding manuals in general:

- A. I prefer that a manual on a software product be as comprehensive as possible; physical size is of little importance. _____
- B. I prefer that information on a software product be covered in several small manuals, each covering a certain aspect of the product. Smaller manuals with limited subject matter are easier to work with. _____
- C. I am interested primarily in reference manuals designed for ease of locating specific information. _____
- D. I am interested primarily in user guides designed to teach the user about a product or certain capabilities of a product. _____

VI. We recognize that we have a wide variety of users. Please identify your primary area of interest or activity:

- A. Student _____
- B. Applications programmer _____
- C. Systems programmer _____
- D. How many years programming experience do you have? _____
- E. What languages
  1. Algol _____
  2. Basic _____
  3. Cobol _____
  4. Compass _____
  5. Fortran _____
  6. PL/I _____
  7. Other _____
- F. Have you ever worked on non-CDC equipment? _____
  1. If yes, approximately what percent of your experience is on non-CDC equipment? _____
  2. How do you rate CDC manuals against other similar manuals using the 1-5 ratings. (Example: XYZ Corp. 2 means XYZ manuals are good as compared to CDC manuals.)
     - Burroughs _____
     - DEC _____
     - Hewlett-Packard _____
     - Honeywell _____
     - IBM _____
     - NCR _____
     - Univac _____
     - Other _____ _____

General Comments _____

_____

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

CUT ON THIS LINE

FOLD

FOLD

STAPLE

STAPLE

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Cover	–
Title Page	–
ii thru viii	A
1-1 thru 1-3	A
2-1 thru 2-11	A
3-1 thru 3-10	A
4-1 thru 4-13	A
5-1 thru 5-7	A
6-1 thru 6-8	A
7-1 thru 7-11	A
A-1 thru A-4	A
B-1 thru B-16	A
C-1 thru C-3	A
D-1 thru D-6	A
E-1, E-2	A
F-1	A
G-1	A
H-1, H-2	A
Index-1 thru -8	A
Comment Sheet	A
Return Env.	–
Cover	–

Page	Revision

Page	Revision

PREFACE

CYBER Record Manager Advanced Access Methods (AAM) Version 2 operates under control of the following operating systems:

NOS 1 for the CONTROL DATA® CYBER 170 Models 171, 172, 173, 174, 175; CYBER 70 Models 71, 72, 73, 74; and 6000 Series Computer Systems.

NOS/BE 1 for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, 74; and 6000 Series Computer Systems.

AAM is a part of DMS-170, the data management system that also includes CYBER Database Control System Versions 1 and 2 and Data Description Language Versions 2 and 3. AAM can be used independently of DMS-170.

AAM input and output facilities are available to users of COMPASS assembly language through macro calls. User programs, COBOL, FORTRAN Extended, and PL/I use AAM for input/output operations. The user programs communicate with AAM either through the compiler, using the calls supplied within the languages, or with AAM macros.

Intended as a primary document for COMPASS programmers, this manual presents background information and operational specifications for AAM. COBOL, FORTRAN Extended, and Sort/Merge programmers can use this manual as a source for AAM terminology and concepts; specific language interfaces are detailed in the appropriate reference manuals. The user is assumed to be familiar with the operating system at the installation and with file organization and manipulation.

Information necessary for a complete understanding of AAM use is contained in the following publications:

Publication	Publication Number
NOS/BE 1 Reference Manual	60493800
NOS 1 Reference Manual, Volume 1	60435400
NOS 1 Reference Manual, Volume 2	60445300
CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual	60495700
CYBER Record Manager Version 1 User's Guide	60495800
Common Memory Manager Version 1 Reference Manual	60499200
COMPASS Version 3 Reference Manual	60492600
CYBER Loader Reference Manual	60429800

CDC manuals can be ordered from Control Data Literature and Distribution Services, 8001 East Bloomington Freeway, Minneapolis, MN 55420

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

An interface between user programs and system input/output routines is provided by the Advanced Access Methods (AAM). AAM subsystems exist in the NOS and NOS/BE operating systems. AAM also provides consistent error processing and maintenance of different file organizations.

AAM routines are used by some compilers and are available for user programs. Use of AAM by compilers and user programs extends input/output compatibility to both the system and application program levels.

The primary task of AAM is record input/output for files on supported devices. The various types of records and file organizations must be identified for AAM. These and other file characteristics must be set by the user in the file information table (FIT). The FIT is divided into fields that describe certain aspects of the file. Refer to appendix D for the exact structure of the FIT.

The following terms are relevant to AAM and related systems:

AAM (Advanced Access Methods)

A file manager that processes indexed sequential, direct access, and actual key file organizations and supports the Multiple-Index Processor.

BAM (Basic Access Methods)

A file manager that processes sequential and word addressable file organizations.

CRM (CYBER Record Manager)

A generic term relating to both BAM and AAM as they run under the NOS and NOS/BE operating systems.

MIP (Multiple-Index Processor)

A processor that allows AAM files to be accessed by alternate keys.

AAM supports two types of MIP: initial MIP and extended MIP.

FILE ORGANIZATIONS

Three file organizations are supported by AAM:

Indexed sequential

Records are in order by primary key and can be accessed sequentially or randomly.

Direct access

Records are not in any recognized order and are accessed by key manipulation.

Actual key

Records are accessed by a primary key containing the block and record number within the file.

AAM supports two types of indexed sequential files: initial indexed sequential files and extended indexed sequential files.

AAM MACROS

The file information table is established for the file by the FILE macro encountered at assembly time. The FILE macro establishes the FIT in the field length of the user program at the point at which it is called. This macro can contain only the file name and file organization or it can have user-specified parameters describing the particular file. FIT fields are assumed by AAM through default values when not specified as macro parameters. AAM macros and functions are listed in table 1-1. Macros are grouped according to their associated functions.

The applicability of some AAM macros depends on the file organization established by the user. Table 1-2 indicates the applicability of each macro to the various file organizations and to files processed by MIP.

Macros are discussed according to each file organization in section 4, File Processing. Consequently, material is presented redundantly for the benefit of a programmer who uses this manual to reference a particular file organization. The format of each macro and a general description are presented in section 5, File Processing Macros.

Macro statements are coded in COMPASS format. Each statement can contain a location field, a macro name in the operation field, a variable field, and a comment field. Any field is terminated by one or more blanks. A macro statement begins in character position 1 of an 80-column card image and continues through column 72. Columns 73 through 80 are used for sequencing. Suggested coding conventions are shown in figure 1-1.

The allocation of the columns in COMPASS format is as follows:

1	Comma (continuation), asterisk (comment line), or other (beginning of new statement)
2 thru 9	Location field entry, left-justified
10	Blank
11 thru 16	Operation field entry, left-justified
17	Blank
18 thru 29	Variable field entry, left-justified
30	Beginning of comment

TABLE 1-1. AAM MACROS

Function	Macro	Action Taken
File creation and maintenance	FILE	Creates the file information table (FIT). In addition to this macro, a FILE control statement is available to supply FIT information.
	FETCH	Retrieves the value of a specified field in the FIT.
	STORE	Sets the value of a field in the FIT.
	SETFIT	Sets values in fields in the FIT with values supplied through the FILE control statement.
File initialization and termination	OPENM	Prepares a file for processing.
	FLUSHM	Flushes buffers to bring mass storage files into a state of equilibrium.
	CLOSEM	Terminates file processing.
Data transfer	GET	Transfers data from a file to the working storage area.
	PUT	Transfers data from the working storage area to a file.
File updating	DELETE	Deletes a record from a file.
	REPLACE	Replaces a record in a file.
File positioning	SKIP	Repositions a file backward or forward.
	REWINDM	Rewinds a file to beginning-of-information (BOI).
	SEEK	Provides an overlap between input/output and processing by positioning while processing.
	START	Positions a file to a record that satisfies a specific condition.

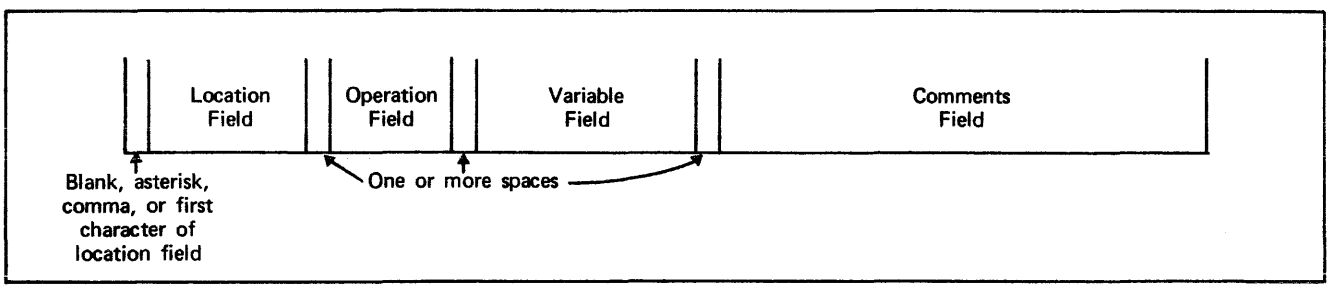


Figure 1-1. COMPASS Format

TABLE 1-2. APPLICABILITY OF MACROS

Macro	File Organization				Initial MIP	Extended MIP
	Initial Indexed Sequential	Extended Indexed Sequential	Actual Key	Direct Access		
CLOSEM	X	X	X	X	X	X
DELETE	X	X	X	X		
FILE	X	X	X	X	X	X
FLUSHM		X				
GET	X	X	X	X	X	X
GETN	X	X	X	X	X	X
GETNR		X				X
OPENM	X	X	X	X	X	X
PUT	X	X	X	X		
REPLACE	X	X	X	X		
REWINDM	X	X	X	X	X	X
SEEK	X	X	X	X	X	X
SETFIT	X	X	X	X	X	X
SKIP	X	X	X		X†	X†
START	X	X			X	X
STORE	X	X	X	X	X	X

†SKIPFL macro only.

FILE STRUCTURES 2

A hierarchical data structure is recognized in a progression from the character level to the largest grouping of data, the file. The AAM user can describe file structure by file organization and record type.

LOGICAL STRUCTURE

The logical structure of an AAM file is user-controlled. The following terms are applicable to the logical file structure and are used throughout this manual:

Record

A record is a group of related characters. A character is represented in six bits as internal display code. A record is the smallest collection of information passed between AAM and the user. The user defines the structure and characteristics of records within a file by declaring a record format. The beginning and ending points of a record are implicit within each format.

Block

A block contains one or more records. Block structure is interwoven with the physical recording format; unlike other logical file structure declarations, the block structure is transparent in use. AAM constructs blocks from the records supplied by the user and supplies the user with records as requested. The user is unaware of block boundaries.

File

A file is a logically connected set of information; it is the largest collection of information that can be addressed by a file name. All data in a file is stored between beginning-of-information (BOI) and end-of-information (EOI).

PHYSICAL STRUCTURE

The following terms pertain to the physical means used to record files:

Input/output device

Any storage medium supported by the operating system.

Rotating mass storage (RMS)

Disk or disk pack.

Mass storage device

Disk, disk pack, or extended core storage (ECS).

PRU device

All mass storage devices. The operating system superimposes a physical structure over the user-declared AAM file structure on all files that reside on PRU devices.

Physical record unit (PRU)

The smallest unit of information that can be transferred between a peripheral storage device and central memory. The PRU size is 640 characters.

Short PRU

A PRU containing fewer than the 640 characters defined for a PRU.

System-logical-record

A group of PRUs terminated by a short or zero-length PRU.

AAM controls the physical file position; the user controls only the logical file position.

FILE ORGANIZATIONS

AAM supports four file organizations: initial indexed sequential, extended indexed sequential, direct access, and actual key. The following paragraphs describe the structure of each file organization.

INITIAL INDEXED SEQUENTIAL FILE STRUCTURE

An initial indexed sequential file consists of a file statistics table, index blocks, and data blocks. Each block is an integral number of PRUs less one central memory word and is treated as a system-logical-record. Both index and data blocks are fixed-length blocks; index blocks need not be the same size as data blocks.

Each record in the initial indexed sequential file is identified by a primary key value. Records are stored in data blocks in increasing primary key sequence. Index blocks contain primary key information used to retrieve any record in the file.

The file statistics table (FSTT) maintains file integrity by preventing user actions that would destroy the file. When the file is created, the user defines the file and key structure that must remain the same for the life of the file. This information is stored in the FSTT and is used to guide processing as long as the file exists. If the user sets a field in the FIT to a value that does not conform to the FSTT, the value is rejected and the job is terminated. The FSTT stores accumulated statistics related to file access; if applicable, it also stores a user-supplied collating sequence for ranking symbolic keys.

The logical structure of an initial indexed sequential file is shown in figure 2-1. The blocks identified as D01 through D09 are data blocks; those identified as I01 through I03 are the first level index blocks and I11 is the primary or second level index block.

The physical structure of an initial indexed sequential file is shown in figure 2-2. FSTT is the file statistics table, D01 through D09 are the data blocks, and I01 through I03 are the index blocks.

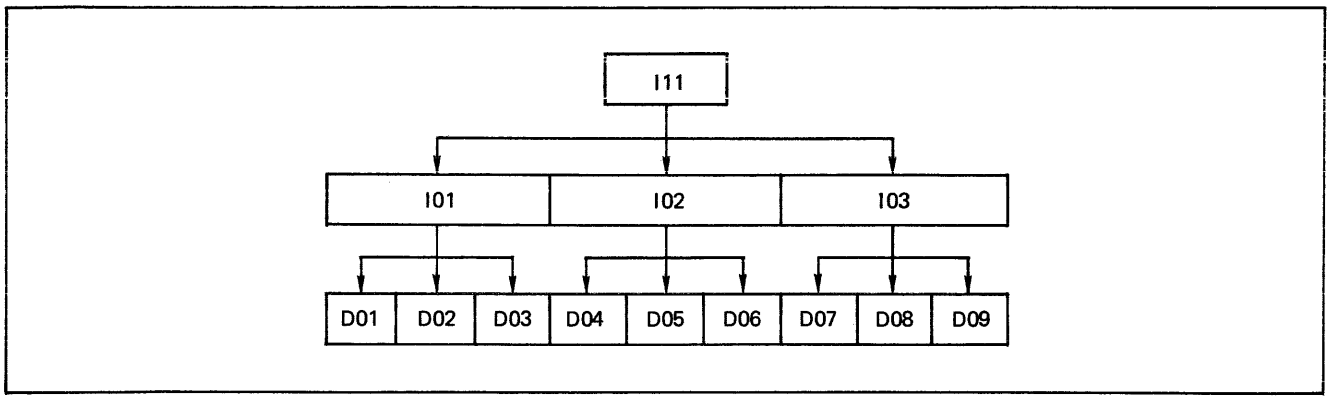


Figure 2-1. Logical Structure of an Indexed Sequential File

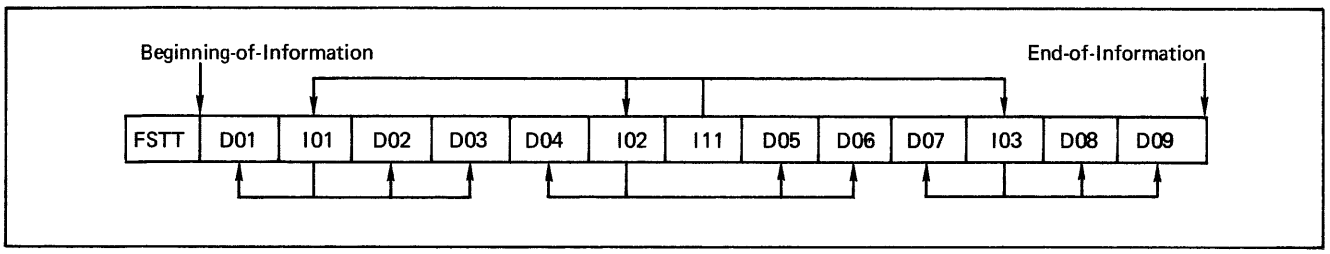


Figure 2-2. Physical Structure of an Indexed Sequential File

When an initial indexed sequential file is created, data block size, index block size, record size, and key characteristics must be specified for AAM to construct the data blocks, index blocks, and key entries for the file. A maximum of 10 active initial indexed sequential files per job step can be processed.

Data Blocks

A data block in an initial indexed sequential file contains a header, an optional checksum word, user records, primary key entries, and padding. The size and characteristics of the data block are determined by the setting of various fields in the FIT when the file is created. The specific FIT fields that are used during file creation are discussed in section 4, File Processing. The formats of the fields are detailed in section 3, File Information Table.

The header in a data block contains a pointer that chains the block in a forward direction to permit sequential reading without an index. It also contains a relative pointer to the first word of unused space in the block and the size of the unused space. The standard block header format is shown in figure 2-3. An installation can choose to increase the size of the header when the system is installed.

User records in a data block can be fixed or variable length. Only whole records can be in a data block; records cannot span blocks. Primary keys (one for each record in the block) are stored separately from records to reduce search time. Records are stored in ascending primary key sequence. The first record in the first data block has the lowest primary key value in the file, and the last record in the last data block has the highest key value.

A primary key entry is stored in the data block for each record in the block. The key entry is an integral number of central memory words. It contains the primary key and a pointer to the corresponding record in the block.

Padding in a data block is the amount of space that is not to be used for writing records during file creation. This space can then be used to insert new records during subsequent runs that update the file. The amount of padding is specified as a percentage of the total block size.

Index Blocks

An index block in an initial indexed sequential file contains a system-supplied header, an optional checksum word, primary key entries, and padding. The size of an index block need not be the same as the size specified for a data block. Other index block characteristics are specified through the FIT. Refer to section 4, File Processing, for the specific FIT fields and to section 3, File Information Table, for the format of the fields.

Index block records are created and maintained by AAM. A primary key entry consists of a primary key value and a PRU number. The primary key value is the lowest key value in the next lower level index block or in a data block; the PRU number points to the beginning of the block. Key entries within an index block are in ascending primary key sequence.

Index blocks are organized into as many levels as necessary to ensure only one index block at the highest or primary level. The maximum number of levels that can exist for a file is specified in the FIT; no more than 63 levels can be specified.

Padding in an index block is the same as in a data block. Data blocks and index blocks do not have to have the same percentage of padding. The default index block padding factor is five percent; a zero value is changed to the default value of five. Any nonzero padding factor less than 100 is acceptable unless data block calculation results in a block that cannot be accommodated in the user field length.

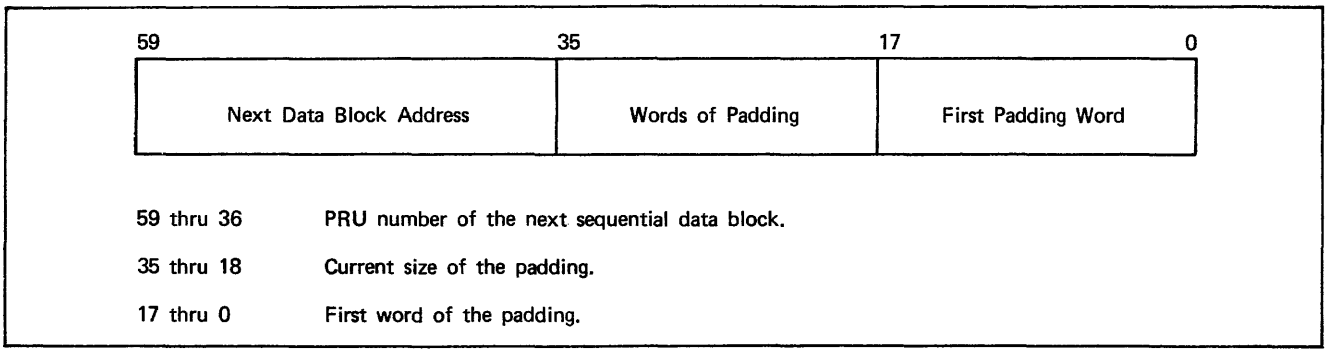


Figure 2-3. Initial Indexed Sequential Block Header Format

EXTENDED INDEXED SEQUENTIAL FILE STRUCTURE

An extended indexed sequential file consists of a file statistics table, index blocks, and data blocks. Each block is an integral number of PRUs less two central memory words and is treated as a system-logical-record. Both index and data blocks are fixed-length blocks and must be the same size.

Each record in the extended indexed sequential file is identified by a unique primary key value. Records are stored in data blocks in increasing primary key sequence. Index blocks contain primary key information used to retrieve any record in the file.

The file statistics table (FSTT) maintains file integrity by preventing user actions that would destroy the file. When the file is created, the user defines the file and key structure that must remain the same for the life of the file. This information is stored in the FSTT and is used to guide processing as long as the file exists. If the user sets a field in the FIT to a value that does not conform to the FSTT, the value is rejected and the job is terminated. The FSTT stores accumulated statistics related to file access; it also stores a default or user-supplied collating sequence for ranking symbolic keys.

The logical and physical structures of an extended indexed sequential file are the same as shown in figure 2-1 and figure 2-2, respectively, for an initial indexed sequential file.

When an extended indexed sequential file is created, data and index block size, record size, and key characteristics must be specified for AAM to construct the data blocks, index blocks, and key entries for the file.

Data Blocks

A data block in an extended indexed sequential file contains a header, user records, record pointers, and padding. The size and characteristics of the data block are determined by the setting of various fields in the FIT when the file is created. The specific FIT fields that are used during file creation are discussed in section 4, File Processing. The formats of the fields are detailed in section 3, File Information Table.

The two-word header in a data block contains a pointer that chains the block in a forward direction to permit sequential reading without an index. It also contains the size of the unused space, a record count, and other flags and counts. An optional checksum can also be included in the header.

User records in a data block can be fixed or variable length. Only whole records can be in a data block; records cannot span blocks. Records are stored in ascending primary key sequence. The first record in the first data block has the lowest primary key value in the file, and the last record in the last data block has the highest key value.

One or more record pointers are stored in a data block. The record pointer is a 30-bit field; two record pointers are stored in a word. The pairs of record pointers are stored at the end of the data block beginning with the last word. The record pointer contains the last word address plus 1 of the record; the address is relative to the beginning of the first record in the block. It also contains the number of trailing characters that are not part of the record and processing flags. If all records in the block are the same length, only one record pointer is needed.

Padding in a data block is the amount of space that is not to be used for writing records during file creation. This space can then be used to insert new records during subsequent runs that update the file. The amount of padding is specified as a percentage of the total block size. The default value of zero percent can be used for files that are expected to grow mainly by sequential inserts or by adding records at the end of the file.

Index Blocks

An index block in an extended indexed sequential file is structured the same as a data block with a system-supplied header, records, a record pointer, and padding. The size of an index block is the same as the size specified for a data block. Other index block characteristics are specified through the FIT. Refer to section 4, File Processing, for the specific FIT fields and to section 3, File Information Table, for the format of the fields.

Index block records are created and maintained by AAM. A record consists of a primary key value and a PRU number. The primary key value is the lowest key value in the next lower level index block or in a data block; the PRU number points to the beginning of the block. Records within an index block are in ascending primary key sequence.

Index blocks are organized into as many levels as necessary to ensure only one index block at the highest or primary level. The maximum number of levels that can exist for a file is specified in the FIT; no more than 15 levels can be specified.

An index block requires only one record pointer because all records in the block are the same length. The record pointer, which is the same as described for data blocks, is stored in the last word of the index block.

Padding in an index block is the same as in a data block. Data blocks and index blocks, however, do not have to have the same percentage of padding. The default index block padding factor is zero percent.

ACTUAL KEY FILE STRUCTURE

An actual key file consists of a file statistics table and a number of data blocks. New data blocks are created at end-of-information as the file grows. Block size can be specified by the user or a default size can be determined by the system. Padding can be defined for data blocks, or block size can be defined to allow for an increase in record size.

The data block contains a fixed number of slots for data records that can be fixed or variable length. The block number and slot number assigned to each record as it is written become the permanent address (primary key) of the record. When a record is written on the file, the primary key can be specified by the user or it can be determined by AAM. If a primary key value of zero is specified by the user, AAM determines where to write the record and returns the block number and slot number to the user.

A maximum of 10 active actual key files per job step can be processed. The logical structure of an actual key file is shown in figure 2-4.

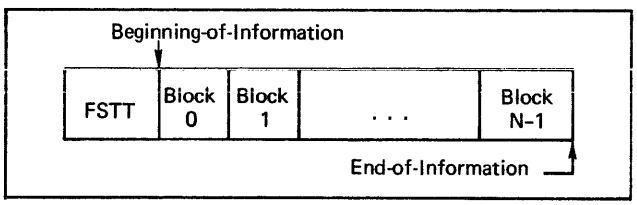


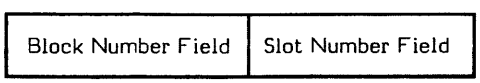
Figure 2-4. Logical Structure of an Actual Key File

The first block is the 63-word file statistics table containing file information and a pointer to end-of-information. The remaining blocks are fixed-length data blocks. Data block format is shown in figure 2-5.

When the block contains overflow record headers, the number of record headers exceeds the number of records in the block. Data records within the block are ordered by slot number with the smallest number being the first record. Record headers are placed at the bottom of the block in inverse order of the data records. A block checksum, if specified for the file, is contained in the last word of each block.

Actual Keys

Records are stored and retrieved by an actual key, which is the primary key. The actual key specifies a data block number and a slot number (record position) within the block. Keys have the following format:



Key length is specified by the user when the file is created. Length can range from 2 to 47 bits; the key must be right-justified within a central memory word. The low-order bits

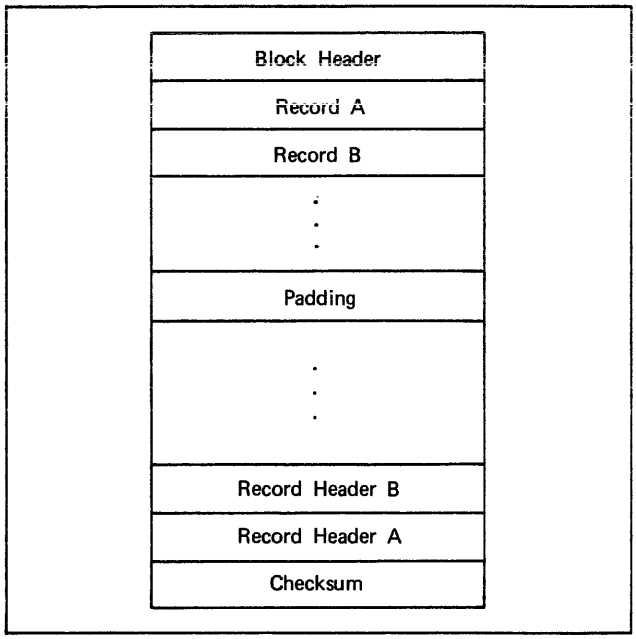


Figure 2-5. Actual Key Data Block Format

are the record position (slot number) and the high-order bits are the block number. The number of low-order bits used for slot numbers is determined by the blocking factor, which is specified before the file is opened. The remaining bits in the key are used to designate the data block number; therefore, the key length determines the maximum file size.

If the specified blocking factor is a power of 2, all integer key values up to the limit set by the key length are permissible for an actual key. If the blocking factor is not a power of 2, some integers are not allowed to be keys. Larger blocking factors (64 versus 8) provide better storage density for files with variable-length records; eight records per block is the default blocking factor. Actual keys need not be contained within the records.

Overflow

Overflow occurs in two ways:

The user specifies the actual key for a write operation and the specified block has insufficient empty space to contain the new record.

The user attempts to replace a record with a new larger record and the block containing the old record has insufficient empty space to contain the new record.

In either case, the record is inserted into a different block that has enough empty space. An overflow record header, which contains a pointer to the record, is placed in the block that would have normally contained the record. This requires two record slots; one contains the overflow record header and the second one contains the record.

Logically, the overflow slot that contains the record is still empty. If a GET macro is issued to retrieve a record from that overflow slot, an error is issued. If a PUT macro is issued to write a record in that slot, the overflow record is moved to a different block and the pointer in its overflow record header is updated. Overflow records always require two accesses to retrieve the record.

The formats for block headers and overflow record headers are shown in figure 2-6. Figure 2-7 shows the structure of a data record. The format of a record header is shown in figure 2-8.

DIRECT ACCESS FILE STRUCTURE

A direct access file contains a file statistics table, home blocks, and (under certain conditions) overflow blocks. All blocks are fixed length. The following terms have specific meaning in relation to direct access files:

Primary key

A primary key is a contiguous bit string that always appears in a direct access record. It is hashed to produce the location of the home data block containing the record.

Hashing

Hashing denotes the method of using primary keys to search for relative home block addresses of direct access records.

Home block

A home block is a block whose relative address is computed by hashing primary keys. A home block contains synonym records whose keys hash to that relative address.

Synonyms

Synonyms are records whose keys hash to the same home block.

Overflow record

An overflow record is a record whose key has been hashed to a home block that is already filled.

Overflow block

An overflow block is the second or subsequent block in a chain that starts at a home block. It contains overflow records and can contain records belonging to more than one overflow chain.

Chain

A chain consists of blocks that are logically connected by forward and/or backward pointers. Home blocks and overflow blocks are chained both forward and backward.

The relative position of records within a direct access file is not important. A record is stored and retrieved by hashing its primary key to produce the relative address of a home block. When a home block is filled, the record can be placed in another home block or in a system-generated overflow block; the placement of the record depends on the overflow record storage option selected by the user.

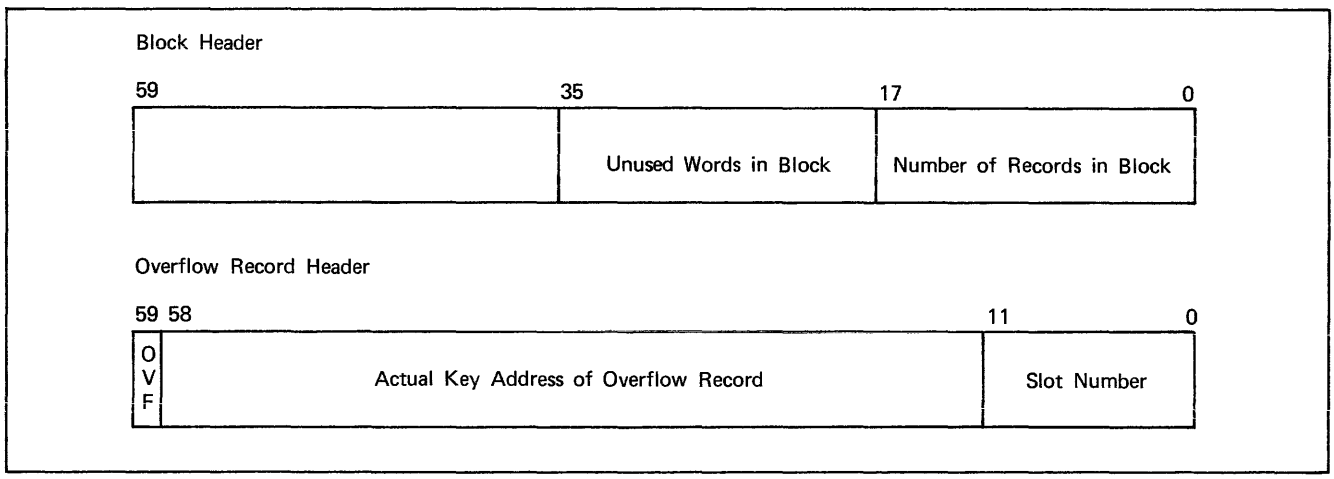


Figure 2-6. Actual Key Block and Overflow Record Header Formats

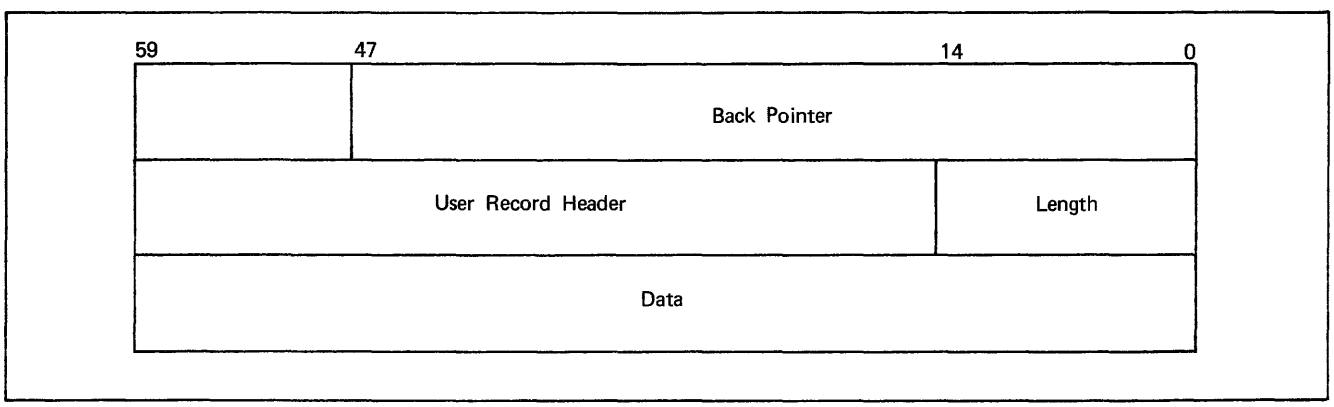
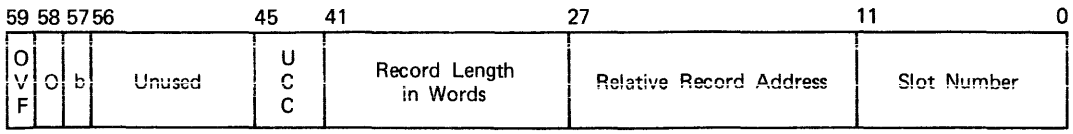


Figure 2-7. Actual Key Data Record Structure



59	Overflow bit, indicating record header format. Zero implies a normal header; 1 indicates an overflow record header.
58	Overflow record bit. If set, the record described by the record header has overflowed. Overflow records contain a one-word back-pointer. A back-pointer is the actual key address of the corresponding overflow record header.
57	User header bit. If set, the record contains the user header. The user can optionally divide records into user header and data portions. If this is done, user header length in words is stored in the rightmost 15 bits of the first word of the record. The header indicator bit (HB) in the FIT is examined on PUTs and REPLACEs to determine if the incoming record is divided into header and data portions. On accesses, the HB field indicates which portion of the record is to be returned.
56 thru 46	Unused.
45 thru 42	Unused character count; specifies the number of characters (0 through 9) that contain no information in the last word of the record. Record lengths are supplied in characters by the user and converted to words and unused characters internally.
41 thru 28	Record length field; specifies the number of words necessary to contain the record. This length includes the back-pointer and user header, if present.
27 thru 12	Relative record address is a pointer to the record described by the record header. The pointer is relative to the first word of the block.
11 thru 0	Slot number; indicates the block slot that the record header is using. Record headers are ordered by this field.

Figure 2-8. Actual Key Record Header Format

The logical structure of a direct access file is shown in figure 2-9. FSTT is the file statistics table, H1 through H6 are the home data blocks, and OV1 through OV3 are the overflow blocks.

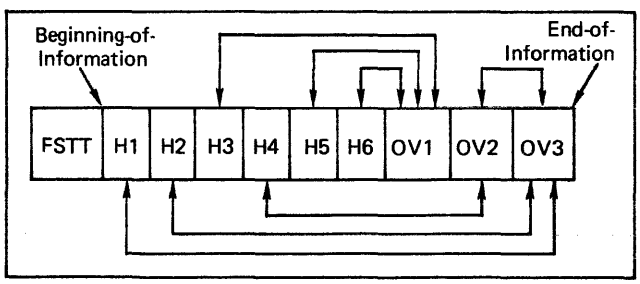


Figure 2-9. Logical Structure of a Direct Access File

File Storage Allocation

Mass storage space is preallocated when a direct access file is opened. Record storage and retrieval are by primary key; the location of a record within a file is determined by hashing the primary key to a relative block address.

Records are grouped in fixed-size blocks according to the results of the primary key hashing. When more records hash to a home block than the block can contain, overflow blocks are created if that option has been selected by the user. Overflow blocks are linked bidirectionally to form a chain.

Extensive analysis of the record key structure, key range, and key distribution is necessary to implement a randomly organized file in an optimum manner. An ideal hashing algorithm distributes records uniformly across all home blocks. Because no single hashing routine can produce optimum results for all data, a user-supplied hashing routine can be used. Hashing routines are discussed in further detail in section 4, File Processing.

File Blocking

Each direct access block (home or overflow) is an integral number of PRUs less one central memory word and is treated as a system-logical-record. An installation parameter determines the number of words at the beginning of each block for the block header. This parameter allows the user to obtain header space. When the system is installed, the user can choose to increase the size of the header. The first word of the header contains 30-bit backward and forward PRU pointers to form an overflow chain. Optionally, the last word of each block contains a block checksum. Records are stored in the remaining words as received, beginning with the word following the last word of the header.

A direct access primary key is a contiguous bit string ranging in length from 1 to 255 characters. It must always appear in the same character position for all records. Records start on word boundaries. The first word of each record is a record header with the format shown in figure 2-10.

RECORD TYPES

Eight external record types are supported; these record types are listed in table 2-1. Except for S type records and W type records, each record type is described in detail in the following paragraphs. AAM considers S and W type records to be U type records.

When records are written on an initial indexed sequential, direct access, or actual key file, the record type specification is used to compute the record length in characters. This length is recorded in the header word that accompanies each record in these files. When the record is read, record type is ignored and the number of characters indicated by the length field in the header is returned to the program.

The numbering conventions for describing a record or the position of a control field or key field in a record are summarized in figure 2-11. All record lengths are specified by character count. Values are normally unsigned positive integers, counting in a decimal system. For extended indexed sequential files, the maximum record length (MRL) field in the FIT must not exceed $10(2^{13}-5)$ characters.

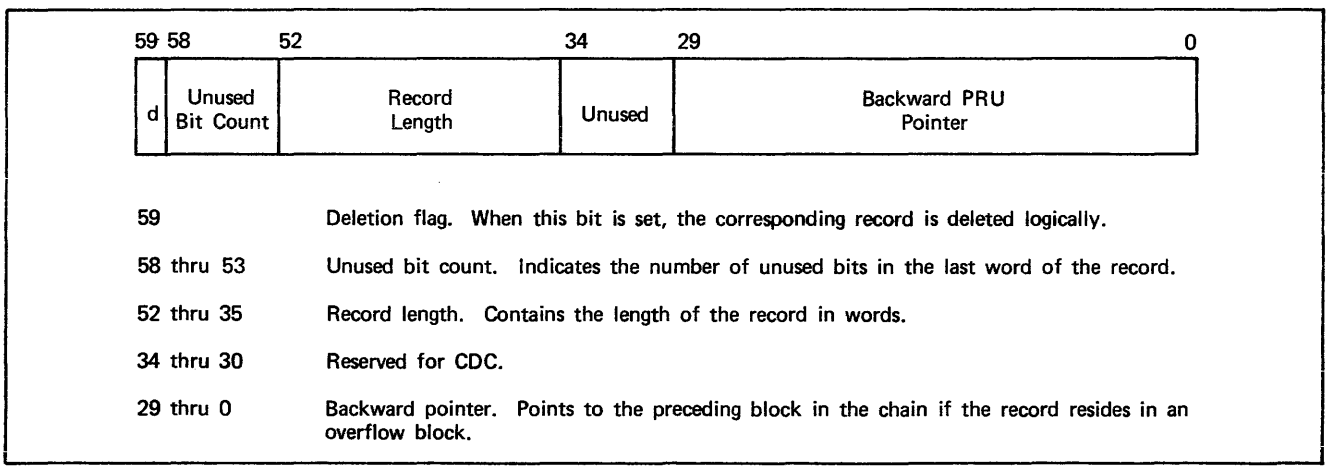
DECIMAL CHARACTER COUNT, D TYPE RECORDS

Records in a file with D type records vary in length. The length of an individual record is specified in a record length field located within the record. The position of the record length field is specified by two fields in the FIT. The length field beginning character position (LP) field indicates the character position (numbering from 0) in which the record length field begins. The length field length (LL) field specifies the number of characters in the record length field, which cannot exceed six characters.

When a D type record is written, the record length field cannot contain a value greater than the value of the maximum record length (MRL) field in the FIT. The maximum length that can be specified in the MRL field is $10(2^{12}-5)$ characters for extended indexed sequential files and $10(2^{17}-1)$ characters for all other files. The length value specified in the record length field is given as right-justified display code filled with zeros or blanks. If the COMP-1 (C1) field in the FIT is set to YES, the record length field is a COMP-1 (binary) field. If the sign overpunch (SB) field in the FIT is set to YES, the record length field is a sign-overpunch field.

The minimum record length (MNR) field in the FIT specifies the minimum number of characters for the D type record. The default value for the MNR field is the sum of the values in the LP and LL fields; however, the MNR field can be set to a greater value.

Figure 2-12 shows an example of a D type record. The record length field is three characters in length (the LL field is set to 3) beginning in character position 22 (the LP field is



59	Deletion flag. When this bit is set, the corresponding record is deleted logically.
58 thru 53	Unused bit count. Indicates the number of unused bits in the last word of the record.
52 thru 35	Record length. Contains the length of the record in words.
34 thru 30	Reserved for CDC.
29 thru 0	Backward pointer. Points to the preceding block in the chain if the record resides in an overflow block.

Figure 2-10. Direct Access Record Header Format

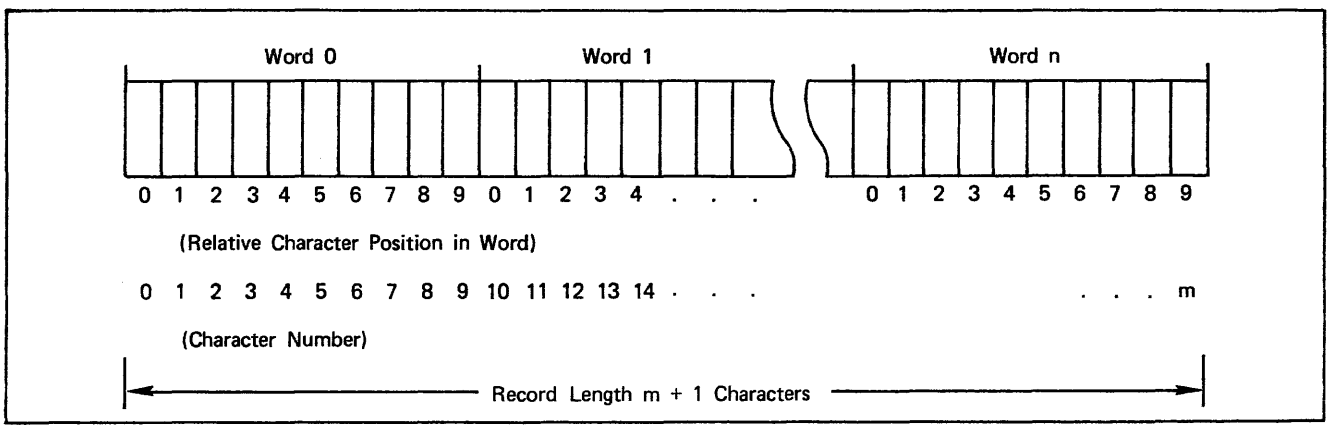


Figure 2-11. Numbering Conventions

set to 22). The minimum number of characters in a record is 25 (the sum of the values in the LL and LP fields).

FIXED LENGTH, F TYPE RECORDS

In a file with F type records, all records are the same length. The number of characters in the F type records is specified by the fixed length (FL) field in the FIT. The maximum record length that can be specified for F type records is $10(2^{13}-5)$ characters for extended indexed sequential files and $10(2^{17}-1)$ characters for all other files; minimum record length is 10 characters. An example of an F type record is shown in figure 2-13; each record in the file contains 200 characters as specified by the FL field.

TABLE 2-1. RECORD TYPES AND LENGTH DESCRIPTIONS

Record Type	Length Description
D - Decimal Character Count	A length field within the record gives the length as character count.
F - Fixed Length	All records are the same fixed length.
R - Record Mark	A record mark character specified by the user terminates the record.
S - System Record	The length is defined by the user.
T - Trailer Count	The fixed-length header contains a trailer count field that specifies the number of fixed-length trailers for the record.
U - Undefined	The length is defined by the user.
W - Control Word	The length is defined by the user.
Z - Zero Byte	The length is determined using the RL or FL field and removing all full words of blanks.

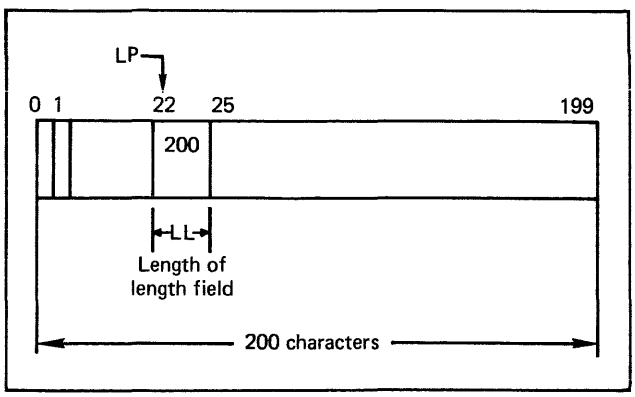


Figure 2-12. D Type Record Example

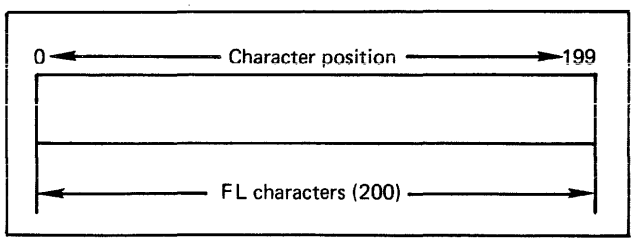


Figure 2-13. F Type Record Example

Any value in the record length (RL) field in the FIT is ignored. When a GET or PUT macro is issued, the value of the fixed length (FL) field in the FIT determines the number of characters that are transferred. A value must be supplied for the FL field before the file can be successfully opened.

RECORD MARK, R TYPE RECORDS

A special delimiting character, called a record mark, terminates R type records. The record mark character, which can be any character of the character set, is selected by the user. The delimiting character is specified in the record mark character (RMK) field in the FIT.

The size of an R type record cannot exceed the number of characters specified by the value of the maximum record length (MRL) field in the FIT. Maximum length that can be specified for R type records is $10(2^{13}-5)$ characters for extended indexed sequential files and $2^{17}-1$ characters for all other files.

When a GET macro is issued, all characters up to and including the record mark character are transferred to the working storage area. If the record mark character is not found within the specified maximum record length, the maximum number of characters is transferred and an excess data error is given.

Issuing a PUT macro causes all characters up to and including the record mark character to be written on the file. If the record mark character is not found within the specified maximum record length, no data is written on the file and an excess data error is given.

Figure 2-14 illustrates the use of R type records. The maximum record length (MRL) field is set to 120 and the record mark character (RMK) field is set to 62_8, which is the default right bracket (]) character. For a file read or write operation, the right bracket character terminates the record.

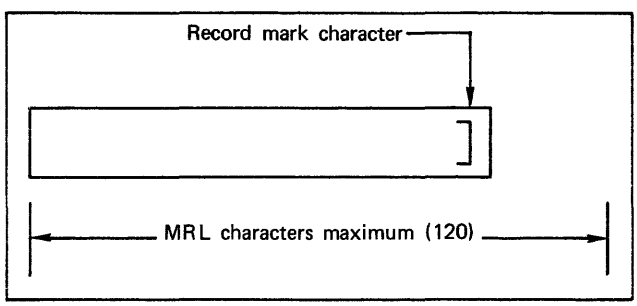


Figure 2-14. R Type Record Example

SYSTEM RECORD, S TYPE RECORDS

When S type records are specified, AAM processes the records as U type records. Refer to the description of U type records.

TRAILER COUNT, T TYPE RECORDS

Records in a file with T type records consist of a fixed-length header and a variable number of fixed-length trailer items. The fixed-length header contains a count field that specifies the number of fixed-length trailer items in the record.

Four fields in the FIT are applicable to T type records and must be specified.

- HL — Header length specifies the number of characters in the fixed-length header.
- TL — Trailer length specifies the number of characters in one fixed-length trailer item.
- CP — Starting character position specifies the character position (numbered from 0) in which the count field begins.
- CL — Count field length specifies the number of characters (one through six) in the count field.

The value in the count field is right-justified display code with zero or blank fill. The COMP-1 (C1) field or the sign overpunch (SB) field in the FIT can be set to YES to change the count field to a COMP-1 or sign-overpunch field.

The count field, which is identified by the CP and CL fields in the FIT, must be located in the fixed-length header portion of the record. The value in the header length (HL) field, therefore, cannot be less than the sum of the values in the CP and CL fields.

The value in the HL field is the logical minimum record length. The maximum length for a record is specified by the maximum record length (MRL) field in the FIT; the value in the HL field cannot exceed the value in the MRL field. Maximum length that can be specified for T type records is $10(2^{13}-5)$ characters for extended indexed sequential files and $10(2^{17}-1)$ characters for all other files. The logical structure of a T type record is shown in figure 2-15.

UNDEFINED, U TYPE RECORDS

Files with U type records have records that are not formatted according to any of the supported record types. The maximum record length (MRL) field in the FIT indicates the maximum length for any record in the file. The maximum record length that can be specified for U type records is $10(2^{13}-5)$ characters for extended indexed sequential files and $10(2^{17}-1)$ characters for all other files.

When a GET or PUT macro is executed, the record length (RL) field in the FIT must be set to indicate the number of characters to be read or written. The value in the RL field cannot exceed the specified maximum record length. AAM maintains record pointers that define the length of the stored record.

CONTROL WORD, W TYPE RECORDS

When W type records are specified, AAM processes the records as U type records. Refer to the description of U type records.

ZERO BYTE, Z TYPE RECORDS

A Z type record is terminated by a 12-bit byte of zeros in the low-order position of the last word in the record. Maximum record size is indicated by the full length (FL) field in the FIT; maximum length that can be specified for Z type records is $10(2^{13}-5)$ characters for extended indexed sequential files and $10(2^{17}-1)$ characters for all other files.

When a record is written, the value of the record length (RL) field determines the processing that takes place. If the RL field is set to a value greater than zero, the end of the record is determined by searching backward from the character position specified by the value of the RL field and removing all full words of blanks.

If the RL field is set to zero when a record is being written, the end of the record is determined by a backward search for the last nonblank character in the working storage area. The search begins in the character position indicated by the full length (FL) field in the FIT; all full words of blanks are removed.

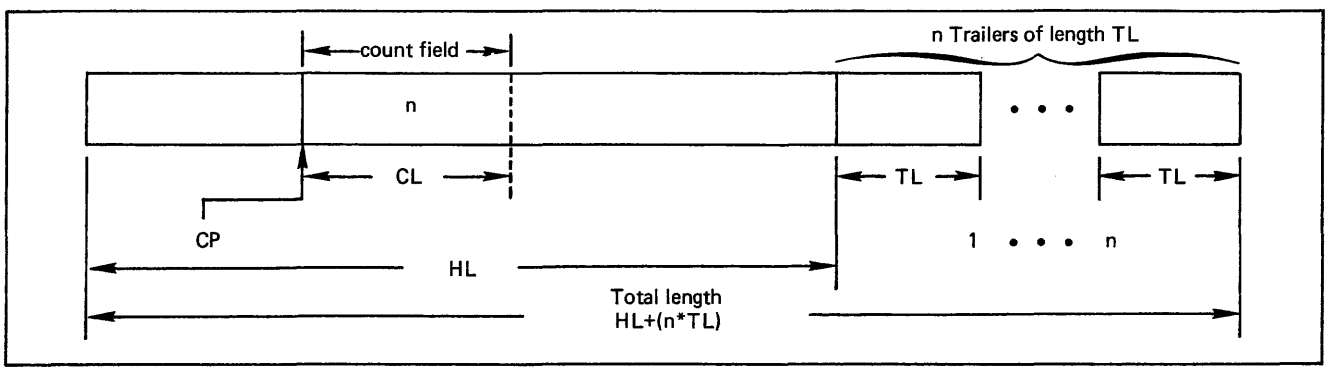


Figure 2-15. T Type Record Format

ALTERNATE KEY INDEX FILE STRUCTURE

An index file is created and maintained by the Multiple-Index Processor (MIP) whenever a data file has alternate keys defined. The index file is automatically created when the data file is created and updated whenever an update to the data file affects the index file.

AAM supports two types of MIP. Initial MIP is used for initial indexed sequential, direct access, and actual key files. Extended MIP is used for extended indexed sequential files.

INITIAL MIP

The index file created and maintained by initial MIP contains an index for each alternate key position defined for the file. Within an index, each alternate key value is associated with a primary key list of records containing that value. The index file is created when the data file is created, or the IXGEN utility can be used to create the index file for an existing data file.

The size of the index file blocks can be specified by the user when the index file is created. Index file block size must always be specified as an integral number of PRUs. A block size of 2 to 8 PRUs is recommended; results are indeterminate if the block size exceeds 8 PRUs.

Each alternate key index is ordered in ascending sequence of alternate key values. The ordering of primary key values within the primary key list associated with an alternate key value can be controlled by the user. The structure of primary key lists can be indexed sequential or first-in first-out. Indexed sequential structure is most efficient. The user can also specify that alternate key values are unique, in which case each primary key list contains only one value.

Figure 2-16 illustrates an initial MIP index file with indexes for two alternate key positions: a four-character symbolic key and a three-digit integer key. For simplicity of illustration, primary keys are one-character keys with values A through Z.

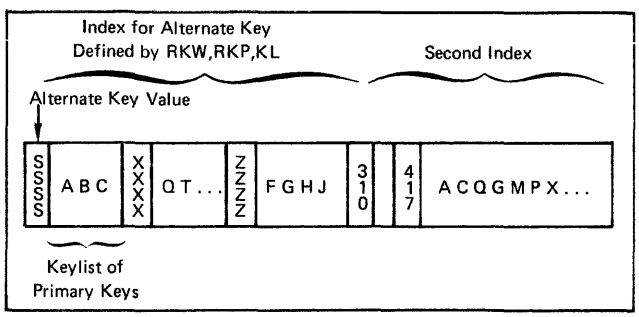


Figure 2-16. Sample Index File, Initial MIP

EXTENDED MIP

The index file created and maintained by extended MIP contains an index for each alternate key position defined for the file. Within an index, each alternate key value is associated with a primary key list of records containing that value. The index file is created when the data file is created, or the MIPGEN utility can be used to create the index file for an existing data file.

When the index file is created, the user can specify the size of the index file blocks in a field in the data file FIT. The block size is increased if necessary to the nearest multiple of 640 characters minus 20. The default size for index file blocks is the data block size.

Each alternate key index is ordered in ascending sequence of alternate key values. The ordering of primary key values within the primary key list associated with an alternate key value can be controlled by the user. The structure of primary key lists can be indexed sequential or first-in first-out. Indexed sequential structure is most efficient. The user can also specify that alternate key values are unique, in which case each primary key list contains only one value.

The index file is structured into three levels: a level 1 main file, level 2 subfiles, and level 3 subfiles. The level 1 main file contains descriptions of the alternate keys. A level 2 subfile contains values for an alternate key and a level 3 subfile contains primary key values for a specific alternate key value. The logical structure of an extended MIP index file is shown in figure 2-17.

The level 1 main file contains descriptions of all the alternate keys defined for the data file. The description of an alternate key includes the position, length, and type of the key as well as information related to sparse keys. Normally, all the descriptions can be contained in one block; however, if more than one block is required, the main file has an indexed sequential structure.

Each level 2 subfile contains all the values for one of the alternate keys. The level 2 subfiles have indexed sequential file organization with index blocks and data blocks. Each record in a data block contains an alternate key value and the first primary key value associated with it. Depending on the amount of available space in the data block and the size of the primary key list, the data block might contain additional primary key values.

A level 3 subfile contains primary key values that cannot be accommodated in the level 2 subfile. If alternate key values are unique, level 3 subfiles are not needed. The structure of the level 3 subfile is either indexed sequential or first-in first-out as specified when the alternate key is defined. Indexed sequential subfiles have index blocks and data blocks. First-in first-out level 3 subfiles have data blocks chained in a forward direction.

The physical structure of an extended MIP index file is shown in figure 2-18. A block in the figure can be either an index block or a data block. The block structure is identical to block structure in an indexed sequential file. All blocks within a subfile are chained together in a forward direction.

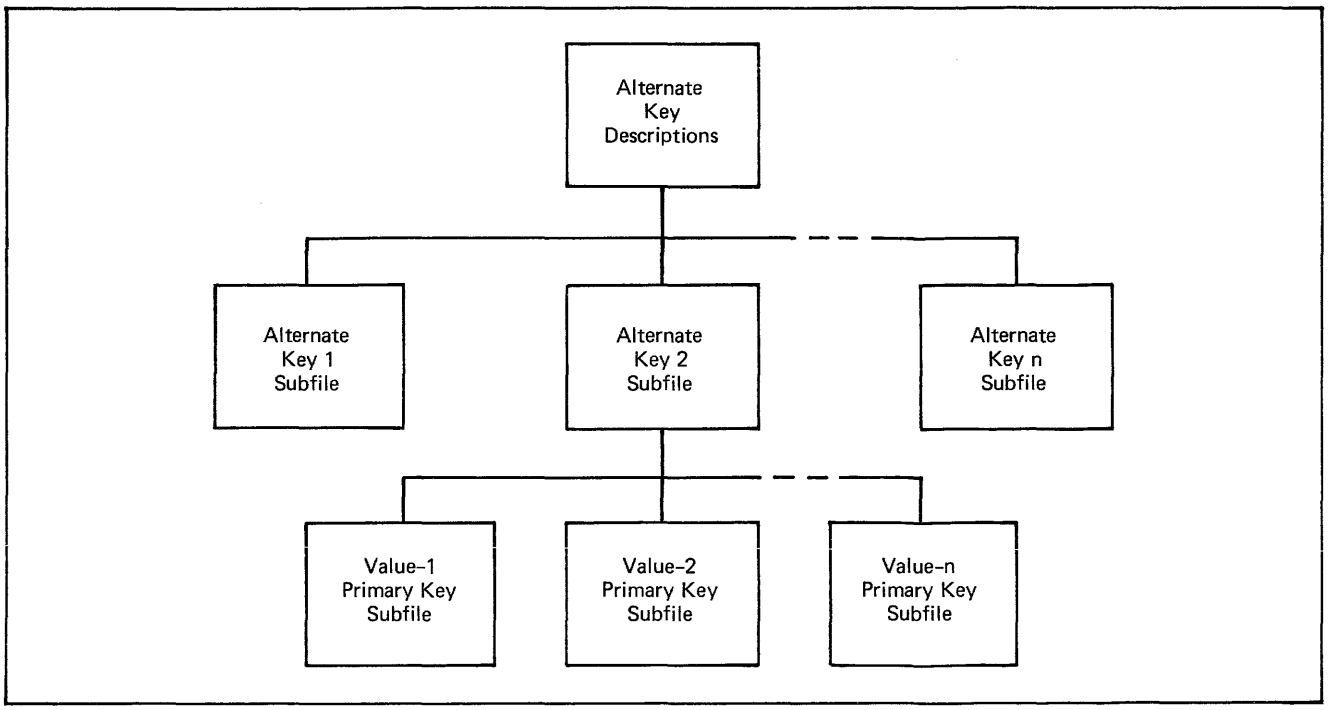


Figure 2-17. Index File Logical Structure, Extended MIP

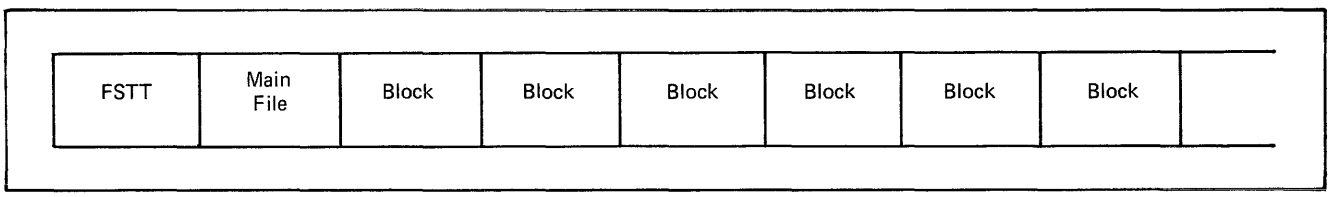


Figure 2-18. Index File Physical Structure, Extended MIP

A file information table (FIT) is required for all AAM files. Information in the table defines the file and specifies how it is accessed. The FILE macro and the FILE control statement are used to create and update the FIT. The FILE macro assembles the FIT in the COMPASS program at the address where the macro is encountered. Pertinent information from the FILE control statement is saved until the file is opened; the saved information is then stored in the FIT and takes precedence over any corresponding preexisting information. A blank FIT, except for addressing information, file organization, and logical file name, could be set up in the user program with definition of file characteristics deferred until the file is opened.

The STORE macro or the FILE control statement can be used to change the setting of fields in the FIT. The fields are identified by the keywords of the FILE macro. The FETCH macro is used to retrieve the contents of a field in the FIT; a FILE macro keyword identifies the field being retrieved.

AAM macros that request file operations can result in amendment of FIT fields. Certain macro operands are stored in FIT fields before the request is performed and values can be stored in FIT fields as a result of processing the request. AAM also maintains certain fields in the FIT to reflect the current state of the file.

FILE MACRO

The FILE macro constructs the file information table at the address where the macro is encountered during assembly; the FIT must be built before the file is opened. The format of the FILE macro is shown in figure 3-1. The interaction between lfn and LFN=axxxxxx is shown in table 3-1.

The FILE macro does not check fields for validity or consistency. If the option specified for a field exceeds the maximum specified size, it is truncated and an assembler warning message is produced.

[lfn] FILE [LFN=axxxxxx] [,keyword=option,] ...

lfn	Symbolic address where the FIT is assembled in the COMPASS program; if the LFN=axxxxxx is omitted or is the same name, logical file name by which the file can be referenced.
LFN	FIT field mnemonic for logical file name; if lfn is omitted, LFN must be specified with axxxxxx.
axxxxxx	Logical file name by which the file can be referenced; if lfn is omitted, symbolic address where the FIT is assembled in the COMPASS program.
keyword	Symbolic name of the FIT field.
option	Selected option of the FIT field.

Figure 3-1. FILE Macro Format

Misspelled or unrecognized parameters generate null parameters; the referenced fields are set to zero. Null parameters are ignored. Warning messages are generated when overlapping fields are specified.

The FILE macro must specify the file organization (FO) mnemonic for an AAM file. Any parameter not applicable to the specified file organization is ignored and an error type 4 is generated during assembly.

The values specified for the FILE macro parameters are assembled into the FIT; parameters can be specified in any order. Table 3-2 shows the FILE macro parameters applicable to each AAM file organization. A detailed explanation of each FIT field that can be specified by the FILE macro parameters follows. The default value is indicated for each field.

BCK Block checksum

BCK=NO (default)

Checksums are not computed during file creation. For a file created with checksums, no checksumming is done during a read operation; however, checksums are computed for blocks written.

BCK=YES

A checksum is computed before each block is written and after it is read. The checksum is part of the block.

BFS Buffer size

BFS=0 (default)

AAM provides the buffer space; the amount of common buffer space is increased by an amount determined by AAM.

BFS=aexp

The buffer size is the number of words specified; maximum is $2^{17}-1$ or 131000 words. If the FWB field is set to zero, AAM increases the amount of common buffer space allocated by BFS.

TABLE 3-1. LFN AND lfn INTERACTION

Statement	COMPASS Location Value	Contents of LFN Field in FIT
A FILE	A	A
FILE LFN=A	A	A
A FILE LFN=A	A	A
A FILE LFN=B	A	B

CDT Collating sequence to display code conversion table; ignored if the DCT field is zero (initial indexed sequential files)

TABLE 3-2. FILE MACRO PARAMETERS BY FILE ORGANIZATION

Parameter	Initial Indexed Sequential	Extended Indexed Sequential	Actual Key	Direct Access
BCK	X	X	X	X
BFS	X	X	X	X
CDT	X			
CL	X	X	X	X
CP	X	X	X	X
CPA		X		
C1	X	X	X	X
DCA		X		
DCT	X	X		
DFC	X	X	X	X
DP	X	X	X	
DX	X	X	X	X
EFC	X	X	X	X
EMK		X		
ERL	X	X	X	X
EX	X	X	X	X
FL	X	X	X	X
FLM	X	X	X	X
FO	X	X	X	X
FWB	X	X	X	X
FWI	X	X	X	X
HB			X	
HL	X	X	X	X
HMB				X
HRL				X
IBL	X			
IP	X	X		
KA	X	X	X	X
KL	X	X	X	X
KP	X	X		X
KT	X	X		
LFN	X	X	X	X
LL	X	X	X	X
LP	X	X	X	X
MBL	X	X	X	X
MKL	X	X		
MNR	X	X	X	X
MRL	X	X	X	X
NDX	X	X	X	X
NL	X	X		
OF	X	X		
ON	X	X	X	X
ORG	X	X		
OVF				X
PD	X	X	X	X
PKA		X		
RB	X	X	X	X
REL	X	X	X	X
RKP		X		X
RKW		X		X
RMK	X	X	X	X
RT	X	X	X	X
SB	X	X	X	X
TL	X	X	X	X
TRC	X			X
WSA	X	X	X	X
XBS		X		
XN	X	X	X	X

CDT=0 (default)

Conversion table is generated from the table specified by the DCT field.

CDT=exp

Conversion table is at the specified address.

CL Trailer count field length (T type records)

CL=0 (default)

For T type records, this field must be defined before the file is opened.

CL=aexp

The length of the trailer count field is the specified number of characters; maximum is 6.

CP Trailer count beginning character position (T type records)

CP=0 (default)

The trailer count field begins in character position 0.

CP=aexp

The specified number is the beginning character position, numbered from 0 on the left; maximum is $10(2^{15}-5)$ for extended indexed sequential files and $10(2^{17}-1)$ for all other files.

CPA Compression/encryption routine number or address (extended indexed sequential files)

CPA=0 (default)

Records are not compressed unless a system routine was specified when the file was previously opened.

CPA=aexp

The specified number identifies the system compression routine to be used; must be less than 100_8.

CPA=exp

The user-supplied compression routine is at the specified address; must not be less than 100_8.

C1 COMP-1 format for length field (D or T type records)

C1=NO (default)

The length field is in coded format.

C1=YES

The length field is in binary (COBOL COMP-1) format.

DCA Decompression/decryption routine address; required if the CPA field specifies a user routine (extended indexed sequential files)

DCA=0 (default)

If the CPA field specifies either no compression or a system compression routine, AAM sets DCA at open time if needed.

DCA=exp

The user-supplied decompression routine is at the specified address; DCA must be specified if a user-supplied compression routine is specified.

DCT Display code to collating sequence conversion table (indexed sequential files)

DCT=0 (default)

CDC conversion tables are used.

DCT=exp

The user-supplied table is at the specified address. For initial indexed sequential files when the CDT field is 0 and for extended indexed sequential files, AAM generates the collating sequence to display code conversion table from the user-supplied table.

DFC Dayfile control

DFC=0 (default)

Only fatal error messages are written on the dayfile.

DFC=1

Error messages are written on the dayfile.

DFC=2

Statistics/notes are written on the dayfile.

DFC=3

Error messages and statistics/notes are written on the dayfile.

DP Data block padding factor (indexed sequential and actual key files)

DP=0 (default)

The installation default value is used for indexed sequential files; 0 for actual key files.

DP=aexp

Padding for the data block is the specified percentage; maximum is 99.

DX End-of-data exit

DX=0 (default)

No end-of-data exit is specified.

DX=exp

The routine at the specified address is entered when an end-of-data condition occurs. The system stores a jump at the first address of the routine and control passes to the first executable statement, which is routine+1.

EFC Error file control

EFC=0 (default)

No messages are written on the error file.

EFC=1

Error messages are written on the error file.

EFC=2

Statistics/notes are written on the error file.

EFC=3

Error messages and statistics/notes are written on the error file.

EMK Embedded key; examined when the file is opened for creation (extended indexed sequential files)

EMK=NO (default)

The key is not part of the record.

EMK=YES

The key is embedded in the record; the RKW, RKP, and KL fields define the position and length of the key.

ERL Trivial error limit

ERL=0 (default)

An indefinite number of trivial errors is permitted; no limit is specified.

ERL=aexp

The specified number is the maximum number of trivial errors allowed before a fatal error occurs; maximum is 511.

EX Error exit

EX=0 (default)

No routine is entered if an error occurs; control is returned to the user's in-line code.

EX=exp

The routine at the specified address is entered when an error occurs. The system stores a jump at the first address of the routine and control passes to the first executable statement, which is routine+1.

FL Fixed length (F type records) or full length (Z type records)

FL=0 (default)

The field must be defined before the file is opened.

FL=aexp

For F type records, the specified number is the record length in characters; 10 through $10(2^{13}-5)$ for extended indexed sequential files and 10 through $10(2^{17}-1)$ for all other files.

For Z type records, the specified number establishes the upper limit of characters or blank padding moved to the working storage area.

FLM File limit

FLM=0 (default)

The file limit is not checked.

FLM=aexp

The file limit cannot exceed the specified number of records.

FO File organization (no default value)

FO=AK

The file has actual key file organization.

FO=IS

The file has indexed sequential file organization.

FO=DA

The file has direct access file organization.

FWB First word address of user-supplied buffer

FWB=0 (default)

AAM provides the buffer space needed.

FWB=exp

The user buffer is at the specified address.

FWI Forced write indicator

FWI=NO (default)

Each buffer is written only when the buffer space is needed for another input/output operation.

FWI=YES

All buffers are written immediately after each operation that modifies the buffer content. This option increases file integrity by keeping the file current; however, performance is degraded as more input/output transfers are required.

HB Header indicator bit (actual key files)

HB=NO (default)

The user header is not returned with the data received.

HB=YES

The user header is returned with the data received.

HL Header length (T type records)

HL=0 (default)

For T type records, this field must be defined before the file is opened.

HL=aexp

The fixed-length portion of the T type records is the specified number of characters; maximum is $10(2^{13}-5)$ for extended indexed sequential files and $10(2^{17}-1)$ for all other files; minimum is the sum of the CP and CL fields.

HMB Number of home blocks (direct access files)

HMB=0 (default)

The field must be defined to open the file.

HMB=aexp

The file contains the specified number of home blocks; maximum is $2^{24}-1$.

HRL Hashing routine location; cannot be changed after file creation (direct access files)

HRL=0 (default)

The system-supplied hashing routine is used.

HRL=exp

The user-supplied hashing routine is at the specified address.

IBL Index block length (initial indexed sequential files)

IBL=0 (default)

The index block size is calculated using the values of the NL, RB, IP, FLM, and KL fields. If either or both of the values of the NL and FLM fields are zero when the file is opened, the default index block size is used.

IBL=aexp

The index block size is the specified number of characters rounded up to an integral multiple of the PRU size minus 10 characters; maximum is $10(2^{17}-1)$.

IP Index block padding factor (indexed sequential files)

IP=0 (default)

For initial indexed sequential files, the installation default value is used (release default is five); for extended indexed sequential files, the default value zero is used.

IP=aexp

The index block padding is the specified percentage; maximum is 99.

KA Key address

KA=0 (default)

No address is specified for a key.

KA=exp

The key value for the record to be processed is at the specified address; for the GETN macro, the key of reference is returned to the specified address.

KL Key length

KL=0 (default)

This field must be defined before the file is opened.

KL=aexp

For actual key files, the key length is the specified number of bits; 2 through 47. For indexed sequential and direct access files, the key length is the specified number of characters. The positive integer that can be

specified for indexed sequential files depends on the key type defined by the KT field. For initial indexed sequential files:

KT=S Symbolic key, maximum is the installation-defined key limit (default is 255).

KT=I Integer key, either 5 or 10 characters must be specified; a 5-character integer key is formed from the lower half of the word.

KT=F Floating point key, the KL field is ignored without comment; key size is always 10 characters.

For extended indexed sequential files:

KT=S or KT=U Symbolic key, maximum is the installation-defined key limit (default is 255).

KT=I Integer key, 10 characters must be specified for the signed binary key.

KP Beginning key position (direct address files and symbolic keys for indexed sequential files)

KP=0 (default)

The key is word-aligned.

KP=aexp

The key begins in the specified character position within the KA field, numbered from 0 on the left; maximum is 9.

KT Key type (indexed sequential files)

For initial indexed sequential files:

KT=S (default)

A symbolic key is a string of alphanumeric characters.

KT=I

An integer key is either 5 or 10 characters in length in either fixed or unnormalized floating point format.

KT=F

A floating point key is 10 characters in length.

For extended indexed sequential files:

KT=S (default)

A collated symbolic key is a string of alphanumeric characters.

KT=I

An integer is a 10-character signed binary key.

KT=U

An uncollated symbolic key is a string of alphanumeric characters.

LFN Logical file name (no default value)

LFN=axxxxxx

The data file logical file name is one to seven characters in length beginning with a letter.

LL Length field length (D type records)

LL=0 (default)

The field must be defined before the file is opened.

LL=aexp

The length of the length field is the specified number of characters; maximum is 6.

LP Length field beginning character position (D type records)

LP=0 (default)

The length field begins in character position 0.

LP=aexp

The length field begins in the specified character position, numbered from 0 on the left; maximum is $10(2^{13}-5)$ for extended indexed sequential files and $10(2^{17}-1)$ for all other files.

MBL Maximum block length; should not be changed after the file is opened

MBL=0 (default)

The installation default size is used.

MBL=aexp

The data block is the specified number of characters in length. The specified size is increased to an integral multiple of PRU size minus two words. MBL should not be specified if a value for the RB field is given for indexed sequential files. If both are set, the value of the RB field is ignored. For extended indexed sequential files, MBL also specifies the length of the index blocks.

MKL Major key length (indexed sequential files, symbolic key type)

MKL=0 (default)

Major key length processing is not specified.

MKL=aexp

The major key length is the specified number of characters; maximum is the KL value. The file is positioned at the first record with a key in which the first specified number of characters matches the major key.

MNR Minimum record length

MNR=0 (default)

The minimum record length is zero characters. Zero length records are not accepted in direct access and actual key files.

MNR=aexp

The minimum record length is the specified number of characters; maximum is the MRL value.

MRL Maximum record length

MRL=0 (default)

This field must be defined before the file is opened for creation.

MRL=aexp

The maximum record length is the specified number of characters; maximum is $10(2^{13}-5)$ for extended indexed sequential files and $10(2^{17}-1)$ for all other files. This establishes the upper limit of characters moved to the working storage area. The field must be specified for OPENM NEW and is returned for OPENM OLD.

NDX Index flag (multiple-index files)

NDX=0 (default)

The data file can be accessed by primary or alternate key.

NDX=1

Only the index file is accessed.

NL Number of index block levels; required only when files are created (indexed sequential files)

NL=0 (default)

The installation default value is used.

NL=aexp

The number specified is the expected number of levels for the file; maximum is 63 for initial indexed sequential files and 15 for extended indexed sequential files. For initial indexed sequential files, the specified value is used to calculate index block size if the value of the IBL field is zero at creation time, and it is used in the allocation of buffer space.

OF Open flag; file positioning at OPENM time (indexed sequential files)

OF=R (default)

The file is rewound.

OF=N (initial indexed sequential files)

The file is not rewound.

OF=E

The file is positioned at end-of-information for extend.

ON Old or new file

ON=OLD (default)

The file is an existing file (FSTT exists).

ON=NEW

The file is being created (FSTT to be established).

ORG Old/new file organization (indexed sequential files)

ORG=OLD (default)

The file organization is initial indexed sequential.

ORG=NEW

The file organization is extended indexed sequential.

OVF Overflow flag (direct access files)

OVF=OVB (default)

Overflow records are stored in home and overflow blocks.

OVF=OVO

Overflow records are stored in overflow blocks only.

OVF=OVH

Overflow records are stored in home blocks only.

PD Processing direction

PD=INPUT (default)

The file is open for input (read).

PD=OUTPUT

The file is open for output (write).

PD=IO

The file is open for input/output (read and write)

PD=NEW

The file is open for a creation run.

PKA Primary key address (extended indexed sequential files)

PKA=0 (default)

The primary key is not returned on an alternate key read operation.

PKA=exp

The primary key is returned to the specified address on an alternate key read operation.

RB Records per block; used in block size calculation

RB=0 (default)

RB is set to 1; the installation default is used if MBL is also zero.

RB=aexp

Blocking factor limit is $2^{12}-1$. For indexed sequential files, RB should not be specified if the MBL field is specified.

REL Key relation; relation of record key to key value at location KA.

REL=1

This specifies an equal (EQ) relation.

REL=2 (not applicable to extended indexed sequential files)

This specifies a less than or equal (LE) relation.

REL=3

This specifies a greater than or equal (GE) relation.

REL=4 (not applicable)

REL=5 (not applicable to extended indexed sequential files)

This specifies a less than (LT) relation.

REL=6

This specifies a greater than (GT) relation.

RKP Relative key position; required if EMK is set to YES (extended indexed sequential, direct access, and multiple-index files)

RKP=0 (default)

The key is word-aligned starting at RKW position.

RKP=aexp

The key begins in the specified position within RKW, numbered from 0 on the left; maximum is 9.

RKW Relative key word; required if EMK is set to YES (extended indexed sequential, direct access, and multiple-index files)

RKW=0 (default)

The key begins in the first word of the record.

RKW=aexp

The key starts in the specified word (numbered from 0) within the record.

RMK Record mark character (R type records)

RMK=0 (default)

The record mark character is the right bracket (]), which is 62_8.

RMK=ccB

The record mark character is the specified octal value (cc); maximum is 77_8.

RMK=1Rx

The record mark character is the specified character (x); any character is valid.

RMK=cc

The record mark character is the specified decimal value (cc); maximum is 63.

RT Record type

RT=W (default)

This specifies a control word record; however, AAM considers this the same as RT=U.

RT=F

This specifies a fixed length record.

RT=R

This specifies a record mark record.

RT=Z

This specifies a zero byte terminated record.

RT=D

This specifies a decimal character count record.

RT=T

This specifies a trailer count record.

RT=U

This specifies an undefined record.

RT=S

This specifies a system-logical-record; however, AAM considers this the same as RT=U.

SB Sign overpunch format for length field (D or T type records)

SB=NO (default)

The length field is in unsigned display code.

SB=YES

The length field uses a COBOL sign overpunch scheme.

TL Length of the trailer portion (T type records)

TL=0 (default)

This field must be defined before the file is opened.

TL=aexp

The length of the trailer portion is the specified number of characters; maximum is $2^{17}-1$.

TRC Trace transaction count (initial indexed sequential and direct access files)

TRC=0 (default)

No tracing is to be performed.

TRC=aexp

If 1 through 31 is specified, the last specified number of transactions are traced prior to termination; if 32 through 63 is specified, all file transactions are traced.

WSA Working storage area address

WSA=0 (default)

No working storage area is specified.

WSA=exp

The working storage area is at the specified address. This field must be set before any file processing macro uses the working storage area. It can be set by the GET, PUT, GETN, GETNR, and REPLACE macros.

XBS Index file block size (multiple-index files, extended MIP)

XBS=0 (default)

The index file blocks are the same size as the data file blocks.

XBS=aexp

The index file blocks are the specified number of characters.

XN Index file name (multiple-index files)

XN=0 (default)

No accesses or updates by alternate key can be performed.

XN=lfn

The index file for alternate key access is the file with the specified logical file name.

FILE CONTROL STATEMENT

The FILE control statement is used to specify file information to update the FIT either when the SETFIT macro is issued or the first time the file is opened in the job step. This run-time control over file specification allows a single program to process files with different record types. Corresponding FIT fields have the value specified on the last control statement encountered.

FILE control statements must be placed before any program call in which the information in the statements is to be used. Because processing of the FILE control statement involves calling a central processor program, it should not be placed within a load set sequence. For example, the FILE control statement should not be placed between the LOAD and EXECUTE control statements.

If more than one FILE control statement appears for a given file, the data on the first control statement can be overwritten by the data on a subsequent control statement when overlapping fields occur in those statements. The FILE control statement conforms to operating system coding conventions.

When an error diagnostic is produced by FILE control statement processing, the entire statement is ignored. FILE control statement diagnostics are written on the dayfile as soon as the error is encountered; diagnostics name the faulty parameter and are self-explanatory. Control is then passed to the next EXIT control statement.

The format of the FILE control statement is shown in figure 3-2. Keywords can be specified in any order. Keywords have the same meanings as described for the FILE macro.

FILE(lfn[=axxxxxx] [,keyword=option] ...)

lfn	Name of the FIT; required.
=axxxxxx	Optional new name for the FIT; allows a file to be requested by a new name without reassembly.
keyword=option	Symbolic name of the FIT field and the option selected.

Figure 3-2. FILE Control Statement Format

If only the lfn and FO parameters appear in the FILE control statement and no subsequent FILE control statement references that file, FIT fields for all succeeding job steps are those specified in the program. If the FILE control statement appears without any parameters, FIT fields for all files revert back to those specified in the program for all succeeding job steps until another FILE control statement is encountered. Except for the USE and OMIT parameters, all parameters valid in a FILE control statement are valid in a FILE macro.

The FILE control statement parameters are listed in table 3-3. The various options for a keyword are separated by the | symbol. If the keyword is selected, one of the options must be selected and the others must be omitted. Parameter values are absolute and generally reference a number of characters. Value formats are denoted as:

n ... n Decimal value

n ... nB Octal value

n ... nW Decimal value, specified in words

Descriptions of the FILE control statement parameters are the same as for the corresponding FILE macro parameters.

RUN-TIME MANIPULATION

The user can communicate with AAM through the FIT without knowing the exact format of the FIT. This is done with the FETCH, STORE, and SETFIT macros; FIT field mnemonics are used in the FETCH and STORE macros.

FETCH MACRO

The FETCH macro retrieves the contents of a specified FIT field by a reference to its mnemonics. The format of the FETCH macro is shown in figure 3-3.

If the specified keyword represents a 1-bit field, it is returned in the sign bit of the X register; the contents of the remainder of the X register are undefined. File names are returned left-justified with zero fill. All other fields are returned right-justified with zero fill.

FIT field mnemonics can be any of the keywords used with the FILE macro or any of the fields listed in figure 3-3. The macro generates code to extract the requested value from the FIT. The code expansion destroys values in user registers Xf, Xm, Af, and Xi (which can be Xf or Xm).

STORE MACRO

The STORE macro places a user-determined value in a specified FIT field at execution time. The format of the STORE macro is shown in figure 3-4.

Most FIT fields listed in appendix D can be set symbolically by the STORE macro. Some fields, such as the file structure parameters, are protected against being changed by the STORE macro. Other fields are not protected but should not be changed after the file has been opened.

A field can be set by using the option with the keyword or by using a register to hold the option as shown in figure 3-5. Examples a and b have the same effect.

TABLE 3-3. FILE CONTROL STATEMENT PARAMETERS

Keyword	Options	Keyword	Options	Keyword	Options
BCK	NO|YES	HL	0|n . . . n|n . . . nB|n . . . nW	OF	R|N|E
BFS	0|n . . . n|n . . . nB	IP	0|nn	OMIT	macro name/macro name/. . .
CF	R|N|U|RET|DET	KL	0|n . . . n|n . . . nB|n . . . nW	ON	OLD|NEW
CL	0|n . . . n|n . . . nB|n . . . nW	KP	0|n . . . n|n . . . nB	ORG	OLD|NEW
CP	0|n . . . n|n . . . nB|n . . . nW	KT	S|I|F|U	PD	INPUT|OUTPUT|IO
C1	NO|YES	LFN	lfn	RB	0|n . . . n|n . . . nB
DFC	0|1|2|3	LL	0|n . . . n|n . . . nB	RKP	0|n . . . n|n . . . nB
DP	0|nn	LP	0|n . . . n|n . . . nB|n . . . nW	RKW	0|n . . . n|n . . . nB
EFC	0|1|2|3	MBL	0|n . . . n|n . . . nB|n . . . nW	RT	W|F|R|Z|D|T|U|S
ERL	0|n . . . n|n . . . nB	MKL	0|n . . . n|n . . . nB|n . . . nW	SB	NO|YES
FL	0|n . . . n|n . . . nB|n . . . nW	MNR	0|n . . . n|n . . . nB|n . . . nW	TL	0|n . . . n|n . . . nB|n . . . nW
FLM	0|n . . . n	MRL	0|n . . . n|n . . . nB|n . . . nW	TRC	0|n . . . n|n . . . nB
FO	IS|DA|AK	NDX	NO|YES	USE	macro name/macro name/. . .
FWI	NO|YES	NL	0|n . . . n|n . . . nB	XN	lfn

FETCH fit,keyword,xi,f,m

fit — Logical file name address of the FIT, or any COMPASS expression giving the FIT address.

keyword — Any of the keywords in the FILE macro, FILE control statement, or any of the following:

BN	Block number
ECT	Trivial error count
ES	Error status
FNF	Fatal/nonfatal flag
FP	File position
LOP	Last operation code
OC	Open/close flag
RC	Record count
RL	Record length
WPN	Write bit

Xi — X register to receive the value of the requested field.

f — Number of the X register used to fetch the FIT word; must be 1 through 5 (default is 5).

m — Number of the X register used as a mask (default is 7).

Figure 3-3. FETCH Macro Format

STORE fit,keyword= { value / option / Ri } ,f,s,m

fit — Address of the FIT or any COMPASS expression giving the address.

keyword — Any keyword described in connection with the FILE macro except OF or RT.

value — integer value associated with the keyword; when the keyword represents a length, it is specified in characters.

option — Option associated with the keyword.

Ri — Any register containing the proper value for the keyword.

f — Number of the X register used to fetch the FIT word; must be 1 through 5 (default is 5).

s — Number of the X register used to store the FIT word; must be 6 or 7 (default is 6).

m — Number of the X register used as a mask (default is 7).

Figure 3-4. STORE Macro Format

```
a.  STORE   fit,RL=10
b.  SX1     10
    STORE   fit,RL=X1
c.  STORE   fit,FO=IS
```

Figure 3-5. STORE Macro Examples

The STORE macro generates code to store the requested value in the FIT. This code expansion destroys the values in user registers Xf, Xs, Xm, Af, As, and Xi (which can be Xf, Xs, or Xm).

SETFIT MACRO

The SETFIT macro sets fields in the FIT according to information provided in the FILE control statement. This normally occurs when the OPENM macro is executed. The SETFIT macro makes it possible for system routines to obtain information, such as run-time buffer requirements, needed by other system routines. The format of the SETFIT macro is shown in figure 3-6.

```
SETFIT  fit

fit     Address of the FIT or register containing the
        address of the FIT.
```

Figure 3-6. SETFIT Macro Format

The SETFIT macro is valid only for a closed file. Any attempt to execute this macro for an open file results in an error. Once the FILE control statement values are placed in the FIT, the macro sets the processed flag (PDF) field to inhibit further FILE control statement processing when the OPENM macro is executed. The flag is cleared during subsequent OPENM processing.

If the buffer size (BFS) field is zero for an existing file, the parameters from the file statistics table are placed in the FIT; the buffer size returned to the BFS field is based on these values. After a buffer is calculated, the open/close (OC) field and first word address of the buffer (FWB) field are cleared.

For a new file, the SETFIT macro should not be issued unless sufficient information exists for buffer calculations. Parameters needed for buffer calculation are shown in table 3-4.

TABLE 3-4. BUFFER CALCULATION PARAMETERS

File Organization	User Must Supply	Parameter	User Can Supply or Default is Used	Parameter
Extended indexed sequential	Key length	KL	Maximum block length	MBL
	Key type	KT	Index block padding factor	IP
	Maximum record length	MRL	Data block padding factor	DP
			Index block specification	NL
			Embedded key	EMK
			Compression routine	CPA
Initial indexed sequential	Key length	KL	Maximum block length	MBL
	Key type	KT	Index block length	IBL
	Maximum record length	MRL	Index block padding factor	IP
			Data block padding factor	DP
			Index block specification	NL
Direct access	Home block number	HMB	Blocking factor	RB
	Key length	KL		
	Maximum block length	MBL or		
	Maximum record length	MRL and		
	Minimum record length	MNR		
Actual key	Maximum block length	MBL or	Blocking factor	RB
	Maximum record length	MRL and		
	Minimum record length	MNR		

This section provides general processing information and explains by file organization the logical operations of processing AAM files. Macros and FIT fields are discussed as applicable to the type of processing for each file organization. The macros and their parameters are described in general in section 5, File Processing Macros. Detailed explanations of the FIT fields are in section 3, File Information Table. Processing of multiple-index files is discussed in section 6, Multiple-Index Files.

GENERAL PROCESSING INFORMATION

Certain processing procedures are common to all AAM file organizations. These procedures are explained in the following paragraphs. Processing unique to each file organization is discussed by file organization.

FILE INFORMATION TABLE

Before an AAM file can be processed, the file information table (FIT) must be established. This provides the name by which the file can be referenced and defines the file structure and processing limitations. The FIT contains fields that are referenced whenever AAM processes the file. FIT fields can be set before file processing by the FILE control statement, FILE macro, SETFIT macro, or STORE macro.

FILE STATISTICS TABLE

A separate creation run is necessary for AAM files. This creation run establishes the file statistics table (FSTT), which becomes a permanent part of the file. The FSTT contains FIT fields that cannot be changed for the life of the file. When the file is opened for processing after its creation run, the FIT fields are automatically established from information in the FSTT of the file.

OPENM MACRO

All files must be initialized using the OPENM macro. Applicable default values are inserted into FIT fields for certain values not supplied before executing the OPENM macro. AAM also performs certain consistency checks on FIT fields when the file is opened. Refer to the OPENM macro description in section 5 for the FIT fields that are checked.

INPUT/OUTPUT MACROS

The GET, GETN, and GETNR macros read records from a file. A working storage area must be established to pass data to the program from a file storage device. The user defines the working storage area (WSA) by supplying an address for the WSA field in the FIT. A GET macro transfers data from the buffer area, which is set up either by the user or by AAM when the file is opened, to the working storage area.

The PUT macro is used to write records to the file. A working storage area must be established to pass data from the program to a file storage device. The PUT macro transfers data from the working storage area to the buffer area, which is set up either by the user or by AAM when the file is opened. The maximum record length (MRL) field in the FIT must be set by the user on a file creation run and becomes a permanent part of the file. The value specified in the MRL field becomes the upper limit on the number of characters that can be transferred.

CLOSEM MACRO

At completion of processing, all files must be closed by the CLOSEM macro. Any remaining records of an output file are written from the buffer to the file storage device, the open/close (OC) field in the FIT is set to closed, and control is returned to the user. Execution of the CLOSEM macro causes the FSTT to be updated; if requested, file statistics are written to the error file ZZZZZEG.

END-OF-DATA ROUTINE

The end-of-data exit (DX) field in the FIT specifies the address of a user routine for processing an end-of-data condition. End-of-data occurs when beginning-of-information (BOI) or end-of-information (EOI) is encountered while attempting a data transfer or positioning operation.

Control is passed to the address (DX)+1; a jump back to the user in-line return code is stored at the DX address. The file position (FP) field indicates the specific end condition (BOI or EOI).

When file position is at EOI, the GETN macro transfers control to the end-of-data exit. If continued GETN macros are issued without repositioning the file, the GETN macro issues an error and transfers control to the error exit (if specified) instead of to the end-of-data exit. No GETN macro that passes control to the end-of-data exit causes data to be transferred to the working storage area. Control is passed to the end-of-data exit only when end-of-information is encountered. The FP field is not set until the file is logically at the end of information.

For indexed sequential and actual key files, control is transferred to the end-of-data exit whenever a SKIP macro encounters EOI or BOI. A trivial error condition is produced by successive SKIP macros after end-of-data has been encountered.

INITIAL INDEXED SEQUENTIAL FILES

The initial indexed sequential file organization is well suited for applications that require reasonably efficient storage and retrieval of records both randomly and sequentially by primary or alternate key. A primary key is an identifier defined by the user for each record within an initial indexed sequential file. Primary and alternate keys can be in any of the following forms:

30-bit integer (5 characters)

60-bit integer (10 characters)

60-bit floating point number (10 characters)

Symbolic (1 to 255 contiguous alphanumeric characters)

The value of the primary key determines the location of the record in the file. Characters within a symbolic (alphanumeric) key are collated according to the standard CDC collating sequence or according to a user-supplied collating sequence. Any user collating sequence has meaning for ranking keys only; it is stored with the user file in the FSTT. Numeric keys are ordered by value. Keys within an initial indexed sequential file can be a part of the record.

FILE CREATION RUN

A separate creation run is necessary for an initial indexed sequential file. This can be done through the FORM utility or through a source program. The FSTT is created when the initial indexed sequential file is created.

The efficiency with which an initial indexed sequential file can be processed is influenced by three fields in the FIT:

BFS	Buffer size
IBL	Index block length
MBL	Maximum block length

On a creation run, the user has the option of specifying these values directly or accepting system defaults calculated by AAM. The ESTMATE utility, which is described in section 7, can be used to calculate suggested values for the MBL, IBL, and BFS fields.

If the MBL field is not specified directly, the value is calculated from the values of the following fields in the FIT:

DP	Data block padding
KL	Key length
MNR	Minimum record length
MRL	Maximum record length
RB	Records per block

If the IBL field is not specified directly, the value is calculated from the values of the index block padding (IP) field and the number of index levels (NL) field. If the IP and NL fields are not specified, a default value of 511 words is used.

Certain fields in the FIT determine the size and characteristics of data and index blocks during file creation. Data blocks and index blocks need not be the same size; padding percentages can also be different. The following FIT fields are used in data block creation:

DP	Data block padding
KL	Key length
KT	Key type
MBL	Maximum block length
MNR	Minimum record length
MRL	Maximum record length
RB	Records per block

The FIT fields used to create the index block are as follows:

IBL	Index block length
IP	Index block padding
KL	Key length
MNR	Minimum record length
MRL	Maximum record length
NL	Number of index levels
RB	Records per block

Certain FIT fields must be set by the user before the file is opened on a creation run; otherwise, a fatal error occurs. These fields can be specified in the FILE control statement, FILE macro, or STORE macro. Any attempt to change these fields after file creation is ignored without comment. The FIT fields that must be set are as follows:

FO	File organization
KL	Key length
KT	Key type
LFN	Logical file name
MRL	Maximum record length

Other FIT fields that must be defined before the file is opened on a creation run can be set by the user or can assume default values. These fields remain the same for the life of the file and attempts to change them are ignored.

CDT	Collating sequence to display code conversion table; default depends on the DCT field
DCT	Display code to collating sequence conversion table; default is CDC conversion table
DKI	Duplicate key indicator; default is no duplicate key processing
DP	Data block padding percentage; release default is 0
IBL	Index block length; default is calculated by AAM
IP	Index block padding percentage; release default is 5
MBL	Maximum block length; default is calculated by AAM
MNR	Minimum record size; cannot exceed value of MRL; default is 0
NL	Number of index levels; maximum is 63; release default is 1
RB	Records per block; should not be specified if MBL is specified; release default is 2

Some FIT fields that are specified before the file is opened for creation are in effect only until another OPENM macro is executed. Attempted changes are ignored without comment or error until the file is opened again; the values in

the FIT are then used to accomplish the open. Default values are assumed without comment if the following fields are not set:

BCK Block checksum; default is no checksums

BFS Buffer size; default is buffer size calculated by AAM

FWB First word address of the buffer; default is buffer address provided by AAM

When records are written to a file on a creation run, the primary keys must be in ascending sequence. The old/new file (ON) field in the FIT must be set to NEW before the file is opened. Only the following macros can be used during a creation run:

OPENM

REWINDM

PUT

CLOSEM

The REWINDM macro must not be issued while records are being inserted into the file.

EXISTING FILE PROCESSING

Initial indexed sequential files must reside on mass storage devices for processing. After file creation, however, the file can be dumped to tape with a COPYBF statement or a permanent file dump routine. The file can be returned later to mass storage for processing.

Open Processing

Before an existing file can be opened, the user must call for construction of the FIT by specifying the logical file name and the file organization. When the file is opened, values from the FSTT are returned to the following FIT fields:

DKI Duplicate key indicator

IBL Index block length

KL Key length

KT Key type

MBL Maximum block length

MRL Maximum record length

NL Number of index levels

A default value is assumed without comment if the following FIT fields are not set before opening the file:

BCK Block checksum; default is no checksums

BFS Buffer size; default is buffer size calculated by AAM

FWB First word address of the buffer; default is buffer address provided by AAM

Two fields in the FIT have no default value and must be set before being used by a file processing macro. If the following fields are not set before required, a fatal error occurs:

KA Key address

WSA Working storage area

Other fields that can be set before the file is opened but need not be set until required by a file processing macro are as follows:

DFC Dayfile control

EFC Error file control

ERL Trivial error limit

EX Error exit

FLM File limit

FWI Forced write indicator

KP Beginning key position

MKL Major key length

The MKL field is reset to zero after execution of a GET or SEEK macro. The other fields remain in effect until changed.

The first time an existing file is opened after its creation run, the old/new file (ON) field in the FIT must be changed from NEW to OLD. This can be done through the FILE macro, FILE control statement, or STORE macro or by specifying any option except NEW in the processing direction (pd) parameter of the OPENM macro.

An existing file can be positioned at end-of-information during open processing. This position is established by specifying the E option in the open flag (of) parameter of the OPENM macro or by setting the open flag (OF) field in the FIT to E through the STORE macro, FILE control statement, or FILE macro before the file is opened.

Read Processing

Records can be read from the file randomly by key value or sequentially by position. The key of reference for a read operation can be the primary key or any alternate key defined for the file. The file can be open for input or for input/output.

The GET macro is used for a random read operation. The relative key word (RKW), relative key position (RKP), and key length (KL) fields in the FIT determine whether the read operation is by the primary key or by one of the alternate keys. The key value at the address specified by the key address (KA) field is used to locate the record to be read. The user must set the KA field to the address of the key value. A trivial error condition results if the specified key is not found in the file; however, the file position is altered to point to where the record should exist. If the key is part of a duplicate set, the first record of the set is returned.

Sequential reading is accomplished by the GETN macro. The GETN macro returns the next sequential record to the working storage area. If the key address (ka) parameter is specified in the macro, the primary key of the record

retrieved is returned to the specified address. The GETN macro must be used to access any duplicate key record other than the first one in the set.

Read-Only Processing

Existing files can be read, but not updated, in a smaller field length by substituting the use of a read-only capsule for the capsule that allows full file processing. Both random and sequential reading are possible with the read-only capability.

When the read-only mode is selected, the file must be open for input. If another AAM file is being processed for input/output in the same job step, the read-only mode must not be selected; if it is selected, an error occurs. The file to be read must not be an empty file. Only the following macros can be issued for the file: OPENM, GET, GETN, SEEK, SKIP, REWINDM, and CLOSEM.

The read-only capability requires the following LDSET control statement (or LDREQ macro from a COMPASS program executed through a terminal):

```
LDSET(SUBST=$RM.IS$-$RM.ISX$)
```

If static loading is being used, the following additional LDSET control statement is required (refer to appendix E for a discussion of static loading):

```
LDSET(SUBST=$SAAM.IS$-$IS.ROEN$)
```

In COBOL programs, the following additional LDSET control statement is required:

```
LDSET(OMIT=IS00OV)
```

Under the NOS operating system, libraries must have been generated with NX=1 before SUBST is used.

Write Processing

New records are added to an existing indexed sequential file with the PUT macro. Records are inserted by primary key value. The user must set the KA field in the FIT to the address of the key value. Execution is faster if the records to be inserted are sorted by primary key in ascending order.

If the file has duplicate primary key values, the position (pos) parameter of the PUT macro determines where the record is inserted within the duplicate key set. If the pos parameter is not specified, the key and the record are placed after the last record in the duplicate key set. The positioning of a record within a duplicate key set is determined by the setting of the pos parameter as follows:

If P is specified and the key given equals the key of the last record referenced, the key and the record are inserted immediately preceding the last record referenced.

If N is specified and the keys are equal, the key and the record are placed immediately following the last record referenced.

If P is specified and the keys are not equal, the key and the record are inserted preceding the first record in the duplicate key set.

If N is specified and the keys are not equal, the key and the record are inserted after the last record in the duplicate key set.

Random Processing

Random processing implies index block manipulation as well as record processing. Maximum efficiency is gained by allowing buffer space for one index block for each index level and space for two data blocks. This number of index blocks allows the primary index block to remain in memory while processing the other index and data blocks. Two data blocks provide input/output/compute overlap.

If no input/output is in progress for the file, a write is initiated for any data block that has been modified as long as the block is not the object of the current macro. This permits a high degree of input/output/compute overlap; however, if the forced write indicator (FWI) field in the FIT is set, each modified block is written immediately.

Major Key Processing

The major key feature is available with the GET, SEEK, and START macros. It allows the user to perform a search on the leading characters of a symbolic key. When the major key length (mkl) parameter is specified in the GET macro, the record returned to the working storage area is the first one encountered with a major key that matches the specified major key value. Presumably, the user wishes to examine a subset of records defined by the major key; the subset is processed using the GETN macro to access the records belonging to the subset.

The START macro can also include the mkl parameter. When it is specified, the file is positioned at the first record containing a major key that matches the specified major key value. A record is not returned to the working storage area by the START macro.

When the major key length (mkl) parameter is specified in the SEEK macro, AAM initiates transfer into the buffer of an index block or the data block containing the first occurrence of the major key. Other program processing can occur while the transfer is taking place.

The file position (FP) field in the FIT can be checked for the status of the block transfer. The FP field has the value 0 if an index block is being transferred or the value 20_8 if a data block is being transferred. If the value of the FP field is 0, another SEEK macro can be issued and a check made of the FP field. This can be done repeatedly until the data block is transferred into the buffer. The GET macro can then be issued to transfer the record containing the first occurrence of the major key from that data block in the buffer to the working storage area. The GET macro can be issued when the FP field contains 0, but then there is no overlap in processing.

Duplicate Key Processing

The user has the option of allowing the existence of duplicate primary keys. If the duplicate key position (POS) field in the FIT is not set, the GET, REPLACE, and DELETE macros reference the first record in a duplicate key set; the PUT macro places the record at the end of a duplicate key set. All records other than the first in the duplicate key set must be accessed with the GETN macro. Duplicate key processing can be selected at any time during the life of the file. Once the option to have duplicate keys is selected, duplicate keys cannot be prohibited.

File Updating

The DELETE macro physically removes the key and its associated record from the file. The key address (KA) field in the FIT must be set to point to the address of the primary key value for the record to be deleted. If the file has duplicate keys and the position (pos) parameter is omitted from the DELETE macro, the first record in the duplicate key set is deleted. The last record referenced is deleted when C is specified for the pos parameter. The key at the address specified by the KA field must equal the key of the record last referenced; otherwise, a trivial error results and the request is ignored.

If the deleted record is the only one in the data block, the block is linked into a chain of deleted data blocks to be used when new data blocks are required for file expansion. If the delete operation results in an empty index block, the block is linked into a chain of deleted index blocks.

The REPLACE macro replaces an existing record with the record in the working storage area. The primary key value for the record in the working storage area must be the same as the primary key value for an existing record. The KA field must be set to point to the primary key for the working storage area record.

The first record in a duplicate key set is replaced when the position (pos) parameter is omitted from the REPLACE macro. The last record referenced is replaced when the pos parameter is set to C. The key at the address specified by the KA field must equal the key of the last record referenced; otherwise, a trivial error occurs and the request is ignored.

File Positioning

When the OPENM macro is executed, positioning of the file depends on the open flag (of) parameter in the macro. If R (rewind) is specified, the file is positioned at the first record, which is the record with the lowest primary key value. If E (end-of-information) is specified, the file is positioned after the last record, which is the record with the highest primary key value. Omitting the parameter causes the current value of the OF field in the FIT to be used. File positioning remains unchanged until one of the following macros is executed: GET, GETN, REWINDM, PUT, REPLACE, DELETE, SKIP, or START.

The GET macro, which accesses a record randomly, alters the file position to the record returned by the macro. The GETN macro, which accesses a record sequentially, advances the file position one logical record and returns that record unless the file is positioned at end-of-information.

The REWINDM macro positions the file to beginning-of-information; execution of the GETN macro then returns the first record in the file. The SKIP macro positions the file forward or backward the specified number of records. After end-of-information has been reached, subsequent forward skips without file positioning cause trivial errors. If a skip count of zero is given, no action is taken.

The START macro positions the file according to a specified key value and key relation; the file is positioned at the record with a key value that is equal to (EQ), greater than or equal to (GE), or greater than (GT) the specified key value. If the specified key value does not exist in the file, the file is positioned at the record with the next greater key value.

Overlap Processing

In response to a user program request for a record, AAM locates the data block by searching the index blocks and transfers the data block from mass storage to the buffer area. The record is then transferred to the working storage area. The execution time to do this can be overlapped with program processing by using the SEEK macro.

The SEEK macro transfers an index or data block from mass storage to the buffer, returning control to the user program at the start of the transfer. The user must check the file position (FP) field in the FIT to determine if an index block (FP set to 0) or a data block (FP set to 20_8) is being transferred. Multiple SEEK macros can be issued until the transfer of the data block is initiated. The user can then issue a macro to process the record originally specified in the SEEK macro. The SEEK macro does not return a record to the working storage area.

EXTENDED INDEXED SEQUENTIAL FILES

The extended indexed sequential file organization is well suited for applications that require reasonably efficient storage and retrieval of records both randomly and sequentially by primary or alternate key. A primary key is a unique identifier defined by the user for each record within an extended indexed sequential file. Primary and alternate keys can be in any of the following forms:

60-bit signed binary (10 characters)

Symbolic (1 to 255 contiguous alphanumeric characters)

Uncollated symbolic (1 to 255 contiguous alphanumeric characters)

The value of the primary key determines the location of the record in the file. Characters within a symbolic (alphanumeric) key are collated according to the standard CDC collating sequence or according to a user-supplied collating sequence. Any user collating sequence has meaning for ranking keys only; it is stored with the user file in the FSTT. Numeric keys are ordered by value. Keys within an extended indexed sequential file can be a part of the record (embedded) or not a part of the record (nonembedded).

FILE CREATION RUN

A separate creation run is necessary for an extended indexed sequential file. This can be done through the FORM utility or through a source program. The FSTT is created when the extended indexed sequential file is created.

The efficiency with which an extended indexed sequential file can be processed is influenced by two fields in the FIT: maximum block length (MBL) and buffer size (BFS). On a creation run, the user has the option of specifying these values directly or accepting system defaults calculated by AAM. The FLBLOK utility, which is described in section 7, can be used to calculate suggested values for the MBL and BFS fields.

If the MBL field is not specified directly, the value is calculated from the values of the following fields in the FIT:

DP	Data block padding
KL	Key length
MNR	Minimum record length

MRL Maximum record length

RB Records per block

A number of fields in the FIT determine the size and characteristics of data and index blocks during file creation. Data and index blocks must be the same size; padding percentages, however, can be different. The following FIT fields are used in data block creation:

DP Data block padding

KL Key length

KT Key type

MBL Maximum block length

MNR Minimum record length

MRL Maximum record length

RB Records per block

The FIT fields used to create the index block are as follows:

IP Index block padding

KL Key length

MBL Maximum block length

MNR Minimum record length

MRL Maximum record length

NL Number of index levels

RB Records per block

Certain FIT fields must be set by the user before the file is opened on a creation run; otherwise, a fatal error occurs. These fields can be specified in the FILE control statement, FILE macro, or STORE macro. Any attempt to change these fields after file creation is ignored without comment. The FIT fields that must be set are as follows:

FO File organization

KL Key length

KT Key type

LFN Logical file name

MRL Maximum record length

If the primary key is embedded in the record, the following FIT fields must also be set:

EMK Embedded key, set to YES

RKP Relative key position; character position within RKW in which the key begins

RKW Relative key word; word in which the key begins

Other FIT fields that must be defined before the file is opened on a creation run can be set by the user or can assume default values. These fields remain the same for the life of the file and attempts to change them are ignored.

DCT Display code to collating sequence conversion table; default is CDC conversion table

DP Data block padding percentage; release default is 0

IP Index block padding percentage; release default is 0

MBL Maximum block length, data and index blocks; default is calculated by AAM

MNR Minimum record size; cannot exceed value of MRL; default is 0

NL Number of index levels; maximum is 15; release default is 1

RB Records per block; should not be specified if MBL is specified; release default is 2

XBS Index file block size; default is data file block size (MBL)

Some FIT fields that can be specified before the file is opened for creation are in effect only until another OPENM macro is executed. Attempted changes are ignored without comment or error until the file is opened again; the values in the FIT are then used to accomplish the open. Default values are assumed without comment if the following fields are not set:

BCK Block checksum; default is no checksums

BFS Buffer size; default is buffer size calculated by AAM

CPA Compression routine address; default is no compression of records

DCA Decompression routine address; default depends on the CPA field

FWB First word address of the buffer; default is buffer address provided by AAM

When records are written to a file on a creation run, the primary keys should be in ascending sequence for a more efficient run. The old/new file (ON) field in the FIT must be set to NEW before the file is opened.

EXISTING FILE PROCESSING

Extended indexed sequential files must reside on mass storage devices for processing. After file creation, however, the file can be dumped to tape with a COPYBF statement or a permanent file dump routine. The file can be returned later to mass storage for processing.

Open Processing

Before an existing file can be opened, the user must call for construction of the FIT by specifying the logical file name and the file organization. When the file is opened, values from the FSTT are returned to the following FIT fields:

KL	Key length
KT	Key type
MBL	Maximum block length
MRL	Maximum record length
NL	Number of index levels
RKP	Relative key position
RKW	Relative key word

The RKW and RKP fields are set to 0 and 10, respectively, if the key is not embedded in the record.

A default value is assumed without comment if the following FIT fields are not set before opening the file:

BCK	Block checksum; default is no checksums
BFS	Buffer size; default is buffer size calculated by AAM
CPA	Compression routine address; default is no compression of records
DCA	Decompression routine address; default depends on the CPA field
FWB	First word address of the buffer; default is buffer address provided by AAM

Two FIT fields have no default value and must be set before being used by a file processing macro. If the following fields are not set before required, a fatal error occurs:

KA	Key address
WSA	Working storage area

Other fields that can be set before the file is opened but need not be set until required by a file processing macro are as follows:

DFC	Dayfile control
EFC	Error file control
ERL	Trivial error limit
EX	Error exit
FLM	File limit
FWI	Forced write indicator
KP	Beginning key position
MKL	Major key length

The MKL field is reset to zero after execution of a GET, SEEK, or START macro. The other fields remain in effect until changed.

The first time an existing file is opened after its creation run, the old/new file (ON) field in the FIT must be changed from NEW to OLD. This can be done through the FILE macro, FILE control statement, or STORE macro or by specifying any option except NEW in the processing direction (pd) parameter of the OPENM macro.

An existing file can be positioned at end-of-information during open processing. This position is established by specifying the E option in the open flag (of) parameter of the OPENM macro or by setting the open flag (OF) field in the FIT to E through the STORE macro, FILE control statement, or FILE macro before the file is opened.

Read Processing

Records can be read from the file randomly by key value or sequentially by position. The key of reference for a read operation can be the primary key or any alternate key defined for the file. The file must be open for input or for input/output.

The GET macro is used for a random read operation. The relative key word (RKW), relative key position (RKP), and key length (KL) fields in the FIT determine whether the read operation is by the primary key or by one of the alternate keys. For a nonembedded primary key, RKW and RKP must be set to 0 and 10, respectively. The key value at the address specified by the key address (KA) field is used to locate the record to be read. The user must set the KA field to the address of the key value. A trivial error condition results if the specified key is not found in the file; however, the file position is altered to point to where the record should exist.

Sequential reading is accomplished by the GETN and GETNR macros. The GETN macro returns the next sequential record to the working storage area. The GETNR macro performs this same function; however, control returns immediately to the user if input/output is required to complete the request. The macro can be issued repeatedly until the transfer of the record is complete, or the input/output status can be monitored for completion before issuing the GETNR macro again.

Write Processing

New records are added to an existing indexed sequential file with the PUT macro. Records are inserted by primary key value. For a nonembedded primary key, the user must set the KA field in the FIT to the address of the key value. Execution is faster if the records to be inserted are sorted by primary key in ascending order.

Random Processing

Random processing implies index block manipulation as well as record processing. If the user cannot allow AAM to use the Common Memory Manager (CMM), maximum efficiency is gained by allowing buffer space for one index block for each index level and space for two data blocks. This number of index blocks allows the primary index block to remain in memory while processing the other index and data blocks. Two data blocks provide input/output/compute overlap. The user can direct AAM to allocate this amount of buffer space by setting the buffer size (BFS) field and by not setting the first word address of the buffer (FWB) field. Refer to appendix G for a detailed description of buffer allocation.

If no input/output is in progress for the file, a write is initiated for any data block that satisfies the following conditions:

The block was altered by the preceding macro.

The block is not the object of the current macro.

This permits a high degree of input/output/compute overlap; however, if the forced write indicator (FWI) field in the FIT is set, each modified block is written immediately.

Major Key Processing

The major key feature is available with the GET, SEEK, and START macros. It allows the user to perform a search on the leading characters of a symbolic key. When the major key length (mkl) parameter is specified in the GET macro, the record returned to the working storage area is the first one encountered with a major key that matches the specified major key value. Presumably, the user wishes to examine a subset of records defined by the major key; the subset is processed using the GETN or GETNR macro to access the records belonging to the subset.

The START macro can also include the mkl parameter. When it is specified, the file is positioned at the first record containing a major key that matches the specified major key value. A record is not returned to the working storage area by the START macro.

When the mkl parameter is specified in the SEEK macro, AAM initiates transfer into the buffer of an index block or the data block containing the first occurrence of the major key. Other program processing can occur while the transfer is taking place.

The file position (FP) field in the FIT can be checked for the status of the block transfer. The FP field has the value 0 if an index block is being transferred or the value 20_8 if a data block is being transferred. If the value of the FP field is 0, another SEEK macro can be issued and a check made of the FP field. This can be done repeatedly until the data block is transferred into the buffer. The GET macro can then be issued to transfer the record containing the first occurrence of the major key from that data block in the buffer to the working storage area. The GET macro can be issued when the FP field contains 0, but then there is no overlap in processing.

File Updating

The DELETE macro physically removes the key and its associated record from the file. The key address (KA) field in the FIT must be set to point to the address of the primary key value for the record to be deleted. If the deleted record is the only one in the data block, the block is linked into a chain of deleted blocks to be used when new blocks are required for file expansion. If the delete operation results in an empty index block, the block is linked into the chain of deleted blocks.

The REPLACE macro replaces an existing record with the record in the working storage area. The primary key value for the record in the working storage area must be the same as the primary key value for an existing record. For a nonembedded primary key, the KA field must be set to point to the primary key for the working storage area record.

File Positioning

When the OPENM macro is executed, positioning of the file depends on the open flag (of) parameter in the macro. If R (rewind) is specified, the file is positioned at the first record, which is the record with the lowest primary key value. If E (end-of-information) is specified, the file is positioned after the last record, which is the record with the highest primary key value. Omitting the parameter causes the current value of the OF field in the FIT to be used. File positioning remains unchanged until one of the following macros is executed: GET, GETN, GETNR, REWINDM, SKIP, or START.

The GET macro, which accesses a record randomly, alters the file position to the record returned by the macro. The GETN macro, which accesses a record sequentially, advances the file position one logical record and returns that record unless the file is positioned at end-of-information. The GETNR macro also advances the file position one logical record when it returns a record.

The REWINDM macro positions the file to beginning-of-information; execution of the GETN or GETNR macro then returns the first record in the file. The SKIP macro positions the file forward or backward the specified number of records; the file is positioned at beginning-of-information or end-of-information if the skip count is too large.

The START macro positions the file according to a specified key value and key relation; the file is positioned at the record with a key value that is equal to (EQ), greather than or equal to (GE), or greater than (GT) the specified key value. If the specified key value does not exist in the file, the file is positioned at the record with the next greater key value.

Overlap Processing

In response to a user program request for a record, AAM locates the data block by searching the index blocks and transfers the data block from mass storage to the buffer area. The record is then transferred to the working storage area. The execution time to do this can be overlapped with program processing by using the SEEK macro or the GETNR macro.

The SEEK macro transfers an index or data block from mass storage to the buffer, returning control to the user program at the start of the transfer. The user must check the file position (FP) field in the FIT to determine if an index block (FP set to 0) or a data block (FP set to 20_8) is being transferred. Multiple SEEK macros can be issued until the transfer of the data block is initiated. The user can then issue a macro to process the record originally specified in the SEEK macro. The SEEK macro does not return a record to the working storage area.

The GETNR macro is used to read records sequentially. If execution of the GETNR macro initiates block transfer to the buffer, control returns immediately to the user. The file position (FP) field can be monitored in the same manner as for the SEEK macro to determine when block transfer is complete. The busy FET address (BZF) field points to an input/output status word that can be monitored to determine when input/output processing is complete. When the FP field is set to 20_8, the record has already been returned as if the last GETNR macro in the series had been a GETN macro.

ACTUAL KEY FILES

The actual key file organization provides fast random access to records in the file. Random access usually requires one access per record. The primary key for a record is its storage location (block number and record number within the block). The user must preserve primary keys if the file is to be accessed randomly by primary key.

FILE CREATION RUN

A separate creation run is necessary for an actual key file. This can be done through the FORM utility or a source program. The FSTT is created when the actual key file is created.

Certain FIT fields must be set by the user before the file is opened on a creation run; otherwise, a fatal error occurs. These fields can be specified in the FILE control statement, FILE macro, or STORE macro. Any attempt to change these fields after file creation is ignored without comment. The FIT fields that must be set are as follows:

- FO File organization
- KL Key length
- LFN Logical file name
- MNR Minimum record length
- MRL Maximum record length
- RT Record type

Two FIT fields that must be defined for file creation can be specified by the user or can assume default values.

- DP Data block padding percentage; release default is 0
- MBL Maximum block length; default is calculated by AAM

If the MBL field is not specified directly, the value is calculated from the values in the following fields:

- MNR Minimum record length
- MRL Maximum record length
- RB Records per block

The value specified for the MBL field must be large enough to hold at least the value specified in the RB field, the block header, checksum (if this option is selected), and the number of average size records specified by the RB field. AAM increases the block size, if necessary, to use mass storage efficiently. Resulting blocks are an integral multiple of physical record unit (PRU) size minus one central memory word.

The following FIT fields must be selected before the file is created if the option is to be used during the life of the file:

- HB Header bit; header appears with user data
- RB Records per block

Some FIT fields that can be specified before the file is opened for creation are in effect only until another OPENM macro is executed. Attempted changes are ignored without comment until the file is opened again; the values in the FIT are then used to accomplish the open. Default values are assumed without comment if the following fields are not set:

- BCK Block checksum; default is no checksums
- BFS Buffer size; default is buffer size calculated by AAM
- FWB First word address of the buffer; default is buffer address provided by AAM

The key value at the address specified by the key address (KA) field in the FIT determines where the record is written. The user can do either of the following:

The key value at location KA can be set to zero; this allows AAM to determine the key value associated with the record.

The key value at location KA can be set to a properly formatted key; this tells AAM where to store the record.

Only the following macros can be used during a creation run:

OPENM

REWINDM

PUT

CLOSEM

EXISTING FILE PROCESSING

Actual key files must reside on mass storage for processing. After file creation, the file can be dumped to tape through the DUMPF utility and reloaded to mass storage for processing with the LOADPF utility. The COPYBF utility can also be used to copy the file to tape and then back to mass storage for processing.

Open Processing

Before an existing file can be opened, the user must call for construction of the FIT by specifying the logical file name and the file organization. When the file is opened, values from the FSTT are returned to the following FIT fields:

- KL Key length
- MBL Maximum block length
- MNR Minimum record length
- MRL Maximum record length
- RB Records per block

A default value is assumed without comment if the following FIT fields are not set before the file is opened:

- BCK Block checksum; default is no checksums
- BFS Buffer size; default is buffer size calculated by AAM
- FWB First word address of the buffer; default is buffer address provided by AAM

Other FIT fields that can be set before the file is opened but need not be set until required by a file processing macro are as follows:

DFC	Dayfile control
DX	End-of-data exit
EFC	Error file control
ERL	Trivial error limit
EX	Error exit
FLM	File limit
FWI	Forced write indicator

The first time an existing file is opened after its creation run, the old/new file (ON) field in the FIT must be changed from NEW to OLD. This can be done through a FIT manipulation macro or by specifying any option except NEW for the processing direction (pd) parameter in the OPENM macro.

Read Processing

The GET macro is used to read records randomly by key value. The key value at the address specified by the key address (KA) field in the FIT is used to locate the record to be read. The GETN macro is used to read the next record in sequence by position. Records can be read by primary key or by any alternate key defined for the file. The file must be open for input or for input/output.

When a record is read, the number of characters retrieved is returned to the record length (RL) field in the FIT. If the requested record is not found, a trivial error results.

The setting of the header indicator bit (HB) field in the FIT determines whether the whole record, including the header, is returned to the working storage area when a record is read. If the HB field is set to YES, the entire record is returned. If the HB field is set to NO, AAM assumes that bits 0 through 14 in the first word of the header specify the number of words the user considers to be the data record header; only the nonheader portion of the record is returned.

Execution of the GETN macro causes the next sequential record to be placed in the working storage area. The first time the GETN macro is issued after the file is opened or after any rewind request, the first record in the file is retrieved. The next GETN macro retrieves the next sequential record. Any empty record position is ignored. Overflow records are returned as they are encountered. An overflow record occupies two slots; the first slot is the one where the record should be and the second slot is the one that actually contains the record. The record is returned when the first slot is encountered. If the key address (ka) parameter is specified in the GETN macro, the primary key value of the record retrieved is returned to the specified address.

Write Processing

The PUT macro is used to add a record to an existing actual key file. The key value at the address specified by the key address (KA) field in the FIT must be unique or zero; otherwise, the request is ignored. When a key value is specified, it must indicate the block number of an existing block or a block number one higher than the highest numbered block currently existing. A key value of zero causes AAM to determine the location for the record; the key value is returned to the user at location KA. If a block cannot accommodate a record with a user-specified key, the record is placed elsewhere by AAM; the value of the original key does not change for user program purposes.

An index for actual key files is not maintained by AAM. For subsequent random reading by primary key value, the user is responsible for preserving primary keys of records written on the file. A multiple-index file can be created to maintain an index for actual key files.

If the header indicator bit (HB) field in the FIT is set to divide a record into a user header portion and a user data portion when a record is read, the record must be written accordingly. The first word in the working storage area must indicate in bits 0 through 14 the number of words in the user header.

File Updating

After a file has been created, records in the file can be deleted or replaced. The DELETE and REPLACE macros are used to update an actual key file.

A record can be eliminated from an existing file with the DELETE macro. The record indicated by the key value at location KA is logically removed from the file and the key is set to zero. The record is physically removed when the space is needed and any remaining records in the block can be relocated. If the requested record cannot be found, the request is ignored and a trivial error results.

The REPLACE macro is used to replace an existing record with a new record. The existing record is specified by the key value at location KA. The new record is in the working storage area. The new record need not be the same size as the record being replaced.

File Positioning

When the OPENM macro is executed, the file is positioned at the first record in the file. File positioning remains unchanged until one of the following macros is executed: GET, GETN, REWINDM, or SKIP. The GET and GETN macros, which are used to read records, position the file at the record retrieved.

The SKIP macro positions the file forward or backward the specified number of records to the beginning of another record. Only small skips should be made because each intervening record is read and counted. The SKIP macro does not return a record to the working storage area.

Skipping stops if beginning-of-information or end-of-information is reached. An informative message is issued if skipping or sequential reading is attempted past the file boundary, but no error exit is taken. Any end-of-data exit is executed only if end-of-information is encountered. A skip count of zero is interpreted as a no-op.

The use of the REWINDM macro is more efficient than extensive backward skipping of records. This macro positions the file to beginning-of-information, which is the start of the user data record with the lowest key.

Overlap Processing

In response to a user program request for a record, AAM determines the block needed and transfers it from mass storage to the buffer area. The specified record is then

transferred to the working storage area. The execution time to do this can be overlapped with program processing by using the SEEK macro.

The SEEK macro transfers the block with the record from mass storage to the buffer, returning control to the user program at the start of the transfer. The program can continue processing. A macro can then be issued to process the record originally specified in the SEEK macro. The SEEK macro does not return a record to the working storage area.

DIRECT ACCESS FILES

The direct access file organization is well suited for applications that require rapid access by key value. Direct access files can be accessed either randomly or sequentially by primary or alternate key; however, records accessed sequentially by primary key are not logically ordered.

FILE CREATION RUN

A separate creation run is necessary for a direct access file. This can be done through the FORM utility, the CREATE utility, or a source program.

Mass storage for a direct access file is preallocated. Before the file is opened on a creation run, the user must specify the size and number of home blocks to be preallocated. The number of home blocks is specified by setting the number of home blocks (HMB) field in the FIT. The key analysis utility, which is described in section 7, can be used to test various home block sizes.

The user has the option of specifying the home block size directly or accepting a system default. The maximum block length (MBL) field is set by the user to specify home block size. If the MBL field is not set by the user, AAM calculates the value for the MBL field from the values in the following FIT fields:

MNR	Minimum record length
MRL	Maximum record length
RB	Records per block; default is 2

A number of fields must be set by the FILE control statement, the FILE macro, or the STORE macro before the file is opened for a creation run. If these fields are specified for an existing file, the new values are ignored without comment. A fatal error occurs if the following fields are not set on a creation run:

HMB	Number of home blocks
KL	Key length
LFN	Logical file name
MNR	Minimum record length
MRL	Maximum record length

The position of the primary key in the record is assumed to begin in the first character position. If the primary key is in another position, the position must be specified before the file is opened. The following FIT fields, which cannot be changed after the file is opened for a creation run, describe the key position:

RKP	Relative key position
RKW	Relative key word

Default values are used without comment if certain FIT fields are not set before the file is opened for a creation run. These fields are effective only until the file is opened again; attempted changes are ignored without comment until another OPENM macro is executed. At that time, the values in the FIT are used to accomplish the open. These fields are as follows:

BCK	Block checksum; default is no checksums
BFS	Buffer size; default is buffer size calculated by AAM
FWB	First word address of the buffer; default is buffer address provided by AAM

The minimum size of the buffer is two data blocks and the FSTT. The maximum size is three data blocks and the FSTT.

Only the following macros can be used on a file creation run:

OPENM

REWINDM

PUT

CLOSEM

Overflow

Overflow records in direct access files are handled in one of three ways. The setting of the overflow (OVF) field in the FIT at file creation time determines the method used for overflow records. The OVF field can be set as follows:

OVF=OVB (default)	Overflow records are stored in overflow blocks or in other home blocks. If conservation of file space is more critical than access time, this option should be selected.
OVF=OVO	Overflow blocks are created to handle any overflow records occurring. If access time is more critical than file space, this option should be selected.
OVF=OVH	Overflow records are stored in home blocks only. An attempt to add a record to a file whose home blocks are filled is disallowed and a trivial error message is issued.

User Hashing Routine

At file creation time, the user has the option of selecting a user hashing routine instead of the supplied hashing routine. This option is controlled by the hashing routine location (HRL) field in the FIT.

If the symbolic entry point name of the user hashing routine is MYHASH, the user should code HRL==XMYHASH in the FILE macro. Parameters needed by the user hashing routine are passed as follows:

```
SA1     ARRAY
RJ      =XMYHASH
```

The array contains the addresses of the following:

ARRAY	Key length (KL)
ARRAY+1	Key address (KA)
ARRAY+2	Number of home blocks (HMB)
ARRAY+3	Hashing result

When the hashing routine completes its computation, the address of the hashing result must be placed in ARRAY+3 and control must be returned to AAM. AAM then converts the value to a relative physical record unit (PRU) number. The user hashing routine could be coded as shown in figure 4-1. Upon return to AAM from any hashing routine, the remainder of the hashed key divided by the value of the HMB field is used as the ordinal of a home data block.

```
MYHASH  DATA    0
        Computation
        BX6     Xi      STORE HASH RESULT
        SA2     A1+3    GET ADDRESS FOR
                        HASHED RESULT
        SA6     X2      STORE HASH RESULT
        EQ      MYHASH  RETURN TO AAM/DA
```

Figure 4-1. User Hashing Routine Example

Supplied Hashing Routine

When the HRL field is not set to the address of a user hashing routine, the system-supplied hashing routine is used. The supplied hashing routine folds the word-aligned key into one word using the integer add instruction. If the folded key is an 18-bit integer or an 18-bit packed integer, no further hashing is done; otherwise, the folded key is hashed using the shift and divide instructions to produce a 48-bit result. This hashed key is the ordinal of a home data block on mass storage.

A prime number of home blocks is recommended when the supplied hashing routine is used. This generally produces a more uniform distribution of records than a nonprime number.

Direct Access File Records

All record types are allowed for direct access files. When creating the file through a source language, W type records are the default. When using the FORM utility or the CREATE utility through FORM, Z type records are the default.

AAM determines the length of each record before it is written to the file. A data record header is generated by AAM; this includes the number of characters in the record. The header and record are then written to the file.

EXISTING FILE PROCESSING

Direct access files must reside on mass storage for processing. After file creation, the file can be dumped to tape using the FORM utility and then reloaded for processing. The DUMPF and LOADPF utilities or the COPYBF utility can also be used.

Open Processing

Before an existing file can be opened, the user must call for construction of the FIT by specifying the logical file name and the file organization. When the file is opened, values from the FSTT are returned to the following FIT fields:

HMB	Number of home blocks
KL	Key length
MBL	Maximum block length
MNR	Minimum record length
MRL	Maximum record length

If the following fields are not set before the file is opened, the default value is assumed without comment:

BCK	Block checksum; default is no checksums
BFS	Buffer size; default is buffer size calculated by AAM
FWB	First word address of the buffer; default is buffer address provided by AAM

Two FIT fields that have no default value must be set before being used by a file processing macro; otherwise, a fatal error occurs. These fields are as follows:

KA	Key address
WSA	Working storage area

A number of FIT fields can be set before the file is opened but need not be set until required by file processing macros. These fields are as follows:

DFC	Dayfile control
DX	End-of-data exit
EFC	Error file control
ERL	Trivial error limit
EX	Error exit
FWI	Forced write indicator
KP	Beginning key position

The first time an existing file is opened after its creation run, the old/new file (ON) field in the FIT must be changed from NEW to OLD. This can be done through a FIT manipulation macro or by specifying any option except NEW in the processing direction (pd) parameter of the OPENM macro.

Read Processing

A direct access file can be read randomly by primary or alternate key using the GET macro. It can also be read sequentially by the GETN macro. The file must be open for input or input/output.

For both the GET and GETN macros, the number of characters read is based on the record length value in the data record header. The value of the record length (RL) field in the FIT is ignored. At the completion of a read operation, the RL field is set to the length of the record returned.

With the GET macro, records are located using the key value at the address indicated by the key address (KA) field in the FIT. If the requested record cannot be found, a trivial error occurs.

The first GETN macro executed after an OPENM or REWINDM macro retrieves the first record in the file. A subsequent GETN macro retrieves the next sequential record. All home blocks are processed first and then any overflow blocks. Intervening GET, REPLACE, and DELETE macros are allowed and do not alter the sequential position of the file. If a PUT macro or a REPLACE macro with a larger size record is followed by a GETN macro, a trivial error results. Any other function has no effect on sequential reading or file positioning.

Read-Only Processing

Existing files with direct access file organization can be read, but not updated, in a smaller field length by using a read-only capsule instead of a full capsule with update capabilities. Both random and sequential reading are possible with the read-only capability.

When the read-only mode is selected, the file must be opened for input. If another AAM file is being processed for input/output in the same job step, the read-only mode must not be selected; if it is selected, an error occurs. The file to be read must not be an empty file. Only the following macros can be issued for the file: OPENM, GET, GETN, SEEK, REWINDM, and CLOSEM. Any updating operation causes a fatal error to be issued.

The direct access file read-only capability requires the following LDSET control statement (or LDREQ macro from a COMPASS program executing through a terminal):

```
LDSET(SUBST=$RM.DA$-$RM.DAX$)
```

If static loading is being used, the following additional LDSET control statement is required:

```
LDSET(SUBST=$SAAM.DA$-$RO$$DA$)
```

Refer to appendix E for a discussion of static and dynamic loading. Under the NOS operating system, libraries must have been generated with NX=1 before SUBST is used.

Write Processing

Records are written to a direct access file with the PUT macro. The user must set the key of the record to be unique. The key also must be in the same position within the record as when the file was established.

File Updating

The DELETE macro logically removes an existing record from a file. The record associated with the specified key is flagged as deleted and the space is available to store another record in the file. If the requested record is not found, a trivial error results and the request is ignored.

The REPLACE macro can be used to replace a record in the direct access file with a record in the working storage area. The record to be replaced is located by hashing the key specified by the relative key word (RKW), relative key position (RKP), and key length (KL) fields in the FIT. Replacement records need not be the same size as the records replaced unless the file is being processed sequentially. A REPLACE macro that changes the record size invalidates further sequential processing.

File Positioning

The REWINDM macro is used to position the file to beginning-of-information. The file must be open when the macro is executed. The REWINDM macro resets the sequential position so that the next GETN macro returns the first record in the direct access file.

Overlap Processing

In response to a user program request for a record, AAM locates the desired home block by hashing the key and then transfers the home block to the buffer area. The specified record is then transferred to the working storage area. The execution time to do this can be overlapped with program processing by using the SEEK macro.

The SEEK macro transfers a home block from mass storage to the buffer, returning control to the user program at the start of the transfer. The program can continue processing. A macro can then be issued to process the record originally specified in the SEEK macro. The SEEK macro does not return a record to the working storage area.

The macros described in this section are used for processing the AAM files established with the FILE macro and FILE control statement. The macros conform to COMPASS syntax; the location, operation, and variable fields are separated by one or more blanks.

In the macro parameter strings, the fit parameter is required; all others are optional and positional. When an optional parameter is omitted, the parameter position must be marked by a comma; however, trailing commas can be omitted. For example, the format of the OPENM macro is:

```
OPENM    fit,pd,of
```

If the pd parameter is not specified in the OPENM macro, the format is:

```
OPENM    fit, ,of
```

The first parameter of every macro (fit) identifies the file information table for the referenced file. If the address specified by the fit parameter is invalid, the results are indeterminate. The fit parameter can specify any of the following:

lfn	Location field name of the first word of the FIT; one through seven alphabetic or numeric characters.
Rn	Any A, B, or X register containing the FIT address.
exp	Any COMPASS expression giving the FIT address.

Only parameters applicable to the file organization specified in the FIT should be set. Supplying parameters applicable to other file organizations could cause erroneous results.

MACRO EXECUTION

The current contents of the FIT are used for macro execution. If a parameter is omitted, the default value is valid only if the respective FIT field has not been previously set to a different value. A field in the FIT can be set by any of the following:

FILE macro parameter

FILE control statement parameter, which can override defaults during open processing

SETFIT macro, which can call for FILE control statement processing without full open processing

Default, which can be set during open processing

Macro parameter that is moved to the FIT before file processing occurs (a zero value in a parameter list moves a zero to the FIT field; a null value does not affect the FIT field)

Registers are not saved or restored. It should be assumed that all registers are destroyed during macro execution.

Static loading for AAM uses the STLD.RM macro and new parameters in the FILE control statement or FILE macro. Refer to appendix E for details on static loading.

The user macros, with the exception of the FETCH, FILE, STLD.RM, and STORE macros, generate code as follows:

When syntax error checking is completed, all nonnull parameters following the FIT address are placed in registers.

Register B6 is set to the end of the macro expansion as the return address.

A jump to the proper AAM entry point is generated in the top of a word; bits indicating which parameters were specified with the macro are set in the bottom of the word.

The FIT address is placed in register A0; if it is already in A0, no code is generated.

Register B1 is set to 1; if B1=1 pseudo-op is in effect, no code is generated.

PROCESSING MACROS

Several macros are available for processing AAM files. These macros are described in this section. The FETCH, FILE, SETFIT, and STORE macros are described in section 3, File Information Table.

CLOSEM MACRO

The CLOSEM macro terminates file processing and positions the file as specified. It should be the last macro issued for a file. The format of the CLOSEM macro is shown in figure 5-1.

When the CLOSEM macro is executed for a file opened for output, any information in the file buffer is written on the file as part of file termination. If unload (U) is specified in the CLOSEM macro, close processing is as follows:

If it is a permanent file, it is detached from the job and returned to the permanent file manager.

Mass storage space assigned to the file is released.

When the CLOSEM macro is executed for a file opened for output, any information in the file buffer is written on the file as part of file termination.

Close processing for a file varies according to the value specified for the cf parameter of the CLOSEM macro.

Rewind (R)

The file is rewound.

CLOSEM fit,cf

fit Address of the FIT.

cf File positioning after close processing:

R	Rewind (default)
N	No rewind
U	Unload; release buffer space and remove name from the active file list.
RET	Return; release buffer space and remove name from the active file list.
DET	Detach; release buffer space and remove name from the active file list.

Only the fit parameter can be specified as a register.

Figure 5-1. CLOSEM Macro Format

No rewind (N)

The file is not rewound.

Unload (U)

The file is rewound. The open/close flag (OC) field in the FIT is cleared. If the file is a permanent file, it is detached from the job and returned to the permanent file manager. Any scratch mass storage space assigned to the file is released.

Return (RET)

The processing is the same as for unload.

Detach (DET)

The file is not rewound. The open/close flag (OC) field in the FIT is cleared.

A CLOSEM request for a file that has never been opened, or a file that has been closed but not unloaded or reopened, has the following effects:

The FIT error status redundant close is set.

File positioning is the same as for an open file.

Control is returned to the error exit.

DELETE MACRO

The DELETE macro removes a record from the file. If the requested record is not found, a trivial error results and the request is ignored. The format of the DELETE macro is shown in figure 5-2.

Applicable parameters by type of file organization for the DELETE macro are as follows:

Initial indexed sequential	fit,ex,ka,kp,pos
Extended indexed sequential	fit,ex,ka,kp
Direct access	fit,ex,ka,kp
Actual key	fit,ex,ka

DELETE fit,ex,ka,kp,pos

fit Address of the FIT.

ex Address of the error routine.

ka Address of the primary key for the record to be deleted.

kp Beginning character position of the primary key.

pos Duplicate key position; can be C (current record) or omitted (first record in the duplicate key set). Applies only when duplicate key processing is allowed for initial indexed sequential files.

Parameters can be specified as registers.

Figure 5-2. DELETE Macro Format

When the DELETE macro is executed, the specified record is either flagged as deleted or physically removed from the file. If the requested record is not found, a trivial error results and the request is ignored. For initial indexed sequential files with duplicate primary keys, a trivial error results if the pos parameter is set to C and the requested key does not equal the key of the current record; the request is ignored.

FLUSHM MACRO

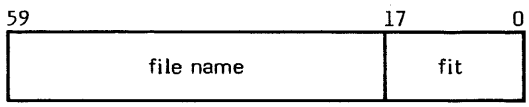
The FLUSHM macro is applicable only to extended indexed sequential files. This macro processes one or more file buffers as if a CLOSEM macro had been issued; the files, however, remain open. Blocks with pending writes and the updated FSTT are written on the file. The format of the FLUSHM macro is shown in figure 5-3.

FLUSHM fitlist

fitlist Address of the list of FIT address entries.

Figure 5-3. FLUSHM Macro Format

The list referenced by the fitlist parameter contains a one-word entry for each file to be flushed. A word of binary zeros terminates the list. The one-word entry is formatted as follows:



The file name, which is specified in display code, is used as a consistency check. The address of the FIT is specified in the lower bits of the word.

GET MACRO

The GET macro retrieves data from a file and delivers it to the working storage area. The file must be open for input or for input/output. The GET macro retrieves a record randomly by key value. The GETN and GETNR macros retrieve records sequentially by file position; the GETNR macro is applicable only to extended indexed sequential files. The formats of the macros are shown in figure 5-4.

GET	fit,wsa,0,ex,ka,kp,mkl
GETN	fit,wsa,ex,ka
GETNR	fit,wsa,ex,ka
fit	Address of the FIT.
wsa	Address of the working storage area to which the user record is returned.
ex	Address of the error routine.
ka	Address of the key for the record to be read.
kp	Beginning character position of the key.
mkl	Major key length in characters; can be used only for a symbolic key in an indexed sequential file.

Parameters can be specified as registers; if parameters are not specified, values in appropriate FIT fields are used.

Figure 5-4. GET, GETN, and GETNR Macro Formats

Applicable parameters by type of file organization for the GET macro are as follows:

Indexed sequential	fit,wsa,0,ex,ka,kp,mkl
Direct access	fit,wsa,0,ex,ka,kp
Actual key	fit,wsa,0,ex,ka

The GET macro transfers a record from a file to the specified working storage area. The location referenced by the ka parameter contains the key value for the record to be read. If no record in the file has a matching key value, a nonfatal error occurs. The record length (RL) field in the FIT is updated to indicate the number of characters in the record retrieved from the file.

If the record is longer than specified by the maximum record length (MRL) field in the FIT, an excess data error occurs. Control is passed to the error exit after transferring to the working storage area the number of characters specified by the MRL field. A record greater than the maximum record length is prevented from overwriting a portion of the calling program or other preserved information. Control is transferred to the user end-of-data exit (DX field in the FIT) by a GET request that detects end-of-information.

The GETN macro is used to read records sequentially. The next record in sequence by position on the file is retrieved and transferred to the specified working storage area.

Applicable parameters by type of file organization for the GETN macro are as follows:

Indexed sequential	fit,wsa,ex,ka
Direct access	fit,wsa,ex
Actual key	fit,wsa,ex,ka

The GETNR macro is applicable only to extended indexed sequential files. This macro causes the next sequential record to be transferred to the working storage area the same as the GETN macro. The difference is that the GETNR macro returns control to the user if the request initiates block transfer to the buffer. The user can continue issuing the GETNR macro until transfer is complete. The file position (FP) field in the FIT is set to 20_8 (EOR) when transfer of the record is complete. While intermediate reads are being performed (index blocks or MIP index file blocks), the FP field is set to 0.

Unnecessary GETNR requests can be avoided by monitoring the status of the input/output processing. The busy FET address (BZF) field in the FIT contains the address of the input/output status word. When the low order bit of the status word is set to 1, input/output processing is complete and a GETNR macro will start a new block read or return a record to the working storage area. If the low order bit is set to 0, a GETNR macro immediately returns control to the user.

OPENM MACRO

Before a file can be read or written, the file must be made available by the OPENM macro. Macros that affect the FIT (FILE, STORE, FETCH, and SETFIT) can be executed before the file is opened. Any file manipulation macro, however, is valid only after the file has been opened. Error procedures are initiated if attempts are made to access an unopened file. The format of the OPENM macro is shown in figure 5-5.

Applicable parameters by type of file organization for the OPENM macro are as follows:

Indexed sequential	fit,pd,of
Direct access	fit,pd
Actual key	fit,pd

The OPENM macro prepares a file for processing by creating and linking all required system tables for a file and by translating user-supplied parameters into appropriate values in the relevant tables. When the OPENM macro is executed, the following events occur:

FILE control statement processing occurs unless it has been suppressed by previous execution of the SETFIT

OPENM	fit,pd,of		
fit	Address of the FIT.		
pd	Type of processing:		
		INPUT	File is opened for read only (default).
		OUTPUT	File is opened for write only.
		I-O	File is opened for read and write.
		NEW	A new file is being created; sets the PD field to OUTPUT and the ON field to NEW.
of	Open flag; file positioning at open time:		
		R	File is rewound before any other open procedures are performed (default).
		E	File is positioned immediately before the end-of-information.

Only the fit parameter can be specified as a register.

Figure 5-5. OPENM Macro Format

macro. The PDF field in the FIT is set by the SETFIT macro to inhibit reprocessing of the FILE control statement. The PDF field is cleared by the OPENM macro.

The FIT is checked for logical consistency; depending on the file organization, additional checks are made for required fields and defaults are supplied where needed.

Buffer parameters are processed.

If no error has been detected, the open/close (OC) flag in the FIT is set to open and control transfers to the user.

Complete open processing occurs when the first OPENM macro in a job step is issued. If a file is closed and then reopened, FIT verification and FILE control statement processing are not repeated if the close flag (CF) field in the FIT is set to R or N.

Any error detected during open processing sets the error status (ES) field in the FIT. If a user error routine is specified by the error exit (EX) field in the FIT, control is transferred to the routine. If the user routine corrects the condition that caused the error and executes another OPENM macro, processing of the file can continue; otherwise, the open/close (OC) field in the FIT indicates the file is not open (set to 0) and further file access is prohibited.

Conditions investigated during FIT consistency checks are listed in table 5-1. Buffer fields are also investigated. The settings of the first word address of the buffer (FWB) field and buffer size (BFS) field determine the method of buffer allocation. If the FWB field is zero, the Common Memory Manager (CMM) must be present or an error occurs. For an extended indexed sequential file, the buffer pool limit is increased by the default amount if the BFS field is also zero; otherwise, it is increased by the amount specified by the BFS field.

When the FWB field is not zero, an error occurs if the BFS field is zero. If the BFS field is also nonzero, the specified buffer space is partitioned into table areas for AAM, blocks for the data file, and (if needed) blocks for the MIP index file. A minimum of two blocks must be allocated for each file or CMM must be present; otherwise, an error occurs. The buffer pool amount must be increased to accommodate two blocks per file.

Data compression can be established for an extended indexed sequential file at any time it is opened. Once data compression is selected, it must be specified for the life of the file. The compression routine address (CPA) and decompression routine address (DCA) fields in the FIT point to the routines to be used for data compression. These fields can originally be specified when the file is created or at any subsequent time the file is opened. Whenever the file is opened after that time, the routine addresses must be supplied in the CPA and DCA fields. Refer to appendix H, Data Compression, for more detailed information.

TABLE 5-1. FIT CONSISTENCY CHECKS

Condition	Action
RT=D, LL=0	Error
RT=T, and CL, HL, or TL=0	Error
RT=Z, FL=0	Error
RT=F, FL=0	Error
RT=T, HL not greater than CL+CP	Error
MRL, MBL=0, BT=K, E	Error

The timing of the setting of the parameters for the processing of each file organization in relation to the OPENM macro is important. These parameters differ for each file organization. The requirements for specific parameters are discussed under open processing for each file organization; refer to section 4, File Processing. The following shows the possible relationships between the OPENM macro and the FIT parameters:

For file creation, certain parameters must be set before executing the OPENM macro; otherwise, a fatal error occurs. If these parameters are specified for an existing file, the new values are ignored without comment.

Certain parameters must be selected before the file is created if the option is to be used during the life of the file.

Certain parameters are optional at file creation. If these parameters are not specified, default values are used. Values specified after file creation are ignored.

Certain parameters must be set prior to open time; otherwise, default values are assumed without comment. These parameters are effective only until another OPENM macro is executed.

Certain parameters need not be set until they are required by file processing commands. Once set, these parameters remain in effect until changed.

Certain parameters have no default and must be set prior to use by a file processing command; otherwise, a fatal error occurs.

PUT MACRO

The PUT macro transfers data from the working storage area to a file. The file must be open for output or input/output. The format of the PUT macro is shown in figure 5-6.

Applicable parameters by type of file organization for the PUT macro are as follows:

Initial indexed sequential	fit,wsa,rl,ex,ka,kp,pos
Extended indexed sequential	fit,wsa,rl,ex,ka,kp
Direct access	fit,wsa,rl,ex
Actual key	fit,wsa,rl,ex,ka

Any errors detected during PUT macro execution cause transfer to the error routine if one is specified. If the error is excess or insufficient data, no data has been transferred; for other errors, the data is unreliable.

PUT fit,wsa,rl,ex,ka,kp,pos

fit	Address of the FIT.
wsa	Address of the working storage area from which the user record is transferred.
rl	Number of characters to be written.
ex	Address of the error routine.
ka	Address of the primary key for the record to be written.
kp	Beginning character position of the primary key.
pos	Relative position of a record for duplicate key processing; applicable only to initial indexed sequential files.
	P Before last record referenced
	N Following last record referenced

Parameters can be specified as registers; if parameters are not specified, values in appropriate FIT fields are used.

Figure 5-6. PUT Macro Format

The length of a record being written is determined by the record length (RL) field in the FIT. For U, S, and W type records, the RL field can be set by the rl parameter in the PUT macro. For F, Z, R, T, and D type records, AAM uses certain FIT fields and the content of the record in the working storage area to determine record length for the RL field; the value of the RL field is determined as follows:

F type records

Record length is taken from the FL field in the FIT.

Z type records

Record length is taken from the RL field in the FIT or from the FL field if the RL field is set to zero. The end of the record is determined by searching backward from the character position specified by the value of the RL or FL field and removing full words of blanks.

R type records

Record length is determined by scanning the record in the working storage area for the terminating record mark character, which is specified by the record mark (RMK) field in the FIT. An error occurs if the record mark is not found within the maximum record length.

T type records

Decimal count is extracted from the record and used to calculate the record length. Length and location of the count field in the record (CL and CP fields), length of the header (HL field), and length of the trailers (TL field) are obtained from the FIT.

D type records

Decimal character record length is extracted from the record. Length and location of the character count field in the record (LL and LP fields) are obtained from the FIT.

In all preceding cases, the length of the record transferred is stored in the RL field in the FIT at the end of the PUT macro operation.

REPLACE MACRO

An existing record in a file is replaced by a record from the working storage area when the REPLACE macro is executed. The new record can be smaller or larger than the original record; however, record length cannot exceed the size specified by the maximum record length (MRL) field in the FIT. The format of the REPLACE macro is shown in figure 5-7.

Applicable parameters by type of file organization for the REPLACE macro are as follows:

Initial indexed sequential	fit,wsa,rl,ex,ka,kp,pos
Extended indexed sequential	fit,wsa,rl,ex,ka,kp
Direct access	fit,wsa,rl,ex
Actual key	fit,wsa,rl,ex,ka

Replacement records need not be the same size as the record being replaced except for a direct access file being processed sequentially. A larger replacement record in a direct access file can cause overflow of records, which leaves the sequential position undefined. If the requested record is not found, a trivial error results and the request is ignored. For initial indexed sequential files with duplicate primary keys, a trivial error occurs if the pos parameter is set to C and the key does not equal the key of the current record; the request is ignored.

REPLACE fit,wsa,rl,ex,ka,kp,pos

fit	Address of the FIT.
wsa	Address of the working storage area with the new record.
rl	Length (in characters) of the new record.
ex	Address of the error routine.
ka	Address of the primary key for the record to be replaced.
kp	Beginning character position of the primary key.
pos	Duplicate key position; can be C (current record) or omitted (first record in the duplicate key set). Applies only when duplicate key processing is allowed for initial indexed sequential files.

Parameters can be specified as registers.

Figure 5-7. REPLACE Macro Format

REWINDM MACRO

The REWINDM macro positions a file to beginning-of-information, which is the beginning of the first data record in the file. The file must be open when the macro is issued. A GETN macro issued immediately after the REWINDM macro returns the first record. The format of the REWINDM macro is shown in figure 5-8.

REWINDM fit	
fit	Address of the FIT or register containing the address.

Figure 5-8. REWINDM Macro Format

SEEK MACRO

Program execution time can be shortened through the use of the SEEK macro, which allows overlapping of central memory processing and input/output activity. The SEEK macro initiates block transfer to the buffer; it does not return a record to the user. The user can then continue processing while the transfer occurs. The format of the SEEK macro is shown in figure 5-9.

SEEK fit,ex,ka,kp,mkl	
fit	Address of the FIT.
ex	Address of the error routine.
ka	Address of the key for the desired record.
kp	Beginning character position of the key.
mkl	Major key length in characters.
Parameters can be specified as registers. If the ex, ka, kp, and mkl parameters are not specified, values in appropriate FIT fields are used.	

Figure 5-9. SEEK Macro Format

Applicable parameters by type of file organization for the SEEK macro are as follows:

Indexed sequential	fit,ex,ka,kp,mkl
Direct access	fit,ex,ka,kp
Actual key	fit,ex,ka

When the SEEK macro is executed, control returns to the user program once a read is initiated. The user program must monitor the file position (FP) field in the FIT to determine when the requested data block is in the buffer and ready to be accessed. The FP field is set to zero if the transfer of an index block has been initiated; it is set to 20_8 (EOR) if a data or home block is being transferred.

For an extended indexed sequential file, the user can also monitor the input/output request to avoid issuing SEEK macros with the same key, which would return immediately because the file was busy. The busy FET address (BZF) field in the FIT is set by AAM and points to an input/output status word. When the low order bit of the status word is set, the current SEEK macro input/output is complete and another operation can be profitably issued for the file.

Normally, the SEEK macro is followed by a macro such as GET or DELETE accessing the record referenced by the SEEK macro. An operation on some other record not already in the buffer can negate the action of the SEEK macro by writing over the data transferred by it. The record is not moved into the working storage area until a GET macro is executed. If a call is made before the seek operation is complete, processing continues reading blocks from the point where the SEEK calls were discontinued.

SKIP MACRO

The SKIP macro repositions an indexed sequential or actual key file in a forward or backward direction a specified number of logical records. It does not return a record to the working storage area. Only small skips are recommended because each record must be read and counted for proper positioning. The format of the SKIP macro is shown in figure 5-10.

SKIPdL fit,count	
d	Direction of skip:
	F Forward
	B Backward
fit	Address of the FIT.
count	Number of logical records to be skipped. A null parameter results in a zero count.
The fit and count parameters can be specified as registers.	

Figure 5-10. SKIP Macro Format

When the SKIP macro is executed, user parameters are checked, records in the file are read, the file is positioned according to the number of records to be skipped, and control returns to the user. A negative skip count is not allowed; the request is ignored and an error is issued. If the skip operation encounters end-of-information or beginning-of-information before the skip count is exhausted, control is transferred to the end-of-data routine with the appropriate file position set.

START MACRO

The START macro positions an indexed sequential file or an alternate key index file to a record that meets a specific condition; the record is not transferred to the working storage area. The file is positioned in the same manner as for a GET macro. The format of the START macro is shown in figure 5-11.

The file is positioned according to the key relation (REL) field in the FIT and the current value at the key address (KA) location. The REL field specifies the desired relation between the value at location KA and the key of the record at which the file is to be positioned. Relations that can be specified are EQ (equal to), GT (greater than), and GE (greather than or equal to). The file is positioned at the beginning of the record that satisfies the relation. If the mkl parameter is specified, the file is positioned relative to the major key specified for an indexed sequential symbolic key.

START fit,ex,ka,kp,mkl

fit	Address of the FIT.
ex	Address of the error routine.
ka	Address of the key for positioning the file.
kp	Beginning character position of the key.
mkl	Major key length in characters.

Parameters can be specified as registers. If the ex, ka, kp, and mkl parameters are not specified, values in appropriate FIT fields are used.

Figure 5-11. START Macro Format

All AAM files have a primary key associated with each record to provide random access to the file. In addition, alternate keys can be defined for records in an AAM file. Alternate keys provide the means to access records by more than one field in a record.

Primary key values must be unique within the file. Alternate keys, which can overlap each other and the primary key, need not have values unique to the record or to the file. Alternate keys must be contained within the minimum record size.

The original data file structure is not affected by alternate key processing. The Multiple-Index Processor (MIP) creates an index file on the creation run for a multiple-index file. On subsequent runs, the index file is updated as necessary when the data file is updated. The index file must be made available to the updating program.

Two Multiple-Index Processors are supported by AAM. Initial MIP processes initial indexed sequential, actual key, and direct access files; extended MIP processes extended indexed sequential files.

For existing AAM files, two utilities are available to assist in creating the index file for alternate key processng: the IXGEN utility for initial MIP and the MIPGEN utility for extended MIP. Refer to section 7, Utilities, for descriptions of the IXGEN and MIPGEN utilities.

INDEX FILE

The index file is created and updated automatically by MIP. It is identified by the index file name (XN) field in the FIT. The index file, which is defined when the file is created, must be made available whenever the data file is updated or is accessed by alternate key. Alternate keys are defined by the user on the creation run.

STORAGE STRUCTURE

The index file contains an index for each alternate key defined for the data file. Within an index, each alternate key value is associated with a keylist of the primary keys for records containing that value.

Each alternate key index is ordered by alternate key value. The ordering of the primary key list for a given index is controlled by the user through a parameter that can be specified when the alternate key is defined by the RMKDEF macro or directive. The ordering of the list is as follows:

If the parameter is omitted or U is specified, each value of the alternate key must be unique. The primary key list for each alternate key value consists of only one primary key value.

If F is specified for the parameter, the ordering of primary key values is first-in first-out. The primary keys are stored in the order in which their corresponding records are created.

If I is specified for the parameter, the primary keys are stored in ascending sequence of primary key values. Numeric keys are in numeric order; symbolic keys are in collating sequence order.

Block Size, Initial MIP

The size of the index file blocks can also be specified by a parameter in the RMKDEF macro or IXGEN directive when the data file is created. The parameter is specified in the macro or directive that defines the primary key. The index file block size must always be specified as an integral number of PRUs. A block size of 2 to 8 PRUs is recommended; results are indeterminate if the block size exceeds 8 PRUs.

Block Size, Extended MIP

The size of the index file blocks is determined when the data file is created. The index block size (XBS) field in the data file FIT specifies the number of characters in a block. A value specified for the XBS field is rounded upward if necessary to the nearest multiple of 640 characters minus 20. The default index file block size is the data file block size.

ALTERNATE KEY SPECIFICATION

Alternate keys are defined when the data file is created. A record can then be accessed by the primary key or by any alternate key defined for the file. For existing files, the IXGEN or MIPGEN utility can be used to define alternate keys and create the index file. (Refer to section 7, Utilities.)

RMKDEF Macro, Initial MIP

On a file creation run, the RMKDEF macro is used to describe the primary key or an alternate key field. The macro must be used once for the primary key and once for each alternate key field in the record; the primary key must be specified first. The RMKDEF macros must be executed after the OPENM macro and before the first PUT macro. The format of the RMKDEF macro for initial MIP is shown in figure 6-1.

Used together, the kg and kc parameters refer to an alternate key that is a repeating group. For example, a repeating group is described in COBOL by an OCCURS n TIMES clause. When the kg parameter is used alone, it refers to the index file block size. The kg parameter should be used alone only when the primary key is being defined. Alternate key fields can overlap in a record; for example, first name, last name, and whole name can all be defined as alternate keys.

RMKDEF Macro, Extended MIP

The RMKDEF macro is used to describe an alternate key field on a file creation run. The macro must be used once for each alternate key field in the record. The RMKDEF macros must be executed after the OPENM macro and before the first PUT macro. An RMKDEF macro that defines the primary key is ignored without comment. The format of the RMKDEF macro is shown in figure 6-2.

The kg and kc parameters refer to an alternate key that is a repeating group. For example, a repeating group is described in COBOL by an OCCURS n TIMES clause. If the same alternate key value occurs more than once in a data record, the primary key is entered in the index only once for that value; therefore, a primary key associated with an alternate key value indicates that the value occurs at least once in the record. Alternate key fields can overlap in a record; for example, first name, last name, and whole name can all be defined as alternate keys.

The nl, ie, and ch parameters are used to define sparse keys. These are alternate keys for which only certain values are of interest to the user. A sparse key is defined by specifying null suppression or sparse control characters.

Null suppression is specified by the nl parameter. The primary key for a record that has a null alternate key value is not included in the alternate key index. A null value is all spaces for a symbolic key or all zeros for a signed binary key.

RMKDEF fit,kw,kp,kl,ki,kf,ks,kg,kc

fit — Address of the FIT for the data file.

kw — Word of the record where the key starts, counting from zero; default is zero.

kp — Beginning character position of the key (0 to 9).

kl — Key length, in characters (1 to 255); default is zero.

ki — 0 (reserved).

kf — Key type:

- 0 Symbolic
- 1 Signed binary
- 2 Unsigned binary

ks — Substructure for each primary key list in the index:

- U Unique (default)
- I Indexed sequential
- F First-in first-out

kg — For a repeating group, number of characters in the group where the key resides. For the primary key definition, the size in PRUs of an index file block.

kc — For a repeating group, number of occurrences; zero if the group is defined in an OCCURS ... DEPENDING ON clause.

Figure 6-1. RMKDEF Macro Format, Initial MIP

The ie and ch parameters are used when indexing of alternate key values is to be controlled by a sparse control character. The one-character field containing the sparse control character must be in the fixed-length portion of the record. The ie parameter specifies whether to include or exclude the alternate key values for records that contain a

RMKDEF fit,kw,kp,kl,ki,kf,ks,kg,kc,nl,ie,ch

fit — Address of the FIT for the data file.

kw — Word of the record where the key starts, counting from zero; default is zero.

kp — Beginning character position of the key:

- 0 to 9 for symbolic key
- 0 for signed binary key

kl — Key length, in characters:

- 1 to 255 for symbolic key
- 10 for signed binary key

ki — 0 (reserved).

kf — Key type:

- 0 Symbolic
- 1 Signed binary
- 2 Uncollated symbolic

ks — Substructure for each primary key list in the index:

- U Unique (default)
- I Indexed sequential
- F First-in first-out

kg — For a repeating group, number of characters in the group where the key resides.

kc — For a repeating group, number of occurrences; zero if the group is defined in an OCCURS ... DEPENDING ON clause.

nl — Null suppression:

- 0 Null values are recorded (default)
- N Null values are not recorded

A null value is all spaces (symbolic key) or all zeros (signed binary key).

ie — Include/exclude sparse control character:

- E Exclude alternate key value if the record contains a sparse control character (default)
- I Include alternate key value if the record contains a sparse control character

ch — Characters that qualify as sparse control characters; up to 36 letters and digits can be specified as a character string.

Figure 6-2. RMKDEF Macro Format, Extended MIP

sparse control character. The ch parameter specifies the sparse control characters applicable to the alternate key being defined.

The sparse control character field is identified by an RMKDEF macro that must appear before the macro defining the alternate key and its sparse control characters. This macro is specified in the following format:

```
RMKDEF fit,kw,kp,0
```

The kw and kp parameters specify the position of the sparse control character. The zero kl parameter indicates that the field is a sparse control character field.

APPLICABLE FIT FIELDS

Several FIT fields are applicable to multiple-index file processing. These fields and their respective uses are as follows:

- FP — File position; when the index file is being accessed, 10_8 indicates the end of primary keys associated with a given alternate key value. For extended MIP, 100_8 indicates the end of the alternate key list.
- FPB — File position bit; when the index file is being accessed, 1 indicates the end of an index associated with a given alternate key position (initial MIP only).
- KL — Key length; number of characters in a primary or alternate key.
- KNE — Key not equal; 1 indicates the key in process is not the same key specified by the KA field. For extended MIP, KNE is set only after an operation for which a GE relation was specified.
- KR — Key value repeat count; when the index file is being accessed, KR indicates the number of occurrences in the record of the key value at location KA (initial MIP only).
- MRL — Maximum record length; when the primary key lists are being retrieved, MRL indicates the length of the working storage area.
- NDX — Index flag; 1 indicates an index only operation; 0 indicates a data record operation.
- PKA — Primary key address; when accessing records by alternate key, the primary key for a record is returned to the specified address (extended MIP only).
- RC — Record count; number of records containing the value of the key at location KA.
- REL — Key relation; relation of the key value at location KA to the key at which the file is positioned; can be EQ, GT, or GE; for initial MIP, LT and LE can also be used.
- RL — Current record length (initial MIP only).
- RKP — Relative key position; character position of a primary or alternate key within the word specified by the RKW field.
- RKW — Relative key word; word in which a primary or alternate key begins.
- XBS — Index file block size; number of characters in an index file block (extended MIP only).
- XN — Index file name; logical file name of the index file.

ALTERNATE KEY PROCESSING

Defining alternate keys for a file allows the user to access records by fields other than the primary key. Two files are involved with alternate key processing. The data file contains records that have unique primary keys. The index file contains alternate key values and their associated primary keys. Both files must be made available to the program. Reading by alternate key can be random or sequential.

ALTERNATE KEY ACCESS

To access a data record by an alternate key, the alternate key position must first be established in the FIT. The relative key word (RKW), relative key position (RKP), and key length (KL) fields must be set for the desired alternate key. These three fields are set for the primary key by open processing; thereafter, the user is responsible for setting them when changing access from primary to alternate key or from one alternate key to another. The index flag (NDX) field in the FIT must be set to zero to access a data record.

The alternate key defined by the RMKDEF macro refers to a position within a record. The GET macro is used to retrieve a record with a specific value in the alternate key position. When the GET macro is executed, the RKW, RKP, and KL fields in the FIT define the alternate key position in the record. The ka, kp, and mkl macro parameters establish the alternate key location that contains the value for the record to be retrieved. The first primary key associated with the alternate key value determines the record returned to the working storage area. The format of the GET macro is:

```
GET    fit,wsa,0,ex,ka,kp,mkl
```

When the GET macro is executed, a record is returned to the location specified by the wsa parameter, the index file is positioned, and the following FIT fields are set:

- KR — Key value repeat count; number of occurrences of the key value in the record (initial MIP only).
- PKA — Primary key address; address of location that contains the primary key of the record retrieved (extended MIP only).
- RC — Record count; number of records that contain the alternate key value.
- RL — Record length; number of characters in the record returned to the working storage area.

The setting of the key relation (REL) field in the FIT determines which record is retrieved as follows:

If the field is set to EQ, the index file is positioned at the alternate key value equal to the value at location KA. The record with the first primary key associated with the alternate key value is returned. If

an equal value is not found, the index file is positioned at the next greater value, the error status (ES) field is set to 506_8, and any specified error exit is taken.

If the field is set to GT, the index file is positioned at the first alternate key value greater than the value at location KA. The record with the first primary key associated with the alternate key value is returned.

If the field is set to GE, the index file is positioned at the first alternate key value greater than or equal to the value at location KA. The record with the first primary key associated with the alternate key value is returned.

Once a GET macro has been executed to establish an index file position, the record for the next primary key in the index can be accessed by the GETN macro. When the index file is positioned past the last primary key in the index, no record is returned to the working storage area, the file position (FP) field is set to EOI, and any specified end-of-data exit is taken. An informative error message is written on the error file ZZZZZEG.

When execution of the GETN macro encounters a new alternate key value, that value is moved to program location KA. Retrieval of the record for the last primary key associated with an alternate key value causes the file position (FP) field in the FIT to be set to 10_8 to indicate the end of a keylist (EOK). The format of the GETN macro is:

```
GETN    fit,wsa,ex,ka
```

Execution of the GETN macro returns a record to the working storage area. For extended MIP, the primary key for the record is moved to the program location indicated by the primary key address (PKA) field in the FIT.

FILE UPDATING

Updating a multiple-index file is basically the same as updating any other AAM file. The only difference is that the logical file name of the alternate key index file must be specified in the FILE control statement by the XN parameter. The index file is automatically updated when a data file update affects the index file.

The PUT and REPLACE macros are used to write and rewrite records. For initial MIP and for extended MIP when the primary key is embedded, it is not necessary to set FIT fields for the primary key; that is, the RKW, RKP, KL, KA, and KP fields do not have to be set. The KA and KP fields must be set for nonembedded keys. The position of the primary key in the record is constant for the file and the address in the working storage area (WSA) field is the address of the record to be written or rewritten.

The DELETE macro is used to delete a record from the file. The RKW, RKP, and KL fields in the FIT do not have to be set; however, the key address (KA) and key position (KP) fields must be set for the primary key because the WSA field is not required for the DELETE macro.

The index file position and the RKW, RKP, and KL fields are not changed by execution of the PUT, REPLACE, or DELETE macro. A series of GETN macro requests can be interrupted by update requests without losing alternate key sequence.

READ-ONLY PROCESSING

Read-only processing is applicable to initial MIP only. Existing multiple-index files can be read, but not updated, in a smaller field length by not loading the routines used for writing multiple-index files. Both random and sequential processing are possible with the read-only capability.

When the read-only mode is selected, the file must be opened for input. If another AAM file is being processed for input/output in the same job step, the read-only mode must not be selected; if it is selected, an error occurs. The file to be read must not be an empty file. Only the following macros can be issued for the file: OPENM, GET, GETN, SEEK, SKIP, SKIPFL, REWINDM, and CLOSEM. All AAM file updating operations are trapped and trivial error 513 (REQUIRED ROUTINES NOT LOADED - RM$MEXB/-RM$MFSQ) is issued.

The read-only capability requires the following LDSET control statement (or LDREQ macro from a COMPASS program executing through a terminal):

```
LDSET(OMIT=$RM$$MEXB$/$RM$$MFSQ)
```

If static loading is being used, the following additional LDSET control statement is required:

```
LDSET(SUBST=$RM$$MIP$-$RM$$MIP2$)
```

Refer to appendix E for a discussion of static and dynamic loading. Under the NOS operating system, libraries must have been generated with NX=1 before SUBST is used.

INDEX FILE POSITIONING

The alternate key index file is positioned when a GET macro accesses a record by alternate key. The index file can also be positioned without returning a record. The START, SKIP, and REWINDM macros change the position of the index file.

START Macro

The START macro positions the index file to the first primary key for a given alternate key value. The value is at the location specified by the key address (KA) field in the FIT. The format of the START macro is:

```
START       fit,ex,ka,kp,mkl
```

The key relation (REL) field in the FIT determines the positioning of the index file in relation to the value at location KA. The REL field has three possible values:

EQ The index file is positioned at the alternate key value equal to the value at location KA. The default for the REL field is EQ. If an equal key value is not in the index, trivial error 506 results.

GT The index file is positioned at the first alternate key value greater than the value at location KA.

GE The index file is positioned at the first alternate key value greater than or equal to the value at location KA. If an equal key value is not in the index, the key not equal (KNE) field in the FIT is set to 1.

After the START macro is executed, the record count (RC) field in the FIT is set to the number of primary keys for the alternate key at which the index file is positioned.

Other Positioning Macros

In addition to the START and GET macros, the index file position is changed by the SKIP and REWINDM macros. When a change is made from one alternate key index to another, the index position is established as follows:

Initial MIP

Index position is reset automatically to the beginning of the index.

Extended MIP

Index position must be established by a REWINDM, GET, or START macro.

The SKIP macro is used to skip forward a number of primary keys from the current position. The format of the SKIP macro is:

```
SKIP        fit,n
```

The index file is positioned at the first primary key in the alternate key index by the REWINDM macro. The format of the REWINDM macro is:

```
REWINDM     fit
```

For initial MIP, execution of the REWINDM macro sets the record count (RC) field in the FIT to the number of primary keys belonging to the first alternate key value.

INDEX FILE PROCESSING

The alternate key index file can be accessed to retrieve information related to the alternate keys. Primary key lists or counts of primary keys for either a single alternate key value or a range of values can be retrieved. Obtaining this information from the index file has no effect on the data file.

In order to access the index file, the index flag (NDX) field in the data file FIT must be set to YES. If the OPENM macro is executed with NDX set to YES, only the index file is opened for processing. The index file must be an existing file at open time. If the NDX field is set to YES when the file is opened, it cannot be reset to NO until after the file has been closed.

MACRO PROCESSING

The index file is accessed through execution of various macros. Only those macros described in the following paragraphs can be used with the index file.

The OPENM macro and the CLOSEM macro open and close the index file. Execution of these macros does not affect the data file.

The REWINDM macro positions the index file at the beginning of the alternate key index from which information is to be retrieved. The alternate key is determined by the relative key word (RKW), relative key position (RKP), and key length (KL) fields in the data file FIT. The file is positioned at the first value for the designated alternate key.

The index file can be positioned at a specific value of an alternate key through execution of the START macro. The RKW, RKP, and KL fields in the FIT specify the alternate key for file positioning. The alternate key value at the location indicated by the key address (KA) field in the FIT and the condition designated by the key relation (REL) field determine the positioning at a specific value within the alternate key index. When the relational condition is EQ, the file is positioned at the alternate key value equal to the value at location KA; if an equal value cannot be found in the index, the file is positioned at the next higher value. For the GT relational condition, the file is positioned at the next higher value than the value at location KA. The GE relational condition causes the file to be positioned at a value equal to or greater than the value at location KA.

The GET macro is used to retrieve the primary keys for an alternate key value. The alternate key to be accessed is determined by the RKW, RKP, and KL fields in the FIT. The alternate key value at location KA and the condition specified in the REL field determine the positioning of the index file. Execution of the GET macro positions the index file at the desired alternate key value and returns as many of its associated primary key values as the working storage area can contain.

The GETN macro can be executed after the GET macro to retrieve additional primary key values associated with the alternate key value. It can also be executed after a REWINDM, START, or SKIPFL macro to begin returning primary key values from the position established by the previous macro. Primary keys are returned to the working storage area until one of the following conditions occurs:

The working storage area is full.

The end of the list of alternate key values is reached (end-of-information).

The index file is positioned at the beginning of a primary key list for an alternate key that is greater than the key at location KA when the value of the REL field in the FIT is GT or GE, or the index file is positioned at the beginning of a primary key list for an alternate key that is equal to the key at location KA when the value of the REL field is GE or EQ.

The key address (KA) field in the FIT must be set for the GETN macro when the index file is being accessed. If primary key list retrieval is to be terminated according to a key value, the KA field must point to the location containing the key value. If the KA field is set to 0, primary key list retrieval terminates only if the working storage area is filled or if end-of-information is reached. This is the same as if the key value at location KA is greater than any possible value for the alternate key.

The SKIPFL macro is used to count the number of primary key values for one or more alternate key values; the primary key values are not returned to the working storage area. The counting can be terminated by a key value in the same manner as the GETN macro. Counting can also be specified for a number of alternate key values or to end-of-information.

FIT FIELDS FOR INDEX FILE PROCESSING

Index file processing involves user setting of several fields in the FIT. In addition, AAM sets certain FIT fields during macro execution. The following FIT fields can be set by the user:

KA — Key address; location of the user-supplied key value for START and GET macros and for GETN and SKIPFL macros that use a key.

KL — Key length; number of characters in the alternate key being accessed.

KP — Key position; position of user-supplied key value at location KA.

MKL — Major key length; number of characters, which is less than the full length of the alternate key, in the user-supplied key value; can be used with indexed sequential symbolic keys only.

MRL — Maximum record length; length of the working storage area in characters; should be a multiple of 10 characters because each primary key value returned begins on a new word boundary.

NDX — Index flag; must be set to 1 for index file access.

REL — Key relation; indicates the relation to be satisfied between the user-supplied key value and the index file key value; possible relations are EQ, GE, and GT; for initial MIP, LE and LT can also be used.

RKP — Relative key position; beginning character position of the alternate key within the word specified by the RKW field.

RKW — Relative key word; word in which the alternate key being accessed begins.

WSA — Working storage area; location into which primary key lists are returned.

The following FIT fields are set by AAM during execution of the START, GET, GETN, and SKIPFL macros:

FP — File position; set to indicate the position of the index file when control returns to the user:

- 0 — Middle of primary key list
- 10_8 — End of primary key list
- 100_8 — End-of-information

KNE — Key not equal; for an operation involving a key, indicates whether or not the current alternate key value matches the user-supplied key value:

- 0 — Equal key values
- 1 — Higher user-supplied key value or end-of-information

MKL — Major key length; reset to 0 after a user-supplied value has been noted.

PTL — Primary key total; number of primary key values transferred to the working storage area during execution of the GET or GETN macro.

RC — Record count; for a START or GET macro, the number of primary keys associated with the desired alternate key value; if the KNE field is set to 1, the number of primary keys associated with the first alternate key value greater than the given one.

RL — Record length; set by the GET, START, SKIPFL, and GETN macros as follows:

- GET — Set to the value in the PTL field.
- START — Set to zero
- SKIPFL — Set to the number of primary key values that have been skipped.
- GETN — Increased by the number of primary key values transferred to the working storage area; cleared on entry only if the file position from the last operation was end-of-keylist (EOK).

COUNT RETRIEVAL

The primary key values associated with a given alternate key value are counted by executing the START macro. The RKP, RKW, and KL fields in the FIT must be set to identify the alternate key. Because a specific alternate key value is involved, the major key length (MKL) field is set to 0 for full length key comparison and the key relation (REL) field is set to equal (EQ). The format of the START macro is as follows:

```
START    fit,ex,ka,kp
```

The fit parameter specifies the address of the data file FIT with which the index file is associated. The file is positioned at the alternate key value that is equal to the value at the location specified by the ka parameter; the record count (RC) field in the FIT contains the number of primary keys associated with the alternate key value. The key not equal (KNE) field is set to zero to indicate that the desired value has been found.

If an equal alternate key value cannot be found, the file is positioned at the next higher key value and the RC field contains the number of primary keys associated with that alternate key value. The KNE field is set to 1 to indicate that the desired key value does not exist in the file.

The file position (FP) field in the FIT is set during execution of the START macro. It is set to 10_8 if the index file is positioned at an alternate key value. If, however, the user-supplied key value is greater than all existing values for the alternate key, the FP field is set to 100_8.

RANGE COUNT RETRIEVAL

The number of primary keys associated with a range of consecutive alternate key values can be determined by executing a REWINDM or START macro and then a SKIPFL macro. The beginning and end of the range can be specified in various ways.

The beginning of the range indicates the first alternate key value for which primary keys are to be counted. The key value is specified as one of the following:

The first alternate key value in the file; execution of the REWINDM macro positions the index file to this point.

The first alternate key value that is not less than a specified value; the REL field in the FIT is set to GE and the START macro is executed to reach this position in the index file.

The first alternate key value that is greater than a specified value; the REL field in the FIT is set to GT and the START macro is executed to reach this position in the index file.

If a major key is specified for the START macro, only the number of characters in the major key are used for comparison. If the REL field is set to EQ or GE, the file is positioned at the first alternate key value with leading characters that match the major key. If no such key exists, the file is positioned at the next logical alternate key value. If the REL field is set to GT, the file is positioned at the first alternate key value with leading characters greater than the major key value.

The end of the range, which is the last alternate key value to be included in the range count, is specified by setting various FIT fields before executing the SKIPFL macro. The last key value is determined as follows:

If the key address (KA) field is set to 0, the last alternate key value in the index is the end of the range.

If the KA field points to a location that contains an alternate key value and the key relation (REL) field is set to GT, all key values not exceeding the value at location KA are included in the count.

If the KA field points to a location that contains an alternate key value and the REL field is set to GE, all key values less than the value at location KA are included in the count.

After the SKIPFL macro is executed, the RL field in the FIT contains the number of primary key values for all the alternate key values within the specified range. Unless the file is positioned at end-of-information, it is positioned the same as after execution of the START macro; however, the record count (RC) field in the FIT is undefined.

PRIMARY KEY LIST RETRIEVAL

The list of primary keys for a specific alternate key value can be retrieved by executing the GET macro. The major key length (MKL) field in the FIT should be set to 0 for a full-length alternate key comparison and the key relation (REL) field should be set to EQ for an equal comparison. When the GET macro is executed, the key not equal (KNE) field is set to 0 if the alternate key value is found in the index file or to 1 if it is not found. The format of the GET macro is:

```
GET     fit,wsa,0,ex,ka,kp,mkl
```

Execution of the GET macro causes the primary key values associated with the alternate key value to be transferred to the working storage area. Transfer of primary key values terminates when the last primary key value has been transferred or when the working storage area has been filled. The following FIT fields indicate the status of the primary key list retrieval:

FP — File position; set to 10_8 when all primary keys have been transferred; otherwise, set to 0.

PTL — Primary key total; number of primary keys transferred to the working storage area.

RC — Record count; total number of primary keys associated with the alternate key value.

RL — Record length; same as the PTL field for the GET macro.

If the FP field is set to 10_8, the entire primary key list has been retrieved. In this case, the PTL, RC, and RL fields contain the same value. The index file is positioned at the beginning of the primary key list for the next alternate key value.

The FP field set to 0 indicates that additional primary keys are associated with the alternate key value. The RC field contains a value greater than the PTL and RL fields, which contain equal values. The remaining primary keys can be retrieved by executing the GETN macro after making the working storage area available for use again and after setting the REL field to GT. Primary keys are transferred until the end of the list is reached or the working storage area is filled. Additional GETN macros can be executed to complete transfer of the primary key list. The FP field in the FIT indicates that the entire list has been transferred when it is set to 10_8; the index file is then positioned at the first primary key for the next sequential alternate key value.

The normal purpose of primary key list retrieval is to determine the primary key values for a specific alternate key value. If the major key length (MKL) field in the FIT is set to a value other than 0, more than one alternate key value could satisfy the condition of the REL field. The GETN macro execution would then continue until the index file is positioned at an alternate key value that does not satisfy the condition specified by the REL field.

Whenever a GETN macro is executed, the RL field is incremented by the number of primary keys transferred to the working storage area; the PTL field indicates the number of primary keys transferred during execution of the macro most recently executed (GET or GETN). The final value in the RL field (when the FP field contains 10_8) should equal the value in the RC field after execution of the GET macro, which should also equal the total of the values in the PTL field after execution of the GET macro and all subsequent GETN macros.

RANGE LIST RETRIEVAL

The primary key lists for a range of consecutive alternate key values can be retrieved through execution of a START macro followed by execution of one or more GETN macros. The beginning of the range of alternate key values is established in the same manner as for the range count retrieval; that is, the REWINDM macro can be used to position the index file to the first alternate key value, or the START macro can be used to position the file to a specific alternate key value.

Once the beginning of the range has been established, the GETN macro is executed to transfer primary keys to the working storage area. To determine when the primary key lists for the range of alternate key values have all been transferred, the file position (FP) field in the FIT must be checked for a value of 10_8 (end-of-keylists) or 100_8 (end-of-information) after execution of the GETN macro.

The end of the range of alternate key values is determined by the setting of certain fields in the FIT:

If the key address (KA) field is set to 0, the end of the range is end-of-information.

If the KA and key position (KP) fields are set to indicate an alternate key value and the key relation (REL) field is set to GT, the end of the range is the alternate key value that is not greater than the one indicated by the KA and KP fields.

If the KA and KP fields are set to indicate an alternate key value and the REL field is set to GE, the end of the range is the last alternate key value that is less than the one indicated by the KA and KP fields.

Whenever the GETN macro is executed, the FP field in the FIT should be checked. If it is equal to 10_8 or 100_8, all the desired primary key lists have been retrieved. If FP is equal to 0, however, the working storage area should be made available for retrieval of more primary keys and another GETN macro should be issued.

Several utility routines are provided for use with AAM files. Utilities are available for:

Printing statistics

Estimating the optimal block and buffer sizes for indexed sequential files

Performing key analysis for direct access files

Creating direct access files

Creating an index file for alternate key access to an existing file

The utilities are called by operating system control statements. File dumping and reloading functions are handled by FORM and permanent file utilities.

INITIAL INDEXED SEQUENTIAL FILES

Two utilities are provided for use with initial indexed sequential files only. These utilities print statistics and suggest block and buffer sizes.

SISTAT UTILITY

The SISTAT utility shows statistical information concerning an initial indexed sequential file since creation time. The information is written to the file OUTPUT or to a user-defined file. Current file statistics are obtained by placing the SISTAT control statement after the control statement that causes program execution. The format of the SISTAT control statement is shown in figure 7-1. An example of the statistical information output by the SISTAT utility is shown in figure 7-2.

ESTMATE UTILITY

The ESTMATE utility is an aid to the user in creating an initial indexed sequential file. Suggested values for the size of the buffer, data blocks, and index blocks are output by this utility. Use of the suggested values adds to the efficiency with which an initial indexed sequential file is processed.

A variety of file description parameters are input to the utility by the user. The utility returns to the file OUTPUT suggested index block size, data block size, and the minimum and suggested buffer space requirements. The format of the ESTMATE control statement is shown in figure 7-3.

SISTAT(lfn,sfn)

lfn	Logical file name of an initial indexed sequential file.
sfn	Logical file name of the statistical information file; default is OUTPUT.

Figure 7-1. SISTAT Control Statement Format

```
                    STATISTIC OUTPUT
                    FILE  GENFILE

FILE FORMAT (IN WORDS)
                    INDEX BLOCK SIZE =         63
                    DATA  BLOCK SIZE =        127
                    KEY TYPE =      I OR F
                    KEY SIZE =              2

TOTAL TRANSACTIONS
          NUMBER OF:
                    INSERTS =          400
                    DELETES =            5
                    REPLACES=            1

STORAGE ALLOCATION
          NUMBER OF:
                    INDEX LEVELS =                 1
                    INDEX BLOCKS =                 1
                    EMPTY INDEX BLOCKS =           0
                    DATA RECORDS =               395
                    DATA  BLOCKS =                26
                    EMPTY DATA BLOCKS =            0

NUMBER OF UNUSED ENTRIES IN THE PRIMARY INDEX =            5

TOTAL MASS STORAGE USED BY SIS FILES =       3456
```

Figure 7-2. SISTAT Utility Output

ESTMATE(NR=n,KS=n,MR=n,MI=n)	
NR	File size (approximate number of records).
KS	Key length in characters; must be 10 for floating point and 5 or 10 for integer.
MR	Maximum record size in characters.
MI	Minimum record size in characters.

Figure 7-3. ESTMATE Control Statement Format

The ESTMATE utility expects directives as the next unexecuted record on the file INPUT. The directive format is shown in figure 7-4. Default values are used for any omitted parameters. If fewer than four parameters are given, the last parameter must be followed by a period.

A number of directive statements can be used for the same file to provide alternative buffer and block lengths. When only one directive statement is input to the ESTMATE utility, the parameters can be included in the ESTMATE control statement, thus making a separate directive statement unnecessary. The parameters that would be added to the control statement are as follows:

NL	Number of index levels; 1 through 63.
BF	Blocking factor (number of records per block).
PI	Index block padding percentage; 0 through 99.
PD	Data block padding percentage; 0 through 99.

Default values are used for any omitted parameters.

The deck structure shown in figure 7-5 generates the two estimates shown in figure 7-6. The ESTMATE control statement indicates a file of 100,000 records that range in size from 500 to 1000 characters; the key size is 20 characters. The first directive statement specifies one index level, five records per block, and no index block padding. The second directive statement specifies two index levels, five records per block, and ten percent index block padding.

*nl,bf,pi,pd	
nl	Number of index levels; 1 through 63.
bf	Blocking factor (number of records per block).
pi	Index block padding percentage; 0 through 99.
pd	Data block padding percentage; 0 through 99.

Figure 7-4. ESTMATE Directive Format

```
job statement
ESTMATE(NR=100000,KS=20,MR=1000,MI=500)
7/8/9
*1,5.
*2,5,10.
6/7/8/9
```

Figure 7-5. ESTMATE Utility, Sample Deck Structure

EXTENDED INDEXED SEQUENTIAL FILES

Two utilities are provided for use with extended indexed sequential files only. These utilities print statistics and suggest block and buffer sizes.

FLSTAT UTILITY

The FLSTAT utility shows statistical information concerning an extended indexed sequential file since file creation time. The information is written to the file OUTPUT or to a user-defined file. The format of the FLSTAT control statement is shown in figure 7-7.

The amount of information output by the FLSTAT utility depends on whether or not an installation option is selected. (Refer to the Installation Handbook for details.) Figure 7-8 shows the output generated for a data file and an index file when the installation option is not selected. Figure 7-9 shows the output generated for the same two files when the option is selected.

FLBLOK UTILITY

The FLBLOK utility is an aid to the user in creating an extended indexed sequential file. Appropriate values are suggested for the size of the buffer and the data and index blocks. Use of the suggested values adds to the efficiency with which an extended indexed sequential file can be processed.

A variety of file description parameters are input to the FLBLOK utility by the user. The utility returns suggested block size and minimum and suggested buffer size to the file OUTPUT. The format of the FLBLOK control statement is shown in figure 7-10.

The FLBLOK utility expects directives as the next unexecuted record on the file INPUT. The format of the directive is shown in figure 7-11. Default values are used for any omitted parameters. If fewer than four parameters are specified, the last parameter must be followed by a period.

Multiple directive statements can be used for the same file to provide alternative buffer and block lengths. When only one directive statement is given to the FLBLOK utility, the parameters can be specified in the FLBLOK control statement; a separate directive statement is then unnecessary. The parameters that would be added to the FLBLOK control statement are as follows:

NL	Number of index levels; 1 through 15.
BF	Blocking factor (number of records per block).
PI	Index block padding percentage; 0 through 99.
PD	Data block padding percentage; 0 through 99.

The deck structure shown in figure 7-12 generates the two estimates shown in figure 7-13. The FLBLOK control statement indicates a file of 100,000 records that range in size from 500 to 1000 characters; the key size is 20 characters. The first directive specifies one index level, five records per block, and no index block padding. The second directive specifies two index levels, five records per block, and ten percent index block padding.

```
ESTMATE(NR=100000,KS=20,MR=1000,MI=500)

THE PERCENTAGE OF PADDING IN THE DATA BLOCK WAS NOT  SPECIFIED
                              THE SIS DEFAULT VALUE OF        0 IS ASSUMED

THE PERCENTAGE OF PADDING IN THE INDEX BLOCK WAS NOT SPECIFIED
                              THE SIS DEFAULT VALUE OF        5 IS ASSUMED

*1,5.

                    INDEXED SEQUENTIAL FILE ESTMATE


NUMBER OF RECORDS=    100000              KEY    SIZE=       20 CHARACTERS
MINIMUM RECORD SIZE=        50 WORDS      MAXIMUM RECORD SIZE=       100 WORDS

                    INDEXED SEQUENTIAL FILE ESTMATE

NUMBER OF RECORDS=    100000              KEY    SIZE=       20 CHARACTERS
MINIMUM RECORD SIZE=        50 WORDS      MAXIMUM RECORD SIZE=       100 WORDS

  NUMBER                          INDEX        DATA        MINIMUM     SUGGESTED
 OF INDEX       ACCESS           BLOCK        BLOCK        BUFFER       BUFFER
  LEVELS         MODE            SIZE         SIZE          SIZE         SIZE
                                 (WORDS)      (WORDS)       (WORDS)      (WORDS)

    1          RANDOM           63167          447         63831        64283
    1        SEQUENTIAL                       447           539          988


ESTMATE(NR=100000,KS=20,MR=1000,MI=500)

THE PERCENTAGE OF PADDING IN THE DATA BLOCK WAS NOT  SPECIFIED
                              THE SIS DEFAULT VALUE OF        0 IS ASSUMED

*2,5,10.

                    INDEXED SEQUENTIAL FILE ESTMATE


NUMBER OF RECORDS=    100000              KEY    SIZE=       20 CHARACTERS
MINIMUM RECORD SIZE=        50 WORDS      MAXIMUM RECORD SIZE=       100 WORDS

  NUMBER                          INDEX        DATA        MINIMUM     SUGGESTED
 OF INDEX       ACCESS           BLOCK        BLOCK        BUFFER       BUFFER
  LEVELS         MODE            SIZE         SIZE          SIZE         SIZE
                                 (WORDS)      (WORDS)       (WORDS)      (WORDS)

    2          RANDOM             511          447          1179         2146
    2        SEQUENTIAL                       447           539          988
```

Figure 7-6. ESTMATE Utility Output

FLSTAT(lfn,sfn)	
lfn	Logical file name of an extended index sequential file.
sfn	Logical file name of the statistical information file; default is OUTPUT.

Figure 7-7. FLSTAT Control Statement Format

DIRECT ACCESS FILES

Two utilities are provided for use with direct access files. These utilities analyze the effectiveness of a hashing routine and create a direct access file. Both utilities require the Common Memory Manager (CMM) to be present.

KEY ANALYSIS UTILITY

The key analysis utility tests hashing routines for effectiveness in producing uniform distribution of record keys in a file. A uniform distribution optimizes processing time. The key analysis utility can be called in either of two ways:

The user can read the input file and call the key analysis utility to process the file on a record-by-record basis.

The key analysis utility can be used as an owncode exit from the FORM utility to process the user file on a record-by-record basis.

The same hashing routine can be used for up to five tests varying the number of home blocks for each test. It is also possible to test up to five hashing routines with the same number of home blocks. The number of synonym records produced by each hashing routine is counted and the resulting information written to a file named KEYLIST. The file KEYLIST must be rewound and copied to the file OUTPUT for the results to be printed. Otuput can show synonym records only, standard deviations only, or both. The format of the output from the key analysis utility is shown in figure 7-14.

The key analysis utility is called through a source program or through the FORM utility. The utility expects a KYAN directive as the next unexecuted record on the file INPUT. The format of the KYAN directive is shown in figure 7-15. The directive begins in column 1. All parameters must be declared; no default values are provided.

If a continuation statement is to be used for the first KYAN or subsequent statement, all 80 columns must be filled. A slash (/) in column 80 indicates continuation to a subsequent statement. A maximum of seven statements can be used. Parentheses must enclose the entire parameter list; no embedded blanks are allowed.

Possible error messages that are printed on the user's dayfile are as follows:

NOT ENOUGH FIELD LENGTH. USE nnnnnn.

The run is terminated because the field length cannot accommodate the internal tables.

```
STATISTICS FOR FILE SIPF


ORGANIZATION------- IS
CREATION DATE------ 08/23/77
DATE OF LAST CLOSE- 08/23/77
TIME OF LAST CLOSE- 15.21.14.

FILE IS NOT MIPPED
COLLATION IS STANDARD

PRIMARY KEY INFORMATION
  KEY IS NOT EMBEDDED
  TYPE -- COLLATED SYMBOLIC
  LENGTH IN CHARACTERS -------- 5

MAXIMUM RECORD SIZE  160
MINIMUM RECORD SIZE  160

TOTAL TRANSACTIONS
  NUMBER OF PUTS ------ 1
  NUMBER OF GETS ------ 0
  NUMBER OF DELETES --- 0
  NUMBER OF REPLACES -- 0
  NUMBER OF GETNEXTS -- 0

NUMBER OF BLOCKS------- 1
NUMBER OF EMPTY BLOCKS- 0
BLOCK SIZE IN PRUS----- 1
NUMBER OF DATA RECORDS- 1

FILE LENGTH IN PRUS  3
NUMBER OF INDEX LEVELS IN USE   0
```

```
STATISTICS FOR FILE INDEXF


ORGANIZATION------- MIP
CREATION DATE------ 08/23/77
DATE OF LAST CLOSE- 08/23/77
TIME OF LAST CLOSE- 15.44.48.

PRIMARY KEY INFORMATION
  KEY IS NOT EMBEDDED
  TYPE -- COLLATED SYMBOLIC
  LENGTH IN CHARACTERS -------- 5

ALTERNATE KEY INFORMATION
  CHARACTERS IN LARGEST KEY-- 20

PRIMARY KEY SUBSTRUCTURES
  NUMBER OF UNIQUE -- 4
  NUMBER OF -IS-   -- 2
  NUMBER OF FIFO   -- 1

NUMBER OF BLOCKS------- 8
NUMBER OF EMPTY BLOCKS- 0
BLOCK SIZE IN PRUS----- 4
NUMBER OF DATA RECORDS- 7

FILE LENGTH IN PRUS  34
MAX NUMBER OF LEVEL 2 INDEX LEVELS  4
MAX NUMBER OF LEVEL 3 INDEX LEVELS  4
```

Figure 7-8. FLSTAT Utility Regular Output

```
STATISTICS FOR FILE SIPF


ORGANIZATION------- IS
CREATION DATE------ 09/13/77
DATE OF LAST CLOSE- 09/13/77
TIME OF LAST CLOSE- 13.58.44.

FILE IS MIPPED
COLLATION IS STANDARD

PRIMARY KEY INFORMATION
  KEY IS NOT EMBEDDED
  TYPE -- COLLATED SYMBOLIC
  LENGTH IN CHARACTERS -------- 5

MAXIMUM RECORD SIZE  160
MINIMUM RECORD SIZE  160

TOTAL TRANSACTIONS
  NUMBER OF PUTS ------ 30
  NUMBER OF GETS ------ 0
  NUMBER OF DELETES --- 0
  NUMBER OF REPLACES -- 0
  NUMBER OF GETNEXTS -- 0
  NUMBER OF SEEKS ----- 0
  NUMBER OF GETNRS ---- 0
  NUMBER OF CIO CALLS - 23

NUMBER OF BLOCKS ------ 11
NUMBER OF EMPTY BLOCKS- 0
BLOCK SIZE IN PRUS----- 1
NUMBER OF DATA RECORDS- 30

FILE LENGTH IN PRUS  13
NUMBER OF INDEX LEVELS IN USE   1

STATISTICS FOR FILE INDEXF


ORGANIZATION------- MIP
CREATION DATE------ 09/13/77
DATE OF LAST CLOSE- 09/13/77
TIME OF LAST CLOSE- 14.35.21.

PRIMARY KEY INFORMATION
  KEY IS NOT EMBEDDED
  TYPE -- COLLATED SYMBOLIC
  LENGTH IN CHARACTERS -------- 5

ALTERNATE KEY INFORMATION
  CHARACTERS IN LARGEST KEY-- 20

PRIMARY KEY SUBSTRUCTURES
  NUMBER OF UNIQUE -- 4
  NUMBER OF -IS-   -- 2
  NUMBER OF FIFO   -- 1

NUMBER OF CIO CALLS  23

NUMBER OF BLOCKS------- 8
NUMBER OF EMPTY BLOCKS- 0
BLOCK SIZE IN PRUS----- 4
NUMBER OF DATA RECORDS- 7

FILE LENGTH IN PRUS  34
MAX NUMBER OF LEVEL 2 INDEX LEVELS  4
MAX NUMBER OF LEVEL 3 INDEX LEVELS  4
```

Figure 7-9. FLSTAT Utility Expanded Output

FLBLOK(NR=n,KS=n,MR=n,MI=n)

NR	File size (approximate number of records).
KS	Key length in characters; must be 10 for signed binary.
MR	Maximum record size in characters.
MI	Minimum record size in characters.

Figure 7-10. FLBLOK Control Statement Format

*nl,bf,pi,pd

nl	Number of index levels; 1 through 15.
bf	Blocking factor (number of records per block).
pi	Index block padding percentage; 0 through 99.
pd	Data block padding percentage; 0 through 99.

Figure 7-11. FLBLOK Directive Format

ILLEGAL PARAMETER IN INPUT CARD

The run is terminated because the KYAN directive contained a bad parameter.

ENTRYi - SYNONYM LIMIT EXCEEDED

More than 4095 records have been hashed to the same home block. Processing terminates on the specific entry but continues on the other entries.

ENTRYi - BAD KEY ENCOUNTERED

A specific key hashes outside the home block area. This key is ignored and processing continues.

MORE THAN 25 BAD KEYS ENCOUNTERED

The run is terminated.

nnnnnD WORDS OF CENTRAL MEMORY USED

```
job statement
FLBLOK(NR=100000,KS=20,MR=1000,MI=500)
7/8/9
*1,5.
*2,5,10.
6/7/8/9
```

Figure 7-12. FLBLOK Utility, Sample Deck Structure

```
FLBLOK,NR=100000,KS=20,MR=1000,MI=500.


$!

THE PERCENTAGE OF PADDING IN THE DATA BLOCK WAS NOT  SPECIFIED
                                THE SIS DEFAULT VALUE OF         0 IS ASSUMED

THE PERCENTAGE OF PADDING IN THE INDEX BLOCK WAS NOT SPECIFIED
                                THE SIS DEFAULT VALUE OF         5 IS ASSUMED

*1,5.

                      INDEXED SEQUENTIAL FILE ESTMATE


NUMBER OF RECORDS=    100000                KEY    SIZE=       20 CHARACTERS

MINIMUM RECORD SIZE=        50 WORDS        MAXIMUM RECORD SIZE=       100 WORDS

  NUMBER                      INDEX           DATA         MINIMUM     SUGGESTED
 OF INDEX       ACCESS        BLOCK           BLOCK        BUFFER       BUFFER
  LEVELS         MODE         SIZE            SIZE          SIZE         SIZE
                             (WORDS)         (WORDS)       (WORDS)      (WORDS)

    1           RANDOM         4862            4862          9979        14745
    1         SEQUENTIAL                       4862          9979        14745


FLBLOK,NR=100000,KS=20,MR=1000,MI=500.


$!

THE PERCENTAGE OF PADDING IN THE DATA BLOCK WAS NOT  SPECIFIED
                                THE SIS DEFAULT VALUE OF         5 IS ASSUMED

*2,5,10.

                      INDEXED SEQUENTIAL FILE ESTMATE


NUMBER OF RECORDS=    100000                KEY    SIZE=       20 CHARACTERS

MINIMUM RECORD SIZE=        50 WORDS        MAXIMUM RECORD SIZE=       100 WORDS

  NUMBER                      INDEX           DATA         MINIMUM     SUGGESTED
 OF INDEX       ACCESS        BLOCK           BLOCK        BUFFER       BUFFER
  LEVELS         MODE         SIZE            SIZE          SIZE         SIZE
                             (WORDS)         (WORDS)       (WORDS)      (WORDS)

    2           RANDOM          446             446          1048         1948
    2         SEQUENTIAL                        446          1048         1498


END OF INPUT FILE ENCOUNTERED
```

Figure 7-13. FLBLOK Utility Output

```
HOME BLOCK                entry1 . . . entry5

     0                     XXX . . .  XXX
     1                     XXX . . .  XXX
     .                       .
     .                       .
     .                       .
     n                     XXX . . .  XXX


STANDARD DEVIATION

                           entry1 . . . entry5

                           XX.XX      XX.XX
```

Figure 7-14. Key Analysis Output

```
KYAN(LFN=axxxxxx,MRL=i,KL=j,RKP=k,RKW=l,
  H1=entry1,hmb1,option1, ... H5=entry5,hmb5,option5)
```

axxxxxx	Logical file name of the file containing the user hashing routines; if the default hashing routine is used, LFN is set to zero.
i	Maximum record length in characters.
j	Key length in characters.
k	Relative key position within relative key word (RKW), counting from 0.
l	Relative key word in which the key begins, counting form 0.
entry1 ... 5	Entry point names of hashing routines to be tested; SDAHASH must be specified to test the system-supplied hashing routine.
hmb1 ... 5	Number of home blocks.
option1 ... 5	Output options: S Synonyms only D Standard deviations only B Synonyms and standard deviations

Figure 7-15. KYAN Directive Format

The key analysis utility can be entered through a source program written in COMPASS, COBOL, or FORTRAN Extended. The field length requirement is the sum of the space needed by the source program, the hashing routines to be tested, AAM, SDAKYAN, and internal tables. The space needed by AAM varies as a function of the input file organization. The number of central memory words required for internal tables is the largest home block value specified.

The key analysis utility has two entry points: SDAKEYH and SDAENDH. The COMPASS user must open the input file and read the records one by one. As each record is read, the user program sets register A1 to point to the location of the key address and issues a return jump to SDAKEYH. A return jump to SDAENDH must be used to terminate use of the KYAN directive.

For a COBOL program, the linkage is as follows:

```
ENTER SDAKEYH data-name.
ENTER SDAENDH.
```

The data name contains the record key and must be an elementary item in the Working-Storage or Common-Storage Section of the COBOL program.

For a FORTRAN Extended program, the linkage is as follows:

```
CALL SDAKEYH(KA)
CALL SDAENDH
```

KA is the address of the record key.

An example of a deck structure using the key analysis utility as an external subroutine is shown in figure 7-16. Hashing routines to be tested are assumed to be in relocatable binary format on a permanent file named MYHASH.

```
JOBX, . . .
ATTACH(MYHASH)
  COMPASS. FTN. COBOL.
LGO.
DISPOSE(KEYLIST,PR)
7/8/9
User program source deck
7/8/9
KYAN(LFN=MYHASH, . . .)
6/7/8/9
```

Figure 7-16. Key Analysis as External Subroutine

CREATE UTILITY

The CREATE utility is available only when Sort/Merge has been installed. This utility can be used to create direct access files through FORM or from a call through a user program. A direct access file is produced more rapidly when the CREATE utility is used than when such a file is produced by reading input and calling AAM to write each record. The CREATE utility should be used for files containing 1000 or more records.

In general, the CREATE utility hashes the key from an input record and prefixes the key to the record. Sort/Merge is then used to sort the hashed keys. After the sort operation, the prefixed keys are removed and the CREATE utility uses AAM to produce the direct access file.

A job using the CREATE utility involves the following:

FILE control statement to describe input and output files

Loader control statement to load the COBOL library for Sort/Merge

CREATE directive on the file INPUT

The format of the CREATE directive is shown in figure 7-17. The second and third parameters are omitted when the default hashing routine is selected. Any operating system separator, as well as embedded blanks, can be used between parameters.

CREATE(lfn,hash,hfl)

lfn	Logical file name of the output file (same as specified in a FILE control statement for a direct access file).
hash	User hashing routine entry point.
hfl	Name of the file containing the hashing routine in relocatable binary form.

Figure 7-17. CREATE Directive Format

All input and direct access file characteristics (other than defaults) must be specified with FILE control statements. For both source program calls and use of the CREATE utility through FORM, the COBOL library and AAM modules must be loaded. If a user hashing routine is used, the routine must also be loaded.

The FILE control statement used to define the direct access file structure must specify the following parameters:

lfn	Logical file name
FO	FO=DA file organization
HMB	Number of home blocks
MNR	Minimum number of characters in any record
MRL	Maximum number of characters in any record
KL	Number of 6-bit characters in the key
BFS	Number of words in the buffer. Default buffer size is 260 words; the buffer must be able to hold at least one home block.

Additional file structure parameters can be included in the FILE control statement.

When the CREATE utility is called by a source program, the user must cause the input file to be read. After each record is read, the user must place the key in the record and give control to the utility at entry point SDACRTU. The key address, the working storage address, and the total record length must be passed to the CREATE utility. At the end of file processing, the user calls CREATE at entry point SDAENDC.

The source program must not reference the direct access file being created. A FILE control statement must be used to describe file structure. If key position is not left-justified at location KA, the relative key position (RKP) must be set in the FILE control statement. The two entry points used in calling the CREATE utility from a source program are SDACRTU and SDAENDC.

The appropriate data name, variable name, or list parameters for key address (KA), working storage area (WSA), and record length (RL) are provided in calls with SDACRTU as follows:

COBOL

```
ENTER SDACRTU USING data-name-1,
data-name-2, data-name-3.
```

FORTRAN Extended

```
CALL SDACRTU(variable-name-1,
variable-name-2,variable-name-3)
```

COMPASS

A pointer to a comparable three-parameter list is stored in register A1; the call to SDACRTU uses a return jump.

The RL field must be specified as an integer in a COMPASS or FORTRAN Extended program. In a COBOL program, the RL field must be specified by a COMP-1 item.

An example of a COBOL source code call to the CREATE utility is shown in figure 7-18; the Identification, Environment, and Data Divisions are assumed. This procedure illustrates that portion of a job in which the user reads each record, enters SDACRTU for hashing, and enters SDAENDC after all records are read to complete direct access file creation.

```
      .
      .
      .
      PROCEDURE DIVISION.
      START.
      OPEN INPUT lfn.
      PERFORM A n TIMES.
    A READ lfn INTO SDA-WSA AT END GO TO B.
      MOVE xx TO RL.
      .
      .
      ENTER SDACRTU USING data-name-1, data-name-2,
          data-name-3.
    B ENTER SDAENDC.
      CLOSE lfn.
      STOP RUN.
```

Figure 7-18. CREATE Call Through COBOL

MULTIPLE-INDEX FILES

Three utilities are provided for use with files processed by the Multiple-Index Processor (MIP). The IXGEN utility creates an index file for processing by initial MIP. The MIPGEN and MIPDIS utilities are used with files processed by extended MIP.

IXGEN UTILITY

The IXGEN utility is used to create an index file for alternate key access to an existing initial indexed sequential, direct access, or actual key file. In addition, this utility can be used to define additional alternate keys for a file or to remove alternate keys from a file. The existing data file must not be an empty file and must have all primary and alternate keys within the records; an initial indexed sequential file cannot have duplicate primary keys. Key specifications can define overlapping fields.

An existing direct access file must use the system-supplied hashing routine because there is no way to specify a user hashing routine for IXGEN. An attempt to use a direct access file with a user hashing routine produces AAM error 171.

A job using the IXGEN utility involves the following:

A FILE control statement to identify the existing data file, its organization, and the logical file name of the index file

An RFL control statement to specify a field length of 65000_8 plus the size of the buffer to process the file (a larger field length improves efficiency; adding 15000_8 is suggested)

An IXGEN control statement to identify the existing data file, the source of additional control information (RMKDEF directives), and the file to receive the listable output

A set of RMKDEF directives on the file INPUT or other file of card images

When the index file is created, the first RMKDEF directive must define the primary key for the existing data file. Each alternate key must also be specified in an RMKDEF directive.

Alternate key definitions can be added to or purged from an existing index file only through the IXGEN utility. Each alternate key to be added to or purged from the index file must be specified in an RMKDEF directive. The primary key must not be specified in an RMKDEF directive for an update run.

The format of the IXGEN control statement is shown in figure 7-19. The format of the RMKDEF directives expected by the IXGEN utility is shown in figure 7-20. The kg and kc parameters used together refer to a key that is part of a repeating group, such as that which results from the COBOL clause OCCURS n TIMES. In the RMKDEF directive for the primary key, the kg parameter can be used to specify the length in PRUs of a block in the index file. The kc parameter has no meaning in this directive.

IXGEN(prifile,directs,outf)

prifile	Logical file name of the existing initial indexed sequential, direct access, or actual key file.
directs	Name of the file containing the RMKDEF directives; optional; default is INPUT.
outf	Name of the file containing listable output from IXGEN; optional; default is OUTPUT.

Figure 7-19. IXGEN Control Statement Format

The structure of primary key lists is specified by the ks parameter. For efficiency in processing, indexed sequential structure is recommended. First-in first-out structure can also be specified; however, the ordering of primary keys generated by the IXGEN utility should not be assumed to be the same order in which the data file records were created.

MIPGEN UTILITY

The MIPGEN utility is used to create an index file for alternate key access to an existing extended indexed sequential file. This utility can also be used to define additional alternate keys for a file or to remove alternate keys from a file. The existing data file must not be an empty file. Key specifications can define overlapping fields.

A job using the MIPGEN utility involves the following:

A FILE control statement to identify the existing extended indexed sequential file, to specify the logical file name of the index file, and to specify the index file block size

RMKDEF(prifile,rkw,rkp,kl,0,kf,ks,kg,kc)

prifile	Logical file name of the existing initial indexed sequential, direct access, or actual key file; required.
rkw	Relative word in the record in which the alternate key begins, counting the first word in the record as 0; required.
rkp	Relative character position within the relative key word (rkw) in which the alternate key begins, counting the first character position in the word as 0; required.
kl	Number of characters in the key, 1 to 255; required.
0	Required to mark position for the reserved field.
kf	Key format; required: 0 Character string, similar to symbolic key type (KT=S) for initial indexed sequential files 1 Signed binary, similar to floating point (KT=F) and integer (KT=I) key types for initial indexed sequential files 2 Unsigned binary, described in COBOL as PICTURE 9; no existing file primary key has this designation 3 Purge alternate key definition from the index
ks	Substructure for each primary key list in the index; optional: U Unique (default) I Indexed sequential; recommended for efficiency in processing F First-in first out
kg	Length in characters of the repeating group in which the key resides when used with kc.
kc	Number of occurrences of the repeating group; zero if the group is defined by an OCCURS ... DEPENDING ON clause.

Figure 7-20. RMKDEF Directive Format, IXGEN Utility

An RFL control statement to specify a field length of 65000_8 plus the size of the buffer to process the file (a larger field length improves efficiency; adding 15000_8 is suggested)

A MIPGEN control statement to identify the existing data file, the source of additional control information (RMKDEF directives), and a list file for output from the utility

A set of RMKDEF directives on the file INPUT or other file of card images

When the index file is created, each alternate key must be defined by an RMKDEF directive.

Alternate key definitions can be added to or purged from an existing index file only through the MIPGEN utility. Each alternate key to be added to or purged from the index file must be specified in an RMKDEF directive.

The format of the MIPGEN control statement is shown in figure 7-21. The format of the RMKDEF directives expected by the MIPGEN utility is shown in figure 7-22. The kg and kc parameters are used together and refer to a key that is a repeating group, such as that which results from the COBOL clause OCCURS n TIMES.

MIPGEN(prifile,directs,lfile)

prifile	Logical file name of the existing extended indexed sequential file.
directs	Name of the file containing the RMKDEF directives; optional; default is INPUT.
lfile	Name of the file that contains the output listing from MIPGEN; optional; default is OUTPUT.

Figure 7-21. MIPGEN Control Statement Format

The structure of primary key lists is specified by the ks parameter. For efficiency in processing, indexed sequential structure is recommended. First-in first-out structure can also be specified; however, the ordering of primary keys generated by the MIPGEN utility should not be assumed to be the same order in which the data file records were created.

The nl, ie, and ch parameters are used to define sparse keys. An alternate key is defined as a sparse key when all values of the key are not desired to be indexed. Sparse keys cause short indexing operations that save disk space, computer time for index file maintenance, and search time. A sparse key is a result of either null suppression or sparse control characters.

The nl parameter specifies null suppression for an alternate key. If null suppression is specified, the alternate key index does not include primary keys for records that have null values for the alternate key. All spaces for a symbolic key and all zeros for an integer key are null values.

The ie and ch parameters are used when indexing of alternate key values is to be controlled by a sparse control character. The one-character field containing the sparse control character must be in the fixed-length portion of the record. The ie parameter specifies whether to include or exclude the alternate key values for records that contain a sparse control character. The ch parameter specifies the sparse control characters applicable to the alternate key being defined; up to 36 letters and digits can be specified as a character string.

The sparse control character field is identified by an RMKDEF directive that must appear before the directive defining the alternate key and its sparse control characters. This directive is specified in the following format:

RMKDEF(prifile,rkw,rkp,0)

The rkw and rkp parameters identify the position of the sparse control character. The zero key length parameter indicates that the field is a sparse control character field.

MIPDIS UTILITY

The MIPDIS utility temporarily or permanently disassociates an index file from its associated extended indexed sequential file. If the primary and alternate key fields are not updated during the disassociation, the index file can be reassociated with the data file.

RMKDEF(prifile,rkw,rkp,kl,0,kf,ks,kg,kc,nl,ie,ch)

prifile	Logical file name of the existing extended indexed sequential file; required.
rkw	Relative word in the record in which the alternate key begins, counting from 0; required.
rkp	Relative beginning character position within the relative key word (rkw), counting from 0; required.
kl	Number of characters in the key, 1 to 255; required.
0	Required to mark position for the reserved field.
kf	Key format, required: 0 or S — Character string 1 or I — Signed binary 2 or U — Uncollated character string 3 or P — Purge alternate key definition from the index
ks	Substructure for each primary key list in the index; optional: U — Unique (default) I — Indexed sequential; recommended for efficiency in processing F — First-in first-out
kg	Length in characters of the repeating group in which the key resides.
kc	Number of occurrences of the repeating group; zero if the group is defined by an OCCURS ... DEPENDING ON clause.
nl	Null suppression; a null value is all spaces (symbolic key) or all zeros (integer key): 0 — Null values are indexed (default) N — Null values are not indexed
ie	Include/exclude sparse control character: I — Include alternate key value if the record contains a sparse control character E — Exclude alternate key value if the record contains a sparse control character
ch	Characters that qualify as sparse control characters; up to 36 letters and digits can be specified as a character string.

Figure 7-22. RMKDEF Directive Format, MIPGEN Utility

Whenever a data file that has an associated index file is opened, a safety lock in the file statistics table requires the index file to be present. The MIPDIS utility removes this requirement.

Disassociation of an index file from the data file is useful under various circumstances. One instance occurs when a data file that has an associated index file is no longer being accessed by alternate key. In this case, the index file is no longer needed and can be disassociated from the data file.

Indexed sequential files are sometimes reorganized to reclaim extraneous padding caused by block splitting and to redistribute it evenly throughout the file. The reorganization is accomplished through either the FORM utility or a user program. The index file can be disassociated from the data file before the reorganization. After the reorganization, the index file is still valid for the data file and can be associated with the data file again by the MIPDIS utility. This eliminates the need to create a new index file through the MIPGEN utility or during the creation of the restructured data file.

While the data file is disassociated, any changes to the primary or alternate key values are not reflected in the index file. This can result in errors when updating or accessing the file by alternate key.

The format of the MIPDIS control statement is shown in figure 7-23. This control statement can be used to disassociate or associate an index file with its data file.

MIPDIS(lfn1,da,lfn2)

lfn1	Logical file name of the data file.
da	Disassociate/associate index file: D Disassociate from data file A Associate with data file
lfn2	Logical file name of the index file; not required for disassociation.

Figure 7-23. MIPDIS Control Statement Format

STANDARD CHARACTER SET A

CONTROL DATA operating systems offer the following variations of a basic character set:

CDC 64-character set

CDC 63-character set

ASCII 64-character set

ASCII 63-character set

The set in use at a particular installation was specified when the operating system was installed.

Depending on another installation option, the system assumes an input deck has been punched either in 026 or in 029 mode (regardless of the character set in use).

Under NOS/BE, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of the job statement or any 7/8/9 card. The specified mode remains in effect through the end of the job unless it is reset by specification of the alternate mode on a subsequent 7/8/9 card.

Under NOS, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of any 6/7/9 card, as described above for a 7/8/9 card. In addition, 026 mode can be specified by a card with 5/7/9 multipunched in column 1, and 029 mode can be specified by a card with 5/7/9 multipunched in column 1 and a 9 punched in column 2.

Graphic character representation appearing at a terminal or printer depends on the installation character set and the terminal type. Characters shown in the CDC Graphic column of the standard character set table are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII-CRT and ASCII-TTY terminals.

Display Code (octal)	CDC Graphic	CDC Hollerith Punch (026)	CDC External BCD Code	ASCII Graphic Subset	ASCII Punch (029)	ASCII Code (octal)
00†	: (colon)††	8-2	00	: (colon) ††	8-2	072
01	A	12-1	61	A	12-1	101
02	B	12-2	62	B	12-2	102
03	C	12-3	63	C	12-3	103
04	D	12-4	64	D	12-4	104
05	E	12-5	65	E	12-5	105
06	F	12-6	66	F	12-6	106
07	G	12-7	67	G	12-7	107
10	H	12-8	70	H	12-8	110
11	I	12-9	71	I	12-9	111
12	J	11-1	41	J	11-1	112
13	K	11-2	42	K	11-2	113
14	L	11-3	43	L	11-3	114
15	M	11-4	44	M	11-4	115
16	N	11-5	45	N	11-5	116
17	O	11-6	46	O	11-6	117
20	P	11-7	47	P	11-7	120
21	Q	11-8	50	Q	11-8	121
22	R	11-9	51	R	11-9	122
23	S	0-2	22	S	0-2	123
24	T	0-3	23	T	0-3	124
25	U	0-4	24	U	0-4	125
26	V	0-5	25	V	0-5	126
27	W	0-6	26	W	0-6	127
30	X	0-7	27	X	0-7	130
31	Y	0-8	30	Y	0-8	131
32	Z	0-9	31	Z	0-9	132
33	0	0	12	0	0	060
34	1	1	01	1	1	061
35	2	2	02	2	2	062
36	3	3	03	3	3	063
37	4	4	04	4	4	064
40	5	5	05	5	5	065
41	6	6	06	6	6	066
42	7	7	07	7	7	067
43	8	8	10	8	8	070
44	9	9	11	9	9	071
45	+	12	60	+	12-8-6	053
46	-	11	40	-	11	055
47	*	11-8-4	54	*	11-8-4	052
50	/	0-1	21	/	0-1	057
51	(	0-8-4	34	(	12-8-5	050
52	)	12-8-4	74	)	11-8-5	051
53	$	11-8-3	53	$	11-8-3	044
54	=	8-3	13	=	8-6	075
55	blank	no punch	20	blank	no punch	040
56	, (comma)	0-8-3	33	, (comma)	0-8-3	054
57	. (period)	12-8-3	73	. (period)	12-8-3	056
60	≡	0-8-6	36	#	8-3	043
61	[	8-7	17	[	12-8-2	133
62	]	0-8-2	32	]	11-8-2	135
63	% ††	8-6	16	% ††	0-8-4	045
64	≠	8-4	14	" (quote)	8-7	042
65	←	0-8-5	35	_ (underline)	0-8-5	137
66	∨	11-0 or 11-8-2†††	52	!	12-8-7 or 11-0†††	041
67	∧	0-8-7	37	&	12	046
70	↑	11-8-5	55	' (apostrophe)	8-5	047
71	↓	11-8-6	56	?	0-8-7	077
72	<	12-0 or 12-8-2†††	72	<	12-8-4 or 12-0†††	074
73	>	11-8-7	57	>	0-8-6	076
74	≤	8-5	15	@	8-4	100
75	≥	12-8-5	75	\	0-8-2	134
76	¬	12-8-6	76	~ (circumflex)	11-8-7	136
77	; (semicolon)	12-8-7	77	; (semicolon)	11-8-6	073

†Twelve zero bits at the end of a 60-bit word in a zero byte record are an end of record mark rather than two colons.

††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank (55_8).

†††The alternate Hollerith (026) and ASCII (029) punches are accepted for input only.

CDC CHARACTER SET COLLATING SEQUENCE									
Collating Sequence Decimal	Collating Sequence Octal	CDC Graphic	Display Code	External BCD	Collating Sequence Decimal	Collating Sequence Octal	CDC Graphic	Display Code	External BCD
00	00	blank	55	20	32	40	H	10	70
01	01	≤	74	15	33	41	I	11	71
02	02	%	63 †	16 †	34	42	v	66	52
03	03	[	61	17	35	43	J	12	41
04	04	→	65	35	36	44	K	13	42
05	05	≡	60	36	37	45	L	14	43
06	06	∧	67	37	38	46	M	15	44
07	07	↑	70	55	39	47	N	16	45
08	10	↓	71	56	40	50	O	17	46
09	11	>	73	57	41	51	P	20	47
10	12	≥	75	75	42	52	Q	21	50
11	13	¬	76	76	43	53	R	22	51
12	14	.	57	73	44	54	]	62	32
13	15	)	52	74	45	55	S	23	22
14	16	;	77	77	46	56	T	24	23
15	17	+	45	60	47	57	U	25	24
16	20	$	53	53	48	60	V	26	25
17	21	*	47	54	49	61	W	27	26
18	22	-	46	40	50	62	X	30	27
19	23	/	50	21	51	63	Y	31	30
20	24	,	56	33	52	64	Z	32	31
21	25	(	51	34	53	65	:	00 †	none†
22	26	=	54	13	54	66	0	33	12
23	27	≠	64	14	55	67	1	34	01
24	30	<	72	72	56	70	2	35	02
25	31	A	01	61	57	71	3	36	03
26	32	B	02	62	58	72	4	37	04
27	33	C	03	63	59	73	5	40	05
28	34	D	04	64	60	74	6	41	06
29	35	E	05	65	61	75	7	42	07
30	36	F	06	66	62	76	8	43	10
31	37	G	07	67	63	77	9	44	11

†In installations using the 63-graphic set, the % graphic does not exist. The : graphic is display code 63, External BCD code 16.

ASCII CHARACTER SET COLLATING SEQUENCE

Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code	Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code
00	00	blank	55	20	32	40	@	74	40
01	01	!	66	21	33	41	A	01	41
02	02	"	64	22	34	42	B	02	42
03	03	#	60	23	35	43	C	03	43
04	04	$	53	24	36	44	D	04	44
05	05	%	63†	25	37	45	E	05	45
06	06	&	67	26	38	46	F	06	46
07	07	'	70	27	39	47	G	07	47
08	10	(	51	28	40	50	H	10	48
09	11	)	52	29	41	51	I	11	49
10	12	*	47	2A	42	52	J	12	4A
11	13	+	45	2B	43	53	K	13	4B
12	14	,	56	2C	44	54	L	14	4C
13	15	-	46	2D	45	55	M	15	4D
14	16	.	57	2E	46	56	N	16	4E
15	17	/	50	2F	47	57	O	17	4F
16	20	0	33	30	48	60	P	20	50
17	21	1	34	31	49	61	Q	21	51
18	22	2	35	32	50	62	R	22	52
19	23	3	36	33	51	63	S	23	53
20	24	4	37	34	52	64	T	24	54
21	25	5	40	35	53	65	U	25	55
22	26	6	41	36	54	66	V	26	56
23	27	7	42	37	55	67	W	27	57
24	30	8	43	38	56	70	X	30	58
25	31	9	44	39	57	71	Y	31	59
26	32	:	00†	3A	58	72	Z	32	5A
27	33	;	77	3B	59	73	[	61	5B
28	34	<	72	3C	60	74	\	75	5C
29	35	=	54	3D	61	75	]	62	5D
30	36	>	73	3E	62	76	^	76	5E
31	37	?	71	3F	63	77	_	65	5F

† In installations using a 63-graphic set, the % graphic does not exist. The : graphic is display code 63.

AAM checks user requests to ensure proper processing. If results are not satisfactory, an error condition exists and the following occurs:

A three-digit octal error code is returned to the error status (ES) field in the FIT.

For a fatal error, the fatal/nonfatal (FNF) field is set in the FIT.

The error exit is taken if the user has set the error exit (EX) field in the FIT.

The dayfile control (DFC) field and the error file control (EFC) field in the FIT determine the disposition of error messages and notes/statistics. Depending on the setting of these two fields, error messages and statistics/notes are written to the dayfile and/or the error file ZZZZZEG.

ERROR COMMUNICATION

Regarding errors, AAM and the user communicate through the following FIT fields:

ECT Trivial error count

ERL Trivial error limit

ES Error status

EX Error exit

The ES field is a 9-bit field that is set to an octal value after AAM has attempted error resolution and is ready to return control to the user. When an attempt is made to execute an input/output request after an error, AAM does not clear the ES field. If the request is not legal, AAM increments the ECT field and proceeds with execution. If a subsequent error is detected, the ES field reflects the most recent error. The user is responsible for clearing the ES field when an error exit (EX) is not supplied and the ES field is checked after every macro call.

FIT fields relevant to error processing and their meanings are as follows:

DFC Dayfile control; set by the user to control the listing of error messages on the dayfile.

- DFC=0 Only fatal error messages to the dayfile (default).
- DFC=1 Error messages to the dayfile.
- DFC=2 Statistics/notes to the dayfile.
- DFC=3 Error messages and statistics/-notes to the dayfile.

EFC Error file control; set by the user to control the listing of error messages on the error file.

- EFC=0 No error file entries (default).
- EFC=1 Error messages to the error file.
- EFC=2 Statistics/notes to the error file.
- EFC=3 Error messages and statistics/-notes to the error file.

ERL Trivial error limit; if not specified, the value is zero, no error account is accumulated, and an indefinite number of trivial errors is permitted; if a value is specified, the job is terminated when the value of the ECT field reaches the value specified for the ERL field.

EX Error exit; an 18-bit field that is interpreted as follows:

- EX=0 No user error routine; control is returned as a normal exit; the ES field is set to an error code. If a fatal error is encountered, the message is output to the dayfile.
- EX≠0 When a fatal or trivial error occurs, control is transferred to EX+1; a jump to the user in-line return address is stored in the EX field and the ES field is set to an error code.

FNF Fatal/nonfatal flag; set to 1 for fatal errors.

ERROR FILE PROCESSING

When the error file control (EFC) field is set to a nonzero value, error messages and/or statistics/notes are written to the error file ZZZZZEG. The error file is always flushed at an abnormal termination. At the completion of a job step, the error file is flushed if all files are closed. The CRMEP control statement can be used to process the error file and control the listing of information from the error file on the output file. The format of the CRMEP control statement is shown in figure B-1.

The parameters, options, and defaults for the CRMEP control statement are listed in table B-1. The first default listed in the table is set if neither the parameter nor the option is specified. The second default listed is set if the parameter is specified without an option. More than one option can be specified with each parameter; more than one parameter can be specified in one CRMEP control statement.

CRMEP(parameter=option$_1$/option$_2$/ ... /option$_n$, ...)

parameter	Mnemonic specifying type of error file processing and listing.
option	Selected setting of the specified parameter.

Figure B-1. CRMEP Control Statement Format

The capability to dump the contents of the FIT to the error file for subsequent processing is provided by the FITDMP macro. When the FITDMP macro is executed, the FIT is written to the error file ZZZZZEG as note number 1000. The error file control (EFC) field in the FIT must be set to 2 or 3 to ensure that notes are written to the error file. The CRMEP control statement can then be used to display the FIT on the output file. The format of the FITDMP macro is shown in figure B-2.

The id parameter is an optional parameter that is used to display an identifier for the FIT dump. The FIT display identifier at the location specified by the id parameter consists of 10 characters of displayable information.

ERROR CONDITION PROCESSING

When an error condition is encountered, the error status (ES) field is set to the appropriate error number. For a trivial error, the trivial error limit (ERL) field set to zero allows unlimited trivial errors. If the ERL field is greater than zero, the trivial error count (ECT) field is incremented and compared with the ERL field as follows:

If the ECT field is less than the ERL field, control is passed to the error exit if specified or to the user's in-line code. If control passes to the in-line code, the user is responsible for checking the error status.

If the ECT field is equal to the ERL field, the ES field is set to 356 (trivial error limit reached) and the fatal/nonfatal (FNF) field is set. Control is returned to the error exit if specified or to the user's in-line code.

When a file is accessed sequentially and end-of-information is encountered, the file position (FP) field is set to indicate EOI and an informative message is issued. If the end-of-data exit (DX) field has been set, the exit is taken. If another access beyond end-of-information is attempted without repositioning the file, a fatal error status is given for an indexed sequential file and a trivial error status is given for direct access and actual key files. If the error exit (EX) field is set, that exit is taken. If the FNF field is set and any AAM function is attempted on the file, a 115 error is generated and the job is aborted.

FITDMP fit,id

fit Address of the FIT to be dumped.

id Address of the FIT display identifier.

Figure B-2. FITDMP Macro Format

TABLE B-1. CRMEP CONTROL STATEMENT PARAMETERS

Parameter	Option	Defaults			Description
		First Default	Second Default	Initial Value	
LO	N or		*		Select notes
	-N/	*		*	Omit notes
	F or	*	*	*	Select fatals
	-F/				Omit fatals
	D or	*	*	*	Select data manager messages
	-D/				Omit data manager messages
	T or		*		Select trivial errors
	-T/	*		*	Omit trivial errors
SF	lfn_1/ lfn_2/ ... lfn_n	all	all	none	Select messages associated with the specified lfns.
OF	lfn_1/ lfn_2/ ... lfn_n	none	none	none	Omit messages associated with the specified lfns.
SN	mno_1/ mno_2/ ... mno_n	all	hardware and parity errors	none	Select only specified message numbers (octal).
ON	mno_1/ mno_2/ ... mno_n	none	142 and 143 only	none	Omit only specified message numbers (octal).
L	lfn	OUTPUT	LIST		Specifies the post-processor output file.
RU			*		Return/unload of error file; performed at end of processing. If RU is not specified, file position of error file remains at EOI at end of processing.

CLASSES OF ERRORS

Syntax errors are diagnosed by AAM; the messages are self-explanatory. System errors are detected by the operating system. Execution errors, occurring during execution of input and output requests, are subdivided into call errors and invalid input/output requests.

CALL ERRORS

Call errors are undetectable parameter errors. For example:

```
GET     X1
```

If register X1 does not contain the valid FIT address, an unpredictable AAM error, mode error, or D00 error can result.

INVALID INPUT/OUTPUT REQUESTS

Requests for illegal input/output operations produce the following general types of errors:

FIT	Content of address given as the FIT address does not pass a test for plausibility. It does not contain a legal logical file name in bits 59 through 18, or the FIT has inconsistencies.
File organization	Input/output requests or specifications illegal on the type of file specified by the file organization (FO) field in the FIT.
Record type	Input/output requests illegal for the record type specified by the record type (RT) field in the FIT.
OPENM/CLOSEM	Input/output requests illegal for files opened or closed as specified by the open/close (OC) field and/or the old/new file (ON) field in the FIT.
Processing direction	Input/output requests that would violate the processing direction limitations specified by the processing direction (PD) field in the FIT.
File position	Input/output requests illegal for the file position given by the file position (FP) field in the FIT.
Last operation	Input/output requests illegal in the context of the last operation.
Key	Attempts to access or write records whose keys are not within the range of keys defined for a file.
Data	Errors in data specification, such as inconsistency between the amount of data requested and the amount actually present, illegal field present in the data, required field is absent, or parity error.
Device	Input/output requests illegal on the device upon which the file resides.

All errors are either fatal or nonfatal. Some nonfatal errors are trivial in that no user action is required. Fatal errors usually indicate incorrect parameter specification and incomplete or contradictory information provided by the user as program errors. A fatal error message is always printed on the dayfile.

Trivial errors are usually data errors, such as attempting to insert a record already in the file or to replace or delete a record that does not exist. If a trivial error message is printed, the key and type of error are part of the error message. The record associated with the trivial error is dropped; however, the file position might be altered.

If the error exit (EX) field in the FIT has been set to the address of an error routine, any error causes a transfer of control to the address in EX+1 for a recovery routine after the error has been resolved. Fatal errors inhibit any further attempts to perform input/output on the file using AAM; such attempts cause the job to terminate. If the EX field is not set, an error sets the error status (ES) field and returns control to the calling program. The ES field is cleared after an error.

AAM is in the user's field length and is subject to destruction by the user.

DIAGNOSTICS

Error messages that can be output by AAM are listed in table B-2. The messages are in order by error code. The table contains the following information:

Code	Octal value corresponding to the error condition.
Message	Diagnostic output; varies depending on the setting of the DFC and EFC fields and the parameters specified in the CRMEP control statement.
Significance	Meaning of the message.
Action	Suggestion for the user to recover from the error condition.
Severity	Type of error; can be any of the following:

F	Fatal
T	Trivial
T/F	Trivial under some conditions, fatal under other conditions

Table B-3 is a list of notes and informative messages that can be output.

TABLE B-2. DIAGNOSTICS

Code	Message	Significance	Action	Severity
001	INVALID FO	File organization must be indexed sequential (IS), direct access (DA), or actual key (AK).	Correct the file organization field.	F
002	FIT/FILE ORGANIZATION MISMATCH	The file organization specified does not match any opened files.	Check to see that the correct file is being processed or that the FO field is specified correctly.	F
006	FIRST BLOCK IS NOT A FSTT	The first block in the file must be the file statistics table (FSTT). For an indexed sequential file, the ORG field must be set for the correct file organization.	If a file is being created, check that the pd parameter is specified in the OPENM macro or the ON field is set to NEW.	F
030	INVALID RT	Record type must be W, S, Z, F, R, T, D, or U; it must conform to other file specifications, such as FO.	Correct the record type field.	T
031	RT=F/Z AND FL=0	For fixed length F or zero byte terminated Z type records, a maximum record length must be specified for the FL field in the FIT.	Specify the FL field.	T
032	RT=T AND HL OR TL=0	For T type records, the header length (HL) must be large enough to hold the trailer count field defined by the CP and CL fields. The length of the trailer count field must be given in the TL field and must be at least one character long.	Correct the header length or the trailer length field.	T
033	RT=D AND LL=0/RT=T AND CL=0	For D type records, the LL field in the FIT must provide the length of the record field that specifies record length.	Specify the length of the D type record length field.	T
		For T type records, the CL field in the FIT must provide the length of the field that specifies the number of trailer items.	Specify the length of the trailer count field of the T type record.	
035	RT=T/D, MRL EXCLUDES CONTROL FIELD	For T and D type records, the record must contain a field identifying record length.	Check that for D type records LP+LL is less than MRL. For T type records, CP+CL must be less than MRL. The position count for LP and CP begins with 0.	T
036	RL INCONSISTENT WITH RECORD DESCRIPTION	For T type records, the fixed header length (HL) must include a field CL characters long, beginning at CP, to identify trailer item count.	Check that the count field is included in HL. The current record is ignored. Position CP is counted from 0.	T
037	RT=D/T AND CL/LL > 6	For D and T type records, the length of the count field must be one to six character positions.	Correct the length of the count field.	T
040	REDUNDANT OPEN	A file must be closed before open processing, such as buffer allocation or FILE control statement processing, takes place. A redundant open call is ignored.	Correct the program to close the file before open processing.	T

TABLE B-2. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
050	NUMBER OF FILES PERMITTED TO BE OPEN SIMULTANEOUSLY HAS BEEN EXCEEDED	The installation defines the number of AAM files that can be open at one time because buffers are limited by central memory available. Default release value is 10 files of each organization.	Check with a local analyst for the limit on the number of files that can be open at one time.	F
051	SETFIT DISALLOWED ON OPEN FILE	Open processing would have already processed the FILE control statement. The SETFIT macro processes FILE control statements without full open processing.	Change the placement of the SETFIT macro.	T
052	FILE NOT CLOSED AFTER LAST UPDATE/CONDITION QUESTIONABLE	The possibility exists that the file has internal errors. The most likely cause is a system crash that prevented closing of the file.	Rerun the program that updated the file.	T
053	NO HOME RECORD	The OLD parameter has been specified when opening an empty direct access file.	Check that the correct file name has been specified, or change the OLD parameter to NEW.	F
054	FILE ILLEGALLY EXTENDED (EOI MOVED)	An existing file has been opened without extend permission and information has been written beyond the old EOI.	Change the program to open with extend permission.	F
055	FILE NONEXISTENT, CANNOT OPEN-OLD	The logical file name specified does not match any existing file.	Check that the logical file name is correctly specified.	F
056	EMPTY FILE OPENED FOR READ-ONLY	When using the read-only processors, the file must be an existing non-empty file because it is opened for a read-only purpose.	Check that the correct file name has been specified, or change the OLD parameter to NEW.	F
060	REDUNDANT CLOSE	A second call to close the file was issued. The operations requested by the CF field are performed before the error is issued.	Correct the program to eliminate the redundant close operation.	T
070	OUTPUT REQUEST, PD=INPUT	A file opened with PD set to INPUT cannot be written. The write statement is ignored.	If the file is to be written, set the PD field in the FIT to OUTPUT or IO before opening the file.	T
071	INPUT REQUEST, PD=OUTPUT	A file opened with PD set to OUTPUT cannot be read. The read statement is ignored.	If the file is to be read, set the PD field in the FIT to INPUT or IO before opening the file.	T
074	MUST HAVE CMM FOR MULTIPLE ACCESS	To have multiple FITs for one file, CMM must be used. The file is not opened.	Correct the program to allow CMM to be loaded.	T
075	UBS MAY NOT BE USED FOR MULTIPLE ACCESS	A file that is to be accessed by more than one FIT cannot have user-supplied buffer space for any of the FITs.	Correct the program to eliminate the user-supplied buffer.	T
100	CANNOT SEQUENTIALLY POSITION BEYOND FILE BOUNDS	A sequential read or SKIPFL is not possible with the file at EOI. A SKIPBL is not possible with the file at BOI.	The file must be repositioned if further access is desired. Repeated access attempts with file at the end cause the fatal error flag to be set.	T/F

TABLE B-2. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
110	FILE NOT OPEN	A file must be opened before it can be read or written. Omission of required FIT field parameters or inconsistencies in specified parameters inhibit open.	Correct the program to open the file before reading or writing, or correct omissions or inconsistencies in FIT fields.	T
115	OUTSTANDING FATAL ERROR ON THE FILE	A fatal error prevents future access to the file with the error, but it does not cause job termination unless the user attempts further operations except CLOSEM on the file.	Correct and rerun.	F
117	PUT OR RELEASE OF LARGER RECORD ILLEGAL AFTER GETN	Sequential read of a direct access file is possible only if the existing records are not disturbed. Writing any new record or increasing existing record size prevents subsequent sequential access.	Correct the program.	T
130	RT=W, BAD CONTROL WORD, FILE DEFECTIVE OR MISPOSITIONED	Record type was specified as W. This message indicates the records being read are not, in fact, W type records.	Check that the existing file is correctly described.	T/F
135	RMS READ PARITY ERROR	The system returned a parity error status after a read.	Recreate the file on a good device.	T/F
136	RMS WRITE PARITY ERROR	The system returned a parity error status after a write.	Recreate the file on a good device.	F
142	EXCESS DATA	In a write, no information is written to the file; the user has supplied RL greater than FL/MRL or the record mark character for an R type record was not found before MRL characters. On a read, no information is transferred to the working storage area; the record length exceeds the FL/MRL defined. For GET macro processing, the following conditions cause an error: Z No zero byte found before FL characters R No record mark found before MRL T,D Control field RL > MRL U RL > MRL F Excess data cannot occur	Correct the inconsistency between the RL and FL or MRL fields.	T
143	INSUFFICIENT DATA	Control information in the record being read (length calculated by fields such as CP and CL) specifies a length for each record. The record existing in the file is smaller than the specified length. All characters available are returned.	No action is required.	T

TABLE B-2. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
146	USER HEADER LENGTH ERROR	The attempted PUT or REPLACE macro is rejected because the user header length is inconsistent with the record length.	Check the user header length and the record length for inconsistencies.	T
147	CHECKSUM ERROR IN DATA OR INDEX BLOCK	There is a conflict between the loading checksum and computed checksum in either the data block or index block.	Notify a system analyst.	F
150	FILE NOT ON RMS	Indexed sequential, direct access, and actual key files must be created on a disk, drum, or family pack.	Correct the control statement to ensure a valid device assignment.	T/F
165	ILLEGAL FILE NAME	The LFN does not consist of one to seven letters and digits, the first being a letter.	Correct the LFN or the FIT address.	F
166	FIT INCOMPLETE – CANNOT CREATE FILE	A required parameter is missing, or information for the FIT field is not specified correctly.	Refer to section 4 of this manual for parameters required during file creation.	F
167	RECORD LENGTH OUTSIDE MIN-MAX RANGE -- REQUEST IGNORED	Minimum and maximum record length, MNR and MRL, establish the absolute record limit for the life of the file. For D or T type records, the control field specified is outside the value specified by the RL field, or it is not within the values specified by the MNR and MRL fields.	Correct the program to write records within the established limit, or recreate the file changing MNR and MRL. Check to see that the CL/CP fields or the LL/LP fields are specified correctly.	T
170	RECORD SIZE EXCEEDS BLOCK SIZE OR IS NEGATIVE	All data blocks or home blocks must hold at least one record plus control information.	Correct the RL or MBL field.	T/F
171	INCORRECT HASHING ROUTINE	The hashing routine used to create a direct access file must be used for all subsequent access.	Check that the correct routine is available to the job or that the HRL field has not been changed. The routine name can be different each time, but the results produced cannot differ.	F
172	ERRONEOUS KL OR RKP FIELD SPECIFIED	The key length (KL) or relative key position (RKP) field is not specified properly for the key type.	Correct the KL or RKP field.	F
174	FIT INCOMPLETE FOR BFS CALCULATION	Record length range MRL and MNR, blocking factor RB, or other key characteristics required for buffer size calculation have been omitted.	Refer to section 3 of this manual for parameters required for BFS calculation.	T
175	REQUESTED DATA OR INDEX BUFFER TOO LARGE	The data block or index block size cannot exceed $2^{17}-1$.	Correct the data or index block size.	F
176	MAXRECSZ IN FSTT EXCEEDS MRL IN FIT, WSA MAY BE TOO SHORT	The MRL field in the FIT is less than the maximum record size recorded in the FSTT.	Correct the inconsistency between the current MRL value and the MRL value used when the file was created.	T

TABLE B-2. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
200	BAD FSTT LINKED TO FIT	The FSTT field in the FIT does not point to a valid FSTT when the file is being closed.	Correct the program to avoid destroying the FSTT field.	T
201	FILE CONTAINS BAD BLOCKS	Some data blocks in the file have checksum or parity errors. Updating is not allowed.	The file should be recreated as soon as possible.	T
202	FILE IS RUINED	The file structure has been destroyed. The file is no longer usable.	The file must be recreated.	T
203	CANNOT UPDATE WITHOUT MIP FILE	The file is a multiple-index file and an update operation has been attempted without having the index file present.	Close the data file. Set the XN field in the FIT to the index file name and reopen.	T
204	KEY POSITION OUT OF RANGE	The starting character position of a key is defined by positions 0 through 9, counting from the left of a word.	Correct the KP field.	F
205	MINIMUM RECORD SIZE OUT OF RANGE	Minimum record length (MNR) must be at least one character but no more than maximum record length (MRL) and must contain the key.	Correct the MNR field.	F
206	KEY NOT CONTAINED WITHIN RECORD	The embedded key must be within the record.	Check for proper RKW, RKP, KL, MNR, and MRL. Minimum and maximum record lengths (MNR and MRL) are in characters; relative key word (RKW) is in words, starting from 0; relative key position (RKP) is in 6-bit fields, 0 through 9, counting from 0 on the left.	F
207	MINIMUM RECORD SIZE EXCEEDS MAXIMUM	Required parameter MRL must be equal to or larger than MNR.	Correct the inconsistency between the MRL and MNR fields.	F
223	CHECKSUM ERROR IN FSTT	A conflict exists between the loading checksum and the computed checksum in the FSTT.	Notify a system analyst.	F
245	FUNCTION NOT VALID FOR THIS FO	The function attempted is not valid for the file organization indicated in the FIT.	Correct the program.	T
250	FILE RMS LIMIT EXCEEDED (AK)	The user has exceeded the mass storage limit as specified in the LIMIT control statement or installation-defined limit.	Correct the problem and rerun.	F
252	SYSTEM RMS LIMIT REACHED	No more mass storage was available for the file.	Consult a system analyst; perhaps the installation parameter limit was exceeded.	T/F
253	FILE LIMIT REACHED – RECORD NOT INSERTED	The number of records currently in the file cannot exceed the limit that the user specified with FLM.	Recreate the file increasing the value of FLM.	T
300	NO READ PERMISSION	To be read, a permanent file must be attached with read (RD) permission.	Attach the file with the required read permission.	F

TABLE B-2. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
301	NO WRITE OR MODIFY PERMISSION	A permanent file requires proper access permissions. Modify (MD) permission is required for any updating operation.	Attach the file with the required modify permission.	F
302	NO EXTEND OR ALLOCATE PERMISSION	A permanent file requires extend (EX) permission before new records can be inserted.	Attach the file with the required extend permission.	F
304	NOT ALLOWED TO CREATE OVERFLOW BLOCKS (DA)	The OVF option selected requires original home blocks to accommodate all records. New records are ignored because all home blocks are full.	Change the OVF option if overflow blocks can exist.	T
324	PROCESSING DIRECTION NOT CONSISTENT WITH REQUEST	A file opened for INPUT cannot be written; a file opened for OUTPUT cannot be read.	Correct the inconsistency between the PD field and the input/output operation.	F
333	ILLEGAL CALL TO DIAGNOSTIC ROUTINE	An unexpected jump to a diagnostic routine has occurred.	Notify a system analyst.	F
334	TOTAL OF OPEN FILES NOT EQUAL TO TOTAL OF FIT ADDRESSES	The system has destroyed system tables.	Reload the program or notify a system analyst.	F
335	HIERARCHY TABLE OVERFLOW	Index level has increased too rapidly for AAM; update operation has not been performed.	For extended indexed sequential files, close and reopen the file. For other files, rerun the program starting with the update transaction that caused the overflow.	T/F
336	BAD FIT ADDRESS	The user or the system has destroyed system tables.	Correct the program and reload it.	F
337	INTERNAL ERROR IN IS/DA/AK 1.X	An internal error has been detected.	Notify a system analyst.	F
345	INSUFFICIENT CMM SPACE AVAILABLE	Not enough CMM space exists to open the file. To open a file requires enough free CMM space to load any rare capsules required, and to allow two of the largest blocks to be in memory at the same time. The file is not opened.	Release some CMM, if any is being used by the user program, or increase the amount of memory available to the job.	T
346	CMM NOT AVAILABLE AND THERE IS NO LIST OF FILES ADDRESS	A new block for the list-of-files cannot be allocated, and the LOF$RM entry point has been cleared.	Correct the program so that the pointer is not destroyed. A default list with 65_8 entries is supplied.	F
347	FDL ERROR CODE . . . ON CAPSULE . . .	Either CMM is not loaded when FDL is called to load a capsule or the AAMLIB file is not valid.	Check the load sequence or map to see if CMM is loaded. Fix the static load calls to load the proper routines. If using local libraries, check for a valid AAMLIB file.	T
352	FILE TO BE CLOSED IS NOT KNOWN	The logical file name specified does not match any existing file.	Check that the logical file name is correctly specified.	T

TABLE B-2. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
354	BUFFER SPACE SUPPLIED IS INSUFFICIENT FOR I/O	A buffer specified by the BFS field must be large enough to hold at least the larger of one block specified by MBL+2 or one physical record unit for the file's resident device.	Increase the BFS value.	T
355	CODE MODULES REQUIRED FOR I/O NOT LOADED	Routines necessary for processing have not been loaded.	Refer to appendix E for the correct loading procedures.	T
356	TRIVIAL ERROR LIMIT REACHED	Error count ECT equals the user-defined error limit ERL, resulting in a fatal error.	Correct the errors.	F
357	UNABLE TO OBTAIN SPACE FOR BUFFER	Required space cannot be allocated. CMM is not available and the FWB field is zero.	Supply a value for the FWB field or delete the OMIT=CMM parameter.	F
370	FATAL I/O ERROR	Either a block with an incorrect length was encountered or the operating system detected an error in the file or in the way the file was being used.	Correct the program.	F
372	FO=IS INDEX STRUCTURE FULL 15 LEVELS	The extended indexed sequential file has filled 15 levels of indexing, which is the maximum allowed. Further updating is not permitted.	Reorganize the file to allow more indexes per block.	F
403	SKIPBL DISALLOWED	A backward skip is not possible for D, R, and T type records.	Correct the program.	T
404	SKIPFL DISALLOWED FOR RT=U	No forward record skip is possible for U type records.	Correct the program.	T
415	ONLY PUT ALLOWED DURING INITIAL CREATION	During file creation, only PUT macros are valid between open and close.	Correct the program to eliminate all macros except PUT.	T
417	CANNOT REPLACE WITH LARGER RECORD IN SEQUENTIAL MODE	The REPLACE statement is ignored.	Correct the program.	T
420	CANNOT REWIND NO-REWIND FILE	The N parameter of the OPENM macro is meaningless because the initial indexed sequential file position is at the start of user records when the file is opened.	Remove the N parameter from the OPENM macro.	T
421	WSA NOT SPECIFIED – REQUEST IGNORED	For read or write, the location of the record in the user field length is required.	Specify the WSA field for the read or write operation.	T
422	SEEK NOT ALLOWED IN SEQUENTIAL MODE	The SEEK macro is ignored because it is not allowed during sequential processing.	Close and reopen the file for random processing if SEEK is desired.	T
424	CANNOT GET IN SEQUENTIAL MODE – GETN ASSUMED	The GET macro cannot be used in sequential mode.	Use the GETN macro.	T
425	CANNOT SKIP BACKWARD IN SEQUENTIAL MODE	The SKIP macro is ignored because backward skips are not allowed in sequential mode.	Correct the program.	T

TABLE B-2. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
426	GETN NOT ALLOWED DURING FILE CREATION – REQUEST IGNORED	On a file creation run, only the PUT macro is allowed between open and close.	Correct the program to eliminate all macros except PUT.	T
427	GET, SEEK INVALID IN SEQ MODE	Opening an indexed sequential file for INPUT establishes a sequential mode of operation in which access by key is prohibited.	Open the file for input/output if GET and SEEK are desired.	T
430	INVALID OP FOR READ ONLY	Read-only mode has been selected for an initial indexed sequential file; updating with PUT, REPLACE, or DELETE is not possible.	Change the PD field and read-only mode.	F
431	SKIP OR GETN WHEN SEEKING	A SEEK sequence on an indexed sequential file must be completed through EOR return to the FP field.	Correct the program.	T
441	MAJOR KEY WITH SYMBOLIC KEYS ONLY	Key type (KT) must be S for major key actions.	Correct the KT field.	F
442	INVALID ACTUAL KEY – REQUEST IGNORED	The key is not valid; the request is ignored.	Correct the KA field.	T
443	COMP-1 KEY HAS INCONSISTENT BIAS – REQUEST IGNORED	The COMP-1 key has been specified incorrectly (initial indexed sequential files).	Correct the COMP-1 key.	T
444	NEW KEY LESS THAN PREVIOUS KEY IN INITIAL CREATION	Records should be sorted by ascending key before an indexed sequential file is created. An out-of-order key is ignored.	Sort the records into ascending sequence.	T
445	KEY NOT FOUND – FILE POSITION ALTERED – REQUEST IGNORED	The entire file was searched, but the key does not exist. File position is one record ahead of the position where the search began.	No action is required.	T
446	DUPLICATE KEY FOUND – FILE POSITION ALTERED – REQUEST IGNORED	A duplicate key has been found. The request is ignored (initial indexed sequential files).	Change the duplicate key indicator if duplicate keys are allowed, or check the key field of the current record.	T
447	KEY ADDRESS NOT SPECIFIED – REQUEST IGNORED	The file cannot be read randomly if a key is not given.	Correct the program to specify the key address (KA) field.	T
452	FILE POSITIONING ERROR	An attempt was made to position the file beyond EOI.	Correct the program to check the FP field or specify the DX field.	F
501	INDEX FILE NOT COMPATIBLE WITH CRM FILE	Information in the file statistics table for a multiple-index file does not agree with index file information.	Check that the proper index file has been specified.	T
502	SPECIFIED KEY NOT DEFINED	The key position specified by the RKW, RKP, and KL fields for an alternate key does not correspond to an alternate key definition in the index file.	Correct the RKW, RKP, or KL field.	T
503	DUPLICATE ALTERNATE KEY ERROR	All alternate key values must be unique if the index structure for a multiple-index file has been specified as unique.	Specify indexed sequential structure if more than one alternate key is to have the same value.	T

TABLE B-2. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
504	SEQUENTIAL OPERATION BEYOND EOI ATTEMPTED	End-of-information has been encountered. No further sequential operations, such as GETN or a system search for a key, are possible until the index file is repositioned by a user statement.	Correct the program.	T
505	ERROR IN RMKDEF PARAMETER	The parameters used with the RMKDEF macro have been specified incorrectly.	Check that letters and digits appear properly; also, that the file name given in RMKDEF corresponds to the name of the file.	F
506	ALTERNATE KEY NOT FOUND	A key value specified does not match any alternate key value in the index file.	Action depends on program processing of keys.	T
507	***AAM MALFUNCTION n ***	For an extended indexed sequential file, an impossible condition has been encountered. This condition probably occurred when part of the executable code of AAM was altered by an agency other than AAM. The code n specifies the condition that has occurred: n=1 FIAAT POSKEY1 bad n=2 FIAAT POSKEY3 bad - FIFO n=3 Intermediate block reached with all keys too low n=4 Attempt to go up from primary n=5 Error in removing one level of hierarchy n=6 Compression buffer size bad n=7 Running total of CMM too high n=10 Index file not opened n=11 Attempt to use a busy FIX cell n=12 Attempt to chain an already chained block n=13 Attempt to read or write PRU 0 n=14 Attempt to write a block being read n=15 UBS free block count bad n=16 Attempt to unchain block not chained n=17 Empty count less than zero	Notify a system analyst.	F
510	INTERNAL ERROR IN MIP 1.X	An internal error has been encountered in initial MIP.	Notify a system analyst.	F

TABLE B-2. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
511	RMKDEF ONLY AFTER OPEN-NEW – IGNORED	The RMKDEF macro can be used only on a creation run.	Correct the program.	T
512	CRM DATA FILE MODIFICATIONS ILLEGAL WITH NDX=YES	If NDX is set to YES, the PUT, DELETE, and REPLACE macros are not allowed.	Correct the program.	T
513	REQUIRED ROUTINES NOT LOADED – RM$MEXB/RM$MFSQ	Read-only processing of multiple-index files requires the LDSET control statement or LDREQ macro (initial indexed sequential and direct access files).	Supply the LDSET control statement or LDREQ macro.	T
514	FILE CONTAINS DUPLICATE PRIMARY KEYS	Alternate keys are not permitted when duplicate primary keys are defined (initial indexed sequential files).	The file cannot be a multiple-index file.	F
515	NO INDEX FILE SPECIFIED	No name has been specified for the XN field on an IXGEN or file creation run for a multiple-index file.	Specify an index file for the XN field.	F
520	CHANGED KEY TYPE	The key type (KT) specified on the file creation run cannot be changed for the life of the file.	Change the KT field.	F
521	CHANGED KEY SIZE	The key length (KL) specified on a file creation run cannot be changed for the life of a file.	Change the KL field.	F
522	KEY TYPE INCORRECT	For an initial indexed sequential file, KT must be S (symbolic), I (integer), or F (floating point number).	Correct the KT field.	F
523	NO KEY DEFINED	Key type (KT), key length (KL), and key address (KA) must be defined.	Define the key fields.	F
524	KEY SIZE ILLEGAL	For initial indexed sequential files, the integer key must be 5 or 10 characters; floating point number keys must be 10 characters; symbolic keys can be 1 to the installation-defined length limit.	Correct the KL field.	F
525	MAJOR KEY SIZE ILLEGAL	MKL must be at least 1 and less than the full key defined by KL.	Correct the MKL field.	F
526	HASHED KEY OUTSIDE HOME BLOCK AREA	The user has changed hashing routines. The hashing routine in use is limited in the range of keys that it can successfully process.	Check the HRL field in the FIT to verify that the correct hashing routine is in use; otherwise, the user should limit the selection of keys to a narrower range.	F
527	ATTEMPT TO REDEFINE SPARSE CONTROL CHARACTER	An RMKDEF directive attempted to redefine the sparse control character (extended indexed sequential files).	Correct the RMKDEF directive.	F
530	PADDING FACTOR OUT OF RANGE	Padding can be specified as 0 to 99 percent.	Correct the padding percentage.	F

TABLE B-2. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
532	FILE ALREADY EXISTS, CANNOT OPEN-NEW	Two files in one program cannot have the same name.	Check the PD field in the FIT or the pd OPENM parameter. The ON field must be changed from NEW for file access after creation run.	F
534	MRL EXCEEDS MAX ALLOWED RECORD SIZE	The value of the MRL field is greater than 81870 characters. The file is not opened.	Correct the MRL field in the FIT.	T
535	NO DECOMPRESSION ROUTINE SUPPLIED	A value for the DCA field was not supplied on OPENM OLD for a file that has user compression. The file is not opened (extended indexed sequential files).	Correct the DCA field in the FIT.	T
536	NO OR WRONG COMPRESSION ROUTINE SUPPLIED	A value for the CPA field was not supplied on OPENM OLD for a file that has user compression. The file is not opened (extended indexed sequential files).	Correct the CPA field in the FIT.	T
540	FIFO KEY SUB-STRUCTURE NOT ALLOWED IN REPEATING GROUPS	For alternate keys in repeating groups, the key must be unique or stored in an indexed sequential substructure.	Correct the RMKDEF directive.	F
541	PURGE ILLEGAL – SPECIFIED ALT KEY NOT KNOWN	An attempt was made through MIPGEN to purge an alternate key that did not exist.	Correct the MIPGEN RMKDEF directive.	F
542	NEW KEYDEF MATCHES ONE ALREADY KNOWN – KEYDEF REJECTED	The key was not defined to be unique.	Correct the MIPGEN RMKDEF directive.	F
543	FILE NOT POSITIONED IN DUPLICATE KEY SET – CANNOT DELETE CURRENT	The file is not positioned at a duplicate key set, the DELETE current is not honored (initial indexed sequential files).	Position the file to a duplicate key set.	T
544	PADDING REQUESTED TOO LARGE	The padding percentage requested would not allow the data block to contain one maximum record on create or would not allow three index records per index block. The file is not opened.	Correct the PD or IP field in the FIT and reopen the file.	T
545	CANT OPEN NEW FOR INPUT	The processing direction must be set to OUTPUT on a file opened as a new file.	Correct the PD or ON field in the FIT and reopen the file.	T
546	PRIMARY KEY NOT FOUND	A primary key in the alternate key index file cannot be found in the data file. The data file and index file have been modified inconsistently.	Disassociate the data file and create a new index file using the MIPGEN utility.	F
547	BAD STRUCTURE FOUND IN FILE	The block being looked at contains an impossible counter or pointer.	Notify a system analyst.	F
550	CANNOT COMPRESS – KEY POSITION INVALID	To compress records, the primary key must either be nonembedded or begin in the first character position. The file is not opened.	Change the key position if the file is to be compressed.	T

TABLE B-2. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
551	REL MUST BE EQ, GT OR GE	An invalid REL value was detected. The operation is not performed.	Set the REL field to a correct value.	T
712	NEGATIVE OR OVER-SIZED ARGUMENT--WSA, SKP, OR LA	One of the parameters indicated was erroneously specified when a macro was issued.	Correct the program.	F
713	NEGATIVE OR OVER-SIZED ARGUMENT--RL, ST, OR LBL	One of the parameters indicated was erroneously specified when a macro was issued.	Correct the program.	F
714	NEGATIVE EX OR DX PARAMETER	A negative value was specified for the EX or DX field.	Correct the program.	F
715	NEGATIVE OR OVER-SIZED ARGUMENT--WA OR KA	Either the WA or KA field was erroneously specified.	Correct the program.	F
716	NEGATIVE OR OVER-SIZED ARGUMENT--PTL OR KP	Either the PTL or the KP field was erroneously specified.	Correct the program.	F
717	NEGATIVE OR OVER-SIZED ARGUMENT--MKL, POS, GPS, OR TRM.	One of the parameters indicated was erroneously specified when a macro was issued.	Correct the program.	F
720	DEVICE CAPACITY EXCEEDED	The CIO read driver has encountered an error.	Check the system dayfile for the specific head driver error.	T
721	ERROR DETECTED BY OPERATING SYSTEM	A system hardware error that cannot be corrected has been encountered.	Check the system dayfile for a system/hardware error message.	T

TABLE B-3. NOTES OR INFORMATIVE MESSAGES

Code	Message	Code	Message
1000	FIT DUMP	1025	DATA BLOCK SIZE AND BLOCKING FACTOR BOTH SET
1001	FILE OPENED	1026	EOI ENCOUNTERED ON SKIP OR GETN
1002	FILE CLOSED	1027	THE KEY IS . . . . .
1003	NUMBER OF INDEX LEVELS	1030	ERROR ENCOUNTERED DURING . . . . .
1004	***NUMBER OF GETS THIS OPEN	1031	One of many general comments output by AAM routines
1005	***NUMBER OF PUTS THIS OPEN	1032	THE KEY IS . . . . . THE KEY IN OCTAL IS . . . . .
1006	***NUMBER OF REPLACES THIS OPEN	1033	***NUMBER OF GET NEXTS THIS OPEN . . . . .
1007	***NUMBER OF DELETES THIS OPEN	1034	***NUMBER OF ACCESSES THIS OPEN . . . . .
1010	***TOTAL DISKAREA*** . . . . . WORDS	1035	***TOTAL NUMBER OF RECORDS . . . . .
1011	GETN REACHED EOI	1036	***TOTAL NO. OF OVERFLOW RECORDS . . . . .
1012	SKIP REACHED FILE BOUNDARY BEFORE EXHAUSTING SKIP COUNT	1037	***NO. OF AVAILABLE PRIMARY INDEX ENTRIES . . . . .
1013	END OF INFORMATION ENCOUNTERED	1040	***RECORDS/HOME-BLK CREATED THIS OPEN . . . . .
1014	BEGINNING OF INFORMATION ENCOUNTERED	1041	***RECORDS/OVF-BLK CREATED THIS OPEN . . . . .
1015	FILE LIMIT REACHED, LINEAR SEARCH FOR SPACE INITIATED	1042	***OVERFLOW BLOCKS CREATED THIS OPEN . . . . .
1016	ILLOGICAL SUCCESSIVE SEEK REQUESTS	1043	***TOTAL NUMBER OF HOME BLOCKS . . . . .
1017	CANNOT CHECKSUM A FILE CREATED WITHOUT CHECKSUMS	1044	***TOTAL NUMBER OF HOME BLOCKS IN USE . . . . .
1020	ILLOGICAL TO CHANGE THE KEY BEFORE SEEK FUNCTION COMPLETED	1137	THE FOLLOWING BLOCK CONTAINS A PARITY ERROR . . . . .
1021	HOME BLOCKS EMPTY--HASHING ROUTINE NOT VERIFIED		
1022	DELETED LAST RECORD		
1023	EMPTY FILE OPENED		
1024	IS ERROR RECOVERY		

GLOSSARY C

AAM (ADVANCED ACCESS METHODS) – A file manager that processes indexed sequential, direct access, and actual key file organizations and supports the Multiple-Index Processor.

ACTUAL KEY – The primary key for a record in a file with actual key organization, which specifies the block number and record position in that block. These keys are usually generated by AAM and returned to the user.

ACTUAL KEY (AK) FILE – A mass storage file in which each record is stored at the location specified by the block and record slot number in the primary key associated with that record.

ALTERNATE KEY – A key other than the primary key by which an indexed sequential, direct access, or actual key file can be accessed.

BAM (BASIC ACCESS METHODS) – A file manager that processes sequential and word addressable file organizations.

BEGINNING-OF-INFORMATION (BOI) – The start of the first user record in a file.

BLOCK – A logical or physical grouping of records to make more efficient use of hardware. All files are blocked. See also Data Block, Home Block, Index Block, and Overflow Block.

BLOCK CHECKSUM – A number used to check that the contents of a data block have not been altered accidentally; a means of ensuring data integrity. Block checksums can be requested for files through use of the BCK parameter in the FILE control statement.

CHARACTER – A letter, digit, punctuation mark, or mathematical symbol forming part of one or more of the standard character sets. Also, a unit of measure used to specify block length, record length, and so forth.

CLOSE – A set of terminating operations performed on a file when input and output operations are complete. All files processed by AAM must be closed.

COMPRESSION – The process of condensing a record to reduce the amount of storage space required. The user can supply a compression routine or use a system-supplied routine. See Decompression.

CREATION RUN – All processing of a file, from open to close, the first time the file is written or made into an AAM file. Files must be created in a separate creation run during which only write operations on the file being created are allowed.

CRM (CYBER RECORD MANAGER) – A generic term relating to the common products BAM and AAM.

DATA BLOCK – A block in which user records are stored in an indexed sequential or actual key file. Data block structure is defined by the user, or AAM defaults are accepted. Contrast with Index Block for indexed sequential files.

DECOMPRESSION – The process of expanding a compressed record to restore it to its original size. The user can supply a decompression routine or use a system-supplied routine. See Compression.

DECRYPTION – The process of condensing and reformatting an encrypted record to restore it to its original size and format. The user supplies a decryption routine. See Encryption.

DEFAULT – A value assumed in the absence of a user-specified value declaration for the parameter involved. Values for many defaults are defined by the installation.

DIRECT ACCESS (DA) FILE – A file containing records stored randomly in home blocks according to the hashed value of the primary key in each record. Files must be mass storage resident. All allocation for home blocks occurs when the file is opened on its creation run. Access is random or sequential.

DIRECTIVES – The instructions that supplement processing defined by a control statement or by a program call for execution of a utility function or member of a product set. Directives do not appear in the control statement record; they are usually in a separate record of the file INPUT or a file referenced in a control statement call. Directives are required for execution of FORM, the CREATE utility, and EDITLIB among others.

ENCRYPTION – The process of expanding and reformatting a record. The user supplies an encryption routine. See Decryption.

END-OF-INFORMATION (EOI) – The end of the last user record in a file.

FIELD – A portion of a word or record; a subdivision of information within a record; also, a generic entry in a file information table identified by a mnemonic.

FIELD LENGTH – The area in central memory allocated to a particular job; the only part of central memory that a job can directly access. Contrasts with mass storage space allocated for a job and on which user's files reside.

FILE – A logically related set of information; the largest collection of information that can be addressed by a file name. It starts at beginning-of-information and ends at end-of-information. Every file in use by a job must have a logical file name.

FILE CONTROL STATEMENT – A control statement that supplies file information table values after a source language program is compiled or assembled but before the program is executed. In applications such as those with a control statement call to the FORM utility, a FILE control statement must be used. Basic file characteristics such as organization, record type, and description can be specified in the FILE control statement.

FILE INFORMATION TABLE (FIT) – A table through which a user program communicates with AAM. For direct processing through AAM, a user must initiate establishment of this table. All file processing executes on the basis of information in this table. The user can set FIT fields directly or use parameters in a file access call that sets the fields indirectly. Some product set members set the fields automatically for the user.

FILE STATISTICS TABLE (FSTT) – A table generated and maintained by AAM to collect statistics about each file. The FSTT is a permanent part of a file and contains information such as organization type, size of blocks, number of current accesses, and so forth.

FLUSHING – The method of processing file buffers and updating the file statistics tables as if close operations had been requested without actually closing the files.

HASHING – The method of using primary keys to search for relative home block addresses of records in a file with direct access storage structure.

HOME BLOCK – A block in a file with direct access storage structure whose relative address is computed by hashing keys. A home block contains synonym records whose keys hash to that relative address. If all the synonym records cannot be accommodated in the home block, an overflow block can be created by the system. A user creating a direct access file must define the number of home blocks with the HMB parameter in the FILE control statement.

INDEX – A series of keys and pointers to records associated with the keys.

INDEX BLOCK – For an indexed sequential file, a block with ordered keys and pointers to the data blocks and other index blocks, forming a directory of the records within a file.

INDEXED SEQUENTIAL (IS) FILE – A file organization in which AAM maintains files in sorted order by use of a user-defined primary key, which need not be within the record. Keys can be integer, floating point (initial indexed sequential files only), or symbolic; access is random or sequential. Files contain index blocks and data blocks.

INSTALLATION OPTION – One of several alternate means of processing that is selected when AAM is installed at a computer installation. Once an option is selected, all subsequent use of AAM is governed by the selection. For all options or limits defined as installation options, the user should consult with a system analyst to determine the valid limits.

INTEGER KEY – A binary key used with indexed sequential files; for initial indexed sequential files, either 30 or 60 bits in length; for extended indexed sequential files, a 60-bit signed binary key. See Symbolic Key.

KEY – A group of contiguous characters or numbers the user defines to identify a record in an AAM file.

KEY ANALYSIS UTILITY – A utility program that provides information about hypothetical record distribution for a file with direct access organization. The utility reads the key of each record in the file and determines the home block where the record would reside.

KEY ENTRY – The format of key information in index and data blocks in an initial indexed sequential file. The user need not be concerned with entries, except to realize that symbolic key length and integer key values can be specified to minimize entry length for improved efficiency.

LDSET – The loader control statement. Various parameters include:

LIB	Make available the named library
USE	Load the routines named
STAT	Static loading requested
OMIT	Inhibit loading of the routines named

LOAD SET – A group of loader control statements beginning with a call that causes information to be loaded into central memory and ending with a call for execution of a loaded program. Nonloader statements must not appear in a load set.

LOGICAL FILE NAME – The name given to a file being used by a job. The name must be unique for the job and must consist of one to seven letters or digits, the first of which must be a letter.

MACRO – A single instruction that when compiled into machine code generates several machine code instructions.

MAINTENANCE RUN – A program or job to update an existing file; technically refers to that part of the job from file open to file close.

MAJOR KEY – The leading characters of a symbolic key in an indexed sequential file.

MASS STORAGE – A disk pack that can be accessed randomly. ECS is not considered mass storage.

MASTER FILE – A file containing information about a set of entities. All information about a single entity constitutes a record in the file. A master file is normally kept up to data by a maintenance run.

MULTIPLE-INDEX FILE – An indexed sequential, direct access, or actual key file that has alternate keys defined.

MULTIPLE-INDEX PROCESSOR (MIP) – A processor that allows AAM files to be accessed by alternate keys.

OPEN – A set of preparatory operations performed on a file before input and output can take place; required for all AAM files.

OVERFLOW BLOCK – A block added to the file by AAM for use when the home blocks in a direct access file are full.

OWNCODE – A routine written by the user to process certain conditions. Control passes automatically to user owncode routines defined in the FIT for:

DX	End-of-data condition
EX	Error condition

PADDING – The free space reserved in a file at creation time to accommodate additional records; specified as a percentage figure.

PERMANENT FILE – A file on a mass storage permanent file device that can be retained for longer than a single job. It is protected against accidental destruction by the system and can be protected against unauthorized access.

PHYSICAL RECORD UNIT (PRU) – The smallest unit of information that can be transferred between a peripheral storage device and central memory. The PRU size is permanently fixed for all mass storage devices.

PRIMARY KEY – A key that must be defined for a file when the file is first created.

PRU DEVICE – A mass storage device in which information has a physical structure governed by physical record units (PRUs).

RANDOM ACCESS – Access method by which any record in a file can be accessed at any time in any order; applies only to mass storage files. See Sequential Access.

RECORD – The largest collection of information passed between AAM and a user program in a single read or write operation. The user defines the structure and characteristics of records within a file by declaring a record format. The beginning and ending points of a record are implicit in each format.

RECORD SLOT NUMBER – The position of a record within a block in an actual key file; specified by the low-order bits of the primary key.

RELEASE SYSTEM – A software system delivered to a customer. In installing a system, the customer, but not an individual applications programmer, can use default values or parameters that differ from the release system.

REWIND – To position a file at beginning-of-information.

SEQUENTIAL ACCESS – A method in which only the record located at the current file position can be accessed. See Random Access.

SPARSE KEY – An alternate key that is used infrequently. Only those alternate key values of interest are included in the index file.

SYMBOLIC KEY – An alphanumeric key used with indexed sequential files; 1 to 255 characters. See Integer Key.

SYNONYM RECORDS – Direct access file records whose primary keys hash to the same home block.

WORKING STORAGE AREA – An area within the user's field length intended for receipt of data from a file or transmission of data to a file.

A file information table (FIT) must be associated with every file that uses AAM. For normal language requirements, compilers generate the FIT automatically; users writing in high level languages need not be concerned with the FIT and its generation. The COMPASS user is responsible for supplying the FIT; the FILE macro is provided to create the FIT. Word and bit designations are illustrated in figure D-1.

The FIT is activated by an OPENM request for the file. After the file is opened, FIT fields can be updated with the FILE control statement or the STORE macro, with information from the processing macros, or by AAM as a result of processing the file. Information in the FIT can be retrieved with the FETCH macro. In figure D-1, the fields enclosed in parentheses can be accessed by the FETCH macro but cannot be changed. If a STORE macro is attempted on these fields, an assembly diagnostic results.

The FIT fields are listed in this appendix by word and bit position. For the convenience of the user, the COMPASS symbols are included with the applicable FIT field values. Generally, any particular file organization or record type requires only a small portion of the total information specified here. The first ten words of the FIT are used by AAM for communicating with the operating system.

Word 0

59-18	LFN	Logical file name of the data file.
17-1		Reserved for CDC.
0	CMPLT	FET complete bit; cannot be changed by the user.

Word 1

59-48	DVT	FET device type; cannot be changed by the user.
47		Reserved for CRM.
46	RDR	Read release.
45-37		Reserved for CDC.
36	FF	OS flush on abnormal termination: 0 Buffer not flushed. 1 Buffer flushed for output file with scratch disposition on abnormal termination.
35-30		Reserved for CDC.
29-24	DC	Disposition code; cannot be changed by the user. Refer to operating system manual for possible settings.
23-18		Length of FIT minus 5; set to 30_{10}.
17-0	FWB	First word address of the user buffer.

Word 2

59-18		Zero-filled field.
17-0		Reserved for CRM.

Word 3

59-18		Zero-filled field.
17-0		Reserved for CRM.

Word 4

59-34		Reserved for CDC.
33-0		Reserved for CRM.

Word 5

59-24		Reserved for CRM/INTERCOM.
23-22	ASCII	ASCII character set bits for INTERCOM terminals (BAM only).
21-0		Reserved for CRM.

Word 6 Reserved for CDC.

Word 7 Reserved for CRM (return address stack).

Word 8 Reserved for CDC (FET extension).

Word 9 Reserved for CDC (label fields).

Word 10

59-36	LBL	Label area length in characters (BAM only).
35	LCR	Label check/creation for input/output tape (BAM only).
34		Reserved for CRM.
33-27	FP	File position (in octal); cannot be changed by the user: 0 Mid logical record 1 BOI Beginning-of-information ≡BOI≡ 2 BOF Beginning-of-file ≡BOF≡ 10 EOK End-of-keylist ≡EOK≡ 20 EOR End-of-record ≡EOR≡ 100 EOI End-of-information ≡EOI≡

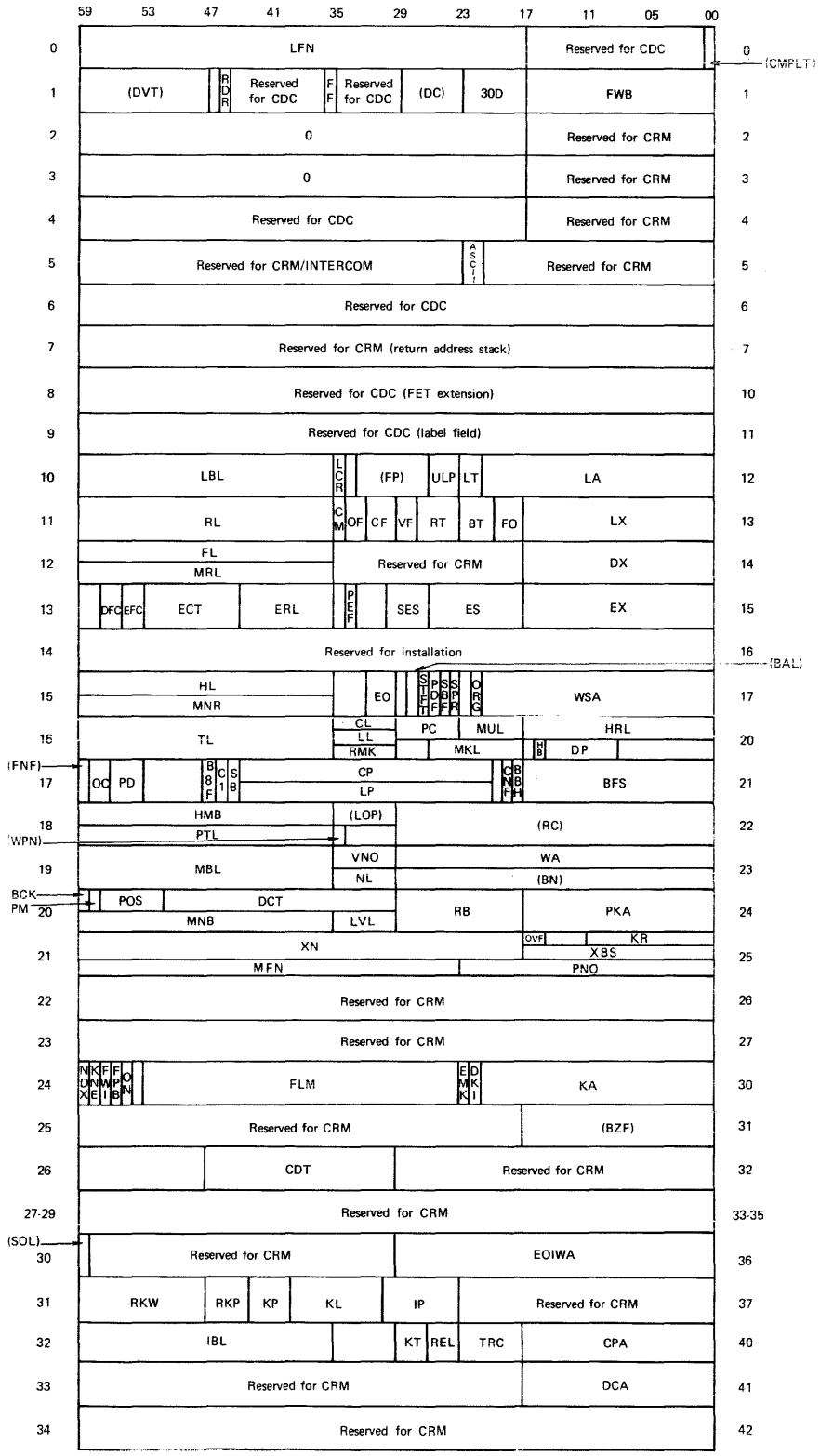


Figure D-1. File Information Table

26-24	ULP	User label processing (BAM only).
23-22	LT	Label type (BAM only).
21-0	LA	Label area address (BAM only).

Word 11

59-36	RL	Current record length in characters.
35	CM	Conversion mode (EC to IC) (BAM only)
34-33	OF	Open flag; positioning of the file at OPENM time:

00		Rewind (default)	≡≡
01	R	Rewind	≡R≡
10	N	No rewind	≡N≡
11	E	Extend	≡E≡

32-30	CF	Close flag; positioning of the file at CLOSEM time:

000		Rewind (default)	≡≡
001	R	Rewind	≡R≡
010	N	No rewind	≡N≡
011	U	Unload	≡U≡
100	RET	Return	≡RET≡
101	DET	Detach	≡DET≡
110	DIS	Disconnect (BAM only)	≡DIS≡

29-28	VF	End-of-volume flag (BAM only).
27-24	RT	Record type:

0000	W	Control word	≡WT≡
0001	F	Fixed length	≡FT≡
0010	R	Record mark	≡RT≡
0011	Z	Zero byte	≡ZT≡
0100	D	Decimal character count	≡DT≡
0101	T	Trailer count	≡TT≡
0111	U	Undefined	≡UT≡
1000	S	System-logical-record	≡ST≡

23-21	BT	Block type (BAM only).
20-18	FO	File organization:

000	SQ	Sequential (BAM only)	≡SQ≡
001	WA	Word addressable (BAM only)	≡WA≡
011	IS	Indexed sequential	≡IS≡
101	DA	Direct access	≡DA≡
110	AK	Actual key	≡AK≡

17-0	LX	Label routine exit address (BAM only).

Word 12

59-36	MRL	Maximum record length in characters; when retrieving primary keys from an alternate key index, working storage area length in characters.
	FL	Fixed length of an F type record, or full length of a Z type record, in characters.
35-18		Reserved for CRM.
17-0	DX	End-of-data exit address.

Word 13

59-58		Reserved for CRM.
57-56	DFC	Dayfile control for error messages:

0	No dayfile messages except fatal errors
1	Error messages to dayfile
2	Statistics/notes to dayfile
3	Errors and statistics/notes to dayfile

55-54	EFC	Error file control:

0	No error file messages
1	Error messages to error file
2	Statistics/notes to error file
3	Errors and statistics/notes to error file

53-45	ECT	Trivial error count.
44-36	ERL	Trivial error limit.
35		Reserved for CRM.
34	PEF	Parity error flag (BAM only).
33-31		Reserved for CRM.
30-27	SES	System parity error severity (BAM only).
26-18	ES	Error status (octal value).
17-0	EX	Error exit address.

Word 14	Reserved for installation.

Word 15

59-36	HL	Header length in characters; T type records.
	MNR	Minimum record length.
35-33		Reserved for CRM.
32-30	EO	Error option (BAM only)
29		Reserved for CRM.

28 BAL Buffer allocated by CRM; cannot be changed by the user.

27 STFT Internal SETFIT flag used for CRM processing.

26 PDF SETFIT macro FILE statement flag; cannot be changed by the user.

25 SBF Suppressed buffer I/O flag (BAM only).

24 SPR Suppress read ahead (BAM only).

23 Reserved for CRM.

22 ORG Old/new file organization:

0	OLD	Initial indexed sequential file organization	≡OLD≡
1	NEW	Extended indexed sequential file organization	≡NEW≡

21-0 WSA Working storage area address.

Word 16

59-36 TL Trailer length in characters; T type records.

35-30 CL Count field length in characters; T type records.

LL Length field length in characters; D type records.

RMK Record mark character; R type records.

29-24 PC Padding character (BAM only).

23-18 MUL Multiple of characters per K or E type block (BAM only).

26-18 MKL Major key length in characters (indexed sequential files).

17-0 HRL Hashing routine address (direct access files).

16 HB User header option (actual key files):

0 Do not return header

1 Return header (default)

15-9 DP Data block padding percent (indexed sequential and actual key files).

Word 17

59 FNF Fatal/nonfatal flag; cannot be changed by the user:

0 Nonfatal

1 Fatal

58-57 OC Open/close flag:

00	Never opened	≡NOP≡
01	Opened	≡OPE≡
10	Closed	≡CLO≡

56-54 PD Processing direction:

000		Input (default)	≡≡
001	INPUT	Input	≡INPUT≡
010	OUTPUT	Output	≡OUTPUT≡
011	IO	Input/output	≡IO≡

53-48 Reserved for CRM.

47 B8F Round PUTs down to *8 bits (BAM only).

46 C1 COMP-1; format for the CL/LL field; T or D type records:

0	NO	Display code	≡NO≡
1	YES	Binary	≡YES≡

45 SB Sign overpunch; overpunch option for CL/LL field; T or D type records:

0	NO	No overpunch	≡NO≡
1	YES	Overpunch	≡YES≡

44-21 CP Trailer count beginning character position (numbered from 0); T type records.

LP Length field beginning character position (numbered from 0); D type records.

20 Reserved for CRM.

19 CNF Connected file flag (BAM only).

18 BBH Buffer below highest high address (BAM only).

17-0 BFS Buffer size in words.

Word 18

59-36 HMB Number of home blocks (direct access files).

PTL Partial transfer length (BAM only); number of keys moved to working storage area for a GET or GETN on an alternate key index.

35-30 LOP Last operation code; cannot be changed by the user (BAM only).

35 WPN Write bit; the upper bit of LOP is a 1-bit subfield that can be accessed separately; cannot be changed by the user:

0 Last operation was not a write

1 Last operation was a write

29-0 RC Record count; count of full records read or written since the file was opened. The count is not adjusted for repositioning and backspacing operations. For a multiple-index file, the number of records with this alternate key value. This field cannot be changed by the user.

Word 19

59-36 MBL Maximum block length in characters.

35-30 VNO Current volume number of the multi-volume sequential file (BAM only).

NL Number of levels of index blocks (indexed sequential files).

29-0 BN Block number of the current block (sequential files); cannot be changed by the user (BAM only).

WA Current position word address, set by GET and PUT macros (BAM only).

Word 20

59 BCK Block checksum:

0	NO	No checksumming of blocks	≡NO≡
1	YES	Checksumming of blocks	≡YES≡

58 PM Processing mode:

0	Random	≡RPM≡
1	Sequential	≡SPM≡

57-52 POS Duplicate key position (initial indexed sequential files):

0 First record in a duplicate key set

1 Current record

51-30 DCT Address of the display code to collating sequence conversion table (indexed sequential files).

59-36 MNB Minimum block length in characters.

29-18 RB Number of records per block (actual key files) or average number of records (indexed sequential and direct access files).

17-0 PKA Primary key address; address to receive primary key on an alternate key access (extended indexed sequential files).

Word 21

59-18 XN Logical file name of the alternate key index file associated with the data file.

17-0 XBS Index file block size (extended indexed sequential files).

59-24 MFN Multifile set name (BAM only).

23-0 PNO Multifile position number (BAM only).

17-16 OVF Direct access file overflow flag:

01	OVO	Overflow blocks only	≡OVO≡
10	OVB	Either overflow or home blocks	≡OVB≡
11	OVH	Home blocks only	≡OVH≡

11-0 KR Key value repeat count; number of times the key value repeats in the current record (initial multiple-index files).

Word 22

59-46 Reserved for CRM.

45-40 LAC Last action performed on the file; used by compiler languages to communicate with each other.

39-36 LNG Last compiler language to have used the file:

0 Unknown

1 COBOL

2 FORTRAN

3 PL/I

4-7 Reserved

35-0 Reserved for CRM.

Word 23 Reserved for CRM.

Word 24

59 NDX Index flag:

0 Data file is accessed

1 Index file is accessed

58 KNE Key not equal (multiple-index files):

0 Key match found

1 No key match found

57 FWI Forced write indicator:

0	NO	No forced write	≡NO≡
1	YES	Forced write	≡YES≡

56 FPB File position bit (system routine use only); or EOI reached random operation (multiple-index files):

0 EOI not reached

1 EOI reached

55 ON Old or new file:

0	OLD	Old file	≡OLD≡
1	NEW	Creation run	≡NEW≡

54 Reserved for CRM.

53-24 FLM File limit, records per file.

23 EMK Embedded key flag (extended indexed sequential files):

0	NO	Key is not part of the record	≡NO≡
1	YES	Key is included in the record	≡YES≡

22	DKI	Duplicate key indicator; indicates duplicate primary key permission (initial indexed sequential files): 0 No duplicate keys 1 Duplicate keys allowed
21-0	KA	Key address of the key value for record processing.

Word 25

59-18		Reserved for CRM.
17-0	BZF	Busy FET address; cannot be changed by the user.

Word 26

59-48		Reserved for CRM.
47-30	CDT	Address of the collating sequence to display code conversion table (initial indexed sequential files).
29-0		Reserved for CRM.

Words 27-29 Reserved for CRM.

Word 30

59	SOL	S/L tape bit; cannot be changed by the user (BAM only).
58-30		Reserved for CRM.
20-0	EOIWA	End-of-information word address (BAM only).

Word 31

59-48	RKW	Relative key word (direct access files and alternate key access of multiple-index files).
47-44	RKP	Relative key position in RKW (direct access files and alternate key access of multiple-index files).
43-40	KP	Beginning character position of the key (indexed sequential and direct access files).
39-31	KL	Key length in characters (indexed sequential and direct access files). Key length in bits (actual key files).
		Primary or alternate key length in bits prior to open of a new multiple-index file; after open, length in characters (actual key files).
30-24	IP	Index block padding percent (indexed sequential files).
23-0		Reserved for CRM.

Word 32

59-42	IBL	Index block length in characters (initial indexed sequential files).
41-30		Reserved for CRM.
29-27	KT	Key type (indexed sequential files):

000		Symbolic (default)	
001	S	Symbolic (default)	≡SKT≡
010	I	Integer	≡IKT≡
011	F	Floating	≡FKT≡
011	U	Uncollated symbolic	≡UKT≡

26-24	REL	File position key relation (indexed sequential and multiple-index files):

1	EQ	Equal	≡EQ≡
2	LE	Less than or equal (initial files only)	≡LE≡
3	GE	Greater than or equal	≡GE≡
4	NE	Not equal (initial files only)	≡NE≡
5	LT	Less than (initial files only)	≡LT≡
6	GT	Greater than	≡GT≡

23-18	TRC	Trace transition count; number of transactions to be traced (initial indexed sequential and direct access files).
17-0	CPA	Compression routine address (extended indexed sequential files).

Word 33

59-18		Reserved for CRM.
17-0	DCA	Decompression routine address (extended indexed sequential files).

Word 34 Reserved for CRM.

AAM has been divided into functional capsules that are loaded by relocatable controlling routines at execution time. This method of dynamic loading requires a program to be compatible with the Common Memory Manager (CMM). Static loading is available for programs that are not compatible; however, static loading could involve a field length penalty of as much as 1400_8 words. AAM uses dynamic loading unless static loading is specified through a control statement or a macro.

More information about the Common Memory Manager and the CYBER Loader can be obtained from their respective reference manuals.

DYNAMIC LOADING

For dynamic loading, all AAM macros reference entry points in the controlling routines. The controlling routines, which process parameters and diagnose certain types of errors, are loaded at relocatable load time or overlay generation time. The controlling routines load and transfer control to the Fast Dynamic Loader (FDL) capsule containing the proper AAM controller in fixed-position fixed-length blocks. The controller then loads the FDL capsules needed to process the macro.

It is important to the dynamic loading scheme that the controlling routines not be overlayed. Unknown results, including bad jump addresses to service routines, occur if these routines are overlayed. To prevent the controlling routines from being overwritten, they must be part of the (0,0) overlay. This can be assured by specifying the FILE macro in the (0,0) overlay.

The OPENM/SETFIT capsule is loaded when the first OPENM or SETFIT macro is encountered. If the SETFIT macro occurs first, the FILE control statement parameters are processed, the dynamic AAM controller capsule is loaded, and control is transferred to that capsule. The required AAM processor capsule is then loaded, the buffer size is calculated, and control is returned to the user.

When the OPENM macro occurs before a SETFIT macro, the SETFIT functions are performed first. Open processing then occurs. The file is opened, FIT consistency checks are performed, and control is returned to the user. The open processing capsule is unloaded when a macro other than OPENM, SETFIT, STORE, or FETCH is encountered. For optimum efficiency in loading, the open processing for all files should be completed before other processing is specified. The AAM processor capsule remains loaded.

When the first macro that requires a buffer is encountered, a buffer is allocated through CMM in a fixed-position fixed-length block. The capsules required to perform the function specified by the macro are loaded; control transfers to the capsules and then back to the user. Generally, the capsules required to process these functions remain in memory until all files requiring them have been closed. Some capsules are loaded while a series of operations are being performed and are unloaded when additional memory space is needed to load another capsule.

The CLOSEM capsule is loaded when the CLOSEM macro is encountered. An additional AAM capsule might be loaded to close the file and release buffer space. The CLOSEM capsule unloads any capsules no longer needed for processing and unloads itself after closing the last file.

The AAM controller capsule, processing capsules, and dynamic buffers are loaded above the highest high address; however, they are not destroyed by overlay swapping. Because of this, it is possible to swap overlays without first closing the AAM files. When the file is other than an extended indexed sequential file and the first I/O processing overlay loaded is read-only, certain precautions are necessary. If the read-only capsules are loaded, a swap to another overlay doing an update might result in an error if the read-only file is not closed before the swap. The presence of the read-only capsule prevents the full processing capsule from being loaded.

AAM contains a trace function that is used primarily for debugging purposes with initial indexed sequential and direct access files. The processing for the trace function is contained in a separate capsule that is loaded only if the trace transaction count (TRC) field in the FIT is set to YES.

STATIC LOADING

Static loading is provided for the cases where the user is managing memory and the program cannot be compatible with CMM. It should only be used as a short term conversion aid. Long term support of static loading is not to be provided. Two methods are available for designating which capsules need to be statically loaded. One method is control statement oriented and the other method is macro oriented.

CONTROL STATEMENTS

Static loading can be specified through the LDSET and FILE control statements. The STAT option must be specified in the LDSET control statement and the USE and OMIT parameters must be specified in the FILE control statement. One FILE control statement must be included for each file to ensure that all necessary routines are loaded. The file organization (FO), record type (RT), and index file name (XN) parameters must be specified on the same or a previous FILE control statement as the USE and OMIT parameters. These three parameters cannot be specified in a FILE control statement following the one that specifies the USE and OMIT parameters.

The USE and OMIT parameters are formatted as follows:

$$USE=mn_1/mn_2/.../mn_n$$

$$OMIT=mn_1/mn_2/.../mn_n$$

In both parameter formats, mn is a macro name. The functions of the USE and OMIT parameters are listed in table E-1. The USE and OMIT parameters can be used in more than one FILE control statement; the results are cumulative. If the STAT option is specified in the LSDET control statement and the USE parameter is not specified in the FILE control statement, no processing capsules are loaded.

In the example shown in figure E-1, the program to write the file ISFILE uses static loading and contains the OPENM, PUT, and CLOSEM macros. The program to read the file ISFILE also uses static loading. The PUT macro is not contained in that program; the OMIT parameter specifies that the capsule for that macro is to be unloaded. The GET macro is contained in the program and the capsule for that macro is to be loaded. The USE parameter is still in effect for the OPENM and CLOSEM macros.

TABLE E-1. USE AND OMIT PARAMETER FUNCTIONS

Parameter	No List of Macros	List of Macros
USE	All capsules are loaded.	Capsules performing functions specified by the macro list are loaded.
OMIT	All previously loaded capsules are unloaded.	Capsules performing functions specified by the macro list are unloaded.

```
    .
    .
    .
FILE(ISFILE,FO=IS,RT=Z,USE=OPENM/PUT/CLOSEM)
LDSET(STAT=ISFILE)

Load set to write the file.

FILE(ISFILE,OMIT=PUT,USE=GET)
LDSET(STAT=ISFILE)

Load set to read the file.
    .
    .
    .
```

Figure E-1. Static Loading Example

The LDSET control statements necessary for read-only processing are discussed for each applicable file organization in section 4, File Processing. As noted in section 4, static loading requires an additional LDSET control statement. An example of read-only processing using static loading is shown in figure E-2.

STLD.RM MACRO FORMAT

Another method of specifying static loading is through the STLD.RM macro. The format of the STLD.RM macro is shown in figure E-3. This macro must be specified once for each file organization.

```
FILE(ISF,FO=IS,RT=F,USE=OPENM/CLOSEM/GET
    .
    .
    .
LDSET(STAT=ISF)
LDSET(SUBST=$RM.IS$-$RM.ISX$)
LDSET(SUBST=$SAAM.IS$-$IS.ROEN$)
LOAD, ...
LGO.
```

Figure E-2. Read-Only Static Loading Example, Initial Indexed Sequential File

```
(fo)  STLD.RM     USERT=(rtlist),
                  USE=(fcnlist),
                  OMIT=(cmm-fdl)
                  ORG=(new-old)

rtlist      Record type list; record types are separated by
            commas.

fcnlist     AAM functions (macro names); functions are
            separated by commas.

cmm-fdl     CMM or FDL; CMM omits CMM and FDL, FDL
            omits FDL only.

new-old     New or old AAM; OLD (default) is initial
            indexed sequential, direct access, or actual key
            file; NEW is extended indexed sequential file.
```

Figure E-3. STLD.RM Macro Format

USE OF LIST-OF-FILES F

The NOS and NOS/BE operating systems maintain a pointer to the list-of-files, which is a table of the name and FET or FIT address of all active files for each control point. This pointer is set and accessed by the SETLOF and GETLOF macros. A complete description of this feature can be found in the operating system reference manual.

AAM maintains and uses this list-of-files. To alter this list, a user must follow a procedure that is compatible with AAM.

AAM maintains an entry point in its relocatable loaded routines called LOF$RM. The content of this entry point is the address of the current list-of-files. The purpose of this pointer is to minimize the number of GETLOF monitor calls required. The user is encouraged to use this pointer instead of calling the GETLOF macro.

If a user program that coexists with AAM moves the list-of-files, it must update the LOF$RM pointer in addition to calling the SETLOF macro. Also, if a user program adds a new entry to the end of the list-of-files, it must ensure that the next word is zero because AAM does not initialize the list-of-files block to zero.

Allocation of buffer space for extended indexed sequential files can be divided into two classes: user buffer space and pooled buffer space. User buffer space is assigned by the user for a particular file during open processing. Pooled buffer space is allocated by AAM using the Common Memory Manager (CMM) when buffer space is needed and can be used by AAM for any file. AAM does not allow pooled buffer space to exceed a value called TARGET. TARGET is set by the open and close procedures to reflect the CMM requirements expected by AAM; a value of zero indicates that CMM is not to be used for buffer allocation.

USER BUFFER SPACE

When a file is opened, buffer allocation is controlled by the user through the first word address of the buffer (FWB) field and the buffer size (BFS) field in the FIT. If both fields are set to a value other than zero, the fields define the user buffer space for the file. AAM partitions the specified buffer space as follows:

Space for the file statistics table (FSTT) (130 words)

If required, space for the alternate key index file FSTT and the file environment table (139 words)

Space for the FIT extension (up to 168 words)

Space for the data file and index file blocks, beginning with the data file

Once allocated to a file, these blocks cannot be used by any other file. If CMM is not allowed, the minimum space that must be allocated is two blocks (three blocks if the file is compressed) for each file. If the minimum space is not allocated, an error message is issued and the file is not opened. If CMM is allowed, TARGET is increased by the amount that the user buffer space needs to meet the minimum requirement.

POOLED BUFFER SPACE

When a file is opened and the FWB and BFS fields are set to zero, AAM increases TARGET by the amount needed for the particular file organization:

Three blocks for a one index level indexed sequential file

Six blocks for a multiple index level indexed sequential file

Two blocks for an actual key or direct access file

If an alternate key index file is also needed, TARGET is increased by the amount of space required for seven index file blocks. For a compressed data file, space for one additional block is added to TARGET. Space for the FSTTs and FIT extension is allocated from CMM, but it is not added to TARGET because the FSTT and FIT extension for a file are not part of pooled buffer space.

The user controls TARGET directly when the FWB field is set to zero and the BFS field is set to a value greater than zero. TARGET is increased by the value of the BFS field. TARGET is then checked to ensure that enough space exists to satisfy the minimum requirements for the file (two blocks for the data file, two blocks for the index file, and one additional data block if the file is compressed).

If the BFS field is zero and the FWB field is set to a value greater than zero, an error condition exists. The file is not opened.

When AAM is being dynamically loaded, TARGET is increased by the size of the largest seldom-used capsule needed to process an opened AAM file. This is done because pooled buffer space is used for seldom-used capsules as well as for blocks.

BUFFER USE

For both user buffer space and pooled buffer space, all blocks in use are bidirectionally chained in two chains. One chain is file oriented and is used to locate all blocks in memory for a given file. The other chain runs through all blocks (and seldom-used capsules); this chain is referred to as the kickout chain. As blocks are used, they are put at the head of this chain. Blocks gradually migrate to the tail of the chain due to lack of use. A few exceptions to this rather simple algorithm exist. One exception is that the primary index block of a file is always moved to the head of the chain when the file is being accessed randomly. Another exception is that when a new data block is read into the buffer, the previous data block is moved to the tail of the chain.

When space is needed for a new block, the kickout chain is scanned from the tail of the chain forward. For user buffer space, the first block encountered that is the same size as the new block and that belongs to the same file is released; the space is allocated for the new block. If CMM is not present, one of the current blocks is found and used. For pooled buffer space, blocks smaller or larger than the requested block are released until the total space released is enough to allow a new block to be allocated without exceeding TARGET. If a block of the same size is encountered during the scanning, that block is released and the space is allocated for the new block.

DATA COMPRESSION AND DATA ENCRYPTION H

Data compression and data encryption are provided for use with extended indexed sequential files. The system-supplied routine or a user-supplied routine can be used for data compression. Data encryption, which requires a user-supplied routine, is handled through the compression routine owncode exits.

Data compression is performed on a record-by-record basis. It is used to compress strings of zeros or blanks in order to shorten the record length. If the compression routine cannot realize a reduction in record length, the record is flagged and stored in its uncompressed state. When a compressed record is read, it must be restored to its original state by a decompression routine. If a record is flagged to indicate it is not compressed, the decompression routine is not called when the record is read.

Data encryption is used to expand and reformat a record. The encrypted record must be no longer than the number of characters specified for the maximum record length (MRL) field in the FIT. When an encrypted record is read, it must be restored to its original size and format by a decryption routine. Because AAM considers compression/decompression and encryption/decryption to be the same thing, the following discussion is related to compression; encryption is mentioned only where differences exist.

Two fields in the FIT are used to designate data compression and decompression. The compression routine address (CPA) field and the decompression routine address (DCA) field specify the number of the system-supplied routine or the address of a user-supplied routine. For data encryption/decryption, user-supplied routine addresses must be specified in the CPA and DCA fields.

The compression and decompression routines are called by AAM with register A1 pointing to the vector shown in figure H-1. If the primary key is not embedded, the key length parameter is zero. Embedded keys are restricted to keys beginning in the first character position in the first word of the record (word 0, character position 0).

The product of a compression or decompression routine is the record produced by the routine. The destination area is a special area set up by AAM to receive the product of the routine.

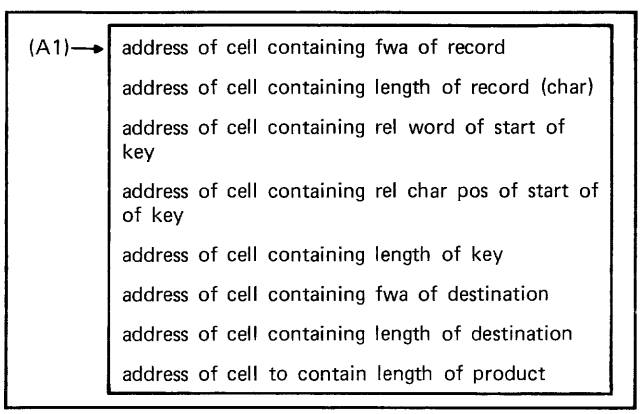


Figure H-1. Vector Used by Compression/Decompression Routines

The compression and decompression routines return either the length of the product (in characters) or a negative number of characters to indicate that the product is too large for the destination area. A decompression routine that produces a record too long for the destination area is the same as a GET macro that reads a record larger than expected by the caller; an error is generated and the record is not transferred.

In contrast to a compression routine, a user-supplied encryption routine could produce a longer record. The record can be expanded up to the number of characters specified for the maximum record length (MRL) field in the FIT. The length of the destination area is always ten characters greater than the value of the MRL field.

The compression routine must store an identifier in the first word of the destination area. This identifier is verified against the identifier stored in the FSTT for the file when compression was first specified. The length of the product of a compression routine includes the identifier length.

The format of the destination area is shown in figure H-2. Words 2 through n are passed to the decompression routine.

When a file is opened on a creation run, the compression address (CPA) field in the FIT is checked. If the value of the field is zero, no data compression is performed during the creation run; an error occurs if the decompression routine address (DCA) field is set. A value greater than zero in the CPA field sets the method of compression for the life of the file.

If the CPA field contains the address of a user-supplied compression routine, the address is the entry point of the routine. The compression routine is called with a dummy record in order to determine the identifier, which is stored permanently in the COMPACT field in the FSTT. The identifier can be anything other than a zero word.

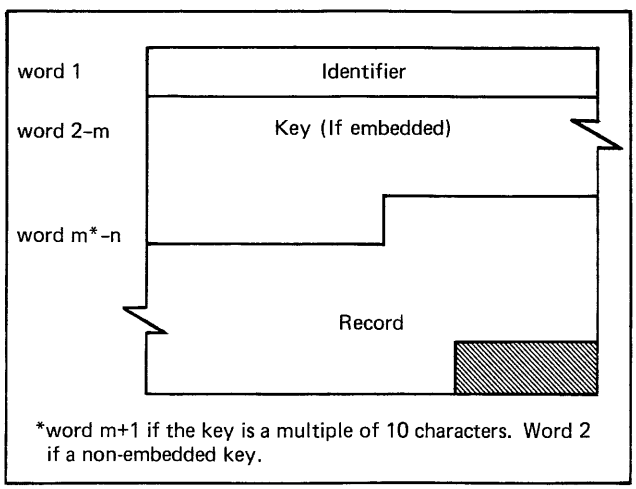


Figure H-2. Destination Area Format

If the CPA field contains the 6-bit integer identifying a system-supplied compression routine, the necessary routines are loaded by the Fast Dynamic Loader (FDL), the addresses are stored in the CPA and DCA fields in the FIT, and the identifier is stored in the COMPACT field in the FSTT. The integer identifying the compression routine is stored in the SYSCOMP field in the FSTT.

When a file is opened after the creation run, the value of the COMPACT field in the FSTT is checked to determine whether or not compression and decompression are required. If the field contains zero, no compression takes place. If the COMPACT field contains a value greater than zero, the SYSCOMP field is checked to determine the method of compression.

If the SYSCOMP field contains a zero value, the addresses in the CPA and DCA fields in the FIT are used when compression or decompression is required.

If the SYSCOMP field contains a nonzero value, the proper system routines are loaded.

The user is responsible for maintaining the correct entry point addresses of user-supplied compression and decompression routines for the CPA and DCA fields in the FIT. If the CPA field is zero, compression is not performed on the record.

When a system-supplied compression routine is specified, the user is responsible for maintaining unchanged the entry point addresses stored in the CPA and DCA fields by the system. Compression can be turned off by the user by storing zeros in the CPA field.

System-supplied compression routine number 1 compresses strings of display coded blanks (55_8), zeros (33_8), and colons (00_8). An escape character, 72_8 (<), signals the beginning of a compressed string. The character following the escape character is divided into two parts: a two-bit character code and a four-bit repeat count. The two-bit character code is as follows:

00	72_8	Escape character (<)
01	33_8	Zero
10	00_8	Colon
11	55_8	Blank

The four-bit repeat count indicates the number of occurrences of the compressed character. The value of the repeat count is three less than the actual number of occurrences of the blank, zero, or colon; however, the repeat count for the escape character (<) is one less than the actual number of occurrences. Up to 18 occurrences of the compressed character are compressed into two characters. For example, 36 consecutive blanks are compressed into four characters. Single and double occurrences of the zero, colon, or blank are not compressed. A single occurrence of the escape character causes an expansion: one character for the < character to signal the beginning of a compressed string and one character for the character code and the repeat count.

INDEX

COMMENT SHEET

CONTROL DATA CORPORATION

TITLE: CYBER Record Manager Advanced Access Methods Version 2 Reference Manual

PUBLICATION NO. 60499300 REVISION A

This form is not intended to be used as an order blank. Control Data Corporation solicits your comments about this manual with a view to improving its usefulness in later editions.

Applications for which you use this manual.

Do you find it adequate for your purpose?

What improvements to this manual do you recommend to better serve your purpose?

Note specific errors discovered (please include page number reference).

CUT ON THIS LINE

General comments:

FROM NAME: _____ POSITION: _____

COMPANY NAME: _____

ADDRESS: _____

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

CUT ON THIS LINE

FOLD

FOLD

STAPLE

STAPLE

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

TITLE: CYBER Record Manager Basic Access Methods
Version 1.5 Reference Manual

PUBLICATION NO. 60495700
REVISION D

DATE: March 31, 1978

REASON FOR CHANGE:

This revision reflects feature CP 091, CYBER Record Manager Basic Access Methods Version 1.5.

INSTRUCTIONS:

Discard the previous edition of this manual and replace with the attached.

CYBER Record Manager Basic Access

Manual Title Methods Version 1.5 Reference Manual Pub. No. 60495700 Rev. D

As part of Control Data's continuing quality improvement program, we invite you to complete this questionnaire so that you may have a more direct influence on the manuals you use.

Please rate this manual for each general and individual category on a scale of 1 through 5 as follows:

1 - Excellent 2 - Good 3 - Fair 4 - Poor 5 - Unacceptable

I. Writing Quality _____

- A. Technical accuracy _____
- B. Completeness _____
- C. Audience defined properly _____
- D. Readability _____
- E. Understandability _____
- F. Organization _____

II. Examples _____

- A. Quantity _____
- B. Placement _____
- C. Applicability _____
- D. Quality _____
- E. Instructiveness _____

III. Format _____

- A. Type size _____
- B. Page density _____
- C. Art work _____
- D. Legibility _____
- E. Printing/Reproduction _____

IV. Miscellaneous _____

- A. Index _____
- B. Glossary _____

V. Please provide a yes or no answer regarding manuals in general:

- A. I prefer that a manual on a software product be as comprehensive as possible; physical size is of little importance. _____
- B. I prefer that information on a software product be covered in several small manuals, each covering a certain aspect of the product. Smaller manuals with limited subject matter are easier to work with. _____
- C. I am interested primarily in reference manuals designed for ease of locating specific information. _____
- D. I am interested primarily in user guides designed to teach the user about a product or certain capabilities of a product. _____

VI. We recognize that we have a wide variety of users. Please identify your primary area of interest or activity:

- A. Student _____
- B. Applications programmer _____
- C. Systems programmer _____
- D. How many years programming experience do you have? _____
- E. What languages
  1. Algol _____
  2. Basic _____
  3. Cobol _____
  4. Compass _____
  5. Fortran _____
  6. PL/I _____
  7. Other _____
- F. Have you ever worked on non-CDC equipment? _____
  1. If yes, approximately what percent of your experience is on non-CDC equipment? _____
  2. How do you rate CDC manuals against other similar manuals using the 1-5 ratings. (Example: XYZ Corp. 2 means XYZ manuals are good as compared to CDC manuals.)
     - Burroughs _____
     - DEC _____
     - Hewlett-Packard _____
     - Honeywell _____
     - IBM _____
     - NCR _____
     - Univac _____
     - Other _____ _____

General Comments _____

_____

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

CUT ON THIS LINE

FOLD

FOLD

STAPLE

STAPLE

60495700

CONTROL DATA CORPORATION

CYBER RECORD MANAGER BASIC ACCESS METHODS VERSION 1.5 REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 1
NOS/BE 1

REVISION RECORD	
REVISION	**DESCRIPTION**
A	Original release.
(11-1-75)	
B	This revision reflects 7000 Record Manager as released under SCOPE 2.1.4: new features include F053
(3/5/76)	connected file flag. The revision also reflects CYBER Record Manager Version 1.4: new features
	include DM 119 FILE control statement cancel; and DM 135 internal changes which do not affect this
	manual. See the list of effective pages.
C	This revision reflects CYBER Record Manager 1.4 at PSR level 452. All references to 7000 Record
(7-1-77)	Manager have been eliminated.
D	This revision reflects feature CP 091, CYBER Record Manager Basic Access Methods Version 1.5.
(3-31-78)	
Publication No. 60495700	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

or use Comment Sheet in the back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Cover	–
Title Page	–
ii thru viii	D
1-1 thru 1-3	D
2-1 thru 2-10	D
3-1 thru 3-9	D
4-1 thru 4-6	D
5-1 thru 5-7	D
6-1 thru 6-8	D
A-1 thru A-4	D
B-1 thru B-10	D
C-1, C-2	D
D-1 thru D-7	D
E-1, E-2	D
F-1	D
G-1	D
Index-1 thru -5	D
Comment Sheet	D
Return Env.	–
Cover	–

Page	Revision

Page	Revision

PREFACE

CYBER Record Manager Basic Access Methods (BAM) Version 1.5 operates under control of the following operating systems:

NOS 1 for the CONTROL DATA® CYBER 170 Models 171, 172, 173, 174, 175; CYBER 70 Models 71, 72, 73, 74; and 6000 Series Computer Systems.

NOS/BE 1 for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, 74; and 6000 Series Computer Systems.

BAM input and output facilities are available to users of COMPASS assembly language through macro calls; user programs, COBOL, FORTRAN Extended, and Sort/Merge use BAM for input/output operations. The user programs communicate with BAM either through the compiler, using the calls supplied within the languages, or with BAM macros.

Intended as a primary document for COMPASS programmers, this manual presents background information and operational specifications for BAM. COBOL, FORTRAN Extended, and Sort/Merge programmers can use this manual as a source for BAM terminology and concepts; specific language interfaces are detailed in the appropriate reference manuals. The user is assumed to be familiar with the operating system at the installation, and with file organization and manipulation.

Information necessary for a complete understanding of BAM use is contained in the following publications:

Publication	Publication Number
NOS/BE 1 Reference Manual	60493800
NOS 1 Reference Manual, Volume 1	60435400
NOS 1 Reference Manual, Volume 2	60445300
CYBER Record Manager Advanced Access Methods Version 2 Reference Manual	60499300
CYBER Record Manager Version 1 Guide for Users of COBOL Version 4	60496000
CYBER Record Manager Version 1 Guide for Users of FORTRAN Extended Version 4	60495900
CYBER Record Manager Version 1 User's Guide	60495800
Common Memory Manager Version 1 Reference Manual	60499200
COMPASS Version 3 Reference Manual	60492600
CYBER Loader Reference Manual	60429800

CDC manuals can be ordered from Control Data Literature and Distribution Services: 8001 East Bloomington Freeway, Minneapolis, MN 55420

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

FIGURES

TABLES

BAM FEATURES 1

BAM provides an interface between user programs and system input/output routines. BAM subsystems exist in NOS/BE and NOS operating systems.

BAM also provides:

Consistent error processing

Accommodation for various labeling conventions

Maintenance of different file organizations

BAM routines are used by some compilers and are available for user programs. Use of BAM by compilers and user programs extends input/output compatibility to both the system and application program levels.

The primary task of BAM is record and block input/output for files on supported devices. Consequently, the various types of records, blocks, and file organizations must be identified for BAM. These and other file characteristics must be set by the user in a file information table (FIT). The FIT is divided into fields that describe certain aspects of the file. Refer to appendix D for the exact structure of the FIT.

REFERENCES

The following terms are relevant to BAM and related systems:

AAM (Advanced Access Methods)

A file manager that processes indexed sequential, direct access, and actual key file organizations and supports the Multiple-Index Processor.

BAM (Basic Access Methods)

A file manager that processes sequential and word addressable file organizations.

CRM (CYBER Record Manager)

Refers to CYBER Record Manager, a generic term relating to both BAM and AAM as they run under NOS/BE and NOS operating systems.

MIP (Multiple-Index Processor)

A processor that allows AAM files to be accessed by alternate keys.

FILE ORGANIZATIONS

Two file organizations are supported by BAM:

Sequential (SQ)

Records are stored in the order in which they were written.

Word addressable (WA)

A group of contiguous words comprise a file. Records are accessed by a word number within the file.

MACROS

A FIT is established for each file by a FILE macro encountered at assembly time. This macro can contain the file name only, or it can have user-specified parameters describing a particular file. The FILE macro establishes the FIT in the using program's field length at the point at which it is called. FIT fields are assumed through default values when they are not specified as parameters in macros. The macros and functions are listed in table 1-1 according to their associated purposes:

File creation and maintenance

File initialization and termination

Data transfer

File updating

File positioning

Boundary conditions

User label processing

The applicability of some macros depends on the file organization established by the user. Table 1-2 presents macros as applicable to sequential and word addressable file organizations, the two supported by BAM.

This manual discusses macro properties and generalizes processing whenever possible. However, explanations are provided in section 4 for each macro according to file organization. Consequently, material is presented redundantly for the benefit of a programmer who uses this manual to reference particular features only.

Macro statements are coded in COMPASS format. Each statement can contain a location field, a macro name in the operation field, a variable field, and a comment field. Any field is terminated by one or more blanks. A macro statement begins at character position 1 of an 80-column card image and continues through column 72. Columns 73 through 80 are used for sequencing. Suggested coding conventions are shown in figure 1-1.

The allocation of the columns in COMPASS format is as follows:

1	Comma (continuation), asterisk (comments line), or other (beginning of new statements)
2 thru 9	Location field entry, left-justified
10	Blank
11 thru 16	Operation field entry, left-justified
17	Blank
18 thru 29	Variable field entry, left-justified
30	Beginning of comments

TABLE 1-1. CYBER RECORD MANAGER MACROS

Function	Macro	Action Taken
File creation and maintenance	FILE	Creates a file information table (FIT). In addition to this macro, a FILE control statement is available to supply FIT information.
	FETCH	Retrieves the value of specified fields in the FIT.
	STORE	Sets values in fields of the FIT.
	SETFIT	Sets values in fields of the FIT with values supplied through the FILE control statement.
File initialization and termination	OPENM	Prepares a file for processing; initiates label processing.
	CLOSEM	Terminates file or volume processing; initiates label processing.
Data transfer	GET	Transfers data from a file to the working storage area.
	PUT	Transfers data from the working storage area to a file.
	CHECK	Determines completion status of input/output operations.
File updating	REPLACE	Replaces a record in a file.
File positioning	SKIP	Repositions a file backward or forward.
	REWINDM	Rewinds the current volume to beginning-of-information (BOI).
Boundary conditions	ENDFILE	Records a partition terminator.
	WEOR	Records a section terminator.
	WTMK	Records a tapemark on a tape file.
User label processing	GETL	Retrieves the next label of a label string and delivers it to the label area.
	PUTL	Writes or checks a label in the label area.
	CLOSEL	Terminates label processing.

TABLE 1-2. MACROS AND RELATED FILE ORGANIZATIONS

Macro	File Organization	
	SQ	WA
CHECK	X	X
CHECKR	X	X
CLOSEL	X	
CLOSEM	X	X
ENDFILE	X	
FETCH	X	X
FILE	X	X
GET	X	X
GETL	X	
GETP	X	
GETWR	X	
OPENM	X	X
PUT	X	X
PUTL	X	
PUTP	X	
PUTWR	X	
REPLACE	X	
REWINDM	X	X
SETFIT	X	X
SKIP	X	
STORE	X	X
WEOR	X	
WTMK	X	

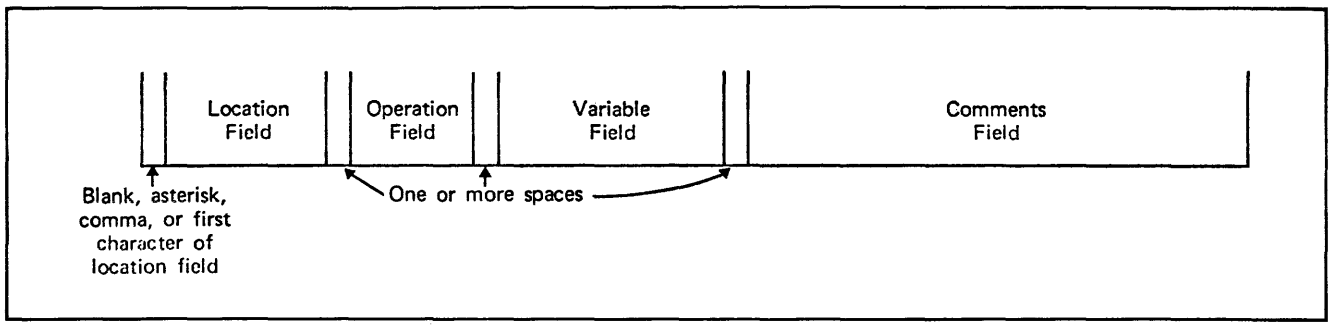


Figure 1-1. COMPASS Format

FILE STRUCTURES 2

There is a hierarchical data structure in a progression from the character level to the largest grouping of data, the file, which can be contained on one or more volumes. The BAM user can describe file structure by file organization (FO), block type (BT), and record type (RT). This section presents these structures. Additionally, many of the file information table fields that must be set by the user are identified. They are explained in detail in section 3.

LOGICAL STRUCTURE

The logical structure of a file is user-controlled. The following definitions describe terms used throughout this manual that are applicable to the logical structure of a file:

Record

A record is a group of related characters. A character is represented in six bits as internal display code. A record or portion thereof is the smallest collection of information passed between BAM and the user. The user defines the structure and characteristics of records within a file by declaring a record format. The beginning and ending points of a record are implicit within each format. Records are grouped into files.

Section

A section consists of one or more records. Generally, a section is less than a partition and greater than a record, but it can be identical to either or both. A section begins with the first record after the end of the preceding section; a section ends when a special record or condition occurs. Only sequential files are grouped into sections.

Partition

A partition consists of one or more sections. Generally, it is less than a file and greater than a section, but it can be identical to either or both. A partition begins with the first record after the end of the preceding partition; a partition ends when a special record or condition occurs. Only sequential files are grouped into partitions.

Block

A block can contain partial records or one or more records. Block structure is interwoven with the physical recording format; unlike other logical file structure declarations, the block structure is transparent in use. Blocks are constructed from the records supplied by the user and the user is supplied with records as required. The user is unaware of block boundaries. Only sequential files are grouped into blocks.

File

A file is a logically connected set of information; it is the largest collection of information that can be addressed by that file name. All data in a file is stored between the beginning-of-information (BOI) and the end-of-information (EOI). Label groups are not considered to be part of file data in the general case.

PHYSICAL STRUCTURE

The following definitions pertain to the physical means used to record files:

Input/output device

Any storage medium supported by the operating system.

Rotating mass storage (RMS)

Disk or disk pack.

Mass storage device

Disk, disk pack, or extended core storage (ECS).

Volume

A volume is a reel of magnetic tape with sequential files. A file can be contained on more than one volume and a volume can contain more than one file.

Level number

A level number can range from 00 to 17_8 and is physically recorded on a physical record unit (PRU) device in an eight-character appendage to a short PRU. A short PRU consisting only of the eight-character level number appendage is called a zero-length PRU. The appendage is neither created by nor returned to the user. The level number value is available in the FIT on some input operations and can be specified by the user on some output operations.

Physical record

A physical record is defined only on magnetic tape; it consists of the data between interrecord gaps. A physical record need not contain a fixed amount of data.

S/L tape

S/L tape must be declared by the user. The physical structure of a file on an S/L tape depends entirely on the logical structures selected by the user; no operating system structure is superimposed. Physical record size is limited only by the buffer size on an L tape; physical record size on an S tape cannot be greater than 5120 characters. On S/L tapes, a block and a physical record are the same.

PRU device

All mass storage devices and non-S/L tapes are PRU devices; a physical structure is superimposed over the user-declared file structure by the operating system on all files that reside on PRU devices.

Physical record unit (PRU)

The smallest unit of information that can be transferred between a peripheral storage device and central memory. The PRU size is permanently

fixed for PRU devices; the PRU concept does not apply to S/L tapes. PRU device sizes are:

Mass storage devices – 640 characters

Binary SI tapes – 5120 characters

Coded SI tapes – 1280 characters (supported under NOS/BE only)

I tapes – 5120 characters (supported under NOS only)

Short PRU

A short PRU contains less than the number of characters defined for a PRU on a PRU device. An eight character level number appendage is always part of a short PRU.

System-logical-record

A system-logical-record is defined only on PRU devices. It consists of a group of PRUs terminated by a short or zero-length PRU. A system-logical-record can be simulated on an S/L tape by writing a series of physical records of the same length as a PRU, followed by a physical record of a length less than a PRU and with a level number appendage. However, because of the installation parameter that defines noise (IP.NOISE=), no PRU smaller than the installation definition or operating system default can be written on an S/L tape. (The default on NOS/BE is 8 characters; the default on NOS is 14 characters.)

BAM controls the physical file position while the user controls only the logical file position. Physical and logical positions are not guaranteed to agree after a given operation unless S type records are being used.

FILE ORGANIZATIONS

BAM supports two file organizations: sequential and word addressable. Once the file organization is set for a BAM file, it must not be changed to an AAM file organization in the same job step. It is possible that the AAM interface routines are not loaded and that internal FIT fields have been initialized based on the BAM file organization. The following is a description of the structure of each organization and its applicable record and block types.

SEQUENTIAL FILES

Sequential files are tape-like in structure. Records are placed in the order of presentation; physically, a record follows the previous record. Given the location of one record, the location of the next record is determined in relation to the given record only. A sequential file can extend across any number of volumes and can be accessed sequentially only.

A sequential file can reside either on a magnetic tape or on mass storage. Tape files, punch card or printer files, and some mass storage files are classified as sequential. A mass storage sequential file is not necessarily maintained internally in sequential order by CIO; however, records are presented to the user in sequential order. All sequential files are blocked through the block type parameter specification, regardless of device type, except for S type records.

The logical structure of a sequential file is shown in figure 2-1. The physical structure of a sequential file is shown under the discussion of the various block types.

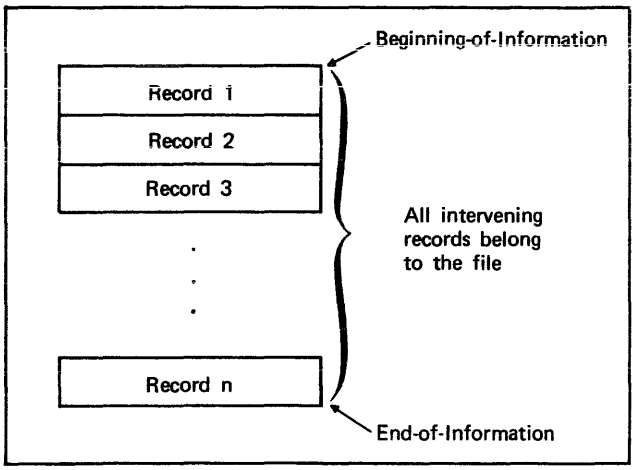


Figure 2-1. Logical Structure of a Sequential File

Block Types for Sequential Files

Sequential file blocking is, essentially, the concept of compressing actual records into contiguous record groups, thereby saving storage that would otherwise be wasted for interrecord gaps. Blocks can be various types, as explained in the following discussion. BAM supports four block types identified as I, C, K, or E. These block types are applicable to sequential files. A summary of block types and physical recording formats is represented in table 2-1.

Internal Blocking Type I

I type blocks begin with a block control word, which contains block and record identification. Contents of the block control word include a pointer to the first record beginning in the block. I type blocks can contain only W type records. Except for the last block of the section, partition, or file which can be shorter; I type blocks are always 5120 characters.

TABLE 2-1. BLOCK TYPE USAGE

Block Type	Physical Recording Format	
	PRU Device	S/L Tape
I	I block size is 5120 characters; section or partition is a single system-logical-record.	I block size is 5120 characters; last block in section, partition, or file can be shorter.
C	C block size is equal to 0 (unblocked) or a multiple of PRU size; section or partition is a single system-logical-record.	C block size equals 5120 characters for S tapes and a maximum of the value of the BFS field minus two for L tapes; last block of section, partition, or file can be shorter.
K	K blocking on PRU devices is prohibited.	Each K block is written as a physical record.
E	E blocking on PRU devices is prohibited.	Each E block is written as a physical record.

A file with I type blocks can be recorded on either a PRU device or an S/L tape. On a PRU device, a short I block is recorded as a short PRU, which is the end of a system-logical-record. On an S/L tape, I type blocks are not an allowable ANSI (American National Standards Institute) interchange format because ANSI does not define W type records.

The block control word format is shown in figure 2-2. Blocks and records are numbered consecutively from 1. The record number includes all records that are physically present whether they are logically present or not. If no record begins in the block, word offset and record number equal zero. The block control word is word zero of the block.

Character Count Block Type C

Each C type block contains the number of characters specified by the value of the maximum block length (MBL) field of the FIT; however, the last block of the section, partition, or file can be shorter. Except for S type records, records can span block boundaries as shown in figure 2-3. C type blocks can contain any record type.

If the MBL field is not specified, the default values are set as follows:

S tapes	MBL=5120 characters
L tapes	MBL=value of the buffer size (BFS) field minus two
PRU devices	MBL=0 (unblocked)

If a value is specified for the MBL field, it can be a maximum of 5120 characters for S tapes and a maximum of the value of the BFS field minus two for L tapes. The most efficient value of the MBL field for PRU devices is 0; however, it can be set to a multiple of PRU size. If the value specified is not 0 or a multiple of PRU size, the value is rounded down to a multiple. The MBL field set to PRU size facilitates parity error recovery for W type records because a boundary condition would exist.

When record type is S and block type is C, any user value for MBL is not changed for files on any device. S type records cannot be blocked. On an S/L tape, one S type record is one tape block. The C type block on an S/L tape is not an allowable ANSI interchange format because BAM does not support the ANSI spanned record type.

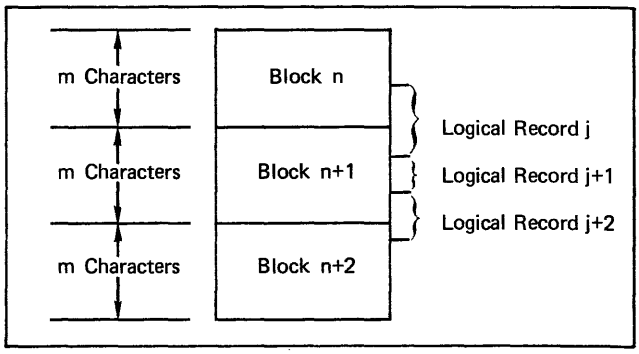


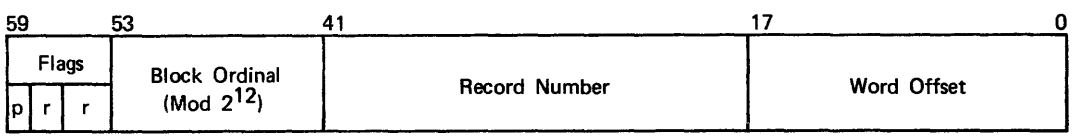
Figure 2-3. C Type Block Structure

Record Count Block Type K

For K type blocks, each variable-length block contains the same number of records. Records cannot span blocks. The last block of a partition or file can contain fewer than the value specified in the number of records per block (RB) field of the FIT. K type blocks are prohibited on PRU devices; they are valid for S/L tapes only. K type blocks can contain any type record except S or W type records.

Padding can be inserted when a K type block is written. The three FIT fields concerned with padding are the padding character (PC), the multiple of characters per block (MUL), and the minimum block length (MNB). The value of the MNB field takes precedence over the value of the MUL field. Padding is inserted so that each block, except possibly the last one on a file or volume, is a multiple of MUL characters and is at least the number of characters specified by the value of the MNB field in length. The last block of a partition can contain fewer than the number of characters specified by the value of the MNB field; padding is not added to the last block because the GET macro cannot distinguish padding from a valid record.

When writing K type blocks, the value of the RB field of the FIT is used to construct blocks of exactly that number of records. When reading K type blocks, each block need not be exactly the number of records specified by the value of the RB field, because blocks are physically delimited and boundaries are readily detected. However, if the RB field of the FIT is set to a value less than the number of records



59	Parity bit, used to maintain odd parity within the control word.
58, 57	Reserved for CDC.
56 thru 54	Reserved for users.
53 thru 42	Ordinal of the current block (modulus 4096).
41 thru 18	Ordinal of the first record beginning in this block (modulus 2^{24}, if necessary).
17 thru 0	Word number of the control word of the first W type record in the block.

Figure 2-2. Block Control Word Format for I Type Blocks

physically present, only the number of records specified by the value of the RB field are returned to the working storage area; other records physically present are assumed to be padding and are not returned to the working storage area.

K type blocks are recorded as tape physical records. To ensure that the last block in a file is interpreted correctly, minimum record size should be greater than noise record size, because it is possible for the last block to contain only a single record.

The K type block is an allowable format for ANSI standard tape interchange. The structure of a K type block is shown in figure 2-4.

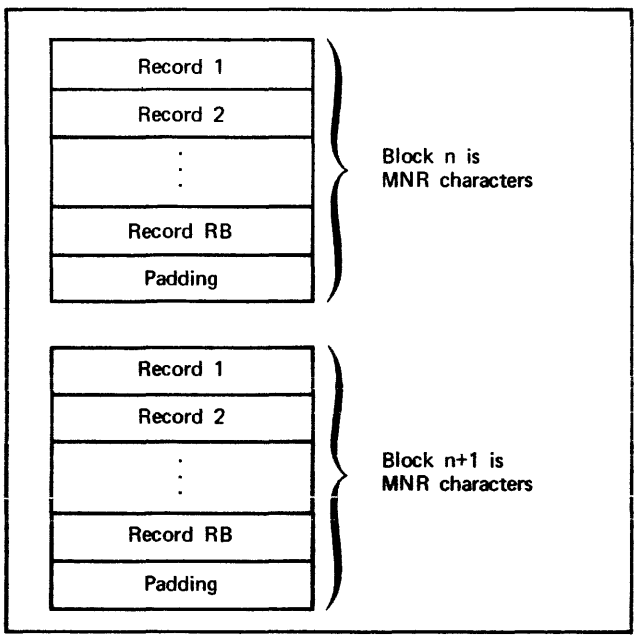


Figure 2-4. K Type Block Structure

Exact Records Block Type E

Each E type block contains an integral number of records, as many whole records as can be contained in the block size, which is the number of characters specified by the value of the maximum block length (MBL) field of the FIT. E type blocks are prohibited on PRU devices; they are valid only for S/L tapes. Any type record, except S or W type records, can be contained in E type blocks.

The value of the minimum block length (MNB) field of the FIT must not be greater than the value of the MBL field. The value of the MBL field must be greater than the value specified in the maximum record length (MRL) field of the FIT.

Padding can be inserted when an E type block is written. The three FIT fields concerned with padding are the padding character (PC), the multiple of characters per block (MUL), and the minimum block length (MNB). The value of the MNB field takes precedence over the value of the MUL field. Padding is inserted so that each block, except possibly the last one on a file or volume, is a multiple of MUL characters and is at least the number of characters specified by the value of the MNB field in length. The last block of a partition can contain fewer than the number of characters specified by the value of the MNB field. To ensure that the last block in a file is interpreted correctly, minimum record length should be greater than noise record size, because it is possible for the last block to contain only a single record.

When specifying E type blocks with padding, the following restriction must be observed or E type blocks can be constructed in which padding cannot be distinguished from data. The value of the MBL field minus the value of the MNB field must be greater than the value of the MRL field minus the value of the minimum record length (MNR) field; and the value of the MUL field must be less than the value of the MNR field.

The E type block is an allowable format for ANSI standard tape interchange. E type block structure is shown in figure 2-5.

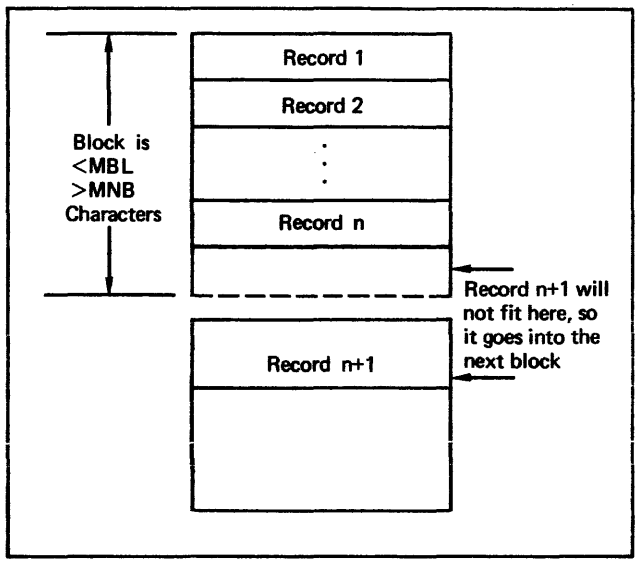


Figure 2-5. E Type Block Structure

File Boundaries

The beginning-of-information is that point in a file before which no data exists. The end-of-information is that point in a file after which no data exists. Table 2-2 shows the various file boundary conditions for sequential files.

Partition Boundaries

A partition begins at beginning-of-information or after a preceding end-of-partition (EOP). A partition ends at end-of-information (EOI) or on the occurrence of an end-of-partition boundary. End-of-partition boundaries vary depending on device, block type, and record type, as shown in table 2-3.

Section Boundaries

A section begins at beginning-of-information, or after a preceding end-of-partition, or after a preceding end-of-section (EOS). A section ends at end-of-information, or end-of-partition, or at the occurrence of an end-of-section boundary. End-of-section boundaries vary depending on device, block type, and record type, as shown in table 2-4.

S type records are a special case for section identification. Although an S type record is defined to be a record terminated by a short PRU of level less than 17, an S type record is never considered to be a section. When S type records are read, the file position (FP) field of the FIT is set to end-of-record, never to end-of-section.

TABLE 2-2. SEQUENTIAL FILE BOUNDARY CONDITIONS

Device	Boundary	
	Beginning-of-Information	End-of-Information
Mass storage	Before the first record written.	After the last record written.
Labeled tape	Between the file header label group and the first record written.	Between the last record written and the file trailer label group.
Unlabeled S/L tape	Between load point and the first record written.	Undefined.
Unlabeled SI tape	Between load point and the first record written.	Between the last record written and the file trailer label group.
Unlabeled I tape	Between load point and the first record written.	Undefined.

TABLE 2-3. END-OF-PARTITION BOUNDARIES

Device	Block Type	Record Type	End-of-Partition Boundary
PRU device	I	W	One-word deleted record pointing back to the last I block boundary; control word with the EOP flag; terminate the system-logical-record with level 0.
	C	W	Control word with an EOP flag; terminate the system-logical-record with level 0.
	C	All but W	Terminate the system-logical-record with level 0; zero-length PRU with level 17.
S/L tape	I	W	Zero-length deleted records to exceed noise record size; one-word deleted record pointing back to the I block boundary; control word with an EOP flag; terminate the block.
	C	W	Zero-length deleted records to exceed noise record size; control word with an EOP flag; terminate the block.
	C,K,E	All but W	Terminate the block; tapemark.

Volume Boundaries

Volume boundaries are defined only on magnetic tape with a sequential file. The user of such files can elect to ignore volume boundaries or to be notified when volume boundaries occur. A volume boundary has no necessary relationship to any logical boundary and can occur at any point within a file. The beginning-of-volume of the first volume is synonymous with beginning-of-information. Thereafter, beginning-of-volume is located before the first data block on second and subsequent volumes. An end-of-volume condition exists when one of the conditions shown in table 2-5 occurs.

TABLE 2-4. END-OF-SECTION BOUNDARIES

Device	Block Type	Record Type	End-of-Section Boundary
PRU device	I	W	One-word deleted record pointing back to the last I block boundary; control word with EOS flags; terminate the system-logical-record with level 0.
	C	W	Control word with EOS flags; terminate the system-logical-record with level 0.
	C	All but W	Terminate the system-logical-record with level less than 17.
S/L tape	I	W	Zero-length deleted records to exceed noise record size; one-word deleted record pointing back to the I block boundary; control word with EOS flags; terminate the block.
	C	W	Zero-length deleted records to exceed noise record size; control word with EOS flags; terminate the block.
	C,K,E	All but W	Terminate the block (undefined on a read).

TABLE 2-5. END-OF-VOLUME BOUNDARIES

Device	End-of-Volume Boundary
Labeled tape	Between the last record on tape and the volume trailer label group.
Unlabeled tape	Between the last record on tape and the volume trailer label group (PRU device only) or the first tapemark after the reflective spot (S/L tapes).
Nonstandard labeled tape	Between the last record on tape and the nonstandard end-of-volume label which is controlled by the user.

WORD ADDRESSABLE FILES

Word addressable files are mass storage files containing continuous data or space for data. Words within the file are numbered from 1 to n, each word containing 10 characters. Data is read or written within the file starting at a word specified by the word number, called the word address.

Reading beyond the current end-of-information limit is not allowed. For writing, word addressable files are automatically extended if the write results in an address beyond the end-of-information. Word addressable files can be accessed either sequentially or randomly by word address. The user should recognize that a sequential read is valid only if data is contiguous. The supplied word address for random access points to a location in the file that is on a word boundary; therefore, all records begin on a word boundary.

Although word addressable files must reside on mass storage for processing, the COPYBR or COPYBF utility can be used to copy a word addressable file to tape. The COPYBR utility is preferable. Any level 17 information written by the copy is ignored when the file is restored to mass storage and a write is occurring. A read of level 17 written by the copy utility returns an end-of-partition status.

Only W, F, and U type records are possible in word addressable files. The logical structure of a word addressable file is shown in figure 2-6.

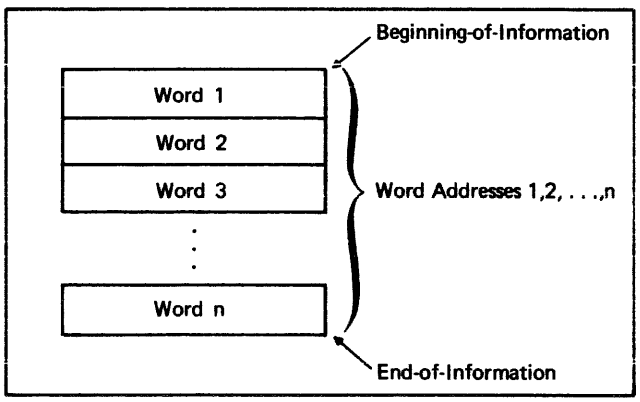


Figure 2-6. Logical Structure of a Word Addressable File

RECORD TYPES

BAM supports eight record types. The eight record types and a corresponding explanation of their lengths are listed in table 2-6.

The numbering conventions for describing a record or the position of a control field in a record are summarized in figure 2-7. All record lengths are specified by character count. Values normally are unsigned positive decimal integers.

The record types allowed for each block type are shown in table 2-7. S type records are not blocked, but block type can be set to C for compatibility with SCOPE 2 type files.

DECIMAL CHARACTER COUNT TYPE D

The record length for D type records is specified in a length field located within the record. The two fields of the FIT which specify the position of the length field are the length field beginning character position (LP) field, numbering from 0, and the length field length (LL) field, which is the number of characters in the length field, one to six characters. The record length specified must be less than or equal to the number of characters specified by the value of the maximum record length (MRL) field of the FIT. Maximum record length that can be specified is $10(2^{17}-1)$ characters. The record length specified in the length field is given as right-justified display code filled with zeros or blanks. The LL field can be COMPUTATIONAL-1 if the C1 field of the FIT is set to YES, or it can be a sign-overpunch field if the sign-overpunch (SB) field of the FIT is set to YES.

TABLE 2-6. RECORD TYPES AND LENGTH DESCRIPTIONS

Record Type	Length Description
Decimal Character Count (D)	Length is given as character count, by the length field contained within the record.
Fixed Length (F)	Fixed length.
Record Mark (R)	Terminated by a record mark character specified by the user.
System Record (S)	Length of the system-logical-record depends on the PRU device; on an S/L tape, one S type record is a physical record.
Trailer Count (T)	Fixed length header followed by a variable number of fixed length trailers; the header contains the trailer count field.
Undefined (U)	Length is defined by the user.
Control Word (W)	Length is contained in a control word prefixed to a record by BAM.
Zero Byte (Z)	Terminated by a 12-bit zero byte in the low-order byte position of a 60-bit word.

TABLE 2-7. RECORD TYPE AND BLOCK TYPE ASSOCIATIONS

Block Type	Record Type							
	F	D	R	T	U	W	Z	S
I						X		
C	X	X	X	X	X	X	X	X
K	X	X	X	X	X		X	
E	X	X	X	X	X		X	

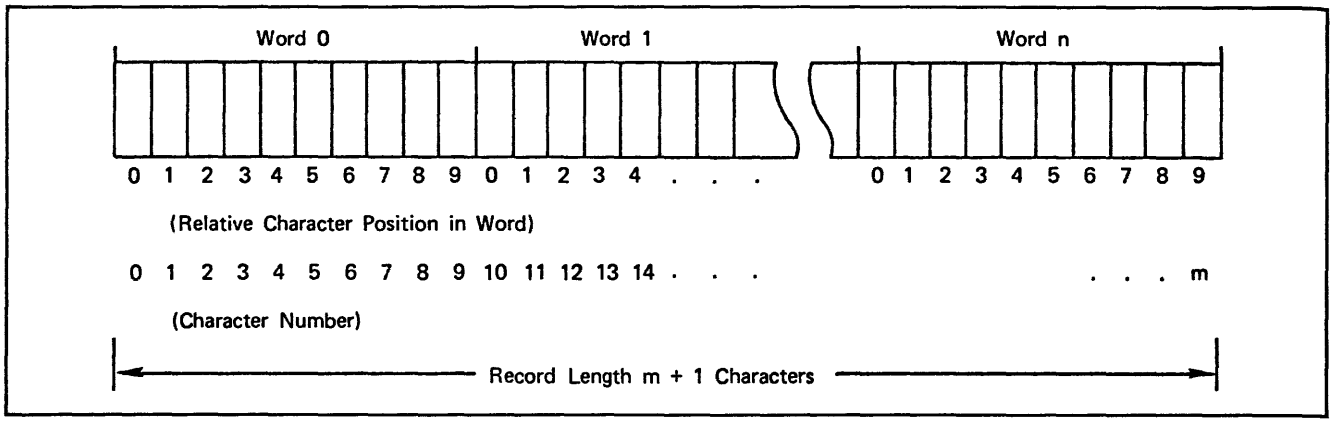


Figure 2-7. Numbering Conventions

For the first GETP or PUTP macro issued for a given record, the minimum number of characters that can be transferred is the value specified in the minimum record length (MNR) field of the FIT. If the user does not supply a value for the MNR field, the sum of the values of the LL field and the LP field is used.

In the example in figure 2-8, the length field is three characters beginning with character position 22. The minimum number of characters that can be transferred for a partial read or write is 25.

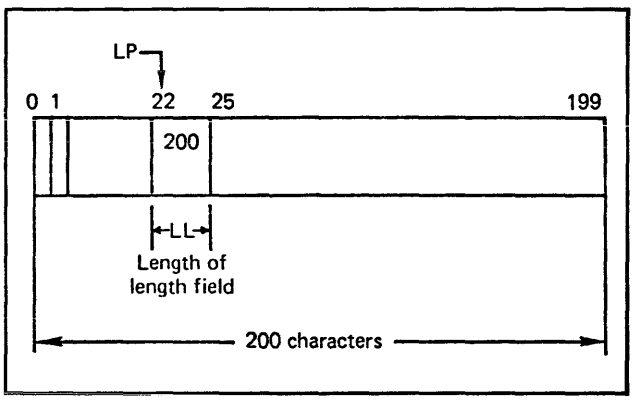
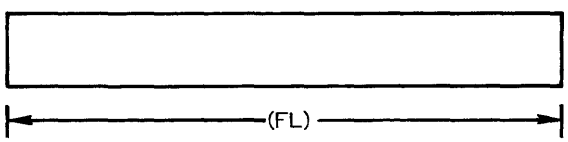


Figure 2-8. D Type Record Example

FIXED LENGTH TYPE F

Fixed length records are defined as records that are the number of characters in length specified by the value of the fixed length (FL) field of the FIT. All records in the file are of equal size. Maximum record length is $10(2^{17}-1)$ characters; minimum length is 10.



Any value in the record length (RL) field of the FIT is ignored, and the number of characters specified by the value of the FL field of the FIT are moved when a GET or PUT macro is issued. A value must be supplied for the FL field for the file to be successfully opened. No padding is supplied on a read.

RECORD MARK TYPE R

The size of the record, which must be less than or equal to the number of characters specified by the value of the maximum record length (MRL) field of the FIT, is specified indirectly by a special delimiting character that terminates each record. The user specifies the delimiting character in the record mark character (RMK) field of the FIT. The same delimiting character is used for each record in the file. This character can be any character of the character set. Maximum record length is $2^{17}-1$ characters.

For a file read, if the delimiting character is not found in the first number of characters specified by the value of the MRL field, that number of characters is moved to the working storage area and an excess data error is given. For writing, if the delimiting character is not found in the first number of characters specified by the value of the MRL field, no data is written to the file and an excess data error is given.

In the example in figure 2-9, MRL=120 and RMK=62_8. The characters are read up to the record mark character.

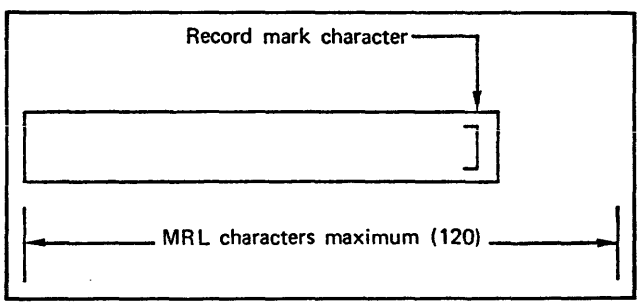


Figure 2-9. R Type Record Example

SYSTEM RECORD TYPE S

On PRU devices, each record is a system-logical-record occupying an integral number of central memory words. On S/L tapes, each record is a tape physical record. The differences in processing of S type records for S/L tapes and PRU devices are shown in table 2-8.

S type records are regarded as word-oriented. When physical blocks are being read from S/L tapes, however, the record length (RL) field of the FIT represents the actual number of characters in the block. For all other cases, the value of the RL field represents the record length rounded upward to a multiple of 10.

An S type record can be created by executing one PUT macro, a series of PUTP macros with a terminating WEOR macro, or a PUTP macro with a TERM parameter. When the WEOR macro is used after a PUTP, level numbers 0 through 16 can be written to terminate the record. Use of levels other than 0 is discouraged, however. If a series of PUTP macros is followed by a PUT, the record written through the PUTP macro is terminated and a new record to satisfy the PUT macro is begun. A user specified value for the RL field causes the current record to be terminated when the number of characters specified has been written.

TABLE 2-8. PROCESSING FOR S TYPE RECORDS

Specification	PRU Device	S/L Tape
Block type	Block type is ignored.	Block type is ignored. Every logical record is one physical record.
Maximum block length	MBL is forced to PRU size of the device.	MBL must be specified by the user and must be greater than MRL.
Record length	RL is rounded up to an integral number of central memory words.	RL specifies the number of characters read or written.
PUT	One system-logical-record of length RL is written, terminated by a level 0.	One physical record of length RL is written (no level number).
PUTP	PTL characters are moved into the buffer (maximum of RL if specified).	PTL characters are moved into the buffer (maximum of RL if specified).
WEOR	Terminate the current record and system-logical-record and write a level 0 through 16.	Terminate the current physical record.
Maximum record length	MRL=0 allows any length record; if MRL≠0 and the record exceeds MRL, an error is given.	MRL must be less than MBL.
GET	MRL must be large enough to contain the entire system-logical-record. If the record exceeds MRL, excess data error is given.	MRL must be large enough to contain the physical record. If the record exceeds MRL, an excess data error is given.
GETP	PTL characters, or the number of characters remaining in the record, are moved from the buffer to WSA.	PTL characters, or the number of characters remaining in the record, are moved from the buffer to WSA.
ENDFILE	Terminate current system-logical-record and write level 0. Write a zero length PRU with level 17.	Terminate the current physical record. Write a tapemark.

A file with unknown format can be specified as having S type records to have a value returned to the RL field after each read; an S tape should be specified on a REQUEST statement (unless the format is known to be SI or I).

TRAILER COUNT TYPE T

T type records consist of a fixed-length base and a variable number of fixed-length trailer items. A count field in the fixed-length base specifies the number of fixed-length trailer items appended to each record. The value recorded in the count field can be display code, right-justified, and zero or blank filled. The fields of the FIT that must be specified for T type records are:

The length of the fixed-length base in the header length (HL) field

The length of fixed-length trailer items in the trailer length (TL) field

The trailer count beginning character position (CP) field, numbered from 0

The count field length (CL) field, one to six characters

The value of the CL field can be COMPUTATIONAL-1 if the C1 field is set to YES, or be a sign-overpunch field if the sign-overpunch (SB) field of the FIT is set to YES. The value of the CP field plus the value of the CL field must be less than or equal to the value of the HL field. The value of the HL field must be less than or equal to the value of the MRL field. Maximum record length that can be specified is $10(2^{17}-1)$ characters. The logical structure of a T type record is shown in figure 2-10.

UNDEFINED TYPE U

This format permits processing of any record type not provided by BAM. The user must supply a value for the record length (RL) field of the FIT for each GET and PUT. The value of the RL field must be less than or equal to the value specified by the maximum record length (MRL) field. Maximum record length that can be specified is $10(2^{17}-1)$ characters.

The RL field of the FIT is altered at the completion of a GET only if an end-of-data has been detected before the number of characters specified by the user in the RL field has been read. The value of the RL field indicates the number of characters transferred.

To read a file with unknown format, an S tape on a REQUEST statement should be specified (unless the format is known to be SI or I tape), and S type records should be specified to have a value returned to the RL field of the FIT after each read.

CONTROL WORD TYPE W

A W type record is any length less than or equal to the number of characters specified by the value of the maximum record length (MRL) field of the FIT beginning at a word boundary (bit 59 of the word). A record is represented in the file as an integral number of central memory words,

prefixed with a record control word supplied by BAM in the format shown in figure 2-11. The control word is written at all block boundaries.

The RL field of the FIT (or the PTL field if a record is written in pieces) must be specified for writing; when reading, the value of the RL field is determined by looking at the control word. Only the characters specified by the value of the RL field are returned to the working storage area on a read. The contents of any unused bits in the last word returned are undefined.

To insure that a tape file with W type records can always be closed, the length of a noise record should be less than 10 characters. W type records cannot be used with E or K type blocks.

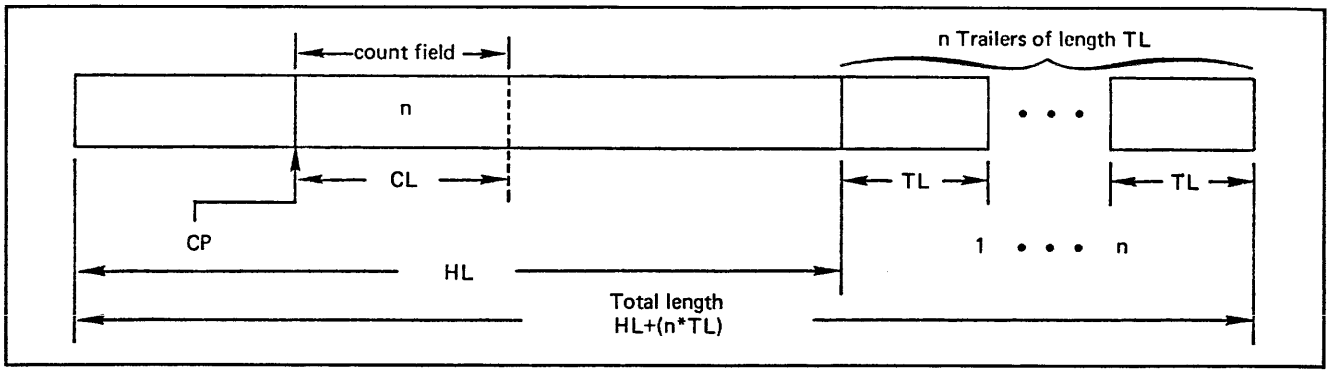
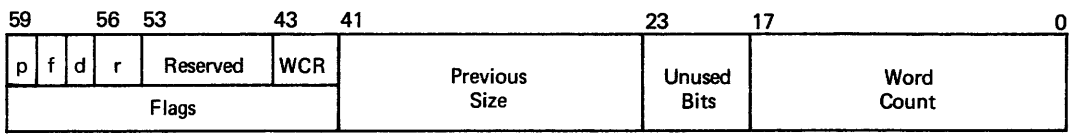


Figure 2-10. T Type Record Format



59 Parity bit. Used to maintain odd parity within the control word.

58 Flag bit. Used in combination with the delete bit to signal the end-of-section or partition. Word count must be zero. The values are:

f	d	
1	1	End-of-section
1	0	End-of-partition
0	1	Deleted record
0	0	Normal record

57 Delete bit. A record, in whose control word this bit is set, is considered to be logically deleted from the file; the record is not passed to the user when the file is read. See the flag bit explanation.

56 thru 54 Reserved for users.

53 thru 44 Reserved for CDC.

43 thru 42 W-continuation record flags. The values are:

43	42	
0	0	Complete record
0	1	First piece of W-continuation record
1	0	Middle piece of W-continuation record
1	1	Last piece of W-continuation record

41 thru 24 The size of the previous record in central memory words including the control word for the record (needed for backspacing). This field is zero if there are no previous records.

23 thru 18 Number of rightmost unused bits in the last word ($0 \leqslant bits \leqslant 59$).

17 thru 0 Number of central memory words necessary to contain the record, not including the control word (word count decimal $\leqslant 2^{17}-2$). If the W-continuation record (WCR) flag is nonzero, the word count refers to only a part of the record.

Figure 2-11. W Type Record Control Word Format

ZERO BYTE TYPE Z

Each record is terminated by a 12-bit byte of zeros in the low-order position of the last word in the record. The full length (FL) field of the FIT must be specified for Z type records. The value of the FL field can be between 1 and $10(2^{17}-1)$.

When a record is read, the zero byte is stripped from the record and blank padding is added to fill the working storage area to (FL+9)/10 words. When Z type records are being read and a zero byte terminator is not found within FL/10+1 words, an excess data error is returned. However, the examination of subsequent characters continues until the first terminator is encountered or a file boundary is reached. If the end-of-information is encountered before the zero byte is found, it is possible the file did not contain Z type records. At the conclusion of a read operation, the RL field of the FIT is set to the number of user characters read, not including blank padding.

When a record is written and the value of the RL field is not zero, the end of the record is determined by searching backwards from the character position specified by the value of the RL field for the first nonblank character. The zero byte is added in the nearest appropriate position. Binary zero-fill is done from the last significant character to the zero byte.

When a record is being written and the value of the RL field is zero, the end of the record is determined by searching backwards from the character position specified by the FL field. When a nonblank character is found, the zero byte is added in the nearest appropriate position. Binary zero-fill is done from the last significant character to the zero byte. If a nonblank character appears in the low-order position of the last word, the record written to the device is one word larger than the physical size of the record in the working storage area, because the nearest appropriate position for the zero byte is in the low-order 12 bits of the word past the character position specified by the FL field. The record on the output device is larger than the value specified in the FL field, but memory is not altered beyond the number of characters specified in the FL field.

If the last character of the record being written is : or % , one blank is appended. If, as a result, the last word of the record contains nine characters, a zero is added to fill out the word and an additional zero word is appended. Z type records give indeterminate results and should not be used on coded 7-track S/L tapes.

A file information table (FIT) is required for all files. Information in this table defines the file and how it is accessed. The FILE macro and FILE control statement are used to create and update the FIT. The FILE macro assembles a FIT in the COMPASS program where the macro is encountered. Pertinent information from the FILE control statement is saved until OPENM time. When the file is opened, the saved information is stored into the FIT and takes precedence over any corresponding preexisting information. A blank FIT, except for addressing information and logical file name, could be set up in the user program with definition of file characteristics deferred until the file is opened.

Fields in the FIT can be changed using the STORE macro or the FILE control statement. The user identifies the fields by the keywords of the FILE macro. Fields in the FIT can be retrieved using the FETCH macro and the keywords of the FILE macro.

Macro requests for file operations can result in amendment of the FIT fields. Certain macro operands are stored in FIT fields prior to performance of the request, and values in FIT fields can be stored as a result of processing the request. Also, certain fields in the FIT are maintained to reflect the current state of the file.

FILE MACRO

The FILE macro constructs the file information table at the address where the macro is encountered during assembly; the FIT must be built before the file is opened. The macro conforms to COMPASS coding conventions. The format of the FILE macro is shown in figure 3-1. The interaction between lfn and LFN=axxxxxx is shown in table 3-1.

The FILE macro does not check fields for validity or consistency. Fields exceeding the maximum specified sizes are truncated; assembler warning messages are produced.

[lfn] FILE [LFN=axxxxxx] [,keyword=option, . . .]

lfn	Symbolic address where the FIT is assembled in the COMPASS program, and logical file name by which the file can be referenced if the LFN=axxxxxx is absent or the same name.
LFN	FIT field mnemonic for logical file name; it must be specified with axxxxxx if lfn is absent.
axxxxxx	Logical file name by which the file can be referenced, and symbolic address where the FIT is assembled in the COMPASS program if lfn is absent.
keyword	Symbolic name of the FIT field.
option	Selected option of the FIT field.

Figure 3-1. FILE Macro Format

Misspelled or unrecognizable parameters generate null parameters, and the fields they reference are set to zero. Null parameters are ignored. Warning messages are generated when overlapping fields are specified.

The FILE macro should specify the file organization mnemonic. Any parameter not applicable to that file organization is ignored and an error type 4 message is generated during assembly.

The values specified for the other FILE macro parameters are assembled into the FIT; they can be specified in any order. Table 3-2 shows which FILE macro parameters are applicable to each file organization. An X indicates appropriate file organizations. (Note that the numbers appearing in parentheses are explained at the end of the table.) A detailed explanation of each FIT field which can be specified by the FILE macro parameters follows:

ASCII ASCII character set bits for INTERCOM terminals.

Absent or ASCII =0

64 character display code.

ASCII=1

95 character ASCII subset.

ASCII=2

128 character ASCII.

BBH Buffer below highest high address. Refer to appendix E for a discussion of the BBH field and loading BAM.

Absent or BBH=NO

Buffer is not below the highest high address.

BBH=YES

Buffer is below the highest high address.

TABLE 3-1. LFN AND lfn INTERACTION

Statement	COMPASS Location Value	Contents of First Word of FIT (lfn)
A FILE	A	A
FILE LFN=A	A	A
A FILE LFN=A	A	A
A FILE LFN=B	A	B

TABLE 3-2. PARAMETERS FOR FILE MACRO BY FILE ORGANIZATION

FILE Macro Parameters	File Organization	
	Sequential (SQ)	Word Addressable (WA)
ASCII	x	
BBH	x	x
BFS	x(1)	x(1)
BT	x	
CF	x	x
CL	x(2)	
CM	x	
CNF	x	
CP	x(2)	
C1	x(2)(4)	
DFC	x	x
DX	x	x
EFC	x	x
EO	x	x
ERL	x	x
EX	x	x
FL	x(3)	x(3)
FO	x	x
FWB	x	x
HL	x(2)	
LA	x	
LBL	x	
LFN	x	x
LL	x(4)	
LP	x(4)	
LT	x	
LX	x	
MBL	x	
MNB	x	
MNR	x	
MRL	x	x
MUL	x	

TABLE 3-2. PARAMETERS FOR FILE MACRO BY FILE ORGANIZATION (Cont'd)

FILE Macro Parameters	File Organization	
	Sequential (SQ)	Word Addressable (WA)
OF	x	
PC	x	
PD	x	x
RB	x(5)	
RMK	x(6)	
RT	x	x
SB	x(2)(4)	
SBF	x	x
SPR	x	
TL	x(2)	
ULP	x	
VF	x	x
WSA	x	x

Notes:

1. Length in words
2. T type records only
3. F and Z type records only
4. D type records only
5. K type blocks only
6. R type records only

BFS Buffer size in words.

Absent or BFS=0

BAM provides the buffer space if necessary; the first word address of the buffer (FWB) field is set to point to the first word address of the space obtained.

BFS=aexp

Buffer size; maximum $2^{17}-1$, or 131000 words. User specifies in words. A practical limit for BFS is $(2^{18}/10) - 1$, or 26200, because this is the largest single move that can be processed.

BT Block type for sequential files; tapes are always blocked.

Absent or BT=I

Internal, block recovery control word; I type.

BT=C

Character count in characters per block; C type.

BT=K

Record count, m records per block; K type.

BT=E

Exact record count; E type.

CF Close flag. File positioning at CLOSEM time.

Absent or CF=R

Rewind

CF=N

No rewind

CF=U

Unload

CF=RET

Return; rewind and unload

CF=DET

Detach; no rewind

CF=DIS

Disconnect terminal file

CL Count field length of a T type record.

Absent or CL=0

No trailer count field defined.

CL=aexp

Length in characters of the trailer; maximum 6.

CM Converison mode.

Absent or CM=NO

No conversion.

CM=YES

Conversion between external and internal code for sequential tape files.

CNF Connect file flag.

Absent or CNF=NO

Normal file input/output.

CNF=YES

Terminal file.

CP Trailer count beginning character position of T type record.

Absent or CP=0

Beginning character position is zero.

CP=aexp

Beginning character position, numbered from zero on the left; maximum $10(2^{17}-1)$.

C1 COMP-1; format for the length field for D and T type records.

Absent or C1=NO

Field is display code.

C1=YES

Field is binary (COBOL COMP-1).

DFC Dayfile control.

Absent or DFC=0

Except for fatal errors, no dayfile messages are written.

DFC=1

Error messages are written on the dayfile.

DFC=2

Notes are written on the dayfile.

DFC=3

Errors and notes are written on the dayfile.

DX End-of-data exit routine address. The system stores a jump at the first address of the routine and control passes to the first executable statement, which is routine+1.

Absent or DX=0

No routine is specified.

DX=exp

Address of the routine to be entered when an end-of-data condition occurs.

EFC Error file control.

Absent or EFC=0

No error file messages are written.

EFC=1

Error messages are written on the error file.

EFC=2

Notes are written on the error file.

EFC=3

Errors and notes are written on the error file.

EO Error option for parity error processing.

Absent or EO=T

Terminate the file.

EO=D

Drop bad data.

EO=A

Accept bad data.

EO=TD

Terminate the file and display the block containing the parity error on error file ZZZZZEG.

EO=DD

Drop bad data and display the block containing the parity error on error file ZZZZZEG.

EO=AD

Accept bad data and display the block containing the parity error on error file ZZZZZEG.

ERL Trivial error limit.

Absent or ERL=0

No trivial error limit; an indefinite number of trivial errors is permitted.

ERL=aexp

Maximum number of trivial errors allowed before a fatal error occurs; maximum 511.

EX Error exit routine address. The system stores a jump at the first address of the routine and control passes to the first executable statement, which is routine+1.

Absent or EX=0

No routine is entered if an error occurs, control is returned to the user's in-line code.

EX=exp

Address of the error exit routine to be entered when an error occurs.

FL Fixed length for F type records; full length for Z type records.

Absent or FL=0

Must be defined for open.

FL=aexp

Record length in characters for F type records, 10 through $10(2^{17}-1)$. For Z type records, 1 through $10(2^{17}-1)$; establishes the upper limit of characters or blank padding moved to the working storage area.

FO File organization.

Absent or FO=SQ

Sequential file

FO=WA

Word addressable file

FWB First word address of the buffer. If FWB is not provided by the user, the minimum buffer needed or the amount specified by the BFS field is provided.

Absent or FWB=0

No user-supplied buffer.

FWB=exp

Address of the buffer.

HL Header length; length of the fixed length portion of a T type record.

Absent or HL=0

Must be defined for open.

HL=aexp

Header length in characters cannot be less than CP+CL; maximum $10(2^{17}-1)$.

LA Label area address.

Absent or LA=0

No area specified.

LA=exp

First word address of the label area.

LBL Label area length.

Absent or LBL=0

No label area length specified.

LBL=aexp

Length in characters; maximum 900.

LFN Logical file name.

LFN=axxxxxx

axxxxxx is a one- to seven-character name beginning with a letter.

LL Length field length of a D type record

Absent or LL=0

Must be defined for open.

LL=aexp

Length in characters; maximum 6.

LP Beginning character position of the length field for a D type record.

Absent or LP=0

Beginning character position is zero.

LP=aexp

Beginning character position numbered from zero on the left; maximum $10(2^{17}-LL-1)$.

LT Label type.

Absent or LT=UL

Unlabeled

LT=S

ANSI standard

LT=NS

Nonstandard

LT=ANY

Any

LX Label routine exit.

Absent or LX=0

No user label processing routine supplied.

LX=exp

Address of the user-supplied label processing routine.

MBL Maximum block length in characters; should not be changed after OPENM.

Absent or MBL=0

The default depends on block type:

BT=K	error
BT=E	error
BT=I	MBL forced to 5120
BT=C	MBL forced to 5120 characters for S tapes and BFS minus two for L tapes; PRU devices considered unblocked

MBL=aexp

Length of data block in characters. For K and E type blocks with Z type records, MBL must not be less than FL + 10. For I type blocks, any MBL is overridden.

MNB Minimum block length for sequential file K and E type blocks.

Absent or MNB=0

No minimum block length specified.

MNB=aexp

Minimum block length in characters; maximum MBL.

MNR Minimum record length of sequential file records.

Absent or MNR=0

Minimum length is zero.

MNR=aexp

Minimum record length in characters; maximum MRL.

MRL Maximum record length of D, R, T, U, and W type records.

Absent or MRL=0

No maximum record length; any record length is acceptable for PUT. No data is moved for GET.

MRL=aexp

Maximum length in characters; maximum $10(2^{17}-1)$. Establishes the upper limit of characters moved to the working storage area.

MUL Multiple of characters per block in which sequential file K and E type blocks are written.

Absent or MUL=0

Characters per block is a multiple of 2.

MUL=aexp

Characters per block is a multiple of aexp; maximum 63.

OF Open flag. File positioning at OPENM time.

Absent or OF=R

Rewind.

OF=N

No rewind.

OF=E

Position at end-of-information for extend.

PC Padding character for sequential file K and E type blocks. Specified in display code. PC must not be the same as the record mark character.

Absent

Padding character is 76_8.

PC=ccB

Padding character is octal value cc; maximum 77_8.

PD Processing direction.

Absent or PD=INPUT

Input (read).

PD=OUTPUT

Output (write).

PD=IO

Input-output (read and write).

RB Records per block in a sequential file K type block.

Absent or RB=0

RB set to 1.

RB=aexp

Blocking factor limit is $2^{12}-1$.

RMK Record mark character in display code. Used as the delimiting character with R type records. RMK must not be the same as the padding character.

Absent or RMK=0

Record mark is 62_8.

RMK=ccB

Record mark is octal value cc; maximum 77_8.

RMK=1Rx

Record mark is x; any character.

RMK=cc

Record mark is decimal value cc; maximum 63.

RT Record type.

Absent or RT=W

Control word

RT=F

Fixed length

RT=R

Record mark

RT=Z

Zero byte type

RT=D

Decimal character count

RT=T

Trailer count

RT=U

Undefined

RT=S

System-logical-records

SB Sign overpunch; COBOL sign overpunch option for the length field for D and T type records.

Absent or SB=NO

Unsigned display code.

SB=YES

Sign-overpunch scheme used.

SBF Suppress buffer flag. Suppresses allocation of buffers and circular buffering. The GETWR and PUTWR functions do not require circular buffers for sequential files with S type records or files with K type blocks and the RB field set to 1. If all the records of a word addressable file are multiples of PRU size and start on PRU boundaries, the circular buffer is not used.

Absent or SBF=NO

Allocates buffers from the information given in the FWB and BFS fields.

SBF=YES

No buffer space is allocated.

SPR Suppress read ahead.

Absent or SPR=NO

Read ahead.

SPR=YES

Read only one block at a time.

TL Trailer length of a T type record.

Absent or TL=0

Must be defined for open.

TL=aexp

Specified in characters; maximum $2^{17}-1$.

ULP User label processing. (See section 6.) Specifies conditions that transfer control to the user label processing routine.

Absent or ULP=NO

None

ULP=V

VOL/EOV

ULP=F

HDR/EOF

ULP=VF

VOL/HDR/EOV/EOF

ULP=U

UVL/UHL/UTL

ULP=VU

VOL/EOV/UVL/UHL/UTL

ULP=FU

HDR/UHL/EOF/UTL

ULP=VFU

All

VF Volume close flag. Volume positioning at CLOSEM time.

Absent or VF=U

Unload

VF=R

Rewind

VF=N

No rewind

WSA Working storage area address. Must be set before any file processing command uses the working storage area. It can be set by macros GET, PUT, and REPLACE.

Absent or WSA=0

No working storage area specified.

WSA=exp

Address of the working storage area.

FILE CONTROL STATEMENT

With the FILE control statement, the user specifies file information to update the FIT when the SETFIT macro is issued, or the first time the file is opened in the job step. This run-time control over file specification allows a single program to process files with different record or block types. Corresponding FIT fields have the value specified on the last control statement encountered.

FILE control statements must be placed before any program call in which the information on them is to be used. Because processing of the FILE control statement involves calling a central processor program, it should not be placed within a load set sequence, for example, between a LOAD and an EXECUTE. If more than one FILE control statement appears for a given file, the data on the first control

statement can be overwritten by the data on a subsequent statement when overlapping fields occur on those statements. The FILE control statement conforms to operating system coding conventions.

If an error diagnostic is produced by FILE control statement processing, the entire statement is ignored. FILE control statement diagnostics are written on the dayfile as soon as the error is encountered; they name the faulty parameter and are self-explanatory. Control is passed to the next EXIT if an error occurs in FILE control statement processing.

The FILE control statement format is shown in figure 3-2. FILE control statement keyword options can be specified in any order. Keywords have the same meaning as described for the FILE macro.

If only the lfn parameter appears in the FILE control statement, the FIT fields for that file revert back to those specified in the program for all succeeding job steps, unless another FILE control statement references that file. If the FILE control statement appears without any parameters, FIT fields for all files revert back to those specified in the program for all succeeding job steps until another FILE control statement is encountered.

The FILE control statement parameters are listed in table 3-3. The various options for a keyword are separated by the | symbol. If the keyword is selected, one of the options must be selected and the others must be omitted.

FILE (lfn[=axxxxxx] [,keyword=option] . . .)

lfn	Name of a FIT; required.
=axxxxxx	Optional new name for the FIT; allows a file to be requested by a new name without reassembly.
keyword=option	Symbolic name of the FIT field and the option selected.

Figure 3-2. FILE Control Statement Format

Parameter values are absolute; generally they refer to number of characters. Value formats are denoted as:

- n . . . n Value is decimal
- n . . . nB Value is octal
- n . . . nW Value is decimal, specified in words

Parameter values for the FIT fields that can be set by the FILE control statement are the same as the parameter values for the FILE macro. The parameter values for the FIT fields that can be set by the FILE control statement but not by the FILE macro are as follows:

LCR Label check/creation. Must be specified by the user.

LCR=E

Existing label is read and checked.

LCR=N

New label is written.

MFN Multifile set name.

MFN=axxxxxx

axxxxxx is the one- to seven-character name beginning with a letter.

PNO Multifile position number. Specifies the position number of the member file on the multifile set.

PNO=aexp

aexp is the position number in display code.

RUN-TIME MANIPULATION

The user can communicate with BAM through the FIT without knowing the exact format of the FIT. This is done with the FETCH, STORE, and SETFIT macros, using the FIT field mnemonics.

TABLE 3-3. FILE CONTROL STATEMENT PARAMETERS

Keyword	Options	Keyword	Options	Keyword	Options
ASCII	0|1|2	FO	SQ|WA	OMIT	macro name/macro name/. . .
BBH	NO|YES	HL	0|n . . . n|n . . . nB|n . . . nW	PC	0|n . . . n
BFS	0|n . . . n|n . . . nB	LBL	0|n . . . n|n . . . nB|n . . . nW	PD	INPUT|OUTPUT|IO
BT	I|C|K|E	LCR	E|N	PNO	0|n . . . n|n . . . nB
CF	R|N|U|RET|DET|DIS	LFN	lfn	RB	0|n . . . n|n . . . nB
CL	0|n . . . n|n . . . nB|n . . . nW	LL	0|n . . . n|n . . . nB	RMK	0|nnB|nn
CM	YES|NO	LP	0|n . . . n|n . . . nB|n . . . nW	RT	W|F|R|Z|D|T|U|S
CNF	NO|YES	LT	S|NS|UL|ANY	SB	NO|YES
CP	0|n . . . n|n . . . nB|n . . . nW	MBL	0|n . . . n|n . . . nB|n . . . nW	SBF	NO|YES
C1	NO|YES	MFN	file name	SPR	NO|YES
DFC	0|1|2|3	MNB	0|n . . . n|n . . . nB|n . . . nW	TL	0|n . . . n|n . . . nB|n . . . nW
EFC	0|1|2|3	MNR	0|n . . . n|n . . . nB|n . . . nW	ULP	NO|V|F|VF|U|VU|FU|VFU
EO	T|D|A|TD|DD|AD	MRL	0|n . . . n|n . . . nB|n . . . nW	USE	macro name/macro name/. . .
ERL	0|n . . . n|n . . . nB	MUL	0|n . . . n|n . . . nB	VF	U|R
FL	0|n . . . n|n . . . nB|n . . . nW	OF	R|N|E		

FETCH

The FETCH macro retrieves the contents of a specified FIT field by a reference to its mnemonics. The macro format is shown in figure 3-3.

FIT field mnemonics can be any of the keywords used with the FILE macro, or any of the fields listed in figure 3-3. The macro generates code to extract the requested value from the FIT. The code expansion destroys values in user registers Xf, Xm, Af, and Xi (which can be Xf or Xm).

STORE

This macro places a user-determined value in a FIT field at execution time. The format of the STORE macro is shown in figure 3-4. The STORE macro generates code to store the requested value in the FIT. This code expansion destroys the values in user registers Xf, Xs, Xm, Af, As, and Xi (which can be Xf, Xs, or Xm).

FETCH fit,keyword,Xi,f,m

fit	Logical file name address of the FIT, or any COMPASS expression giving the FIT address. If fit is Xf or Xm, its contents are changed upon return.
keyword	Any of the keywords in the FILE macro, FILE control statement, or any of the following (when the keyword represents a length, the length is returned as characters):

BN	Block number
ECT	Error count
ES	Error status (equivalent to IRS)
FNF	Fatal error flag
FP	File position field
IRS	Error code
LOP	Last operation
OC	Open/close status
PEF	Parity error flag
PTL	Partial transfer length
RC	Record count
RL	Record length
SES	System error severity
VNO	Volume number
WA	Current word address
WPN	Write bit

f	Number of the X register used to fetch the FIT word. Must be 1 through 5 (default is 5).
m	Number of the X register used as a mask (default is 7).
Xi	X register to receive the value of the requested field. If keyword represents a 1-bit field, it is returned in the sign bit. Keywords that are file names are returned left-justified with zero fill; otherwise, the keyword is returned right-justified with zero fill.

Figure 3-3. FETCH Macro Format

STORE fit,keyword= { option / Xi } ,f,s,m

fit	Logical file name address of the FIT, or any COMPASS expression giving the address or a tag.
keyword	Any keyword described in connection with the FILE macro, except OF, BT, or RT.
option	Options associated with the keyword.
Xi	X register containing the proper code for the keyword. When the keyword represents a length, it is specified in characters.
f	Number of the X register used to fetch the FIT word. Must be 1 through 5 (default is 5).
s	Number of the X register used to store the FIT word. Must be 6 or 7 (default is 6).
m	Number of the X register used as a mask (default is 7).

Figure 3-4. STORE Macro Format

Most FIT fields listed in appendix D can be set symbolically by STORE. Some fields are protected against a STORE; others, such as the structure of a sequential file, are not protected but should not be changed after the file has been opened.

A parameter can be set by using the option with the keyword, or using a register to hold the option as shown in figure 3-5. Examples a and b have an identical effect, just as c and d have an identical effect.

```
a.  STORE fit,RL=10

b.  SX1   10
    STORE fit,RL=X1

c.  STORE fit,FO=SQ

d.  SX1   0
    STORE fit,FO=X1
```

Figure 3-5. STORE Macro Examples

SETFIT

The SETFIT macro sets fields in the FIT. The macro format is shown in figure 3-6. The SETFIT macro makes FILE control statement information available without the need for complete OPENM processing. This makes it possible for system routines to obtain information, such as run-time buffer requirements, needed by other system routines. Also, SETFIT allows the user to cause FILE control statement processing when it would not otherwise occur. Values in all user registers are destroyed.

SETFIT is valid only for a closed file. Once FILE control statement values are placed in the FIT, the macro sets the FILE control statement processed flag (PDF) field of the FIT to inhibit further FILE control statement processing during OPENM. The flag is cleared during subsequent OPENM processing.

If SETFIT is issued and the user setting for the buffer size (BFS) field is zero, the BFS field is set to the buffer size normally allocated, based on other FIT values.

SETFIT fit

fit	Address of the FIT, or an X register containing the address of the FIT.

Figure 3-6. SETFIT Macro Format

This section explains the logical operations of processing a sequential or word addressable file, and explains macros as applicable to each file organization. For a general explanation of all macros and a detailed listing of their parameters, refer to section 5.

Before a file can be processed, the user must establish a file information table (FIT). Establishing the FIT sets a name by which the file can be referenced and defines the file structure and processing limitations. This table contains fields that are referenced whenever BAM processes the file. FIT fields can be set prior to file processing by the FILE control statement, FILE macro, SETFIT macro, and STORE macro.

SEQUENTIAL FILES

In addition to the file manipulation macros, the following macros can be used to process a sequential file:

CHECK,CHECKR

CLOSEM

ENDFILE

GET,GETP,GETWR

OPENM

PUT,PUTP,PUTWR

REPLACE

REWINDM

SKIPdu

WEOR

WTMK

All record types are applicable for sequential files. Except for S type records, records in a sequential file are physically grouped into blocks. Once the user has defined the record and block type, BAM performs all the manipulations required for block construction. Sequential files can reside on mass storage devices or magnetic tape; files with K or E type blocks can reside only on S/L tapes.

OPEN PROCESSING

All files must be initialized using the OPENM macro. Before opening a file, however, the user must call for construction of the FIT by specifying the logical file name. The file organization can also be specified, but the default is sequential.

The record type (RT) and block type (BT) fields, and any other fields needed to describe record and block type, must also be specified before a new file can be opened. For certain systems files, BAM forces the values of the RT, BT, and FL fields of the FIT, as shown in table 4-1.

Consistency checks are performed on certain FIT fields when the file is opened the first time in a job step. Table 5-1 in section 5 lists the fields that are checked for consistency. If a file is closed and then reopened and the close flag (CF) field of the FIT is set to R or N, consistency checks and complete FILE control statement processing are not repeated.

The following fields can be specified prior to opening a file, but need not be set in the FIT until they are required by file processing commands; they can change at any time during a subsequent file processing run:

DX	End-of-data exit; default is no end-of-data routine
EX	Error exit; default is no error routine
ERL	Trivial error limit; default is an indefinite number of trivial errors permitted
DFC	Dayfile control; default is only fatal errors listed
EFC	Error file control; default is no error messages

If label processing is specified, it is initiated during OPENM processing. A conflict between labels specified on the REQUEST statement and the label type (LT) field causes an informative dayfile message and inhibits user label processing. When a labeled file is opened, label checking and creation is based on the label check/creation (LCR) field of the FIT. Refer to section 6 for further information about label processing.

INPUT/OUTPUT PROCESSING

The GET and PUT macros and variations of these macros read and write files. A working storage area must be established to pass data to and from the program and a file storage device. The user defines the working storage area (WSA) by supplying an address for the WSA field of the FIT. This is normally done when the GET or PUT macro is issued. A GET macro transfers data from the buffer area to the working storage area. A PUT macro transfers data from the working storage area to the buffer area.

If only the GETWR, PUTWR, REWINDM, and SKIP macros are to be used for files with logical and physical records equivalent, the suppress buffer flag (SBF) field of the FIT

TABLE 4-1. SYSTEM FILES FORCED VALUES

System File	Forced Values
lfn=INPUT	RT=Z, BT=C, FL=80
lfn=OUTPUT	RT=Z, BT=C, FL=140
lfn=PUNCH	RT=Z, BT=C, FL=80
lfn=PUNCHB	No forced value

can be set to YES. The file must have S type records, or K type blocks with one F or U type record per block. If these restrictions are observed, field length requirements are reduced and central processor time required for each input/output operation is reduced. The elapsed time required to obtain input/output overlap with processing is dependent on the use of the CHECK or CHECKR macro. If the restrictions are not observed, processing advantages do not apply and the use of CHECK or CHECKR is redundant.

Input Processing

The maximum record length (MRL) field of the FIT must be set by the user for reading a file. When a record is transferred from the buffer to the working storage area, if the MRL is zero no data is transferred. If the MRL field is not zero, that value becomes the upper limit for the number of characters transferred even if the record exceeds that length.

Records in a sequential file are read in the order that they occur in the file. They can be read as whole or partial records.

The GET macro reads whole records. The record length (RL), record count (RC), and block number (BN) fields of the FIT are updated during processing. Data transfer always starts at the next record available. If a GET macro is issued when the file is positioned at midrecord because of a prior GETP macro, a skip is made to the record boundary before beginning the GET operation. When the GET macro encounters any end-of-data condition, control is passed to the end-of-data routine.

If the amount of data indicated by the W control word or by the contents of a length or record mark character field is greater than the value specified by the MRL field, the record is truncated to the number of characters specified by the MRL field and an excess data error is returned. If the amount of data is less than the value specified by the fixed length (FL) field on F type records or less than the indicated record length on other types, an insufficient data error is returned.

At the conclusion of a successful read operation, the value of the RL field is the same as the value specified for the RL field for the operation requested. At the conclusion of a read with an insufficient data error, the RL field reflects the number of characters transferred to the working storage area.

The GETP macro transfers part of a record to the user working storage area. The partial transfer length (PTL) field specifies the number of characters to be transferred. At the end of the GETP operation, the PTL field indicates the number of characters actually transferred. The value of the PTL field at transfer completion is the same as the transfer requested unless a record boundary or error condition is encountered.

If the GETP operation initiates record transfer, the EOR flag in the file position (FP) field of the FIT is cleared. When the last data of the record is transferred, the EOR flag is reset. A GETP operation does not cross record boundaries.

The GETP macro transfers characters from the beginning of a record or from the next character available in the record. If the SKIP parameter is specified, however, transfer begins at the start of the next record if current position is within a record. The SKIP parameter is ignored if current position is at the beginning of a record. When the first GETP macro for a record is issued, the RL field is cleared. At the completion of each GETP operation, the RL field is updated to indicate the number of characters read so far.

For U type records, the RL field must be used to specify total record length prior to issuing the first GETP macro for the record. If the length of an S type record is unknown, the user must make a series of GETP requests for PTL characters, where PTL is the length of the working storage area. When the first GETP macro is executed, the FP field of the FIT is set to zero to indicate position in the midst of a logical record. When a subsequent GETP macro completes record retrieval, the EOR flag of the FP field is set, and the length of an S type record becomes known. Consequently, the user must check the FP field for EOR to determine when the record boundary has been reached.

For D and T type records, the first GETP macro for a record must initiate transfer of at least the number of characters specified by the value of the minimum record length (MNR) field. For R type records, the GETP macro is not valid.

S type records can be larger than $2^{23}-1$ characters. In this case, RL is mod 2^{22}.

The GETWR macro initiates the transfer of data in units of words and transfers control to the user. The GETWR macro is intended for use in conjunction with the suppress buffer option. Refer to GETWR processing in section 5 for a complete description of the macro.

Output Processing

The PUT, PUTP, and PUTWR macros write data to a sequential file. An existing file can have records added to it after the previous EOI.

The MRL field need not be set to execute a PUT or variation of a PUT. When a record is transferred from the working storage area to the buffer and the MRL field is set to zero, any number of characters can be written. If the MRL field is not zero, that value becomes the upper limit on the number of characters that can be transferred.

The PUT macro writes an entire record. Data transferred by the PUT macro is written immediately following the last data written to the file. Each PUT operation creates a new record. On R type records, the user must place the record mark character in the record. On D and T type records, the user must set the control fields. The record count (RC) and block number (BN) fields are updated when record and block boundaries are crossed.

The PUTP macro transfers part of a record from the working storage area. The user must set the PTL field to specify the number of characters to be written. The execution of the first PUTP macro begins a new record. The second PUTP macro writes characters immediately after the last character written. The RL field can be specified for the first PUTP macro for S, U, Z, and W type records. If the RL field is zero for Z type records, the value of the FL field is used. For all other record types, the value of the RL field is determined by BAM. When the number of characters equal to the RL value has been transferred, the record is terminated. S, U, and W type records can use the TERM parameter on the last PUTP to terminate the record. An indefinitely long S type record can be written by using a series of PUTP macros followed by a WEOR of any level, or a PUTP macro with the TERM parameter specified. A PUT macro following a series of PUTP requests that did not complete a record is an error.

The PUTWR macro initiates the transfer of data in units of words and transfers control to the user. The PUTWR macro is intended for use in conjunction with the suppress buffer option. Refer to GETWR processing under the GET macro discussion in section 5 for a complete description of the macro.

A file can be updated using the REPLACE macro. The REPLACE macro replaces the last record read with a record from the working storage area. The replacement record has the same record length as the record being replaced, and it must be a mass storage file. The record type can be W or F only; the block type must be C.

Processing 9-Track Binary S/L Tapes

Nine-track tapes must record multiples of eight bits; however, BAM deals exclusively in six bit characters. If the data being written is not a multiple of eight bits, the driver rounds it up to the next multiple of eight bits. If the data being read from the 9-track device is not a multiple of six bits, BAM rounds it up to the next multiple of six bits. If the file is repeatedly copied, a block can contain up to three extraneous undefined six bit characters before it is a multiple of six and eight.

To compensate for this, the user of S type records can either set the maximum record length (MRL) field to three characters larger than the actual data size or ignore the excess data errors. For record types other than S, the user can specify a value of greater than three for the minimum record length (MNR) field; BAM then ignores three or less extraneous characters at the end of the block.

To avoid the extraneous characters when the user is processing eight bit data in S type records, the record length (RL) field should specify a value rounded up to the next multiple of six, and the B8F field should be set to YES. This causes BAM to write the next lower multiple of eight bits to the device.

FILE POSITIONING

The REWINDM macro repositions a mass storage file to the BOI. REWINDM positions labeled tapes to a point after the labels at the beginning of the first file volume. REWINDM positions unlabeled tapes to the load point of the volume currently mounted.

The SKIPdu macro repositions an existing sequential file forward or backward. The user must specify the direction of the skip, the type of units to be skipped, and the number of units to be skipped.

Backward Skipping

A file positioned at unit number m with a skip count of n is positioned to unit m minus n upon completion of the skip backward. Positioned at a unit means ready to read beginning at that unit. The positioning of a file after a SKIPBu of two units is shown in figure 4-1.

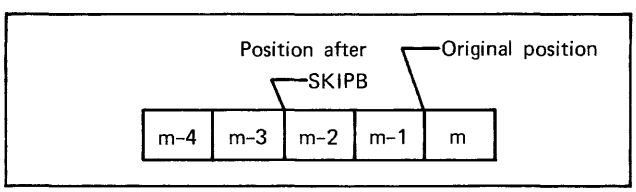


Figure 4-1. SKIPBu Positioning

If an input file is positioned at midrecord when a SKIPBu macro is issued, operation is as if the file were positioned at the end of that unit. If a file is positioned at midrecord when a SKIPBu macro with a zero count is issued, the file is positioned to the start of that unit. A SKIPBL macro after the execution of a PUTP macro that did not terminate a record is an error.

No automatic volume switching occurs when a SKIPBu macro is issued for a multivolume tape file. An error results if the load-point is reached. If a boundary condition is detected before the skip count is exhausted, control is transferred to the end-of-data routine with the appropriate file position set. The file is left positioned immediately before the delimiter. The boundary conditions are:

SKIPBL Section, partition, beginning-of-volume

SKIPBP Partition

SKIPBu Beginning-of-information, load point on a tape file

The restrictions on SKIPBu, with respect to record and block type, are as follows:

SKIPBL is not supported for T, R, U, and D record formats, or for K and E type blocks.

If SKIPBL is attempted when a file residing on a PRU device with C type blocks and F type records is positioned at EOS, EOP, or EOI, it is not possible to determine the exact record boundary. If the fixed length (FL) field is not a multiple of 10, positioning can be unpredictable (not a record boundary).

Forward Skipping

A file positioned at unit number m with a skip count of n is positioned to unit m plus n upon completion of the skip forward. Positioned at a unit means ready to read that unit. If a file is positioned at midrecord when a SKIPFu macro with a zero count is issued, the file is positioned forward to the unit boundary. If a file is positioned in the middle of a record when a SKIPFu macro with a non-zero count is issued, the file is positioned forward to the unit boundary, and then positioned forward the number of units specified. A SKIPFL macro is not allowed with U type records. An output file cannot be positioned forward.

If a boundary condition is detected before the skip count is exhausted, control is transferred to the end-of-data routine with the appropriate file position set. The file is left positioned immediately after the terminator. The boundary conditions are:

SKIPFL Section, partition

SKIPFP Partition

SKIPFu End-of-information

CLOSE PROCESSING

At completion of processing, a file must be closed by the CLOSEM macro. Any remaining records of an output file are written from the buffer to the file storage device; the open/close flag (OC) field of the FIT is set to closed; the action designated by the close flag (CF) field of the FIT is performed; and control is returned to the user.

It is important that all files be closed. During normal termination, the error file is flushed when the last file in a job step is closed. Therefore, error information can be lost if all files are not closed.

End-of-Data Processing

End-of-data occurs when an input/output data transfer or positioning operation is attempted and there is no more data or space on the file, partition, section, or volume because one of the following end conditions was encountered:

End-of-information

End-of-partition

End-of-section

The end-of-data exit (DX) field specifies the address of a user routine for processing an end-of-data condition. When an end-of-data condition exists, control is passed to the address (DX)+1. A jump back to the user in-line return code is stored at the DX address. The file position (FP) field specifies the end condition that caused the transfer of control to the end-of-data exit.

The only requests permitted for sequential files opened for input, after file position EOI has been set, are CLOSEM, REWINDM, and SKIPBu. The only requests permitted for I-O sequential files, after file position EOI has been set, are CLOSEM, SKIPBu, REWINDM, ENDFILE, or PUT.

A GET operation that transfers control to the end-of-data exit does not transfer data to the working storage area. Transfer of control to a user's data exit is an empty GET in that no more data remains; therefore, an end-of-data condition exists. The FP field is not set until a file is logically at the position specified.

Caution must be taken with short records, since PRU devices always contain blocks which are a multiple of 10 characters. EOS, EOP, and EOI are not always correctly detected on a file on a PRU device with F, R, U, D, or T type records and C type blocks when the value of the RL field is less than 10 characters. The padding that has been added to the final block of the file can be greater than or equal to the length of the record. The EOI is not recognized and the padding is processed as valid data.

File Boundary Processing

The CLOSEM macro must be issued to ensure proper EOI processing. The buffer is flushed and, except for unlabeled S/L tapes, an EOI is written to the file. A CLOSEM request for an OUTPUT or I-O sequential file can cause trailer records to be written for W type record files. A deleted zero-length record is written on OUTPUT or I-O sequential files.

Label processing is performed, if appropriate. Label processing performed on I-O sequential files is controlled by the last operation on the file. If the operation was output, labels are created. If the operation was input, labels are checked. On any input labeled file, label checking is performed only if the end-of-information or end-of-volume has been reached. Control is transferred to the user-supplied label routine, if one has been specified.

The CLOSEM VOLUME request forces volume switching to the next reel of a multivolume file. If a value is not supplied for the CF field with the CLOSEM macro, the value in the volume close flag (VF) field of the FIT is used. The current volume number (VNO) field of the FIT is incremented when volumes are switched.

The following actions occur when N is used to specify no rewind on a multivolume file opened for OUTPUT:

Unlabeled S/L Tapes

Two tapemarks are written.

Volume is rewound or unloaded (N parameter is overridden).

New volume is requested by the system and checked.

Data transfer continues on the new volume.

Unlabeled SI, X, B, or I Tapes

Default tapemark and EOV1 label is written.

Volume is rewound or unloaded (N parameter is overridden).

New volume is requested by the system and checked.

Data transfer continues on the new volume.

Standard Labeled S/L and SI, X, B, or I Tapes

If the user has issued a CLOSEM/VOLUME causing the buffer to be flushed, or if the system has detected an end-of-tape, the following occurs:

Control is passed to label routine exit (LX) if defined.

EOV labels are written.

Volume is rewound or unloaded.

Control passes to the LX address if defined.

New volume is requested and checked.

BOV labels are written.

Data transfer continues.

Divisions larger than a record can be specified by issuing a macro to write an end-of-section, end-of-partition, or end-of-information. A partition can be terminated with the ENDFILE macro. Before the EOP is written, the buffer is flushed. The results of ENDFILE depend on the format of the file as described under a description of the macro in section 5.

A section can be terminated by using the WEOR macro. Before the EOS is written, the buffer is flushed. The results of WEOR depend on the format of the file, as shown under a description of the macro in section 5.

The purpose of the WTMK macro is to write tapemarks in nonstandard label processing. It should not be used elsewhere.

TERMINAL FILE PROCESSING

BAM uses a specialized capsule for processing files on terminal devices. It processes Z and S type records. W and U type records can also be specified, but they are processed as S type records. D, T, R, and F type records cannot be specified.

If the files device type is terminal, BAM sets the connect file (CNF) field of the FIT to YES during open processing. If the user sets the CNF field to YES, BAM connects the file.

The user need not reserve buffer space for terminal files. BAM uses one file to write data to the terminal. If the first file written is OUTPUT, that becomes the name of the file used; otherwise, the file used is ZZZZZOU. The user defined FIT is used for reading; data is read directly to the working storage area.

Under the NOS/BE operating system, the type of character set used can be specified by setting the ASCII field of the FIT. If this field is nonzero, the record length (RL) field is still treated as the number of six bit characters to be read or written, but blank stripping and padding is done using a 12-bit ASCII blank (0040B) instead of a six bit display code blank (55B). This ensures no extraneous display code blank (55B) being added or removed from ASCII files. BAM leaves one blank (either 55B or 0040B) on each record written to separate question/answer interaction.

Under the NOS/BE operating system, an input file must be terminated with a %EOF to insure an end-of-data exit. A %EOR is treated as a blank record.

A terminal file can be closed and disconnected by setting the cf parameter to DIS with the CLOSEM macro. The file must be reopened with the CNF field set to NO to be used as a local disk file.

Programs doing terminal I/O and using static loading must use the special names TGET and TPUT with the USE parameter on the FILE control statement to load the special terminal I/O capsule. Refer to appendix E for a discussion of static and dynamic loading.

WORD ADDRESSABLE FILES

In addition to the FIT manipulation macros, only the following macros can be used to process word addressable files:

CLOSEM
GET
OPENM
PUT

Word addressable files must reside on mass storage. Only record types F, U, and W can exist in word addressable files.

OPEN PROCESSING

All files must be initialized using the OPENM macro. Default values are inserted into FIT fields for certain values not supplied prior to open processing.

When a file is opened as a new or existing file, the user must have previously set the record type (RT) field of the FIT, or the default of W type records is set. For F or Z type records, the fixed length (FL) field must also be set.

The following FIT fields can be set before the file is opened and should not be changed until another open is executed:

PD — Processing direction; default is INPUT

FWB — First word address of the buffer; default address is supplied

BFS — Buffer size; default of minimum space is provided except when the suppress buffer flag (SBF) field is set to YES

The following FIT fields need not be set until they are required by file processing commands and can be changed at any time:

EX — Error exit; default is no error routine

DX — End-of-data exit; default is no end-of-data routine

MRL — Maximum record length; default is 0

Certain consistency checks are performed on FIT fields when the file is opened. Table 5-1 in section 5 lists the fields that are checked for consistency.

INPUT/OUTPUT PROCESSING

The GET and PUT macros read and write files. A working storage area must be established to pass data to and from the program and a file storage device. The user defines the working storage area (WSA) by supplying an address for the WSA field of the FIT. This is normally done when the GET or PUT macro is issued. A GET macro transfers data from the buffer area to the working storage area. A PUT macro transfers data from the working storage area to the buffer area.

If all the records of a word addressable file are multiples of PRU size and start on PRU boundaries, the circular buffer is not used to process the records. The suppress buffer flag (SBF) field can be set to YES, and no buffer is allocated. If a record is encountered that is not a multiple of PRU size and does not start on a PRU boundary, an error is issued.

BAM uses the word address (WA) field of the FIT to determine where to read or write data. When a file is opened as a new file, the WA field is set to 1. It is updated after every read or write. If a sequential read or write is desired, the WA field need not be reset by the user.

Any mass storage file can be processed as a word addressable file. Allowances must be made for short PRUs and level numbers; these can be present as a result of previous system processing. An attempt to retrieve word addresses between the end of the short PRU and the start of the next PRU returns an invalid word address error. A read that continues past the short PRU returns an insufficient data error. A read of a level 0 to 16 indicator returns an end-of-section; a read of a zero-length level 17 returns an end-of-partition.

Writing a record into any part of a short PRU causes that PRU to be rewritten as a full PRU without comment. End-of-section or end-of-partition status no longer exists. These files cannot have been written as word addressable files but must have been written as sequential files.

The end-of-data exit (DX) field specifies the address of a user routine for processing an end-of-data condition. An end-of-data exit is taken on an end-of-section or end-of-partition in a W control word. Control is passed to the address (DX)+1. A jump back to the user in-line return code is stored at the DX address. The file position (FP) field specifies the end condition that caused the transfer of control to the end-of-data exit. A read at the end-of-information takes an end-of-data exit with the file position (FP) field of the FIT set to EOI.

Input Processing

The maximum record length (MRL) field of the FIT must be set by the user for reading a file. When a record is transferred from the buffer to the working storage area and the MRL field is zero, no data is transferred. If the MRL field is not zero, that value becomes the upper limit for the number of characters transferred even if the record exceeds that length.

A file is read by the GET macro. The RL field must be set for U type records only. After the GET macro is executed, the RL field contains the number of characters read. W type records are actually one word longer than the RL value returned to allow for the control word. The user must allow for this when calculating the value for the WA field for random access. When a W type record is read, only RL characters are returned to the working storage area. The control word is not returned. If the amount of data indicated by the W type record control word or by the contents of a length field is greater than the value of the MRL field, the record is truncated to the number of characters specified by the value of the MRL field and an excess data error is returned.

Output Processing

The MRL field need not be set to execute a PUT macro. Any number of characters can be written when a record is transferred from the working storage area to the buffer, if the MRL field is set to zero. If the MRL field is not zero, that value becomes the upper limit on the number of characters that can be transferred.

A file is written by the PUT macro. The RL field must be set for U and W type records. The length specified need not be a multiple of 10; however, writing always begins at the left on a word boundary. If the previous write was not a full word, the rightmost character positions are undefined and the next write begins on a new word.

If the value of the WA field is beyond the EOI of the current file, the file is automatically extended and all indications of the previous EOI are gone. Word addressable files are extended in multiples of PRU's. BAM maintains a pointer to the physical EOI but not to the user EOI. If the contents of the file do not require a complete multiple of a PRU, the physical EOI and the user EOI are different.

CLOSE PROCESSING

At completion of processing, a file must be closed by the user with the CLOSEM macro. Any remaining records of an output file are written from the buffer to the file storage device; the open/close flag (OC) field of the FIT is set to closed; and control is returned to the user.

Macros are used for processing the files established with the FILE macro and control statement. An alphabetical listing of all macros and their parameters in COMPASS format is included in this section.

DESCRIPTIVE CONVENTIONS

The macros conform to COMPASS syntax. The location, operation, and variable fields are separated by one or more blanks. In the macro parameter strings, the fit parameter is required. All others are optional and positional. When optional parameters are omitted, their positions must be marked by commas; trailing commas can be omitted.

For example, the format of the OPENM macro is:

```
OPENM fit,pd,of
```

If the pd parameter is not used when the OPENM macro is issued, the format is:

```
OPENM fit,,of
```

The first parameter of every macro identifies the file information table for the referenced file. If the address specified by the fit parameter is invalid, the results are indeterminate. It can be specified by any of the following:

lfn	Location field name of the first word of the FIT, one through seven alphabetic or numeric characters
Rn	Any A, B, or X register containing the FIT address
exp	Any COMPASS expression giving the address of the FIT

When elements are stacked in braces { }, one must be chosen; the others must be omitted. Only parameters applicable to the file organization set in the FIT should be specified. Supplying parameters applicable to the other file organization could cause erroneous results.

MACRO EXECUTION

The current contents of the FIT are used for macro execution. Because the last value set in the FIT is used for execution, default values identified in the macro parameter lists are valid only if the FIT fields have not been changed previously. FIT fields can be set by any of the following:

FILE macro parameters

FILE control statement parameters, which can override defaults during open

A SETFIT macro, which can call for FILE control statement processing without full open processing

Individual fields, which can be set by the STORE macro before or after open

Defaults, which can be set during open

Parameters specified in processing macros that are moved to the FIT before file processing occurs (a zero value in a parameter list moves a zero to the FIT; a null value does not affect the FIT)

The user should presume all registers are destroyed during macro execution. Registers are not saved or restored.

The user macros, with the exception of FETCH, FILE, CLOSEL, STLD.RM, and STORE, generate code as follows:

When checking for syntax errors is completed, all nonnull parameters after the FIT address are placed in registers.

Register B6 is set to the end of the macro expansion as the return address.

A jump to the proper BAM entry point is generated in the top of a word; bits indicating which parameters were specified with the macro are set in the bottom of the word.

The FIT address is placed in register A0; if it is already in A0, no code is generated.

Register B1 is set to 1; if B1=1 pseudo-op is in effect no code is generated.

CHECK

The primary use of the CHECK macro is to check the completion status of input/output operations initiated by GETWR or PUTWR. It can also be used to check input/output completion status after any macro is issued. This macro is applicable to sequential and word addressable files. The file is checked for input/output activity. If active, the job is placed in recall until activity ceases; control is returned to the user. If the file has no input/output activity, control is returned to the user. Data and error exits are suppressed, so the user should examine the file position (FP) and error status (ES) fields of the FIT before continuing.

When the CHECK macro is used to ensure completion of a GETWR request, the RL field contains the record length when CHECK is complete. If an S or L tape is being read, the value of the RL field is the actual number of characters in the record. For S type records on other devices, however, the value of the RL field is the record length rounded upward to a multiple of 10.

When the CHECKR macro is used, the status of the input/output activity is checked and control is returned immediately to the user. The job is not put in recall. If input/output activity is complete, control is returned to a location tag; otherwise, control is returned to the user following the CHECKR.

The formats of the CHECK and CHECKR macros are shown in figure 5-1.

CHECK fit
CHECKR tag_1,fit

fit Address of the FIT.

tag_1 Designates the location to receive control when input/output activity is complete.

Parameters can be specified as registers.

Figure 5-1. CHECK and CHECKR Macro Formats

CLOSEM

The CLOSEM macro terminates file processing and positions the file as specified. It should be the last macro issued for a file. The CLOSEM macro is applicable to both file organizations. Format of the CLOSEM macro is shown in figure 5-2.

When the CLOSEM macro is executed for a file open for output, any information in the file buffer is written to the file device as part of file termination. For sequential files on tape, appropriate label processing occurs during close. Refer to section 6, Label Processing, for a complete description of file and volume label processing.

Close processing for a file varies according to the value specified for the cf parameter of the CLOSEM macro, as follows:

Rewind

The file is rewound.

CLOSEM fit,cf,typ

fit Address of the FIT.

cf Positions the file after close processing:

R	Rewind (default if a FILE close)
N	No rewind
U	Unload (default if a VOLUME close); if a FILE close, release buffer space and remove name from active file list
RET	Return; rewind and unload tape; release buffer space and remove name from active file list
DET	Detach; no rewind; release buffer space and remove name from active file list
DIS	Disconnect; disconnect terminal file and remove name from active file list

typ Type of close to be performed:

FILE	Closes the file; file processing is terminated (default).
VOLUME	Processing on the current volume is terminated, and volumes are switched; the volume number is incremented, and file processing can continue on the new volume without OPENM.

Only the fit parameter can be specified as a register.

Figure 5-2. CLOSEM Macro Format

No rewind

The file is not rewound.

Unload

The file is rewound. The open/close flag (OC) field of the FIT is cleared. If it is a permanent file, it is detached from the job and returned to the permanent file manager. Any unit record file (OUTPUT, PUNCH, or a file that has had DISPOSE performed) is detached from the job and printed or punched. A magnetic tape is unloaded, but the device is not returned to the system. Any scratch mass storage space assigned to the file is released.

Return

The processing is the same as for unload, except that for a tape file, the device is returned to the system.

Detach

The file is not rewound. The open/close flag (OC) field of the FIT is cleared.

Disconnect

The open/close flag (OC) field of the FIT is cleared. The file is disconnected from the terminal.

A CLOSEM request for a file that has never been opened, or a file that has been closed but not unloaded or reopened, has the following effects:

The FIT error status redundant close is set.

File positioning is the same as for an open file.

Control is returned to the error exit.

No label processing is performed.

If a file is closed and then reopened, FIT verification and FILE control statement processing are not repeated if the CF field is set to R or N. Therefore, FIT fields such as BT, RT, and FO should not be changed when the file is reopened. To have FIT verification and FILE control statement processing repeated, the file must have been closed with the CF field set to U, RET, DET, or DIS.

ENDFILE

The ENDFILE macro writes an end-of-partition on a file opened for output or input/output. It is applicable for sequential file organization only. Format of the ENDFILE macro is shown in figure 5-3.

ENDFILE

fit Address of the FIT or register containing the address.

Figure 5-3. ENDFILE Macro Format

For W type records, the ENDFILE macro writes a control word with an end-of-partition flag, and the current PRU or block is terminated. For S/L devices when record type is not W, the ENDFILE macro terminates the current block and writes a tapemark. For PRU devices when record type is not W, the ENDFILE macro terminates the current system-logical-record with a short PRU level 0, and writes a zero-length PRU level 17.

Multiple ENDFILE macros execute as encountered. ENDFILE calls in midrecord are only allowed for files with S type records; for other record types, the end-of-partition is not written and a nonfatal error is issued.

GET

The GET macro retrieves data from a file and delivers it to the working storage area. It is allowed with files opened for input or input/output only. This macro has several forms, which are shown in figure 5-4.

GET	fit,wsa,rl,{dx / ex}, wa
GETWR	fit,wsa,rl
GETP	fit,wsa,ptl,dx,,SKIP
fit	Address of the FIT.
wsa	Address of the working storage area to which the user record is delivered.
ptl	Partial transfer length; number of characters to be transferred.
rl	Record length in characters. Required for U type records only.
dx	Address of the end-of-data routine.
ex	Address of the error routine.
SKIP	Advances to the beginning of the next record before transferring data.
wa	Word address on word addressable files where reading is to start. Word addresses begin with 1.

Parameters (except SKIP) can be specified as registers; if parameters are not specified, values in appropriate FIT fields are used (except GETWR where all parameters are required).

Figure 5-4. GET, GETWR, and GETP Macro Formats

The GET macro transfers a record from a file to the specified working storage area. Lengths are specified and returned in characters. It is applicable for both file organizations.

Applicable parameters by type of file organization for GET are:

Sequential	fit,wsa,rl,dx,
Word addressable	fit,wsa,rl,ex,wa

The following FIT fields are updated during GET processing:

RL Actual length of the record read is returned. Length is specified in characters. For Z type records, the number of significant characters is returned.

RC Record count is updated each time GET reads a record.

For record types other than U, control information in the record or FIT fields is used to determine record length. If the GET request encounters a record longer than the length specified in the maximum record length (MRL) field in the FIT, an excess data error occurs. The number of characters specified by the MRL field are transferred, the remaining characters are skipped, and control passes to the error exit. A record greater than the value specified by the MRL field is prevented from overwriting a portion of the calling program or other preserved information. Control is passed to the user end-of-data exit by a GET request that detects a section or partition boundary, or the end of the file.

The GETWR macro initiates the transfer of data in units of words, and transfers control to the user. GETWR is intended for use in conjunction with the suppress buffer option. The suppress buffer flag (SBF) field of the FIT can be set by a FILE control statement. If the SBF field of the FIT is set to YES, the data is transferred directly to the working storage area, not to the buffer. If the SBF field is set to NO, the data is transferred through the buffer to the working storage area. To check for completion of the operation, the CHECK or CHECKR macro must follow.

The GETWR macro is applicable for sequential files only. The working storage area and the record length must be specified. When reading or writing small S type records to or from an S or L tape, it is sometimes advantageous to set the SBF field to NO, thus gaining nonstop input/output at the expense of buffer space.

When the SBF field is set to NO, all applicable FIT parameters must still be supplied for GET or PUT operations. Also, any data or error exits specified for GET or PUT operations are taken if the SBF field is set to NO. If the SBF field is set to YES, no data or error exits are taken.

The GETP macro transfers partial records in lengths specified by the ptl parameter; it can be used to transfer an arbitrary amount of data from a record. GETP is applicable for sequential files only.

OPENM

Before a file can be read or written, it must be made available by an OPENM macro. Macros that affect the FIT (FILE, STORE, FETCH, and SETFIT) can be used before the OPENM macro. Any file manipulation macro, however, is valid only after the file has been opened. Error procedures are initiated if attempts are made to access an unopened file.

OPENM is applicable to both file organizations. Format of the macro is shown in figure 5-5.

OPENM prepares a file for processing by creating and linking all required system tables for a file, by translating user-supplied parameters into appropriate values in the relevant tables, and by interfacing with label processing. When OPENM is executed, the following events occur:

FILE control statement processing occurs if it has not been suppressed by SETFIT execution. FILE control statement processing can be initiated by SETFIT prior to OPENM. If so, SETFIT sets the PDF field in the FIT to inhibit reprocessing of the FILE control statement. OPENM execution clears the PDF field.

The FIT is checked for logical consistency. Conditions investigated are listed in table 5-1. Depending on the file organization, additional checks can be made for required fields and other defaults supplied.

Buffer parameters are processed.

A read ahead is performed on sequential files opened for input.

Label processing is initiated if appropriate for a sequential file.

OPENM fit,pd,of

fit — Address of the FIT.

pd — Specifies type of processing:

- INPUT — File is opened for read only (default)
- OUTPUT — File is opened for write only
- I-O — File is opened for read and write

of — Open flag; specifies file positioning at open time:

- R — File is rewound before any other open procedures are performed (default)
- N — No file positioning is done before other open procedures
- E — For sequential files, the file is positioned immediately before the EOI to allow extensions to a mass storage file; for permanent word addressable files, the user must issue an EXTEND function if the file is opened with an E position.

Only the fit parameter can be specified as a register.

Figure 5-5. OPENM Macro Format

TABLE 5-1. FIT CONSISTENCY CHECKS

Condition	Action
RT=D, LL=0	Error
RT=T, and CL, HL, or TL=0	Error
RT=Z, FL=0	Error
RT=F, FL=0	Error
RT=T, HL not greater than CL+CP	Error
OF=E, file is not mass storage	Error
FO=LB	Error
Invalid BT field	Error
BT=I, RT≠W	Error
BT=K, RB=0	Default, RB=1
BT=K, MBL=0	Error
MRL, MBL=0, BT=K, E	Error
BT=K, E, file is not S/L device	Error
BT=K, E, RT=W	Error

If no error has been detected, the open/close flag (OC) field in the FIT is set to open and control transfers to the user.

Complete open processing occurs when the first OPENM macro in a job step is issued. If a file is closed and then reopened, FIT verification and FILE control statement processing are not repeated if the close flag (CF) field of the FIT is set to R or N.

Any error detected during open processing sets the error status (ES) field of the FIT. If a user error routine has been specified by the EX field, control passes to that routine. If the user routine corrects the condition that caused the error and executes another open, processing can continue; otherwise, the OC flag reflects 0 (not open) and further file access is prohibited.

Buffer fields are investigated when a file is opened. If the FWB field is zero (no buffer address supplied), an address is allocated. If the BFS field is zero (no buffer size supplied), the minimum space required is calculated and the value is stored in the BFS field. Although BAM sets the buffer pointers in the FIT during OPENM processing, buffer allocation does not actually take place until the first macro requiring a buffer is issued. If the SBF field has been set to YES to suppress buffering, no buffer is allocated.

The timing in relation to specifying file processing parameters and open processing is important. These parameters differ for each file organization. Section 4 lists the requirements for the specific parameters by file organization. The following shows the possible relationships between the OPENM macro and the parameters:

Certain parameters must be set in the FIT with the FILE macro, FILE control statement, or the STORE macro prior to open time; otherwise, a default value is assumed without comment. These parameters are effective only until another open is executed; attempted changes are ignored without comment or error until another open is executed. At that time, the current values in the FIT are used to accomplish the open.

Certain parameters need not be set in the FIT until they are required by file processing commands. Once set, they remain in effect until changed.

Certain parameters have no default and must be set in the FIT to avoid a fatal error prior to use by a file processing command.

PUT

The PUT macro transfers data from working storage to a file; it is allowed for files opened for output or input/output only. This macro has three forms, which are shown in figure 5-6.

The PUT macro transfers a record from working storage to a file. It is applicable for both file organizations.

Applicable parameters by type of file organization for PUT are:

Sequential	fit,wsa,rl,ex
Word addressable	fit,wsa,rl,ex,wa

The rl parameter need not be specified for files with record types F, Z, T, D, and R. Instead, record length for these formats is determined by BAM using fields in the FIT and

PUT	fit,wsa,rl,ex,wa
PUTWR	fit,wsa,rl
PUTP	fit,wsa,ptl,ex,,rl,TERM
fit	Address of the FIT.
wsa	Address of the working storage area.
rl	Number of characters to be written, or for PUTWR the number of words.
ptl	Partial transfer length; number of characters to be transferred.
ex	Address of the error routine.
wa	Word address.
TERM	Signals a record is to terminate with this PUTP; used only with W, S, or U type records.

Parameters can be specified as registers; if parameters are not specified, values in appropriate FIT fields are used (except PUTWR where all parameters are required).

Figure 5-6. PUT, PUTWR, and PUTP Macro Formats

the content of the record in the working storage area. The value of the RL field for F, Z, T, D, and R type records is determined as follows:

F Record length is taken from the FL field of the FIT.

Z If the rl parameter is nonzero in the PUT macro or the RL field of the FIT is nonzero, the end of the record is determined by searching backwards from the character position specified by the value of the RL field. If the rl parameter is not supplied and the RL field is zero, the end of the record is determined by searching backwards from the character position specified by the value of the FL field. A zero byte terminator is appended from that point. Intervening characters are binary zero filled.

R Record length is determined by scanning the record in the working storage area for the terminating record mark character (RMK) which was specified in the FIT. An error occurs if the record mark is not found within the maximum record length.

T Decimal count is extracted from the record and used to calculate the record length. Count field length (CL), trailer count beginning character position (CP), header length (HL), and trailer length (TL) are obtained from fields in the FIT.

D Decimal character record length is extracted from the record. Length field length (LL) and length field beginning character position (LP) are obtained from fields in the FIT.

In all preceding cases, the transferred record length is stored in the RL field of the FIT at the end of the PUT operation.

The RL field must be specified for U, S, and W type records with PUT requests. Lengths specified by the user for W and S format records exclude the record control word and level number appendage. They are supplied by BAM. S type records on a PRU device are always an integral number of words (multiple of ten 6-bit characters) in storage. The value specified by the RL field is rounded upward, if necessary. A level 0 appendage is recorded for each completed PUT operation for S type records. For S/L tapes, the number of characters specified by the RL field are written as one tape block.

For any word addressable files, the word address (WA) field in the FIT is updated to reflect the next available word address; therefore, such files can be written sequentially.

Any errors during PUT or PUTP processing cause transfer to the error routine if one has been specified. In the case of excess or insufficient data errors, no data has been transferred. In the case of other errors, data is unreliable.

The PUTP macro is used to create a single record from a series of write requests. It transfers partial records in lengths specified by the ptl parameter. It can be used to transfer an arbitrary amount of data to a record. By changing the wsa parameter from call to call, portions of the same record can be transferred from different parts of central memory. The PUTP macro is applicable for sequential files only. It is not allowed for R type records.

The PTL field indicates the number of characters to be transferred from the working storage area to the record under construction. The PTL field of the FIT is used for any PUTP operation not containing a ptl parameter value in the macro.

The PTL field must be set for the PUTP macro that initiates a new record. If the record length is specified, it becomes the maximum number of characters possible in the record and is used to determine an excess data error condition. If a PUTP request supplies data that would exceed the record length, or if any other macro requests file action prior to completion of the record, a fatal error condition occurs.

The termination of a record being constructed by a series of PUTP operations is recognized by the total record length (RL) set by the first PUTP macro specified, or by the presence of the TERM parameter to signify the last partial write for this record. For S, U, and W type records using the PUTP macro, the RL field can be set to zero and the TERM parameter used.

The user can make a WEOR request to signal the end of an S type record created by a sequence of PUTP requests. The level number specified by the WEOR macro can be 0 through 16; only level 0 should be written. Levels 1 through 16 exist to support downward compatability in certain system programs. The ENDFILE, REWINDM, CLOSEM, SKIPB, WTMK, and PUT macros also cause termination of a record (block) by adding a level 0. If Z type records are written by the PUTP macro, trailing blanks are suppressed only with the record portion of the last partial transfer.

The PUTWR macro initiates the transfer of data in units of words, and then transfers control to the user. Because the operation might not be complete, the CHECK or CHECKR macro must follow. The PUTWR macro is valid only for sequential files and is intended for use in conjunction with the suppress buffering option. Refer to the GETWR discussion in this section. The working storage area and the record length must be specified with the PUTWR macro.

REPLACE

The REPLACE macro replaces the last record read with a record from the working storage area. It is applicable to sequential mass storage files with C type blocks and F or W type records. Format of the REPLACE macro is shown in figure 5-7.

REPLACE fit,wsa,,ex	
fit	Address of the FIT.
wsa	Address of the working storage area with the new record.
ex	Address of the error routine.
All parameters can be specified as registers.	

Figure 5-7. REPLACE Macro Format

Replacement records must be the same size as the record replaced. If the requested record is not found, a trivial error results and the request is ignored.

REWINDM

The REWINDM macro positions an unlabeled or nonstandard labeled tape file to the beginning of the current volume. A mass storage file or labeled tape is rewound to beginning-of-information. It is applicable to both file organizations. Format of the REWINDM macro is shown in figure 5-8.

REWINDM fit	
fit	Address of the FIT or register containing the address.

Figure 5-8. REWINDM Macro Format

The file need not be open when the REWINDM macro is issued. If the last operation was a write, buffers are cleared and end-of-information written before a file is rewound.

SKIPdu

The SKIPdu macro repositions a file in a forward or backward direction. It is applicable to sequential files only. Format of the SKIPdu macro is shown in figure 5-9. SKIPBL is not supported for T, R, U, and D type records or K and E type blocks; SKIPFL is not supported for U type records.

The SKIPdu macro checks user parameters, reads from the assigned device, positions according to the specified unit to be skipped, and returns control to the user. The SKIPdu macro does not return a record to the working storage area. If a boundary condition is detected before the skip count is exhausted, control is transferred to the end-of-data routine with the appropriate file position set.

A SKIPdu macro call transfers control to the end-of-data routine under the following conditions:

SKIPFL encounters end-of-information

SKIPFL or SKIPBL encounters end-of-partition

SKIPFP or SKIPBP encounters level 17 or a tapemark

SKIPFL or SKIPBL encounters end-of-section

SKIPdu	fit,count
d	Direction of skip:
	F Forward
	B Backward
u	Units to be skipped:
	L Logical records
	P Physical records or system-logical-records of level 0
	F Tapemark or level 17 on PRU devices
fit	Address of the FIT.
count	Number of units to be skipped. A null parameter results in a zero count.
The count and fit parameters can be specified as registers.	

Figure 5-9. SKIP Macro Format

SKIPBL encounters beginning-of-volume

SKIPBu detects the load point on a tape file

SKIPFF or SKIPFP encounters end-of-information

SKIPdP, SKIPdF, and SKIPBL do not detect parity errors. SKIPFL does detect parity errors. A negative skip count is not allowed; the request is ignored, and an error is issued.

If a file is positioned at midrecord when a SKIPdu macro is issued, processing is as follows:

SKIPFu,fit,0	The file is positioned forward to the unit boundary.
SKIPBu,fit,0	The file is positioned backward to the unit boundary.
SKIPFu,fit,n	The file is positioned forward to the unit boundary and then forward n units.
SKIPBu,fit,n	The file is positioned forward to the unit boundary and then backward n units.

An output file can be positioned backward only. If the previous operation was a PUT, the file is terminated before reverse motion is initiated.

WEOR

The WEOR macro is used to terminate a section. The macro format is shown in figure 5-10. WEOR writes an end-of-section for sequential files, if applicable, as shown in table 5-2.

For S type records, a read of EOS returns an EOR value to the file position (FP) field; the EOS value is never returned. For K, E, or C type blocks on an S/L device, an EOS cannot be detected by the GET macro.

WEOR fit,lvl

fit Address of the FIT.

lvl Level number to be appended. Default is 00; lvl can be any octal value from 00 to 16.

All parameters can be specified as registers.

Figure 5-10. WEOR Macro Format

For S type records, the WEOR macro can be used to terminate the system-logical-record being constructed by a series of PUTP macros. The WEOR macro terminates the current record and appends a level number.

For W type records, the file must be on a record boundary to write an end-of-section control word. The record count is updated. The WEOR macro writes a deleted, zero-length record with the flag bit set.

WTMK

The WTMK macro is provided to record a tapemark, or level 17, in nonstandard label processing. It is applicable to sequential file organizations only. Format of the WTMK macro is shown in figure 5-11.

WTMK fit

fit Address of the FIT or register containing the address.

Figure 5-11. WTMK Macro Format

The WTMK macro does not flush the buffer. It checks user parameters, terminates the current block, records the tapemark on S/L tapes and a level 17 for files residing on PRU devices. Control is then returned to the user. The block number (BN) field of the FIT is not cleared to zero.

TABLE 5-2. WEOR PROCESSING

Device	End-of-Section		Boundary Written
	Block Type	Record Type	
PRU device	I	W	One-word record pointing back to the last I block boundary. Control word with EOS flags; terminate the block with level 0
	C	W	Control word with EOS flags; terminate the block with level 0.
	C	All but W	Terminate the block with level not greater than 16_8.
S/L tape	I	W	Zero-length deleted records to exceed noise record size; one-word record pointing back to the I block boundary; control word with EOS flags. Terminate the block.
	C	W	Zero-length deleted records to exceed noise record size. Control word with EOS flags. Terminate the block.
	C,K,E	All but W	Terminate the block.

Tape label processing takes place when a sequential file on magnetic tape is opened or closed. File labeling conventions facilitate the exchange of magnetic tapes between installations. Recording a file using any labeling convention has meaning only for sequential files. The tape formats supported under the NOS operating system are: SI binary, I, and S/L. The tape formats supported under the NOS/BE operating system are: SI coded and binary and S/L.

LABEL DEFINITIONS

The three basic classes of labeling conventions are standard labeled files, nonstandard labeled files, and unlabeled files.

STANDARD LABEL

A standard labeled file is recorded with label groups appended to the data. The content of the labels and the format of the file so recorded conform to the American National Standards X3.27-1969, Magnetic Tape Labels for Information Interchange. Standard label processing applies only to sequential files on magnetic tape.

A label group is composed of a number of 80-character blocks separated by an interrecord gap. The label group is separated from the data records in the file by a hardware tapemark. The three types of label groups are volume/-header group, end-of-file group, and end-of-volume group. The position of these groups in relation to file data is shown in figure 6-1. Table 6-1 shows the contents of each label defined by ANSI.

NONSTANDARD LABEL

A nonstandard label is a descriptive record appended to data according to a set of rules other than the ANSI standard convention. BAM allows nonstandard labels to be written, for processing by the user, for sequential files on all devices.

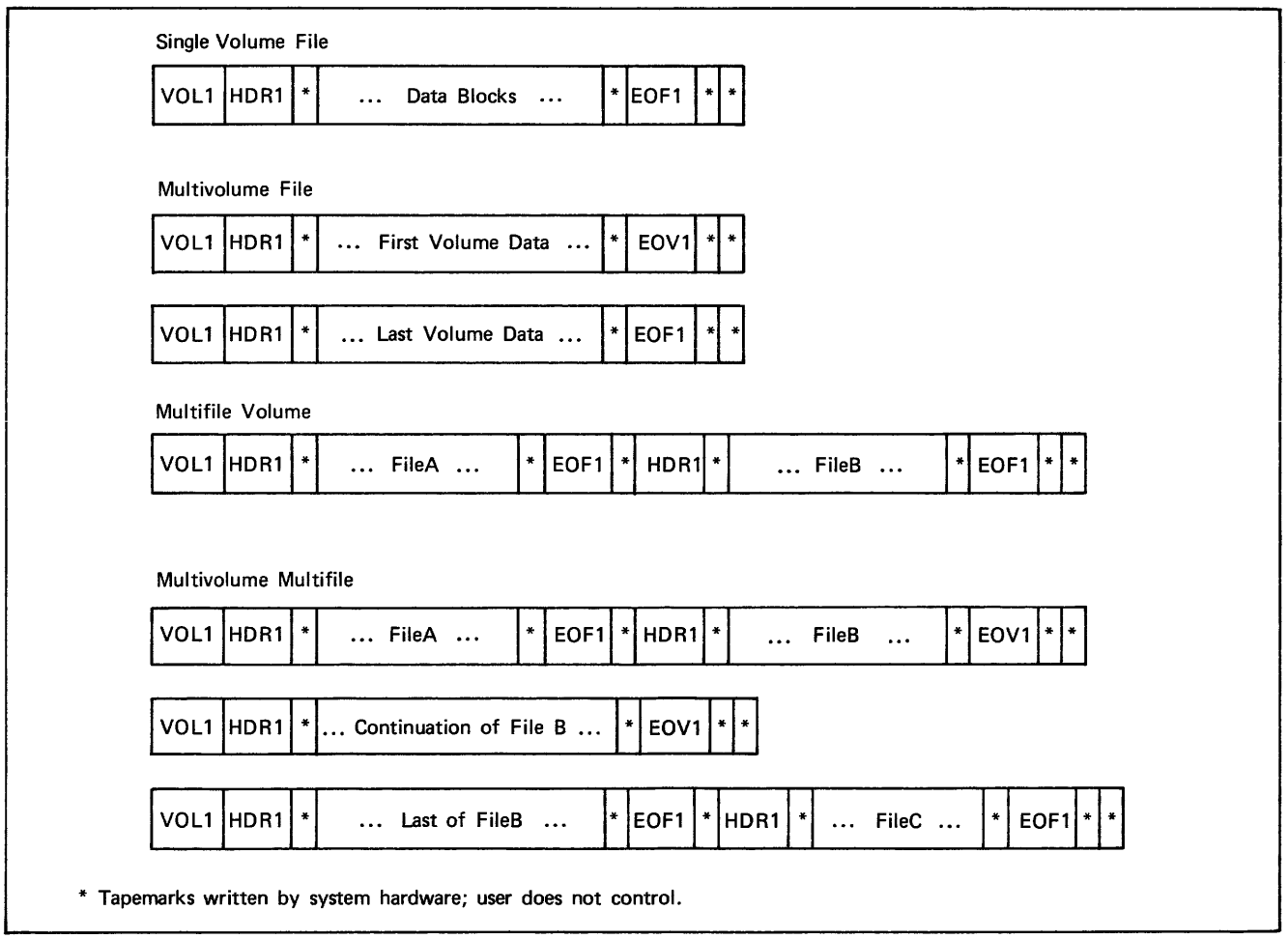


Figure 6-1. Standard Label Tape Formats

TABLE 6-1. ANSI STANDARD LABELS

Label	Character Position	Field	ANSI Name (System Name)	Length	Contents	Default Written	Checked On Input
Volume header	1-3	1	Label Identifier	3	VOL	VOL	Yes
	4	2	Label Number	1	1	1	Yes
	5-10	3	Volume Serial Number	6	Any characters	As typed from console	Yes if file assigned
	11	4	Accessibility	1	Space	Space	No
	12-31	5	Reserved	20	Spaces	Spaces	No
	32-37	6	Reserved	6	Spaces	Spaces	No
	38-51	7	Owner ID	14	Any characters	Spaces	No
	52-79	8	Reserved	28	Spaces	Spaces	No
	80	9	Label Standard Level	1	1	1	No
First file header	1-3	1	Label Identifier	3	HDR	HDR	Yes
	4	2	Label Number	1	1	1	Yes
	5-21	3	File Identifier (File Label Name)	17	Any characters	Spaces	Yes
	22-27	4	Set Identification (Multifile Set Name)	6	Any characters	Volume serial number of first reel of the set	No
	28-31	5	File Section Number (Reel Number)	4	4 digits indicating number of volume in the file	0001	Yes
	32-35	6	File Sequence Number (Position Number)	4	4 digits indicating number of file in multifile set	0001	Yes
	36-39	7	Generation Number	4	(Not used by the operating system)	Spaces	No
	40-41	8	Generation Version Number (Edition Number)	2	2 digits indicating the edition of the file	00	Yes
	42-47	9	Creation Date	6	Space followed by 2 digits for year, 3 digits for day	Current date is used	Yes
	48-53	10	Expiration Date	6	Same as field 9	Same as field 9	Yes
	54	11	Accessibility	1	Any characters	Space	No
	55-60	12	Block Count	6	Zeros	Zeros	Yes
	61-73	13	System Code	13	Any characters	Spaces	No
	74-80	14	Reserved	7	Spaces	Spaces	No
Additional file header	1-3	1	Label Identifier	3	HDR	HDR	Yes
	4	2	Label Number	1	2-9	2-9	Yes
	All other fields are not checked on input; they are written as received from the user.						
First end-of-file	1-3	1	Label Identifier	3	EOF	EOF	Yes
	4	2	Label Number	1	1	1	Yes
	5-54	3-11	Same as corresponding HDR1 label fields				
	55-60	12	Block Count	6	6 digits indicating number of data blocks since the last HDR label group		Yes
	61-80	13-14	Same as corresponding HDR1 label fields				

TABLE 6-1. ANSI STANDARD LABELS (Cont'd)

Label	Character Position	Field	ANSI Name (System Name)	Length	Contents	Default Written	Checked On Input
Additional end-of-file	1-3	1	Label Identifier	3	EOF	EOF	Yes
	4	2	Label Number	1	2-9	2-9	Yes
	All other fields are not checked on input; they are written as received from the user.						
First end-of-volume	1-3	1	Label Identifier	3	EOV	EOV	Yes
	4	2	Label Number	1	1	1	Yes
	All other fields are identical to EOF1 label.						
Additional end-of-volume	1-3	1	Label Identifier	3	EOV	EOV	Yes
	4	2	Label Number	1	2-9	2-9	Yes
	All other fields are not checked on input; they are written as received from the user.						
USER	1-3	1	Label Identifier	3	3-letter code: UVL, UHL, or UTL		Yes
	4-80	Any characters. Content of these fields is not checked on input; it is written as received from the user.					

UNLABELED

An unlabeled file has no system descriptive records at the beginning of the file. The first block of the file is treated as a data block. An unlabeled file on an SI or I tape has a system-processed trailer label. The presence of this label allows end-of-information to be defined. Multivolume processing is done automatically by the operating system for SI or I tapes.

On an S/L tape, no system trailer label exists and end-of-information is undefined. On input, a tapemark encountered after the end-of-tape reflective spot signals end-of-volume and the operating system switches volumes. The formats of unlabeled magnetic tape files are shown in figure 6-2.

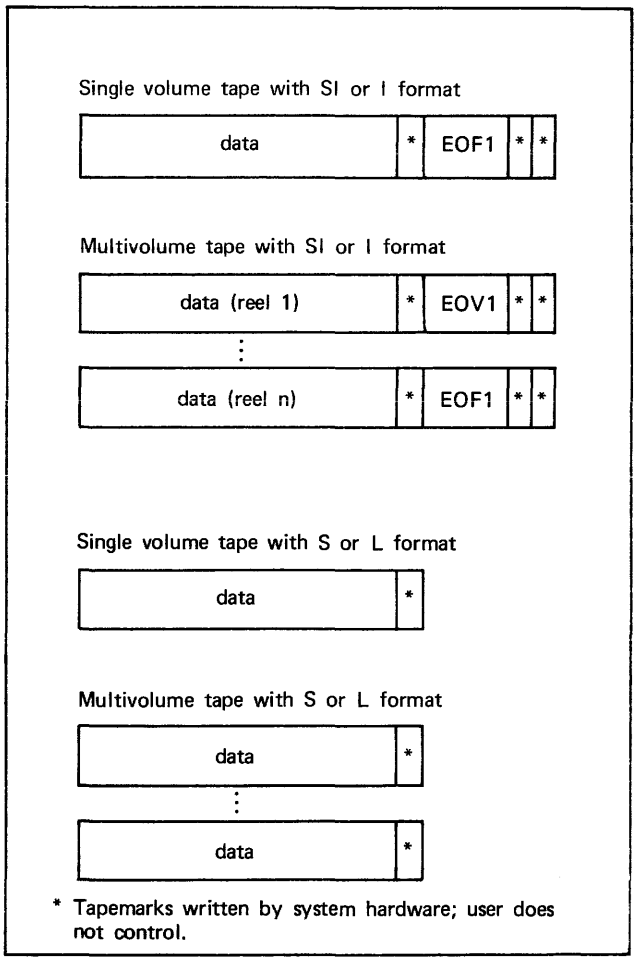


Figure 6-2. Unlabeled Tape Format

LABEL PROCESSING FIT FIELDS

The following FIT fields are used during label processing:

LT Label type. LT is determined when the file is opened, based on parameters on the FILE macro or control statement. If label type is unspecified, user label processing is not allowed.

- LT=S Standard
- LT=UL Unlabeled
- LT=NS Nonstandard
- LT=ANY Unspecified

LCR Label check/creation. LCR is determined when the file is opened, based on the parameters of the FILE macro or control statement. It must be specified by the user.

- LCR=E Existing label is read and checked
- LCR=N New label is written

LA Label area address. Labels are delivered into the area as a result of the GETL macro. Labels are fetched and submitted for processing as a result of the PUTL macro. If LA is zero, no user label processing can be done.

LBL Label area length in characters.

LX Label routine exit address. Control is passed to the LX routine for user label processing at certain file positions, depending on the contents of the ULP field.

ULP User label processing. Types of label processing that are available for standard labels. Any specification for ULP except NO is acceptable. For nonstandard labels when ULP is not set to NO, the user must process all labels.

ULP=NO	No user label processing
ULP=V	User volume label processing
ULP=F	User file label processing
ULP=U	User label processing (UHL, UTL, UVL)
ULP=VF	Combination of V and F
ULP=VU	Combination of V and U
ULP=FU	Combination of F and U
ULP=VFU	Combination of V, F, and U

DECLARING LABEL TYPE

Before a file is opened, the user must set the label type (LT) and label check/creation (LCR) fields of the FIT with a FILE macro or control statement. The equivalent information must be specified on a LABEL or REQUEST control statement. Refer to the appropriate operating system reference manual for a complete description of these control statements. Under the NOS operating system, a file with nonstandard label type must be declared unlabeled on the REQUEST or LABEL statement.

A standard labeled multifile set is specified by putting the MF and MFN parameters on the REQUEST control statement and setting the MFN field of the FIT. The format of a multifile set is explicitly specified by the ANSI standard. User label processing of any of the label groups is supported as described in the User Label Processing Macros subsection. A multifile set can be created and read only if the user supplies all the labels.

Tapes with earlier standard labels, Z labels, can be read but not written. They must be identified by the Z parameter on the REQUEST or LABEL control statement for the operating system to read these files. Under the current ANSI standard, density of label data is the same as that of subsequent data. Earlier standards allowed data recording density to be specified by character 12 of the VOL1 label.

STANDARD LABEL PROCESSING

The VOL1, HDR1, EOV1, and EOF1 labels are always processed to ensure adherence to ANSI standards and LABEL statement parameters. VOL1, EOF1, and EOV1 are written by the system with default values. Any user values are ignored without comment. A tapemark is written to terminate label groups on standard labeled output tapes.

User processing is allowed on ANSI-defined labels when label type (LT) is declared standard and the user label processing (ULP) field is set to identify a label group. Working knowledge of standard magnetic tape label structures is necessary for the following label processing discussions. The notations UTL(f) and UTL(v) indicate all the user trailer (UTL) labels that can follow EOF and EOV labels. UHL(a) and UVL(a) indicate all the user header (UHL) or user volume (UVL) labels that can follow the HDR and VOL1 labels.

A file is initialized by the operating system when the LABEL control statement is encountered. Consequently, user label processing should not be attempted when a LABEL control statement is used.

User label processing is controlled by the FIT fields LX, LA, and ULP. Control passes to a user label processing exit if the user has specified user label processing (ULP on the FILE control statement or macro), and the position of the standard labeled file is such that there are labels to be processed. User label processing (ULP) can be specified for some label groups and not others. If LX is zero, the system supplies the requisite label.

Label processing capabilities are provided by the GETL, PUTL, and CLOSEL macros. GETL retrieves the next label of a label string and delivers it to the label area. Labels are written by PUTL. CLOSEL terminates label processing. More detail is given under User Label Processing Macros.

INPUT TAPE USER PROCESSING

Existing standard labels can be checked when the processing direction (PD) field of the FIT is set to INPUT, or I-O with the LCR field set to E. Labels can be retrieved with GETL; each GETL returns an 80-character label to LA. If the value of the LBL field is less than 80, LBL characters are retrieved and an error flag is set. If LBL exceeds 80, an error flag is set and 80 characters are delivered to LA.

OPENM of Input Tape

If the value of the LX field is zero, the header label group is processed automatically by the operating system. If the LX field is nonzero, control is given to the user's routine twice during OPENM processing. At the first exit, the open/close flag (OC) field of the FIT is set to not open and the PUTL macro can be issued to have the system perform a label check. At the second exit, the OC field is set to open and the GETL macro can be issued.

The labels that can be retrieved during OPENM processing are shown in table 6-2 in the order in which successive GETL macros would retrieve them. Label processing is not allowed for an OPENM with no rewind.

CLOSEM of Input Tape File

If the LX field is zero, EOF1 is processed automatically by the operating system. If LX is nonzero, control is passed to the user label processing routine twice. At the first exit, the OC field of the FIT is set to open and the PUTL macro can be issued to have the system perform a label check. At the second exit, the OC field is set to closed, and trailer labels can be retrieved with the GETL macro. The labels available depend on the contents of the ULP field as shown in table 6-3.

TABLE 6-2. INPUT FILE LABELS ACCESSED AT OPENM

ULP	Labels Retrieved by GETL
V	VOL1
F	HDR1-9
U	UVL(a), UHL(a)
VF	VOL1, HDR1-9
VU	VOL1, UVL(a), UHL(a)
FU	HDR1-9, UHL(a), UVL(a)
VFU	VOL1, UVL(a), HDR1-9, UHL(a)

TABLE 6-3. INPUT FILE LABELS ACCESSED AT CLOSEM

ULP	Labels Retrieved by GETL
V	None
F	EOF1-9
U	UTL(f)
VF	EOF1-9
VU	UTL(f)
FU	EOF1-9, UTL(f)
VFU	EOF1-9, UTL(f)

CLOSEM of Input Tape Volume

If the LX field is zero, EOV1 is processed automatically by the operating system. Volumes are switched and header labels are processed automatically. If the LX field is nonzero, control is passed to the user at address LX when the file position (FP) is beginning-of-volume (BOV) because CLOSEM VOLUME could have been issued midreel.

When an EOV occurs because the label group indicating end-of-tape has been reached by a GET or SKIPFL macro, control is transferred to the LX address at EOV and BOV. The user must differentiate between these file positions in the label routine of the program.

When the file position is EOV, the labels that can be retrieved by the GETL macro are those listed in table 6-4. When the file position is BOV, the labels that can be retrieved by the GETL macro are those listed in table 6-5.

OUTPUT TAPE USER PROCESSING

A new standard label can be written when the processing direction (PD) field is set to OUTPUT or when the PD field is set to I-O and the LCR field is set to N.

In a user label routine, labels can be written with the PUTL macro. Each PUTL takes one 80-character label from address LA and writes it to the file. The 80 characters must be correctly formatted or an error results. If the length of the label area or the label address is zero, no user labels are written, but default VOL1, HDR1, EOF1, or EOV1 label is supplied and an error is returned.

TABLE 6-4. INPUT FILE LABELS ACCESSED AT CLOSEM VOLUME (EOV)

ULP	Labels Retrieved by GETL
V	EOV1-9
F	None
U	UTL(v)
VF	EOV1-9
VU	EOV1-9, UTL(v)
FU	UTL(v)
VFU	EOV1-9, UTL(v)

TABLE 6-5. INPUT FILE LABELS ACCESSED AT CLOSEM VOLUME (BOV)

ULP	Labels Retrieved by GETL
V	None
F	HDR1-9
U	UVL(a), UHL(a)
VF	HDR1-9
VU	VOL1, UVL(a), UHL(a)
FU	HDR1-9, UHL(a), UVL(a)
VFU	VOL1, UVL(a), HDR1-9, UHL(a)

VOL1, HDR1, EOF1, and EOV1 are ANSI-required labels and the operating system ensures that these labels are written to the file. The user can supply the labels in any order and the operating system reorders them. User label processing is allowed on output only if the OPENM macro with the rewind option is used.

OPENM of Output Tape

If the LX field is zero, default VOL1 and HDR1 labels are supplied automatically. If the LX field is nonzero, control is passed to the user label processing routine. Labels that can be written are indicated in table 6-6. Each PUTL macro writes one label.

TABLE 6-6. OUTPUT FILE LABELS WRITTEN AT OPENM

ULP	Labels Written by PUTL
V	None
F	HDR2-9
U	UVL(v), UHL(f)
VF	HDR2-9
VU	UVL(v), UHL(f)
FU	HDR2-9, UHL(f), UVL(v)
VFU	UVL(v), HDR2-9, UHL(f)

CLOSEM of Output Tape File

If the LX field is zero, a default EOF1 is supplied automatically, and any user EOF1 is ignored. If the LX field is nonzero, control is passed to the user label processing routine. Labels that can be written are shown in table 6-7.

TABLE 6-7. OUTPUT FILE LABELS WRITTEN AT CLOSEM

ULP	Labels Written by PUTL
V	None
F	EOF2-9
U	UTL(f)
VF	EOF2-9
VU	UTL(f)
VFU	EOF2-9, UTL(f)

CLOSEM of Output Tape Volume

If the LX field is zero, default EOV1, VOL1, and HDR1 labels are supplied. If the LX field is nonzero, control is passed to address LX when the file position is EOV and BOV. In either case volume switching is automatic.

When the file position is EOV, the labels that can be written with the PUTL macro are those listed in table 6-8. EOV1 is always supplied by the operating system because its content must be an image of HDR1. If a user issues a PUTL macro for an EOV1, it is ignored. When the file position is BOV, the labels that can be written with the PUTL macro are those listed in table 6-9.

TABLE 6-8. OUTPUT FILE LABELS WRITTEN AT CLOSEM VOLUME (EOV)

ULP	Labels Written by PUTL
V	EOV2-9
F	None
U	UTL(v)
VF	EOV2-9
VU	UTL(v), EOV2-9
VFU	EOV2-9, UTL(v)

TABLE 6-9. OUTPUT FILE LABELS WRITTEN AT CLOSEM VOLUME (BOV)

ULP	Labels Written by PUTL
V	None
F	HDR2-9
U	UVL(a), UHL(a)
VF	HDR2-9
VU	UVL(a), UHL(a)
FU	HDR2-9, UHL(a), UVL(a)
VFU	UVL(a), HDR2-9, UHL(a), EOV2-9

NONSTANDARD LABEL PROCESSING

Nonstandard label processing is entirely the responsibility of the user. This type of label processing is available for sequential files on all devices.

The nonstandard labels can be header and/or trailer labels. Header labels appear between the beginning-of-information and a user-defined point. Trailer labels appear between some other user-defined point and end-of-volume or end-of-information. The delimiting and processing of nonstandard labels is the user's responsiblity.

INPUT FILE USER PROCESSING

Each GETL macro retrieves the number of characters of data specified by the label area length (LBL) field, or fewer characters, from a physical record and delivers them to address LA. If a tapemark or level 17_8 is reached, the GETL macro returns with an end-of-labels file position and no data is transferred.

For an input file, control is passed to the label processing routine during OPENM processing or CLOSEM processing when the file is positioned at end-of-partition. The nonstandard label can then be retrieved with the GETL macro.

During CLOSEM processing of an input volume, control is passed to the user at address LX. The CLOSEM macro should be called when the user has determined that end-of-volume processing is required. File position should be end-of-section or end-of-partition, and labels should be separated from data. If an end-of-data is encountered during forward reading, control is passed to the end-of-data exit (DX) routine if present. End-of-volume labels must be processed in the end-of-data routine before CLOSEM is called. The user has the option of issuing a CLOSEM VOLUME/FILE at this time. If CLOSEM VOLUME is issued, volumes are switched automatically and control is passed to the user at address LX at load-point.

If the system closes an input volume, automatic volume switching takes place only at the first tapemark after the reflective spot. Control is passed to address LX at BOV for label processing.

OUTPUT FILE USER PROCESSING

Each PUTL macro delivers the number of characters specified by the LBL field from the label area (LA) to the input/output device. The data is formatted as one physical record. The user can use the WTMK macro for writing record delimiters. Delimiters are not required; processing is entirely up to the user. When the system closes the volume because the reflective spot has been encountered, the output buffer is not flushed. If the user closes the volume, the buffer is flushed before any label processing.

For an output file, control is passed to address LX during OPENM and CLOSEM processing. The user can then write labels with the PUTL macro.

For an output volume, control is passed to address LX twice during CLOSEM processing. The first time is for creation of trailer labels, and the second time is for the creation of header labels.

Automatic volume swapping occurs after the tape reflective spot is encountered. In this case, label processing is available only at BOV.

USER LABEL PROCESSING MACROS

Macros provide label processing capabilities. The macros provided retrieve labels (GETL), submit labels for writing or checking (PUTL), and terminate user label processing (CLOSEL). They are applicable only for sequential files.

GETL

The GETL macro retrieves the next label of a label group and delivers it to the label area. Format of the macro is shown in figure 6-3.

GETL fit,la,lbl

fit	Address of the FIT.
la	Address of the label area; holds the label fetched by the GETL macro.
lbl	Length (in characters) of the label area.

All parameters can be specified as registers.

Figure 6-3. GETL Macro Format

During OPENM and CLOSEM processing, entry is made into the label routine and labels appropriate to the current file position are made available to the user via the GETL macro. The GETL macro validates the contents of certain FIT fields and ensures the legality of the call. The file organization (FO) field must be set to sequential (SQ). The label type (LT) field must be set to standard (S) or to nonstandard (NS) with the ULP field set to other than NO. The processing direction (PD) field must be set to INPUT or to I-O with the LCR field set to E. A check is made that the LBL field is nonzero, and that the label area is specified. If the labels are standard, the file must be a tape file and the user label processing flags must be set (the ULP field not set to NO).

If labels are standard, the number of characters specified by the LBL field are moved to the user label area at LA. If the LA field has not been set either previously or by the GETL macro, an error exit is taken. If the number of characters specified by the LBL field is greater than 80, only 80 characters are retrieved. If the LX field is zero, no label processing routine exists. The ULP field is used in conjunction with the file position (FP) field to determine what type of label is to be retrieved.

When the GETL macro is issued for standard labels and no errors are detected, the user label processing flags are checked to determine what types of labels are appropriate. The next appropriate label is moved to the label area at LA. If none exists, the end-of-labels flag is set and control is returned to the user label routine. If the value of the LBL field is other than 80, an error status is set. If the LBL field is greater than 80, only 80 characters are moved to the label area at LA. If the LBL field is less than 80, only the number of characters specified by the LBL field are moved to the label area. Labels are retrieved in sequential order. For example, at beginning-of-information with the ULP field set to F, the labels on a file containing HDR1, HDR2, and HDR3 labels would be available in the order HDR1, HDR2, HDR3. Each call to GETL would retrieve only one label.

When the GETL macro is issued for nonstandard labels and no errors are detected, a physical record is read and the number of characters specified by the LBL field are moved to the label area at LA. If the physical record is larger than the LBL field, only the number of characters specified by the LBL field are moved. If the physical record is smaller than the LBL field, as many characters as possible are moved and the number of characters moved are returned in the LBL field.

PUTL

The PUTL macro writes a label. Format of the macro is shown in figure 6-4.

PUTL fit,la,lbl

fit	Address of the FIT.
la	Address of the label area; contains the label to be written on the file.
lbl	Length (in characters) of the label area.

All parameters can be specified as registers.

Figure 6-4. PUTL Macro Format

During OPENM and CLOSEM processing, entry is made into the label routine. At this time, labels appropriate to the current file position are submitted to be written on an output file. The PUTL macro validates the contents of certain FIT fields to ensure the legality of the call. The file organization (FO) field must be set to sequential (SQ). The label type (LT) field must be set to standard (S) or to nonstandard (NS) with the ULP field set to other than NO. The processing direction (PD) field must be set to OUTPUT or to I-O with the LCR field set to N. Additional checks are made that the LBL field is nonzero, and that a label area is specified. If the labels are standard, the file must be a tape file and the user label processing flags must be set to other than NO.

The ULP field is used in conjunction with the file position (FP) field to determine if the label being submitted is legal at the present file position. The first three characters of the label at LA are used to determine the type of label: VOL, HDR, EOV, or EOF. If the LA field has not been set either previously or by the PUTL macro, an error exit is taken.

For standard labels, if no errors have been detected, each call to the PUTL macro examines the label at LA, keying on the first four characters. The ULP field is checked to see if the submission of the label at the current file position is allowed. At beginning-of-information with the ULP flag set to F, submission of a VOL1 label would not be allowed.

For nonstandard labels, if no errors have been detected, the number of characters specified by the LBL field are taken from the label at LA and written to the file as a physical record.

The PUTL macro can be used on an input type file with standard labels to have the system perform a label check. The LT field must be set to S and the PD field set to INPUT or to I-O with the LCR field set to E. At the first label exit taken during OPENM and CLOSEM processing, a PUTL macro can be issued. This causes the labels to be moved from the label area LA to the label buffer. The system then compares this label to the input file label. If they are unlike, the file cannot be opened and a fatal error occurs.

CLOSEL

The CLOSEL macro terminates label processing and returns control to OPENM or CLOSEM processing. CLOSEL must be called to terminate user label processing because it is the only way for the user to return control to BAM. Format of the CLOSEL macro is shown in figure 6-5.

CLOSEL fit

fit Address of the FIT or register containing the address.

Figure 6-5. CLOSEL Macro Format

The CLOSEL macro is used to exit a label processing routine and return to the calling routine for continued processing. Entry into the label processing routine is made at various times during OPENM and CLOSEM processing. Generally, on input type files (the PD field is set to INPUT or to I-O with the LCR field set to E), entry is made when the labels are made available for checking. On output type files, entry is made to the label processing routine to allow the user to submit labels to be written on the file.

On nonstandard end-of-volume and end-of-file labels, label processing must be performed by the user at the end-of-data exit (DX) address. This address is taken at the tapemark before the nonstandard label. In this case, when the CLOSEL macro is issued, it returns control in-line after the CLOSEL macro.

STANDARD CHARACTER SET A

CONTROL DATA operating systems offer the following variations of a basic character set:

CDC 64-character set

CDC 63-character set

ASCII 64-character set

ASCII 63-character set

The set in use at a particular installation was specified when the operating system was installed.

Depending on another installation option, the system assumes an input deck has been punched either in 026 or in 029 mode (regardless of the character set in use).

Under NOS/BE, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of the job statement or any 7/8/9 card. The specified mode remains in effect through the end of the job unless it is reset by specification of the alternate mode on a subseqeunt 7/8/9 card.

Under NOS, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of any 6/7/9 card, as described above for a 7/8/9 card. In addition, 026 mode can be specified by a card with 5/7/9 multipunched in column 1, and 029 mode can be specified by a card with 5/7/9 multipunched in column 1 and a 9 punched in column 2.

Graphic character representation appearing at a terminal or printer depends on the installation character set and the terminal type. Characters shown in the CDC Graphic column of the standard character set table are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII-CRT and ASCII-TTY terminals.

STANDARD CHARACTER SETS

Display Code (octal)	CDC			ASCII		
	Graphic	Hollerith Punch (026)	External BCD Code	Graphic Subset	Punch (029)	Code (octal)
00†	: (colon)††	8-2	00	: (colon)††	8-2	072
01	A	12-1	61	A	12-1	101
02	B	12-2	62	B	12-2	102
03	C	12-3	63	C	12-3	103
04	D	12-4	64	D	12-4	104
05	E	12-5	65	E	12-5	105
06	F	12-6	66	F	12-6	106
07	G	12-7	67	G	12-7	107
10	H	12-8	70	H	12-8	110
11	I	12-9	71	I	12-9	111
12	J	11-1	41	J	11-1	112
13	K	11-2	42	K	11-2	113
14	L	11-3	43	L	11-3	114
15	M	11-4	44	M	11-4	115
16	N	11-5	45	N	11-5	116
17	O	11-6	46	O	11-6	117
20	P	11-7	47	P	11-7	120
21	Q	11-8	50	Q	11-8	121
22	R	11-9	51	R	11-9	122
23	S	0-2	22	S	0-2	123
24	T	0-3	23	T	0-3	124
25	U	0-4	24	U	0-4	125
26	V	0-5	25	V	0-5	126
27	W	0-6	26	W	0-6	127
30	X	0-7	27	X	0-7	130
31	Y	0-8	30	Y	0-8	131
32	Z	0-9	31	Z	0-9	132
33	0	0	12	0	0	060
34	1	1	01	1	1	061
35	2	2	02	2	2	062
36	3	3	03	3	3	063
37	4	4	04	4	4	064
40	5	5	05	5	5	065
41	6	6	06	6	6	066
42	7	7	07	7	7	067
43	8	8	10	8	8	070
44	9	9	11	9	9	071
45	+	12	60	+	12-8-6	053
46	-	11	40	-	11	055
47	*	11-8-4	54	*	11-8-4	052
50	/	0-1	21	/	0-1	057
51	(	0-8-4	34	(	12-8-5	050
52	)	12-8-4	74	)	11-8-5	051
53	$	11-8-3	53	$	11-8-3	044
54	=	8-3	13	=	8-6	075
55	blank	no punch	20	blank	no punch	040
56	, (comma)	0-8-3	33	, (comma)	0-8-3	054
57	. (period)	12-8-3	73	. (period)	12-8-3	056
60	≡	0-8-6	36	#	8-3	043
61	[	8-7	17	[	12-8-2	133
62	]	0-8-2	32	]	11-8-2	135
63	%††	8-6	16	%††	0-8-4	045
64	≠	8-4	14	" (quote)	8-7	042
65	←	0-8-5	35	_ (underline)	0-8-5	137
66	∨	11-0 or 11-8-2†††	52	!	12-8-7 or 11-0†††	041
67	∧	0-8-7	37	&	12	046
70	↑	11-8-5	55	' (apostrophe)	8-5	047
71	↓	11-8-6	56	?	0-8-7	077
72	<	12-0 or 12-8-2†††	72	<	12-8-4 or 12-0†††	074
73	>	11-8-7	57	>	0-8-6	076
74	≤	8-5	15	@	8-4	100
75	≥	12-8-5	75	\	0-8-2	134
76	¬	12-8-6	76	~ (circumflex)	11-8-7	136
77	; (semicolon)	12-8-7	77	; (semicolon)	11-8-6	073

†Twelve zero bits at the end of a 60-bit word in a zero byte record are an end of record mark rather than two colons.

††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank (55_8).

†††The alternate Hollerith (026) and ASCII (029) punches are accepted for input only.

CDC CHARACTER SET COLLATING SEQUENCE									
Collating Sequence Decimal/Octal		CDC Graphic	Display Code	External BCD	Collating Sequence Decimal/Octal		CDC Graphic	Display Code	External BCD
00	00	blank	55	20	32	40	H	10	70
01	01	≤	74	15	33	41	I	11	71
02	02	%	63 †	16 †	34	42	∨	66	52
03	03	[	61	17	35	43	J	12	41
04	04	→	65	35	36	44	K	13	42
05	05	≡	60	36	37	45	L	14	43
06	06	∧	67	37	38	46	M	15	44
07	07	↑	70	55	39	47	N	16	45
08	10	↓	71	56	40	50	O	17	46
09	11	>	73	57	41	51	P	20	47
10	12	≥	75	75	42	52	Q	21	50
11	13	¬	76	76	43	53	R	22	51
12	14	.	57	73	44	54	]	62	32
13	15	)	52	74	45	55	S	23	22
14	16	;	77	77	46	56	T	24	23
15	17	+	45	60	47	57	U	25	24
16	20	$	53	53	48	60	V	26	25
17	21	*	47	54	49	61	W	27	26
18	22	-	46	40	50	62	X	30	27
19	23	/	50	21	51	63	Y	31	30
20	24	,	56	33	52	64	Z	32	31
21	25	(	51	34	53	65	:	00 †	none†
22	26	=	54	13	54	66	0	33	12
23	27	≠	64	14	55	67	1	34	01
24	30	<	72	72	56	70	2	35	02
25	31	A	01	61	57	71	3	36	03
26	32	B	02	62	58	72	4	37	04
27	33	C	03	63	59	73	5	40	05
28	34	D	04	64	60	74	6	41	06
29	35	E	05	65	61	75	7	42	07
30	36	F	06	66	62	76	8	43	10
31	37	G	07	67	63	77	9	44	11

†In installations using the 63-graphic set, the % graphic does not exist. The : graphic is display code 63, External BCD code 16.

ASCII CHARACTER SET COLLATING SEQUENCE

Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code	Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code
00	00	blank	55	20	32	40	@	74	40
01	01	!	66	21	33	41	A	01	41
02	02	"	64	22	34	42	B	02	42
03	03	#	60	23	35	43	C	03	43
04	04	$	53	24	36	44	D	04	44
05	05	%	63†	25	37	45	E	05	45
06	06	&	67	26	38	46	F	06	46
07	07	'	70	27	39	47	G	07	47
08	10	(	51	28	40	50	H	10	48
09	11	)	52	29	41	51	I	11	49
10	12	*	47	2A	42	52	J	12	4A
11	13	+	45	2B	43	53	K	13	4B
12	14	,	56	2C	44	54	L	14	4C
13	15	-	46	2D	45	55	M	15	4D
14	16	.	57	2E	46	56	N	16	4E
15	17	/	50	2F	47	57	O	17	4F
16	20	0	33	30	48	60	P	20	50
17	21	1	34	31	49	61	Q	21	51
18	22	2	35	32	50	62	R	22	52
19	23	3	36	33	51	63	S	23	53
20	24	4	37	34	52	64	T	24	54
21	25	5	40	35	53	65	U	25	55
22	26	6	41	36	54	66	V	26	56
23	27	7	42	37	55	67	W	27	57
24	30	8	43	38	56	70	X	30	58
25	31	9	44	39	57	71	Y	31	59
26	32	:	00†	3A	58	72	Z	32	5A
27	33	;	77	3B	59	73	[	61	5B
28	34	<	72	3C	60	74	\	75	5C
29	35	=	54	3D	61	75	]	62	5D
30	36	>	73	3E	62	76	^	76	5E
31	37	?	71	3F	63	77	_	65	5F

†In installations using a 63-graphic set, the % graphic does not exist. The : graphic is display code 63.

All user requests are checked to ensure proper processing. If results are not satisfactory, an error condition exists and the following occurs:

A three-digit octal error code is returned in the error status (ES) field of the FIT.

For a parity error, a severity level is set in the system parity error severity (SES) field.

For a fatal error, the fatal/nonfatal flag (FNF) field is set in the FIT.

Action indicated by the user setting of the error option (EO) field takes place, as discussed elsewhere in this section.

An error exit is taken if the user has set the error exit (EX) field of the FIT.

Error messages and notes are written to the dayfile and/or the ZZZZZEG error file depending on the values of the dayfile control (DFC) and error file control (EFC) fields.

ERROR COMMUNICATION

Regarding errors, the user and the error processor communicate through FIT fields ES, EX, EO, ERL, ECT, and PEF. The error status (ES) field is a 9-bit field set to an octal value after an attempt at error resolution is made and control is ready to be returned to the user. When an attempt is made to execute an input or output request after an error, the ES field is not cleared. If the request is not legal, the trivial error count (ECT) is incremented, and execution proceeds. If a subsequent error is detected, the ES field reflects the most recent error. The user is responsible for clearing the ES field if an error exit (EX) is not supplied, but instead the ES field is checked after every macro call.

FIT fields and their meaning relevant to error processing are:

FNF Fatal/nonfatal flag; set to 1 for fatal errors.

PEF Parity error flag; set to 1 for parity errors.

SES System parity error severity; set to the severity level of the parity error. The levels have meaning as shown in table B-1.

EX Error exit; an 18-bit field, interpreted as follows:

EX=0 No user error routine; control is returned as a normal exit; the ES field is set with an error code. If the value of EX is zero and a fatal (F) error is encountered, the message is put on the dayfile.

TABLE B-1. TYPES OF PARITY ERRORS

Value	Severity	Explanation
1	Read parity error level 1	Recovery to record boundary is possible. The number of bad records and blocks is known. BAM can recover.
2	Read parity error level 2	Recovery to record boundary is possible. The number of bad blocks is known but not the number of lost records. BAM can recover.
3	Read parity error level 3	Recovery to record boundary is possible. The number of bad records and blocks is unknown. BAM can recover.
4	Read parity error level 4	Recovery to record boundary is not possible. Fatal, BAM cannot recover.
5	Write parity error level 1	Irrecoverable tape write parity error. CLOSEM VOLUME recommended.
6	Write parity error level 2	Irrecoverable tape write parity error. CLOSEM VOLUME cannot be executed.

EX≠0 If a fatal or trivial error occurs, control is transferred to EX+1; a jump to the user in-line return address is stored in the EX field, and the ES field is set.

ERL Trivial error limit which can be specified by the user.

ERL=0 Limit not specified; no error count is accumulated. The number of trivial errors permitted is indefinite.

ERL≠0 The job is terminated when the value of the ECT field reaches the value of the ERL field.

EO Error option; the EO field is used in conjunction with parity errors. If the TD, AD, or DD option is used and the EFC field is set to 3, the block containing the parity error is dumped to the error file for display by the error processor. The EO field is interpreted as follows:

EO=T/TD All parity errors are fatal.

EO=A/AD — All parity errors should be disregarded (the bad data read as if it were good), but the ES field is set to 137 and control is passed to the error exit (EX) routine at the end of the record. If another error occurs when trying to read bad data, error 137 is overwritten by the next error; however, the parity error flag (PEF) remains set.

EO=D/DD — The block in which the parity error occurs is dropped and BAM attempts to find the start of the next good record. If successful, the error exit is taken with the ES field set to 137, the SES field set to 3, and the FNF field set to zero. The content of the working storage area is undefined, and the file is positioned in front of the next good record. If unsuccessful, the error exit is taken with the ES field set to 137, the SES field set to 3, and the FNF field set to 1 (fatal).

DFC — Dayfile control. This field is set by the user to control the listing of error messages on the dayfile.

- DFC=0 — No dayfile messages except fatal errors (default).
- DFC=1 — Error messages to the dayfile.
- DFC=2 — Notes to the dayfile.
- DFC=3 — Error messages and notes to the dayfile.

EFC — Error file control. This field is set by the user to control the listing of error messages on the error file.

- EFC=0 — No error file entries (default).
- EFC=1 — Error messages to the error file.
- EFC=2 — Notes to the error file.
- EFC=3 — Error messages and notes to the error file.

The system message disposition (SDS) and extended diagnostic (EXD) fields of the FIT, which were part of a previous version of the error processor, are replaced by the DFC and EFC fields. If the SDS or EXD fields are used with the FILE macro, a warning assembly diagnostic is issued and no comparable values are placed in the DFC and EFC fields. If they are used with the FETCH or STORE macro, they are translated into compatible values for the DFC and EFC fields. The SDS field set to YES is equivalent to the DFC field set to 2. The EXD field set to YES is equivalent to the EFC field set to 1.

ERROR PROCESSING

If the EFC field is set to nonzero, the CRMEP control statement can be used to process the ZZZZZEG error file and control the listing of error messages on the output file. The error file is always flushed when abnormal termination occurs. At the completion of a job step, the error file is flushed if all files are closed. The format of the CRMEP control statement is shown in figure B-1. The parameters, options, and defaults for the CRMEP control statement are listed in table B-2. The first default is set if neither the parameter nor the option is specified. The second default is set if the parameter is specified without an option. More than one option can be specified with each parameter, and more than one parameter can be specified on one CRMEP control statement.

CRMEP (parameter=option$_1$/option$_2$/ ... /option$_n$, ...)

parameter	Mnemonic specifying type of error file processing and listing.
option	Selected setting of the specified parameter.

Figure B-1. CRMEP Control Statement Format

The FITDMP macro can be used to capture the contents of FIT fields for display by the post error processor. When the FITDMP macro is executed, the FIT, and the FIT display identifier if the id parameter is specified, are written to the ZZZZZEG error file. The CRMEP control statement can then be used to display the FIT on the output file. The format of the macro is shown in figure B-2. The FIT display identifier, which can be up to ten characters, identifies the particular fit dump. The id parameter specifies the location of the display identifier.

FITDMP fit,id

fit	Address of the FIT.
id	Address of the FIT display identifier.

Figure B-2. FITDMP Macro Format

To ensure that notes are written to the error file, the EFC field of the FIT must be set to 2 or 3. Note number 1000 is reserved for user FIT dumps.

Upon encountering an error condition, the error status (ES) field is set to the appropriate error number, the trivial error count (ECT) field is incremented, and it is compared with the trivial error limit (ERL) field. If the ERL field is zero, unlimited errors are allowed and the ECT field is not incremented in the FIT. If the value of the ERL field is nonzero and the ECT field is less than the ERL field, control passes to the error exit (EX) routine if defined, or back to the user's in-line code if the EX field is zero. In the latter case, it is the user's responsibility to check the error status. If the ERL field is nonzero and the value of the ECT field is equal to the value of the ERL field, the ES field is set to 356 (trivial error limit reached). The fatal/nonfatal (FNF) flag is set and another message is written. Control is returned as described above. If the FNF flag is set and any other function is attempted on the file, a 115 error is generated and the job is aborted.

CLASSES OF ERRORS

Syntax errors are diagnosed. The messages are self-explanatory. System errors are detected by the operating system. Execution errors, occurring during execution of input and output requests, are subdivided into call errors and invalid input/output requests.

TABLE B-2. CRMEP CONTROL STATEMENT PARAMETERS

Parameter	Option	First Default	Second Default	Description
LO	N		X	Select notes.
	-N	X		Omit notes.
	F	X	X	Select fatal error messages.
	-F			Omit fatal error messages.
	D	X	X	Select data manager messages.
	-D			Omit data manager messages.
	T		X	Select trivial error messages.
	-T	X		Omit trivial error messages.
SF	$lfn_1/lfn_2/ \ldots /lfn_n$	All	All	Select messages associated with specified files.
OF	$lfn_1/lfn_2/ \ldots /lfn_n$	None	None	Omit messages associated with specified files.
SN	$mno_1/mno_2/ \ldots /mno_n$	All	Hardware and parity errors	Select only specified message numbers.
ON	$mno_1/mno_2/ \ldots /mno_n$	None	Error messages 142 and 143 only	Omit only specified message numbers.
L	lfn	OUTPUT	LIST	Specify output file name.
RU	blank			Return unload of error file performed at end of processing. Error file position at EOI at end of processing.
	0	X		

CALL ERRORS

Call errors are undetectable parameter errors, such as:

```
GET     X1
```

If register X1 does not contain the valid FIT address, an unpredictable BAM error or mode error can result.

INVALID INPUT/OUTPUT REQUESTS

Requests for illegal input/output operations produce the following general types of errors:

FIT	Content of address given as the FIT address does not pass a test for plausibility. It does not contain a legal logical file name in bits 59 through 18, or the FIT has inconsistencies.
File organization	Attempts to issue input/output requests or specifications are illegal on the type of file specified in the FO field of the FIT.
Block type	Attempts to issue input/output requests are illegal for the block type specified in the BT field of the FIT.
Record type	Attempts to issue input/output requests are illegal for the record type specified in the RT field of the FIT.
OPENM/ CLOSEM	Input/output requests are illegal for files opened or closed as specified in the OC and/or ON fields of the FIT.
Processing direction	Input/output requests that would violate the processing direction limitations specified in the PD field of the FIT.

File position — Input/output requests are illegal for the file position given by the FP field of the FIT.

Last operation — Input/output requests are illegal in the context of the last operation; for example, a read after a write on tapes.

Key — Attempts to access or write records whose keys are not within the range of keys defined for a file. This includes attempts to access sequential files by keys.

Data — Errors in data specification, such as inconsistency between the amount of data requested and the amount actually present, illegal field present in the data, required field absent, or parity errors.

Device — Attempts to execute an input/output request are illegal on the device upon which the file resides.

Label — Label information submitted by the user does not correspond with the existing label, or the label is incorrectly formatted.

All errors are fatal or nonfatal. Some nonfatal errors are trivial in that no user action is required. Fatal errors usually indicate incorrect parameter specification and incomplete or contradictory information which is a user program error. A fatal diagnostic is always printed on the dayfile.

If an EX field has been specified in the FIT, any error causes a transfer of control to the address in EX+1 for a recovery routine after the error has been resolved. Fatal errors inhibit any further attempts at input/output on the file. Such attempts cause the job to terminate. In the absence of a value in the EX field, errors set the ES field and return control to the calling program. The ES field is not cleared after an error.

BAM is in the user's field length and is subject to destruction by the user.

DIAGNOSTICS

Table B-3 is a list of notes or informative messages.

TABLE B-3. NOTES OR INFORMATIVE MESSAGES

Code	Message
1000	USER FIT DUMP .....
1137	THE FOLLOWING BLOCK CONTAINS A PARITY ERROR .....

Table B-4 contains the following:

Code — Octal value corresponding to the error condition.

Message — Diagnostic output which varies depending on the values of the DFC and EFC fields, and the parameters specified with the CRMEP control statement.

Significance — Meaning of the message.

Action — Suggestion for the user to correct the error condition.

Severity — Type of error; can be any of the following:

F Fatal

T Trival

T/F Trival under some conditions, fatal under others

TABLE B-4. DIAGNOSTICS

Code	Message	Significance	Action	Severity
001	INVALID FO	File organization must be sequential (SQ) or word addressable (WA).	Correct the file organization field.	F
002	FIT/FILE ORGANIZATION MISMATCH	The file organization specified does not match any opened files.	Check to see that the correct file is being processed, or that the FO field is specified correctly.	F
020	INVALID BT	Block type must be I, C, K, or E.	Correct the block type field.	T
022	W RECORDS DISALLOWED ON BT=E/K	W type records cannot be written for E or K type blocks.	Correct the record type or block type field.	T
025	BT=I, RT NE W	I type blocks require W type records.	Correct the block or record type field.	T
026	SQ BTS REQUIRE MBL	Maximum block length must be specified for SQ files with K or E type blocks.	Specify the maximum block length field.	T

TABLE B-4. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
030	INVALID RT	Record type must be W, S, Z, F, R, T, D, or U; it must conform to other file specifications, such as block type or file organization.	Correct the record type field.	T
031	RT=F/Z AND FL=0	For fixed length F or zero-byte terminated Z type records, a maximum record length must be specified in the FL field of the FIT.	Specify the maximum record length field.	T
032	RT=T AND HL OR TL=0	For T type records, the header length (HL) must be large enough to hold the CL that defines the length of the trailer count field. The length of the trailer count field must be given in TL and must be at least one character long.	Correct the header length or the trailer length field.	T
033	RT=D AND LL=0/RT=T AND CL=0	For D type records, the LL field of the FIT must provide the length of the record field that specifies record length.	Specify the length of the D type record length field.	T
		For T type records, the CL field of the FIT must provide the length of the field that specifies the number of trailer items.	Specify the length of the trailer count field of the T type record.	
035	RT=T/D, MRL EXCLUDES CONTROL FIELD	For T and D type records, the record must contain a field identifying record length.	Check that for D type records LP+LL is less than MRL. For T type records, CP+CL must be less than MRL. The position count for LP and CP begins with 0.	T
036	RL INCONSISTENT WITH RECORD DESCRIPTION	For T type records, the fixed header length (HL) must include a field CL characters long, beginning at CP, to identify trailer item count.	Check that the count field is included in HL. The current record is ignored. Position CP is counted from 0.	T
037	RT=D/T AND CL/LL>6	For D and T type records, the length of the count field must be one to six character positions.	Correct the length of the count field.	T
040	REDUNDANT OPEN	A file must be closed before open processing, such as user label processing for sequential files or buffer allocation, takes place. A redundant open call is ignored.	Correct the program to close the file before open processing.	T
047	OPEN EXTEND ON TAPE FILE	The E option for OPENM is valid only for a sequential file on mass storage.	Change the E option for the OPENM macro.	T
051	SETFIT DISALLOWED ON OPEN FILE	Open processing would have already processed the FILE control statement. The SETFIT function processes FILE control statements without full open processing.	Change the placement of the SETFIT macro.	T
060	REDUNDANT CLOSE	A second call to close the file was issued. The operations requested by the CF field are performed before the error is issued.	Correct the program to eliminate the redundant close operation.	T
070	OUTPUT REQUEST, PD=INPUT	A file opened with pd set to INPUT cannot be written. The write statement is ignored.	If the file is to be written, store OUTPUT or IO in the PD field of the FIT prior to opening the file.	T

TABLE B-4. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
071	INPUT REQUEST, PD=OUTPUT	A file opened with pd set to OUTPUT cannot be read. The read statement is ignored.	If the file is to be read, store INPUT or IO in the PD field of the FIT before opening the file.	T
104	UNABLE TO FLUSH BUFFER	A parity or system error might exist in an output sequential file just prior to a close request that requires the buffer to be flushed.	Rerun the program.	T
110	FILE NOT OPEN	A file must be opened before it can be read or written. Omission of required FIT field parameters or inconsistencies in parameters specified inhibit open.	Correct the program to open the file before reading or writing; or, correct omissions or inconsistencies of FIT fields.	T
111	NO CHECK ON LAST REQUEST	The CHECK or CHECKR macro must be issued after each GETWR or PUTWR macro.	Correct the program to issue the CHECK or CHECKR macro.	T
113	GET/PUT CANNOT BE USED IF SBF=YES	If file organization is sequential, only the GETWR or PUTWR macros can be used if the SBF field is set to YES.	Correct the program to use the GETWR or PUTWR macro or set the SBF field to NO.	T
115	OUTSTANDING FATAL ERROR ON THE FILE	A fatal error prevents future access to the file with the error, but it does not cause job termination unless the user attempts further operations on the file.	Correct and rerun.	F
116	GET FOLLOWS AN OUTPUT OPERATION. FO=SQ	A sequential file cannot be read immediately after a write.	Continue writing, or reposition the file before a read. The current read statement is ignored.	T
120	INVALID KEY/WORD ADDRESS/RECORD NUMBER	Word address for a word addressable file must be less than EOI for GET macros.	Correct the word address field.	T
130	RT=W BAD CONTROL WORD, FILE DEFECTIVE OR MISPOSITIONED	Record type was specified as W. This message indicates the records being read are not, in fact, W type records.	Check that the existing file is correctly described.	T/F
135	RMS READ PARITY ERROR	The system returned parity error status after reading a word addressable file.	Recreate the file on a good device. If the error persists, report it to a systems analyst.	T/F
136	RMS WRITE PARITY ERROR	The system returned parity error status after writing a word addressable file.	Recreate the file on a good device. If the error persists, report it to a systems analyst.	F
137	SQ READ PARITY ERROR	A parity error occurred while reading a sequential file.	Check the SES field of the FIT for severity and retry. (See the beginning of this appendix.)	T/F
140	SQ WRITE PARITY ERROR	A parity error occurred while writing a sequential file.	Check the SES field of the FIT for severity and retry. (See the beginning of this appendix.)	F
141	EXCESS DATA IS FATAL TO PUTP	The value of the RL field is greater than the value of the MRL field during a series of PUTP macros. The error is fatal because part of the bad record is already in the file.	Correct the program.	F

TABLE B-4. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
142	EXCESS DATA	In a write, no information is written to the file. For a read, MRL characters are transferred to the working storage area and remaining record characters skipped. On a read, the record length exceeds FL/MRL defined. For GET processing, the following conditions cause an error. Record types: W RL in control word >MRL Z No zero byte found before FL characters R No record mark found before MRL T,D Control field RL>MRL S MRL reached before level number encountered U RL >MRL F Excess data cannot occur On PUT processing, the record mark character for an R type record was not found before MRL characters, or the user has supplied RL >MRL/FL.	Correct the inconsistency between the RL and FL or MRL fields.	T
143	INSUFFICIENT DATA	Control information in the record being read (record length in a W type control word, or length calculated by fields such as CP and CL) specifies a length for each record. The record existing in the file is smaller than the specified length. All characters available are returned. For I and C type blocks, an end-of-section was encountered before the record terminated. For K and E type blocks, the block end occurred before the record ended. The data transferred through PUTPs is less than FL for an F type record.	No action is required.	T
144	INCOMPLETE PARTIAL PUT SEQUENCE	The previous record was not complete.	Correct the program.	F
150	FILE NOT ON RMS	Word addressable files must be created on a disk, drum, or family pack.	Correct the control statement to ensure a valid device assignment.	T/F
152	LT=S, DT=RMS	Standard labels, which conform to ANSI standards, can exist only on tape files. Label processing statements are ignored because the file is assigned to rotating mass storage.	Correct the inconsistency between label type and device type fields.	T
154	BT=K/E ON PRU TYPE DEVICE	K or E type blocking is possible only for files on S or L tapes.	Change block type, or add an S or L parameter to the REQUEST control statement.	T
157	S-TAPE BUT MBL >5120 CHARACTERS	Maximum block length for S tapes is 5120 characters.	Change MBL to an allowable value or use an L tape.	T

TABLE B-4. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
162	INVALID CONVERSION	The CM field of the FIT must not be YES for W type records.	Change the conversion mode field.	T
165	ILLEGAL FILE NAME	The LFN does not consist of one to seven letters and digits the first being a letter.	Correct the LFN or the FIT address.	F
167	RECORD LENGTH OUTSIDE MIN-MAX RANGE – REQUEST IGNORED	For D or T type records, the control field specified is outside the value specified by the RL field, or not within the values specified by the MNR and MRL fields.	Check to see that the CL/CP fields or the LL/LP fields are specified correctly.	T
170	RECORD SIZE EXCEEDS BLOCK SIZE OR IS NEGATIVE	For K and E type blocking, records cannot be split between blocks. Individual records must be smaller than the block defined by MBL or the maximum block allowed on the device.	Correct the RL or MBL field.	T/F
173	INVALID RL/PTL/MBL	The record length, partial transfer length, or block size is specified incorrectly.	Correct the RL, PTL, or MBL field.	T
207	MINIMUM RECORD SIZE EXCEEDS MAXIMUM	Required parameter MRL must be equal to or larger than MNR.	Correct the inconsistency between the MRL and MNR fields.	F
245	FUNCTION DISALLOWED ON THIS FO	The macro issued is not valid for the file organization specified in the FIT.	Correct the program.	T
254	PARTIALS NOT SUPPORTED FOR FO=WA	The GETP and PUTP macros cannot be issued for a word addressable file.	Correct the program to use the GET or PUT macro.	T
255	RECORD SPECIFICATION NOT COMPATIBLE WITH SBF=YES	For word addressable files with the SBF field set to YES, the RL field must be a multiple of PRU size and the WA field must be a multiple of PRU size plus one.	Correct the program to specify correct values for the RL and WA fields, or set the SBF field to NO to allow a buffer to be allocated.	T
300	NO READ PERMISSION	To be read, a permanent file must be attached with RD permission.	Attach the file with the required read permission.	F
301	NO WRITE OR MODIFY PERMISSION	A permanent file requires proper access permissions. MD permission is required for any updating operation.	Attach the file with the required write permission.	F
302	NO EXTEND OR ALLOCATE PERMISSION	A permanent file requires extend (EX) permission before new records can be inserted.	Attach the file with the required extend permission.	F
312	INVALID LABEL GROUP	Labels that can be accessed are affected by the current file position. Header labels, for example, cannot be accessed at end-of-information.	Check that file position is consistent with label action requested.	F
315	FILE ORGANIZATION IS NOT SEQUENTIAL	Standard labels can be used only with sequential files on tape.	Check that file organization is consistent with label type.	F
316	TOO MANY LABELS	The number of labels that can be written is limited by ANSI standards.	Correct the program.	F
320	INVALID LABEL SEQUENCE	The ULP option controls the type of labels that can be accessed.	Remove conflicts between ULP and the type of label.	F
325	STANDARD LABELS NOT ALLOWED ON MASS STORAGE	LT=S is valid only for tape files.	Correct the inconsistency between label type and device.	F

TABLE B-4. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
326	GETL/PUTL ILLEGAL ON UNLABELED FILE	A tape file must have a label declared on a REQUEST or LABEL control statement before user label access is possible.	Change the label type field.	T
327	GETL ATTEMPTED BEYOND END OF LABELS	Tapemarks separating data and labels stop label processing.	Correct the program.	F
330	INVALID PARAMETER VALUE (LA, LBL, ULP)	LA must be zero or an address in a user program. LBL must indicate the length of the label area, 0 to 900 characters. ULP options are V, F, U, VU, VF, FU, VFU, and NO.	Check GETL or PUTL parameters of FIT fields.	F
332	FILE REQUEST LABEL TYPE DISAGREES WITH LT FIELD OF FIT	When a REQUEST control statement specifies a labeled tape, the user must set LT to S.	Correct the inconsistency between the REQUEST control statement and the label type field.	T
345	INSUFFICIENT CMM SPACE AVAILABLE	Not enough CMM space exists to open the file. To open a file requires enough free CMM space to load any rare capsules required, if any, and to allow two of the largest blocks to be in memory at the same time. The file is not opened.	Release some CMM, if any is being used by the user program, or increase the amount of memory available to the job.	T
346	CMM NOT AVAILABLE AND THERE IS NO LIST OF FILES ADDRESS	A new block for the list-of-files cannot be allocated, and the LOF$RM entry point has been cleared.	Correct the program to not destroy the pointer. A default list with sixty-five entries is supplied.	F
347	FDL ERROR	Either CMM is not loaded when FDL is called to load a capsule, or the BAMLIB file is not valid.	Check the load sequence or map to see if CMM is loaded. Fix the static load calls to load the proper routines. If using local libraries, check for a valid BAMLIB file.	T
352	FILE TO BE CLOSED IS NOT KNOWN	The logical file name specified does not match any existing file.	Check that the logical file name is correctly specified.	T
354	BUFFER SPACE SUPPLIED IS INSUFFICIENT FOR I/O	A buffer specified by BFS must be large enough to hold at least the larger of one block specified by MBL+2 or one physical record unit for the file's resident device.	Increase the BFS value.	T
355	CODE MODULES REQUIRED FOR I/O NOT LOADED	Routines necessary for processing have not been loaded.	Refer to appendix E for correct loading procedures.	T
356	TRIVIAL ERROR LIMIT REACHED	Error count ECT equals the user-defined error limit ERL, resulting in a fatal error.	Correct the errors.	F
357	UNABLE TO OBTAIN SPACE FOR BUFFER	Required space has not been allocated. CMM is not available, and the FWB field is zero.	Supply a value for the FWB field or remove the OMIT=CMM parameter.	F
370	FATAL I/O ERROR	Either a block with an incorrect length was encountered or the operating system detected an error in the file or in the way the file was being used.	Correct the program.	F
403	SKIPBL DISALLOWED	A backward skip is not possible for D, U, R, and T type records or K and E type blocks.	Correct the program.	T

TABLE B-4. DIAGNOSTICS (Cont'd)

Code	Message	Significance	Action	Severity
404	SKIPFL DISALLOWED FOR RT=U	No forward record skip is possible for U type records.	Correct the program.	T
406	REPLACE ATTEMPTED ON TAPE FILE	For sequential files, the REPLACE macro can be used only on disk files.	Copy the file to disk.	T
407	FO=SQ REPLACE ATTEMPTED WHEN FP≠EOR	The REPLACE macro must be preceded by a GET macro or a GETP macro of a full record.	Correct the program to read a full record before the REPLACE macro is issued.	T
410	FO=SQ REPLACE ATTEMPTED WHEN LOP≠GET	For sequential files, the record to be replaced must be read before the REPLACE macro is issued.	Correct the program.	T
411	FO=SQ REPLACE ENCOUNTERED EOS/EOP/BOI	The GET or REPLACE macro did not work properly.	Notify a system analyst.	T
412	FO=SQ REPLACE ILLEGAL FOR THIS RT – USE RT=F/W	For sequential files, the REPLACE macro can only be used with W or F type records.	Correct the program.	T
413	FO=SQ REPLACE ILLEGAL FOR THIS BT – USE BT=C	For sequential files, the REPLACE macro can only be used with C type blocks.	Correct the program.	T
452	FILE POSITIONING ERROR	An attempt was made to position the file beyond EOI.	Correct the program to check the FP field or specify the DX field.	F
712	NEGATIVE OR OVERSIZED ARGUMENT--WSA, SKP, OR LA	One of the parameters indicated was erroneously specified when a macro was issued.	Correct the program.	F
713	NEGATIVE OR OVERSIZED ARGUMENT--RL, ST, OR LBL	One of the parameters indicated was erroneously specified when a macro was issued.	Correct the program.	F
714	NEGATIVE EX OR DX PARAMETER	A negative value was specified for the DX or EX field.	Correct the program.	F
715	NEGATIVE OR OVERSIZED ARGUMENT--WA OR KA	Either the WA or KA field was erroneously specified.	Correct the program.	F
716	NEGATIVE OR OVERSIZED ARGUMENT--PTL OR KP	Either the PTL or KP field was erroneously specified.	Correct the program.	F
717	NEGATIVE OR OVERSIZED ARGUMENT--MKL, POS, GPS, OR TRM	One of the parameters indicated was erroneously specified when a macro was issued.	Correct the program.	F
720	DEVICE CAPACITY EXCEEDED	The CIO read driver has encountered an error.	Check the system dayfile for the specific read driver error.	T
721	ERROR DETECTED BY OPERATING SYSTEM	A system hardware error has been encountered that cannot be handled.	Check the system dayfile for a system/hardware error message.	T

ADVANCED ACCESS METHODS (AAM) – A file manager that processes indexed sequential, direct access, and actual key file organizations and supports the Multiple-Index Processor.

BASIC ACCESS METHODS (BAM) – A file manager that processes sequential and word addressable file organizations.

BEGINNING-OF-INFORMATION (BOI) – The start of the first user record in a file. System information, for example tape labels of sequential files, can appear before the beginning-of-information.

BLOCK – A logical or physical grouping of records to make more efficient use of hardware. Only sequential files are blocked. One of the following block types must be specified by the programmer: C, I, K, or E.

CHARACTER – A letter, digit, punctuation mark, or mathematical symbol forming part of one or more of the standard character sets. Also, a unit of measure used to specify block length, record length, and so forth.

CLOSE – A set of terminating operations performed on a file when input and output operations are complete. All files processed by BAM must be closed.

CYBER RECORD MANAGER (CRM) – A generic term relating to the common products BAM and AAM.

DEFAULT – A value assumed in the absence of a user-specified value declaration for the parameter involved. Values for many defaults are defined by the installation.

END-OF-INFORMATION (EOI) – The end of the last user record in a file. Trailer labels are considered to be past the end-of-information. End-of-information is undefined for unlabeled S or L tapes.

FIELD – A portion of a word or record; a subdivision of information within a record; also, a generic entry in a file information table identified by a mnemonic.

FIELD LENGTH – The area in central memory allocated to a particular job; the only part of central memory that a job can directly access. Contrasts with mass storage space or tapes allocated for a job and on which user's files reside.

FILE – A logically related set of information; the largest collection of information that can be addressed by a file name. It starts at beginning-of-information and ends at end-of-information. Every file in use by a job must have a logical file name.

FILE CONTROL STATEMENT – A control statement that supplies file information table values after a source language program is compiled or assembled but before the program is executed. Basic file characteristics such as organization, record type, and description can be specified in the FILE control statement.

FILE INFORMATION TABLE (FIT) – A table through which a user program communicates with BAM. For direct processing through BAM, a user must initiate establishment of this table. All file processing executes on the basis of information in this table. The user can set FIT fields directly or use parameters in a file access call that sets the fields indirectly. Some product set members set the fields automatically for the user.

INSTALLATION OPTION – One of several alternate means of processing that is selected when BAM is installed at a computer installation. Once an option is selected, all subsequent use of BAM is governed by the selection. For all options or limits defined as installation options, the user should consult with a system analyst to determine the valid limits.

I TAPE – A magnetic tape with recording format of physical records containing the contents of 0 to 512 central memory words of binary information. I tapes are only supported under the NOS operating system.

JANUS – A group of routines in the NOS/BE operating system that controls the unit record equipment including card readers, line printers, and card punches. Files with names of INPUT, OUTPUT, PUNCH, and PUNCHB are JANUS files.

KEY – Information used to identify a record.

LDSET – The loader control statement. Various parameters include:

LIB	Make available the named library
USE	Load the routines named
STAT	Static loading requested
OMIT	Inhibit loading of routines named

LOAD SET – A group of control statements beginning with a call that causes information to be loaded into central memory and ending with a call for execution of a loaded program. Nonloader statements must not appear in a load set.

LOGICAL FILE NAME – The name given to a file being used by a job. The name must be unique for the job and must consist of one to seven letters or digits, the first of which must be a letter.

L TAPE (LONG STRANGER TAPE) – A 7-track or 9-track, labeled or unlabeled magnetic tape with blocks containing more than 5120 characters. Normally written by other than CYBER 170-compatible systems.

MACRO – A single instruction which when compiled into machine code generates several machine code instructions.

MAINTENANCE RUN – A program or job to update an existing file; technically refers to that part of the job from file open to file close.

MASS STORAGE – A disk pack that can be accessed randomly. ECS is not considered mass storage.

MASTER FILE – A file containing information about a set of entities; all information about a single entity constitutes a record in a file. A master file is normally kept up to date by a maintenance run.

OPEN – A set of preparatory operations performed on a file before input and output can take place; required for all BAM files.

OWNCODE – A routine written by the user to process certain conditions. Control passes automatically to user owncode routines defined in the FIT for:

DX	End-of-data condition
EX	Error condition
LX	Tape label processing

PARTITION – A group of sections beginning with the first record after the end of the preceding partition and ending with a special record or condition, dependent on the block and record type and storage device. Generally, a partition is greater than a section and less than a file, but it can be equal to either or both.

PERMANENT FILE – A file on a mass storage permanent file device that can be retained for longer than a single job. It is protected against accidental destruction by the system and can be protected against unauthorized access.

PHYSICAL RECORD – On magnetic tape, information between interrecord gaps. It need not contain a fixed amount of data.

PHYSICAL RECORD UNIT (PRU) – The smallest unit of information that can be transferred between a peripheral storage device and central memory. The PRU size is permanently fixed for all mass storage devices and SI, X, and I tapes; the concept does not apply to S/L tapes.

PRU DEVICE – An SI or I format tape or a mass storage device in which information has a physical structure governed by physical record units (PRUs).

RANDOM ACCESS – Access method by which any record in a file can be accessed at any time. Applies only to mass storage files with an organization other than sequential.

RECORD – The largest collection of information passed between BAM and a user program in a single read or write operation. The user defines the structure and characteristics of records within a file by declaring a record format. The beginning and ending points of a record are implicit in each format.

RELEASE SYSTEM – A software system delivered to a customer is the release system. In installing a system, the customer, but not an individual applications programmer, can use default values or parameters that differ from the released system.

REWIND – To position a file at beginning-of-information.

SCOPE 2 – An operating system on the CONTROL DATA CYBER 70 Model 76 and 7600 Computer Systems. 7000 Record Manager runs under SCOPE 2.

SECTION – A division internal to a sequential file. Recognition of a section boundary is affected by block type, record type, and file residence. A section is a group of records beginning with the first record after the end of the preceding section and ending with a special record or condition, dependent on the block and record type and storage device. Generally, a section is greater than a record and less than a partition, but it can be equal to either or both. Sections are not defined on K and E type blocks.

SEQUENTIAL ACCESS – A method in which only the record located at the current file position can be accessed. See Random Access.

SEQUENTIAL (SQ) FILE – A file with records in the physical order in which they were written. No logical order exists other than the relative physical record position.

S TAPE (STRANGER TAPE) – A magnetic tape with recording format of physical records containing the contents of 512 central memory words of information.

SI TAPE – A magnetic tape with recording format of physical records containing the contents of 0 to 512 central memory words of binary information or 0 to 128 words of coded information. Coded SI tapes are not supported under the NOS operating system.

VOLUME – A reel of magnetic tape or a disk pack is a volume. A given file can encompass more than one volume.

WORD ADDRESS – The relative location of the first word of a record in a word addressable file. Specified as the WA field of the file information table on a call for a read or write operation.

WORD ADDRESSABLE (WA) FILES – Word addressable files are mass storage files containing continuous data or space for data. Words within word addressable files are numbered from 1 to n, each word containing 10 characters. Retrieving or writing of data at any given word within the file is specified by the word number, called the word address.

WORKING STORAGE AREA – An area within the user's field length intended for receipt of data from a file or transmission of data to a file.

A file information table (FIT) must be associated with every file that uses BAM. For normal language requirements, compilers generate the FIT automatically; users writing in higher level languages do not need to be concerned with FITs and their generation. It is the COMPASS user's responsiblity to supply the FIT; BAM provides the FILE macro, which creates the table.

Word and bit designations of the FIT fields are illustrated in figure D-1. The fields enclosed in brackets can be accessed by the FETCH macro but cannot be changed. If a STORE macro is attempted on these fields, an assembly diagnostic results.

The FIT is activated by an OPENM request for the file. After a file is opened, the contents of the FIT can be updated with the FILE control statement or the STORE macro, with information from the processing macros, or by BAM as a result of processing the file. Information in the FIT can be retrieved with the FETCH macro.

The meanings of the FIT fields by word and bit are as follows. For convenience of the user, the COMPASS symbols are included with the applicable FIT fields. The first ten words of the FIT are used by BAM for communicating with the operating system. Generally, for any particular file organization, record, or block type, only a small portion of the total information specified here is required.

Word 0

59-18	LFN	Logical file name of the data file.
17-1		Reserved for CDC.
0	CMPLT	FET complete bit; cannot be changed by the user.

Word 1

59-48	DVT	FET device type; cannot be changed by the user.
47		Reserved for CDC.
46	RDR	Read release.
45-37		Reserved for CDC.
36	FF	OS flush on abnormal termination: 0 Buffer not flushed. 1 Buffer flushed for output file with scratch disposition on abnormal termination.
35-30		Reserved for CDC.
29-24	DC	Disposition code; cannot be changed by the user. Refer to operating system manual for possible settings.
23-18		Length of FIT minus 5; set to 30_{10}.
17-0	FWB	First word address of the user buffer.

Word 2

59-18		Zero-filled field.
17-0		Reserved for CRM.

Word 3

59-18		Zero-filled field.
17-0		Reserved for CRM.

Word 4

59-34		Reserved for CDC.
33-0		Reserved for CRM.

Word 5

59-24		Reserved for CRM/INTERCOM.
23-22	ASCII	ASCII character set bits for INTERCOM terminals. 0 64 character display code 1 95 character ASCII subset 2 128 character ASCII
21-0		Reserved for CRM.

Word 6 Reserved for CDC.

Word 7 Reserved for CRM (return address stack).

Word 8 Reserved for CDC (FET extension).

Word 9 Reserved for CDC (label fields).

Word 10

59-36	LBL	Label area length in characters.
35	LCR	Label check/creation for input/output tape: 0 N Create new labels ≡NLCR≡ 1 E Check existing labels ≡ELCR≡
34		Reserved for CRM.

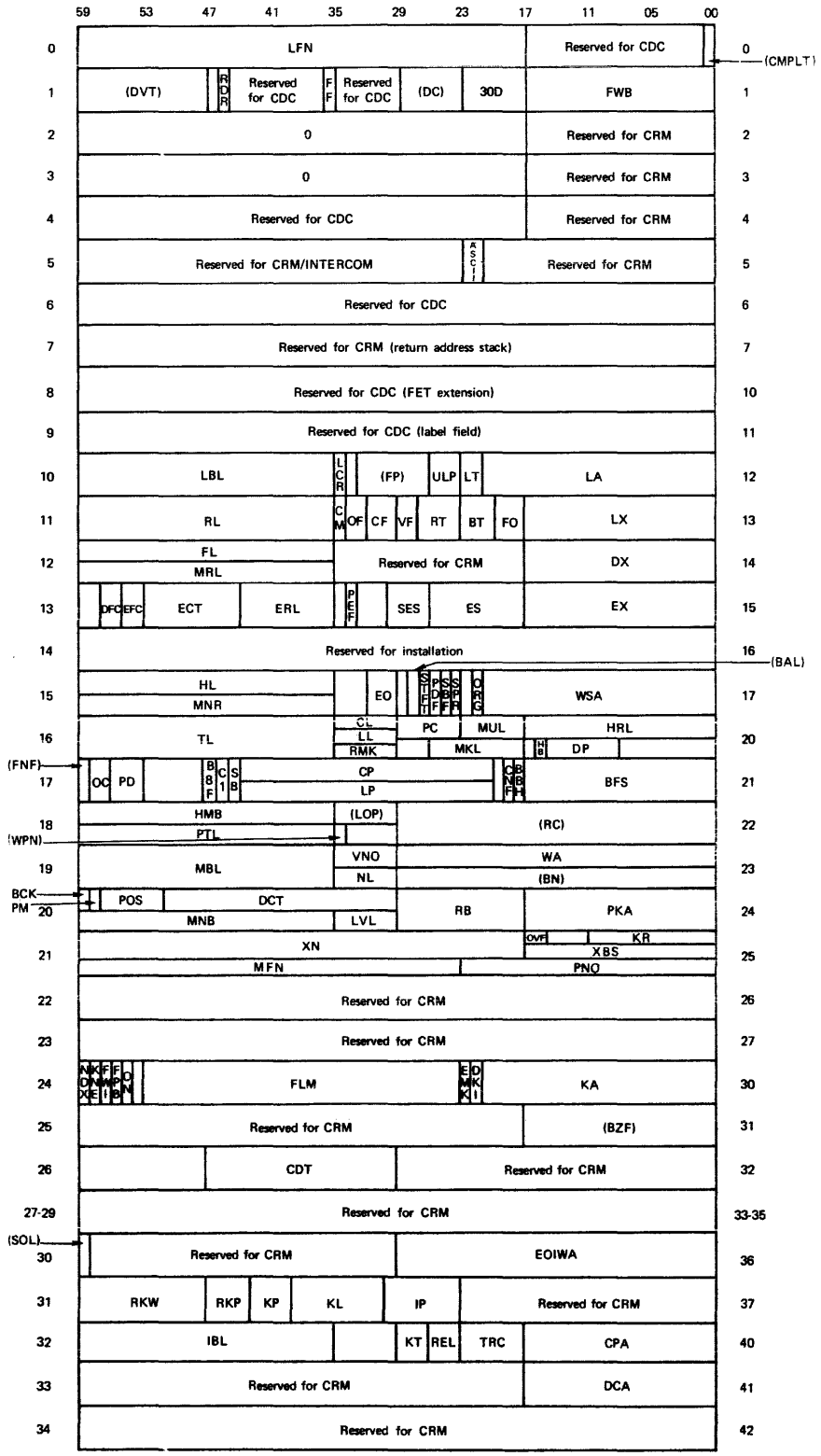


Figure D-1. File Information Table

33-27 FP File position (in octal); cannot be changed by the user:

0		Mid logical record	
1	EOL BOI	End-of-label group Beginning-of-information	≡EOL≡ ≡BOI≡
2	BOF BOV	Beginning-of-file Beginning-of-volume Only set on SKIPBU in connection with DX.	≡BOF≡ ≡BOV≡
4	EOV	End-of-volume	≡EOV≡
10	EOS	End-of-section	≡EOS≡
20	EOR	End-of-record	≡EOR≡
40	EOP	End-of-partition	≡EOP≡
100	EOI	End-of-information	≡EOI≡

26-24 ULP User label processing:

000		None	≡NOP≡
001	V	VOL/EOV	≡VP≡
010	F	HDR/EOF	≡FP≡
011	VF	VOL/HDR/-EOF/EOV	≡VFP≡
100	U	UVL/UHL/-UTL	≡UP≡
101	VU	VOL/UVL/-UHL/EOV/UTL	≡VUP≡
110	FU	UVL/HDR/-UHL/EOF/-UTL	≡FUP≡
111	VFU	All	≡VFUP≡

23-22 LT Label type:

00	S	ANSI standard	≡S≡
01	NS	Nonstandard	≡NS≡
10	UL	Unlabeled (default)	≡UL≡
11	ANY	Any	≡ANY≡

21-0 LA Label area address.

Word 11

59-36 RL Current record length in characters.

35 CM Conversion mode; convert sequential tape files from external to internal code:

0	NO	No conversion	≡NO≡
1	YES	Conversion	≡YES≡

34-33 OF Open flags; positioning of the file at OPENM time:

00		Rewind (default)	≡≡
01	R	Rewind	≡R≡
10	N	No rewind	≡N≡
11	E	Extend	≡E≡

32-30 CF Close flags; position at file close:

000		Rewind (default)	≡≡
001	R	Rewind	≡R≡
010	N	No rewind	≡N≡
011	U	Unload	≡U≡
100	RET	Return	≡RET≡
101	DET	Detach	≡DET≡
110	DIS	Disconnect	≡DIS≡

29-28 VF Volume close flag; position of the file at end-of-volume:

00		Unload (default)	≡≡
01	R	Rewind	≡R≡
10	N	No rewind	≡N≡
11	U	Unload	≡U≡

27-24 RT Record type:

0000	W	Control word	≡WT≡
0001	F	Fixed length	≡FT≡
0010	R	Record mark	≡RT≡
0011	Z	Zero byte	≡ZT≡
0100	D	Decimal character count	≡DT≡
0101	T	Trailer count	≡TT≡
0111	U	Undefined	≡UT≡
1000	S	System-logical-record	≡ST≡

23-21 BT Block type:

000		Internal (default)	≡≡
001	I	Internal	≡IT≡
010	C	Character count	≡CT≡
011	K	Record count	≡KT≡
100	E	Exact records	≡ET≡

20-18 FO File organization:

000	SQ	Sequential	≡SQ≡
001	WA	Word addressable	≡WA≡
011	IS	Indexed sequential (AAM only)	≡IS≡
101	DA	Direct access (AAM only)	≡DA≡
110	AK	Actual key (AAM only)	≡AK≡

17-0 LX Label routine exit address.

Word 12

59-36	MRL	Maximum record length in characters.
	FL	Fixed length of an F type record, or full length of a Z type record, in characters.
35-18		Reserved for CRM.
17-0	DX	End-of-data exit address.

Word 13

59-58		Reserved for CRM.
57-56	DFC	Dayfile control for error messages: 0 No dayfile messages except fatal errors 1 Error messages to dayfile 2 Notes to dayfile 3 Errors and notes to dayfile
55-54	EFC	Error file control: 0 No error file messages 1 Error messages to error file 2 Notes to error file 3 Errors and notes to error file
53-45	ECT	Trivial error count.
44-36	ERL	Trivial error limit.
35		Reserved for CRM.
34	PEF	Parity error flag: 0 No error 1 Parity error
33-31		Reserved for CRM.
30-27	SES	System parity error severity: 1 Read parity error level 1 2 Read parity error level 2 3 Read parity error level 3 4 Read parity error level 4 5 Write parity error level 1 6 Write parity error level 2
26-18	ES	Error status (octal value).
17-0	EX	Error exit address.

Word 14 — Reserved for installation.

Word 15

59-36	HL	Header length of a T type record in characters.
	MNR	Minimum record length.
35-33		Reserved for CRM.
32-30	EO	Error option: 000 T Terminate file ≡T≡ 001 D Drop erroneous data ≡D≡ 010 A Accept ≡A≡ 100 TD Terminate file and display data ≡TD≡ 101 DD Drop erroneous data and display data ≡DD≡ 110 AD Accept erroneous data and display data ≡AD≡
29		Reserved for CRM.
28	BAL	Buffer allocated by CRM; cannot be changed by the user.
27	STFT	Internal SETFIT flag used for CRM processing.
26	PDF	SETFIT macro FILE statement flag: 0 FILE control statement not processed before OPENM 1 FILE control statement was processed before OPENM
25	SBF	Suppressed buffer I/O flag: 0 NO Buffer I/O ≡NO≡ 1 YES Suppress buffer I/O ≡YES≡
24	SPR	Suppress read ahead: 0 NO Read ahead/write behind (buffered sequential I/O) ≡NO≡ 1 YES No read ahead/no write behind (unbuffered sequential I/O) ≡YES≡
23		Reserved for CRM.
22	ORG	Old/new file organization field (AAM only).
21-0	WSA	Working storage area address.

Word 16

59-36	TL	Trailer length in characters; T type record.
35-30	CL	Count field length in characters; T type records.
	LL	Length field length in characters; D type records.
	RMK	Record mark character; R type records.
29-24	PC	Padding character for sequential files.

23-18	MUL	Multiple of characters per K or E type block.
26-18	MKL	Major key length in characters (AAM only).
17-0	HRL	Hashing routine address (AAM only).
16	HB	User header option (AAM only).
15-9	DP	Data block padding percent (AAM only).

Word 17

59	FNF	Fatal/nonfatal flag; cannot be changed by the user: 0 Nonfatal 1 Fatal
58-57	OC	Open/close flag: 00 Never opened ≡NOP≡ 01 Opened ≡OPE≡ 10 Closed ≡CLO≡
56-54	PD	Processing direction: 000 Input ≡≡ 001 INPUT Input ≡INPUT≡ 010 OUTPUT Output ≡OUTPUT≡ 011 IO Input/-output ≡IO≡
53-48		Not used.
47	B8F	Round PUTs for S type records down to *8 bits; used in FORM and 8-bit subroutines: 0 NO Round up to 6 bits ≡NO≡ 1 YES Round down to 8 bits ≡YES≡
46	C1	COMP-1; format for the CL/LL field for T or D type records: 0 NO Display code 1 YES Binary
45	SB	Sign overpunch; overpunch option for CL/LL field for T or D type records: 0 NO No overpunch 1 YES Overpunch
44-21	CP	Trailer count beginning character position field of a T type record (numbered from 0).
	LP	Length field beginning character position of a D type record (numbered from 0).
20		Reserved for CRM.
19	CNF	Connect file flag: 0 NO File not connected to terminal ≡NO≡ 1 YES File connected to terminal ≡YES≡
18	BBH	Buffer below highest high address (HHA): 0 NO Buffer not below HHA 1 YES Buffer below HHA
17-0	BFS	Buffer size in words.

Word 18

59-36	HMB	Number of home blocks (AAM only).
	PTL	Parital transfer length, set by the GETP or PUTP macro.
35-30	LOP	Last operation code; the high order bit of LOP is a write bit, indicating whether the last operation wrote data to the file; cannot be changed by the user: 01 OP OPENM ≡OP≡ 02 CM CLOSEM ≡CM≡ 03 GE GET or GETP ≡GE≡ 43 PU PUT or PUTP ≡PU≡ 56 RP REPLACE ≡RP≡ 04 SE SEEK (AAM only) ≡SE≡ 05 SF SKIPF ≡SF≡ 46 DE DELETE ≡DE≡ 07 GN GETN (AAM only) ≡GN≡ 47 WE WEOR ≡WE≡ 10 RE REWINDM ≡RE≡ 11 GL GETL ≡GL≡ PL PUTL ≡PL≡ 12 SB SKIPB ≡SB≡ 13 CL CLOSEL ≡CL≡ 63 WK WTMK ≡WK≡ 74 EN ENDFILE ≡EN≡
35	WPN	Write bit. The upper bit of LOP is a 1-bit subfield that can be accessed separately. If the last operation was a write, it is set. This field cannot be changed by the user.
29-0	RC	Record count. Count of full records read or written since the file was opened. The count is not adjusted for repositioning and backspacing operations. This field cannot be changed by the user.

Word 19

59-36	MBL	Maximum block length in characters.
35-30	VNO	Current volume number of the multi-volume sequential file.
	NL	Number of index levels of blocks (AAM only).
29-0	BN	Block number of the current block (sequential files); cannot be changed by the user.
	WA	Current position word address, set by GET and PUT macros.

Word 20

59	BCK	Block checksums (AAM only).
58	PM	Processing mode (AAM only).
57-52	POS	Duplicate key position (AAM only).
51-30	DCT	Address of the display code to collating sequence conversion table (AAM only).
59-36	MNB	Minimum block length in characters.
29-18	RB	Number of records per K type block in sequential files.
17-0	PKA	Primary key address (AAM only).

Word 21

59-18	XN	Logical file name of the alternate key index file associated with the data file (AAM only).
17-0	XBS	Index file block size (AAM only).
59-24	MFN	Multifile set name.
23-0	PNO	Multifile position number; position number of member file on multifile set.
17-16	OVF	Direct access file overflow flag (AAM only).
11-0	KR	Key value repeat count (AAM only).

Word 22

59-46		Reserved for CRM.
45-40	LAC	Last action performed on the file; used by compiler languages to communicate with each other.
39-36	LNG	Last compiler language that used the file:

0	Unknown
1	COBOL
2	FORTRAN
3	PL/I
4-7	Reserved

35-0		Reserved for CRM.

Word 23 Reserved for CRM.

Word 24

59	NDX	Index flag (AAM only).
58	KNE	Key not equal (AAM only).
57	FWI	Forced write indicator (AAM only).
56	FPB	File position bit (system routine use only):

0	EOI not reached
1	EOI reached

55	ON	Old or new indexed sequential, direct access, or actual key file (AAM only).
54		Reserved for CRM.
53-24	FLM	File limit, records per file (AAM only).
23	EMK	Embedded key flag (AAM only).
22	DKI	Duplicate key indicator (AAM only).
21-0	KA	Key address (AAM only).

Word 25

59-18		Reserved for CRM.
17-0	BZF	Busy FET address; cannot be changed by the user.

Word 26

59-48		Reserved for CRM.
47-30	CDT	Address of the collating sequence to display code conversion table (AAM only).
29-0		Reserved for CRM.

Words 27-29 Reserved for CRM.

Word 30

59	SOL	S/L tape bit; cannot be changed by the user.
58-30		Reserved for CRM.
20-0	EOIWA	Word address at EOI for word addressable files.

Word 31

59-48	RKW	Relative key word (AAM only).
47-44	RKP	Relative key position in RKW (AAM only).
43-40	KP	Beginning character position of the key (AAM only).

39-31	KL	Key length in characters (AAM only).
		Key length in bits (AAM only).
		Primary or alternate key length (AAM only).
30-24	IP	Index block padding percent (AAM only).
23-0		Reserved for CRM.

Word 32

59-42	IBL	Index block length in characters (AAM only).
41-30		Reserved for CRM.
29-27	KT	Key type (AAM only).
26-24	REL	File position key relation (AAM only).
23-18	TRC	Trace transaction count; number of transactions to be traced (AAM only).
17-0	CPA	Compression routine address (AAM only).

Word 33

59-18		Reserved for CRM.
17-0	DCA	Decompression routine address (AAM only).

Word 34 Reserved for CRM.

In order to reduce field length, BAM has been divided into functional capsules which are loaded by relocatable controlling routines at execution time. This method of dynamic loading requires a program to be compatible with Common Memory Manager (CMM). Static loading is available for programs that are not compatible; however, static loading could involve a field length penalty of as much as 1400_8 words. Unless static loading is specified, BAM uses dynamic loading.

More information about Common Memory Manager and the CYBER Loader can be obtained from their respective reference manuals.

DYNAMIC LOADING

For dynamic loading, all macros reference entry points in the controlling routines. The controlling routines, which process parameters and diagnose certain types of errors, are loaded at relocatable load time or overlay generation time. The controlling routines load and transfer control to the Fast Dynamic Loader (FDL) capsule needed to process the macro in fixed-position fixed-length blocks.

It is important to the dynamic loading scheme that the controlling routines not be overlayed. Unknown results, including bad jump addresses to service routines, result if these routines are overlayed. To prevent the controlling routines from being overwritten, they must be part of the (0,0) overlay.

The OPENM/SETFIT capsule is loaded when the first OPENM or SETFIT macro is encountered. If the SETFIT macro is encountered first, the FILE control statement parameters are processed, buffer size is calculated, and control is returned to the user.

When the OPENM macro is encountered, the SETFIT functions are performed if there has not been a previous SETFIT macro. OPENM processing then occurs. The file is opened, FIT consistency checks are performed, label processing occurs, and control is returned to the user. If label processing is required, the controlling routine loads the GETL/PUTL capsule when the first GETL or PUTL macro is encountered. The open and label processing capsules are unloaded when a macro other than OPENM, SETFIT, GETL, PUTL, STORE, or FETCH is encountered. Therefore, for optimum efficiency in loading, the open processing for all files should be completed before other processing is specified.

When the first macro is encountered that requires a buffer, a buffer is allocated through CMM in a fixed-position fixed-length block. If the buffer below highest high address (BBH) field of the FIT is set to YES, CMM is requested to allocate the buffer below the highest high address (HHA). The HHA is the end of the longest overlay. If the BBH field is set to YES, the file must be closed with the CF field set to U, RET, or DET before another overlay is loaded. If the BBH field is set using the FILE macro, references are issued to the additional CMM routines necessary to process this feature. However, if the BBH field is set using the STORE macro, the FILE control statement, or some other means, the user must reference the additional CMM routines. This can be done by using either the LDSET pseudo-op or the LDSET control statement as follows:

```
LDSET  USE=CMM.AGR
LDSET(USE=$CMM.AGR$ . . .)
```

The capsules required to perform the function specified by the macro are then loaded; control transfers to the capsules and back to the user. Except for the SKIP capsules, the capsules required to process these types of functions remain in core until all files requiring them have been closed. The capsules required for SKIP are loaded while a series of skips is being performed and unloaded when a macro other than SKIP is encountered.

The CLOSEM capsule is loaded when the CLOSEM macro is encountered. It closes the file and buffer space is released if the CF field is set to U, RET, or DET; this must be specified if the BBH field is set to YES. The CLOSEM capsule unloads any capsules no longer needed for processing and unloads itself after it closes the last file.

STATIC LOADING

Static loading is provided in cases where the user is managing memory. It should only be used as a short term conversion aid. Long term support of this feature is not to be provided. There are two methods for designating which capsules need to be statically loaded; one is control statement oriented, and one is macro oriented.

STATIC LOADING WITH CONTROL STATEMENTS

To specify static loading with control statements, the option STAT must be specified on the LDSET control statement; the USE and OMIT parameters must be specified on the FILE control statement. A FILE control statement must be used for each file to insure that all necessary routines are loaded. The FO, RT, and BT parameters must be specified on a previous FILE control statement or on the same FILE control statement as the USE and OMIT parameters. They cannot be specified on a FILE control statement following the FILE control statement which specified the USE and OMIT parameters.

The formats of the USE and OMIT parameters are:

$USE=mn_1/mn_2/\ldots/mn_n$

$OMIT=mn_1/mn_2/\ldots/mn_n$

where mn is a macro name. Terminal users must use TGET and TPUT to load special terminal I/O capsules. The functions of the USE and OMIT parameters are listed in table E-1. The USE and OMIT parameters can be used on more than one FILE control statement for one file; the result is cumulative. If the STAT option is specified on the LDSET control statement and no USE parameter is specified on the FILE control statement, no functions are loaded.

In the example shown in figure E-1, the program to write the file ATAPE uses static loading and contains the macros OPENM, PUT, CLOSEM, and ENDFILE. The program to read the file ATAPE also uses static loading. The macros PUT and ENDFILE are not contained in that program; the OMIT parameter specifies that those capsules are not to be loaded. The GET macro is contained in the program, and the capsule for that macro is to be loaded. The USE parameter is still in effect for the macros OPENM and CLOSEM.

STATIC LOADING WITH THE STLD.RM MACRO

The STLD.RM macro is another method of specifying static loading. (The LDST.RM macro, which was valid in CYBER Record Manager, is treated as a no-op.) The format of the STLD.RM macro is shown in figure E-2. It must be specified once for each file organization.

TABLE E-1. USE AND OMIT PARAMETER FUNCTIONS

Parameter	No List of Macros	List of Macros
USE	All capsules are loaded.	Capsules performing functions specified by the macro list are loaded.
OMIT	All previously loaded capsules are removed.	Capsules performing functions specified by the macro list are removed.

```
.
.
.
FILE(ATAPE,FO=SQ,RT=Z,BT=C,USE=OPENM/PUT/
    CLOSEM/ENDFILE)
LDSET(STAT=ATAPE)
Load set to write file.
FILE(ATAPE,OMIT=PUT/ENDFILE,USE=GET)
LDSET(STAT=ATAPE)
Load set to read file.
.
.
.
```

Figure E-1. Static Loading Example

(fo) STLD.RM USERT=(rt_1,rt_2, . . . ,rt_n),
USEBT=(bt_1,bt_2, . . . ,bt_n),
USE=(mn_1,mn_2, . . . , mn_n),
OMIT=(CMM or FDL)

fo File organization.

rt Record types of files.

bt Block types of files.

mn Macros used in program.

CMM or FDL CMM omits CMM and FDL.
FDL omits FDL.

Figure E-2. STLD.RM Macro Format

USE OF LIST-OF-FILES

The NOS and NOS/BE operating systems maintain a pointer to the list-of-files, which is a table of the names and FIT addresses of all active files for each control point. This pointer is set and accessed by the SETLOF and GETLOF macros. A complete description of this feature can be found in the NOS or NOS/BE reference manual.

BAM maintains and uses this list-of-files. To alter this list, a user must follow a procedure that is compatible with BAM.

BAM maintains an entry point in its relocatably loaded routines called LOF$RM. The content of this entry point is the address of the current list-of-files. The purpose of this pointer is to minimize the number of GETLOF monitor calls required. The user is encouraged to use this pointer instead of calling the GETLOF macro.

If a user program that coexists with BAM moves the list-of-files, it must update the LOF$RM pointer in addition to calling the SETLOF macro. Also, if a user program adds a new entry to the end of the list-of-files, it must insure that the next word is zero because BAM does not initialize the list-of-files block to zero.

For interactive jobs, BAM puts the file that it uses for output to connected files, either OUTPUT or ZZZZZOU, in the first word of the list-of-files table. This is a requirement of the NOS operating system. If a file name is put in the first word of the list, the user cannot depend on that name remaining in the first word. If a user program uses BAM through a terminal under the NOS operating system, it cannot write to a terminal file that is not a BAM file in the same job step. The user program cannot move or destroy the ZZZZZOU entry in the first word of the list-of-files.

The following tape formats are interchangeable between 7000 Record Manager and BAM:

X-mode tapes created on early 6000 series SCOPE systems are supported for read-only purposes under 7000 Record Manager as X type record files.

Binary files having S type records, or Z type records and C type blocks, are interchangeable, provided the value of the maximum block length (MBL) field is 5120.

Files having W, F, U, D, T, R, or Z type records and I, C, K, or E type blocks are interchangeable (except for Z type records with C type blocks), provided such files are accessed via BAM on S/L devices.

The file formats that are not interchangeable are as follows:

7000 Record Manager does not read 7-track coded Z type record tapes.

7000 Record Manager does not read L tapes having a block length greater than the individual station limits.

7000 Record Manager does not correctly read a file having W, F, U, D, T, or R type records recorded on other than S or L tapes.

7000 Record Manager requires macro parameters placed in registers to be in X registers; BAM macro parameters can be in any user registers.

BAM does not read a tape file with C or I type blocks if the value of the MBL field is not equal to 5120.

BAM does not read a tape having embedded tapemarks. (WTMK under 7000 Record Manager does write a tapemark rather than a level 17 on a file with S type records or with Z type records and C type blocks. For interchangeability, use of WTMK is not recommended; the ENDFILE macro should be used instead.)

BAM does not read other than an L tape if the value of the MBL field is other than 5120.

Refer to the table on labeling conventions (section 6) for additional information on labels.

COMMENT SHEET

CONTROL DATA CORPORATION

TITLE: CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual

PUBLICATION NO. 60495700 REVISION D

This form is not intended to be used as an order blank. Control Data Corporation solicits your comments about this manual with a view to improving its usefulness in later editions.

Applications for which you use this manual.

Do you find it adequate for your purpose?

What improvements to this manual do you recommend to better serve your purpose?

Note specific errors discovered (please include page number reference).

CUT ON THIS LINE

General comments:

FROM NAME: _____ POSITION: _____

COMPANY NAME: _____

ADDRESS: _____

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

CUT ON THIS LINE

FOLD

FOLD

STAPLE

STAPLE