Honeywell

LEVEL 6

SOFTWARE

GCOS 6 DATA FILE ORGANIZATIONS AND FORMATS

SERIES 60 (LEVEL 6)

GCOS 6
DATA FILE ORGANIZATIONS AND FORMATS

SUBJECT

Changes and Updates to the Manual

SPECIAL INSTRUCTIONS

This manual supersedes Rev. 1 of the manual dated June 1978. Change bars (▌) indicate new and changed information; asterisks (*) denote deletions.

SOFTWARE SUPPORTED

This publication supports Release 100 of the Series 60 (Level 6) GCOS 6 MOD 600, MOD 400 (Release 120), and MOD 200 (Release 100) Operating Systems. See the manual directory of the latest appropriate *System Concepts* manual for information as to later releases supported by this manual.

PREREQUISITE INFORMATION

Understanding this manual and using it effectively require that the user have a thorough knowledge of the appropriate *System Concepts* manual.

ORDER NUMBER

CB05, Rev. 2

November 1978

Honeywell

Preface

This manual describes the GCOS 6 data file organizations and formats. Unless stated otherwise, the term GCOS refers to the GCOS 6 software; the term Level 6 refers to the Series 60 (Level 6) hardware on which the software executes.

Section 1 defines the terms used in the manual and explains some general file concepts.

Section 2 describes the file organizations that are available for the storage of data.

Section 3 describes the physical structure (i.e., format) of records.

Section 4 describes the physical structure of files.

Section 5 describes the volume formats.

Appendix A defines the contents of all headers.

Appendix B shows the ASCII and EBCDIC character sets, as well as the hexadecimal equivalents of each character.

Appendix C describes the keys that can be used with the various file organizations.

File No.: 1S33 CB05

MANUAL DIRECTORY

The following publications constitute the GCOS 6 manual set. See the "Manual Directory" of the appropriate *System Concepts* manual for the current revision number and addenda (if any) of the relevant operating-system-specific publications.

Order No.	*Manual Title*
CB01	*GCOS 6 Program Preparation*
CB02	*GCOS 6 Commands*
CB03	*GCOS 6 Communications Processing*
CB04	*GCOS 6 Sort/Merge*
CB05	*GCOS 6 Data File Organizations and Formats*
CB06	*GCOS 6 System Messages*
CB07	*GCOS 6 Assembly Language Reference*
CB08	*GCOS 6 System Service Macro Calls*
CB09	*GCOS 6 RPG Reference*
CB10	*GCOS 6 Intermediate COBOL Reference*
CB11	*GCOS 6 BASIC Reference*
CB12	*GCOS 6 Entry-Level COBOL Reference*
CB13	*GCOS 6 Entry-Level FORTRAN Reference*
CB20	*GCOS 6 MOD 400 System Concepts*
CB21	*GCOS 6 MOD 400 Program Execution and Checkout*
CB22	*GCOS 6 MOD 400 Programmer's Guide*
CB23	*GCOS 6 MOD 400 System Building*
CB24	*GCOS 6 MOD 400 Operator's Guide*
CB27	*GCOS 6 MOD 400 Programmer's Pocket Guide*
CB28	*GCOS 6 MOD 400 Master Index*
CB30	*Remote Batch Facility User's Guide*
CB31	*Data Entry Facility User's Guide*
CB32	*Data Entry Facility Operator's Quick Reference Guide*
CB33	*Level 6/Level 6 File Transmission Facility User's Guide*
CB34	*Level 6/Level 62 File Transmission Facility User's Guide*
CB35	*Level 6/Level 64 (Native) File Transmission Facility User's Guide*
CB36	*Level 6/Level 66 File Transmission Facility User's Guide*
CB37	*Level 6/Series 200/2000 File Transmission Facility User's Guide*
CB38	*Level 6/BSC 2780/3780 File Transmission Facility User's Guide*
CB39	*Level 6/Level 64 (Emulator) File Transmission Facility User's Guide*
CB40	*2780/3780 Workstation Facility User's Guide*
CB41	*HASP Workstation Facility User's Guide*
CB42	*Level 66 Host Resident Facility User's Guide*
CB43	*Terminal Concentration Facility User's Guide*
CB44	*Interactive Function User's Guide*
CB48	*3270 Workstation Facility User's Guide*
CB50	*GCOS 6 MOD 600 System Concepts*
CB51	*GCOS 6 MOD 600 Program Execution and Checkout*
CB52	*GCOS 6 MOD 600 Programmer's Guide*
CB53	*GCOS 6 MOD 600 System Building*
CB54	*GCOS 6 MOD 600 Administrator's Guide*
CB55	*GCOS 6 MOD 600 Transaction Driven System*
CB56	*I-D-S/II Data Base Administrator's Guide*
CB57	*I-D-S/II Data Base User's Guide*

CB58	*GCOS 6 MOD 600 Operator's Guide*
CD46	*Display Formatting and Control*
CD47	*GCOS 6 MOD 200 System Concepts*
CD48	*GCOS 6 MOD 200 Application Development Guide*
CD49	*GCOS 6 MOD 200 Operator's Guide*
CD50	*GCOS 6 MOD 200 HASP Workstation Facility User's Guide*

In addition, the following publications provide supplementary information:

Order No.	*Manual Title*
AS22	*Honeywell Level 6 Minicomputer Handbook*
AT04	*Level 6 System and Peripherals Operation Manual*
AT97	*MLCP Programmer's Reference Manual*
FQ41	*Writable Control Store User's Guide*

Contents

Figures

Tables

Section 1
Data File Concepts

A data file is a collection of logically-related data organized in a consistent and usable manner. The operating system supports a set of file organizations that make it possible to store and subsequently locate and retrieve data easily and efficiently. These organizations, which are described in Section 2, are:

- UFAS Sequential
- UFAS Indexed
- UFAS Relative
- Fixed-relative/(BES-compatible)
- UFAS Integrated (MOD 600 only)

Use of standard file organizations ensures:

- An orderly exchange of data among system and application programs
- A consistent file system interface for all programs
- A coherent means of developing application programs

In addition to file organizations, standard record, file, and volume formats are provided to ensure efficient use of the storage devices containing your data files; these formats are described in Sections 3, 4 and 5.

FIELDS AND RECORDS

A record is a collection of logically-related fields, which, in turn, are made up of meaningful data consisting of bit patterns that can be translated into alphabetic, numeric, and special characters in a standard character set. In this system, the ASCII character set is the standard set. (Refer to Appendix B.)

Figure 1-1 illustrates two records. The first is an employee record containing employee-number, name, address, zip-code, and start-date fields; the second is a payroll record containing employee-number, current-salary, planned-increase-percentage, planned-increase-dollar, increase-effective-date, and promotion-indicator fields.

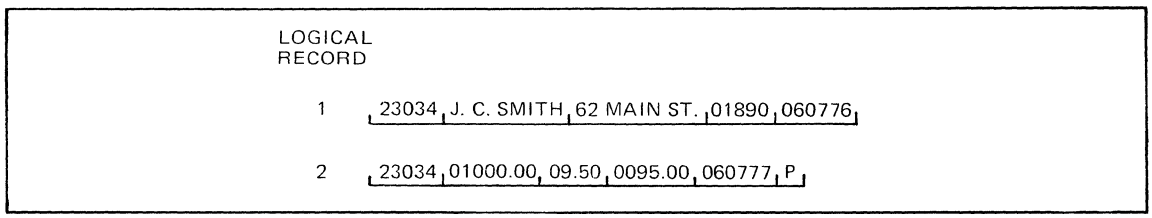


Figure 1-1. Sample (Logical) Records

As illustrated above, the data portion of records can have different formats, and records can have varying numbers of fields (of varying lengths) and mixed types of data. However, it is recommended that the same fields (e.g., name field) always contain the same type of data (e.g., alphabetic) in a single type of record (e.g., employee record); if this is not done, the application program must validate the field before processing it. To summarize, the format chosen should be consistent with the predominant usage of the field.

Records are transferred between main memory and the storage medium in logical units. When records are stored on disk device (except for fixed-relative files), these units-of-transfer are called "control intervals"; when tape-resident, they are called "blocks."

A control interval is the smallest unit of data that can be transferred between memory and a disk device. Its length is fixed at a multiple of 256 bytes, which is a compatible size for all disk devices.[1] A block is the smallest unit of data that can be transferred between memory and a tape device. Unlike control intervals, blocks may vary in length for a given file. Control intervals and blocks are described in greater detail in Section 2.

UFAS FILES

A UFAS (Unified File Access System) file is a collection of one or more records. In order to store and subsequently retrieve records, it is necessary for a predictable relationship to exist between the records and their location in the file; this relationship is called the file organization. UFAS files are transportable across all levels of Series 60 software except fixed-relative files which are only processable via Level 6 or BES software.

ACCESSING RECORDS IN FILES

A distinction is often made between the organization and accessibility of data in a file. While it is true that the organization of data in a file essentially determines the framework within which the file may be accessed by programs, the file's actual accessibility is largely a function of the language in which your programs are written; every language provides a set of instructions, commands, or verbs that can be used to access files of a specific data organization.

Basically, there are only two methods available for accessing records in a file:

- Sequential (for magnetic tape or disk volumes)
- Direct (for disk volumes)

Sequential access means that each record is read one after another, in logical succession, until the desired record is found. For sequential and relative files, this is the order in which records are physically encountered, one after another; for indexed sequential files, "logical succession" is defined as the order of ascending value of the primary key. On the other hand, direct access is used to retrieve a record directly on the basis of its location on the storage device; this requires that a technique such as the use of a key, which is described in Appendix C, be used when storing the record.

*

IDENTIFYING FILES

Data stored by one program in a file may be used or updated by other programs in the same application or by programs in another application; this may be done in one run or in a different run at a later time. In general, time elapses between a file's creation and its use; during this time, the file may be removed or the volume may be physically dismounted from the device.

In an environment where several programs running concurrently request access to files, it is necessary to ensure that each program is granted access only to the correct file so that other files are protected from unauthorized usage or inadvertent destruction. In addition, it is important that the system be able to locate, identify, and establish the required security of each file (see Access Control later in this section).

Based on information (such as file identification and attributes) that you establish when creating a file, the system builds headers, which are referenced by the system when the file is accessed on the file management level to ensure that the correct volume is mounted and that the file is accessible. (Appendix A describes the contents of these headers.)

[1]Fixed-relative diskette files may have 128-byte control intervals.

VOLUMES

One or more files can be stored on an external device (e.g., disk or tape) called a volume. Files can be stored in volumes as follows:

- A volume may contain only one file, whether or not it occupies all available space in the volume; this arrangement is called a monofile volume.
- A volume may contain more than one file, provided the last file in the volume does not go beyond the space available on that volume; this is called a multifile volume.
- A single file may occupy two or more volumes, provided the storage device is tape; this is called a multivolume file. With this arrangement, a second file cannot be contained on the volume that contains the end of the preceding file.
- A volume may contain a number of files with the last file continuing on another volume; this is called a multivolume multifile.

Note:

In this manual, volume and file structures that apply to magnetic tape apply to 9-track magnetic tape only. 7-track magnetic tape is supported under the File System, but only at the Storage Management level; volume and file structures for 7-track tape are the responsibility of the user.

The physical characteristics of the volume determine, to a large extent, the number of files that a volume can contain and the organization of those files.

Figure 1-2 illustrates the file-volume relationships described above.

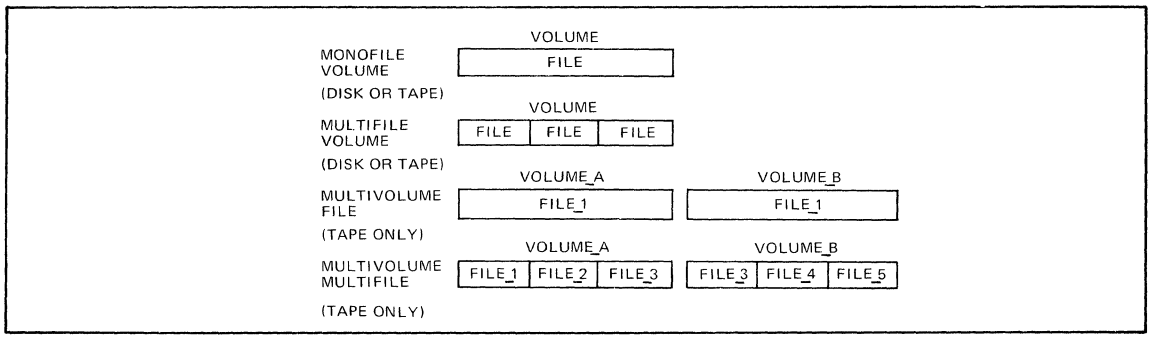


Figure 1-2. File-Volume Relationships

The manner in which records in a file can be organized in a volume is dictated by the access capability of the storage device. Basically, there are only two ways to store data in a volume:

- Physically sequential (on magnetic tape or disk volumes)
- Directly (on disk volumes)

These two methods of storing data are described briefly below; the file organizations that utilize these methods are described in detail in Section 2.

Records are considered to be stored sequentially when they are placed one after another in external storage in the same physical order in which they are presented; the relationship between a record and its location in a file is implicit only in that the record follows one specific record and precedes another specific record in the sequence.

Records are considered to be stored directly when they are placed in external storage independently of the placement of the previous record, rather than in physical order.

EXTERNAL STORAGE DEVICES

The principal external devices available for storing data are:

- Disk units (diskettes, cartridge disks, cartridge module disks, and mass storage units)
- Magnetic tape units (MOD 400 and MOD 600 only)

Although the concepts of records and files also apply to unit record devices (such as card readers, card punches, and printers), and communication devices, the major devices for data storage are the disk and tape units; therefore, the discussion in this manual concentrates on those two types of devices, although the formats of data stored on unit record devices are described in Section 3.

External storage devices provide the medium on which large amounts of data can be permanently stored, and the capability to retrieve and process that data when required. Several characteristics provide a basis on which a comparison of different storage devices can be made. These include storage capacity, access time, data transfer rate, and cost.

CHARACTERISTICS OF EXTERNAL STORAGE DEVICES

The capacity of a storage device is the amount of data that can be stored on it. Part of this capacity may be used by the device itself for error checking and synchronizing information, and by the system for storing administrative information; of the remaining capacity, some may be used for storing control- and file organization-related information. The remaining capacity available for storing data also depends on factors such as record formats (see Section 2).

Access time is the time required to locate the beginning of the data to be processed. For magnetic tapes, this is the time needed to move the tape to the beginning of the next record; for disks, it consists of seek time (which is the time needed to position the read/write head onto a particular track) and latency time (which is the time it takes the disk to rotate until the desired record reaches the read/write head). Once the beginning of the desired record is reached, it is transferred between external storage and main memory at a transfer rate specific to the particular device.

Another characteristic of an external storage device is its access capability. Records on magnetic tape units are accessed in the sequence in which they physically exist on the tape; to access a given record, each record must be read in sequence and identified, until the desired one is found. On disk units, however, it is possible to set up an addressing scheme (utilizing, for example, a primary key), and subsequently access the desired record by moving the appropriate read/write head directly to that record's cylinder/track address.

CHOOSING AN EXTERNAL STORAGE DEVICE

Selection of the external storage device for a particular file is greatly influenced by the file organization (see Section 2) that you choose; for example, if the file organization requires that records be accessible directly on the basis of their addresses, a disk unit is the best available choice. When the file organization permits use of either disk or tape, the choice is often made on the basis of the devices available in your installation.

A comparison of disk and magnetic tape units as storage media can be made on the basis of access speed, sharability, handling and storage, and cost.

Speed of access to a record in a file depends not only on the volume itself, but on the organization of the file. For a given data organization, the speed of access to a given record is generally faster for disk than tape, although reading a tape sequentially is much faster than reading a disk sequentially.

Any number of files may be stored on a disk volume, and many programs can access them at the same time; in addition, several programmers can share a disk volume to store their files. However, if several programmers are using a disk volume concurrently, seek time may be very high because there is only one arm for the read/write heads.

On the other hand, a magnetic tape volume has to be dedicated to one user. However, a magnetic tape reel is easier to handle, store, and transport; for this reason, magnetic tape volumes are used extensively as a backup medium for disk files, and as an intermediate medium when data is to be transported between installations.

In general, magnetic tape volumes are suitable as a backup and transportation medium, as a medium for dumps, and for transaction files that are processed in batches; disk volumes are used when online storage capacity for direct processing is required, and when the data to be stored is * of a volatile nature (i.e., when data is constantly manipulated).

Section 2
File Organizations

As stated in Section 1, file organizations establish a relationship between a record and its location in a file; this relationship is essential to the efficient storage and subsequent retrieval of records.

The following file organizations are available:

- UFAS Sequential
- UFAS Indexed
- UFAS Relative
- Fixed relative (BES-compatible)
- UFAS Integrated (MOD 600 only)

Records stored in the above types of files are transferred between the storage medium and main memory in logical units called control intervals (for disk-resident files), blocks (for tape-resident files), and pages (for I-D-S/II areas). This section describes characteristics, including the advantages and disadvantages, of these file organziations, and methods for calculating the size of data-transfer units; Sections 3 through 5 describe and illustrate formats of data records, files, and volumes.

UFAS SEQUENTIAL FILES

A sequential file is a tape-resident (MOD 400 and MOD 600 only) or disk-resident file containing fixed- or variable-length records organized solely on the basis of their successive physical locations within the file. Records stored in this type of file may be referred to as lines, in which case the relative position of the record in the block or control interval is called the line number.

When creating a sequential file, you must specify the size of the longest record in the file; unlike relative-files, records are truly variable in length. Depending on whether the record is tape- or disk-resident, the criteria for determining the record size vary; these differences are discussed below. Regardless of where the file resides, new records can only be appended to an existing file; i.e., they cannot be inserted in unoccupied positions.

Table 2-1 lists the advantages and disadvantages of sequential files. Formats of records stored in sequential files are illustrated and described in Section 3.

TABLE 2-1. ADVANTAGES AND DISADVANTAGES OF UFAS SEQUENTIAL FILES

Storage Medium	Advantages	Disadvantages
Disk	Records can be updated; records can be deleted; records can span control intervals; records can be read directly.	Records cannot be inserted; when a record is deleted, the space becomes unusable.
Tape	Access is extremely fast; tapes are easily transportable.	Records cannot be updated, deleted, inserted, or read directly.
Both	Overhead is relatively low; compatible with all Series 60 sequential files.	Records cannot be inserted; direct access is limited to disk-resident files.

DISK RESIDENT SEQUENTIAL FILES

Records are written in a sequential file in whatever order they are presented. When creating a sequential file, you must specify the size of the longest record, whether it is fixed- or variable-length. Each record includes a 4-byte header that specifies whether or not the record is active, its size, etc.; although the header must be considered part of the record length, it is built and maintained by the system. The following steps may be used to calculate the information required by a command or macro call when the file is created:

1. Calculate the size of the file, as follows:

   file size = record size x number of records

   Sizes are specified as a number of bytes; the record size must include the 4-byte header; allow for additional 4-byte headers, which are required for each segment of spanned records.

   Although space is specified in terms of control intervals, it is allocated in terms of logical sectors (each of which contains one or more physical sectors).
2. Select a control interval size that is a multiple of 256 bytes (i.e., 256, 512, 768, etc.).
3. Calculate the number of control intervals needed to store the entire file as follows:

$$\text{number of control intervals} = \frac{\text{file size}}{\text{control interval size} - 8}$$

   If the result is a fraction, round up to the next higher integer; the control-interval size is reduced by 8 bytes because the system builds and maintains an 8-byte control-interval header for each control interval.

It is not necessary to calculate the number of records that can be stored in a single control interval, because the system splits the last record in a control interval if it does not fit and stores the remainder in the following control interval; this type of record is called a spanned record, and is supported only in disk-resident sequential files. (See Section 3 for a description of spanned records.)

Optionally, a simple key may be used to access records stored in a disk-resident sequential file (see Appendix C). It is a hexadecimal number that specifies both the control interval number (beginning with 1) and the line number (beginning with 0) of a given record; for example, the simple-key value Z'0000050F' specifies the record occupying the sixteenth position in the fifth control interval in a given file.

Records are written in a sequential file in physical sequence only (i.e., in the order in which they are presented); they cannot be written in the file directly (via a simple key). In addition, records stored in a sequential file may be deleted, but new records cannot be inserted in the position occupied by the deleted record; instead, new records can only be appended to an existing file. Finally, records stored in a disk-resident sequential file can be updated (i.e., rewritten) and read sequentially or directly (via a simple key).

TAPE-RESIDENT SEQUENTIAL FILES (MOD 400 AND MOD 600 ONLY)

Sequential files stored on 9-track tape volumes are the same as disk-resident sequential files, except for the following:

- Records cannot be deleted from or inserted into a file; however, records can be appended to a file.
- Records cannot be updated (i.e., rewritten).
- Records cannot span blocks.

When creating a tape-resident sequential file, you must specify the size of the longest record, whether it is fixed- or variable-length. Variable-length records include a 4-byte length-field, supplied by the system; in addition, you may number the blocks in the file, in which case each block contains a 6-byte block sequence number (BSN) field supplied by the system.

UFAS INDEXED FILES

An indexed file is a disk-resident file containing fixed- and variable-length records, stored in data and overflow control intervals, and index control intervals through which records are accessed directly or sequentially. Records stored in this type of file may be referred to as lines, in which case the position of the record in the data or overflow control interval is called the line number.

Index control intervals are allocated and maintained by the system, and vary in number depending on the number of control intervals allocated to the file. There are three types of overflow:

- Free space
- Local overflow control intervals
- General overflow control intervals

Free space is defined when the file is created as a percentage value. When the file is loaded, that percentage of each data control interval is left free for later file expansion; i.e., no data is loaded into it. Records can later be written into free space (or into reused data control interval space); this method is most efficient, since no index changes are required. Local overflow control intervals are interspersed with data control intervals at regular intervals. General overflow control intervals follow the last data control interval in the file.

Overflow control intervals are loaded by the system when there is insufficient space in data control intervals to contain your records. When you present a new record to be written in an existing indexed file, the system first attempts to write the record in a data control interval. If that data control interval is full, the system then attempts to write it in a local overflow control interval associated with that data control interval. If that fails, the system tries to write the record in a general overflow control interval at the end of the file. When a record is written in an overflow control interval, that control interval assumes the format of a data control interval (see Section 3).

When creating an indexed file, you must specify both the size of the longest record in the file and the location and size of a fixed-length primary key embedded in each record. Your calculations must consider the file overhead; each record has an 8-byte header and a 2-byte line offset array entry associated with it, and each control interval has a 12-byte header. In addition, each non-overflow data control interval contains an 8-byte link-record header and a corresponding 2-byte offset array entry. The following steps may be used to calculate the informaton required by the command or macro call when the file is created:

1. Determine the maximum number of records to be written in the file.
2. Determine how much free space is to be left in each data control interval; the system attempts to insert new records into available free space before it writes them in overflow control intervals.
3. Select a data control interval size that is a multiple of 256 bytes (i.e., 256, 512 etc.).
4. Calculate the number of records that can be stored in each data control interval as follows:

$$\text{number of records per data control interval} = \frac{\text{control interval size} - (\text{free space} + 22)}{\text{record size}}$$

   The control interval size is reduced by the amount of free space in each control interval plus an additional 22 bytes to accommodate the 12-byte control-interval header built and maintained by the system, the 8-byte link-record header, and its associated 2-byte line offset array entry; the record size must include the 8-byte header and 2-byte line offset array entry, and is the size of the longest record in the file. If the result is a fraction, round down to the next lower integer.

5. Calculate the number of data control intervals initially required to contain the file as follows:

$$\text{number of data control intervals} = \frac{\text{number of records}}{\text{number of records per data control interval}}$$

   If the result is a fraction, round up to the next higher integer.

6. When local overflow control intervals are selected, determine the interval at which they are to be allocated and then calculate the number of local overflow control intervals as follows:

$$\text{number of local overflow control intervals} = \frac{\text{number of data control intervals}}{\text{local overflow allocation interval}}$$

Add this number (rounded to the next higher integer) to the "number of data control intervals" value.

7. Determine how many control intervals are to be allocated and used as general overflow control intervals; add this to the number calculated in step 5.

Based on the information concerning the size and location of the required primary keys that you provide when creating the file, the system builds and maintains index control intervals that are used to locate specific records when you specify any key operation (i.e., "read-with-key," "write-with-key" or "delete-with-key" command). In addition, the system builds and maintains inventory control intervals to keep track of available space in overflow control intervals.

An indexed file is loaded by opening it in RENEW mode, presenting the records in ascending key value sequence, and closing it (to create the index). Once the records are stored in the file, they may be updated (although the key and record length cannot be modified) or deleted (in which case, the record is physically removed, and the data control interval that contained the deleted record is compacted). New records may be inserted into available space (i.e., free space or space resulting from the deletion of records) in an existing data control interval or stored in overflow control intervals at the end of the file; records may be read directly, based on a primary key value, or sequentially (i.e., in primary-key sequence).

Indexed files use considerable space in the file for indexes and inventory control. Essentially, the total file size is equivalent to the combined sizes of (1) data control intervals, (2) overflow control intervals, and (3) index control intervals (both coarse- and fine-level). These control intervals are all apportioned automatically by the system based on directives given when the file was created. The following calculations may be used to predict how much space is required.

1. $\text{no. of data c.i.'s} = \frac{\text{no. of records}}{\text{no. of rcds. per data c.i.}}$

   If the result is a fraction, round up to the next higher integer.

2. no. of overflow c.i.'s = no. of allocated c.i.'s − no. of data c.i.'s.

   If the number of control intervals initially allocated to the file is not large enough to hold all the records to be written at load time, the "no. of allocated c.i.'s" must include the number of control intervals by which it is dynamically incremented. (See the description of the Create File command and macro call in the *Commands* and *System Service Macro Calls* manuals, respectively.)

   The total number of index control intervals is increased by 2 to compensate for the general overflow inventory control intervals; it is an arbitrary number, but enough for most applications.

   a. The number of fine-level index control intervals is calculated as:

$$\text{no. of fine-level c.i.'s} = \frac{\text{no. of data c.i.'s} + \text{no. of overflow c.i.'s}}{\text{no. of entries per index c.i.}}$$

   If the result is a fraction, round up to the next higher integer; the formula for calculating the "no. of entries per index c.i." is given below.

   b. The number of course-level index control intervals is calculated as:

$$\text{no. of course-level c.i.'s} = \frac{\text{no. of fine-level c.i.'s}}{\text{no. of entries per index c.i.}}$$

If the result is greater than 1, the course-level index contains more than one level. (See Figure 2-1.) Thus, this formula must be expanded as follows for each hierarchial level (this part of the formula must be repeated until a value of 1 is obtained):

$$+ \frac{\text{no. of course-level c.i.'s from immediately interior level}}{\text{no. of entries per index c.i.}}$$

If the "no. of course-level c.i.'s" is a fraction, round up to the next higher integer. The formula for calculating the "no. of entries per index c.i." is given later in this section.

As mentioned above (and illustrated in Section 4), an indexed file can contain more than one level of course-level index. Figure 2-1 illustrates how to determine the number of levels required for the index and the number of course-level index control intervals. Note that the maximum number of levels is 4. If the number of levels exceeds 4, the index control-interval size selected is not large enough to contain the required number of entries; therefore, one of the following must be done:

- Increase the size of the control interval.
- Decrease the size of the key.
- Decrease the number of records loaded into the data control interval portion of the file.

In addition, the maximum number of records that can be written in an indexed file can be determined by using the following formula:

maximum number of records = xy^a

where: x = Number of records per data control interval
y = Number of entries per index control interval
a = Number of index levels

The number of index levels must be from 1 through 4. The number of entries per index control intervals is calculated as:

$$\text{no. of entries per index c.i.} = \frac{\text{c.i. size} - 12}{\text{key size} + 4}$$

For index control intervals, the index control-interval size is reduced by 12 bytes to accommodate the control-interval header. The key size is increased by four to accommodate a field that identifies the number of the control interval containing that key.

If the result is a fraction, round down to the next lower integer.

Table 2-2 lists the advantages and disadvantages of indexed files.

UFAS RELATIVE FILES

A relative file is a disk-resident file containing fixed- or variable-length records that are logically located in positions relative to the beginning of the file; the first record is record 1. Records stored in this type of file may be referred to as lines, in which case the relative position of the record in the control interval is called the line number.

When creating a relative file, you must specify the size of the longest record in the file. Each record includes a 4-byte header that specifies whether or not the record is active, its size, etc; although the header must be considered part of the record length, it is built and maintained by the system. The following steps may be used to calculate the information required by the command or macro call when the file is created:

1. Determine the maximum number of records to be written in the file.
2. Select a control interval size that is a multiple of 256 bytes (i.e., 256, 512 etc.). Although specified in control intervals, space is allocated in terms of logical sectors (i.e., an integral number of physical sectors).

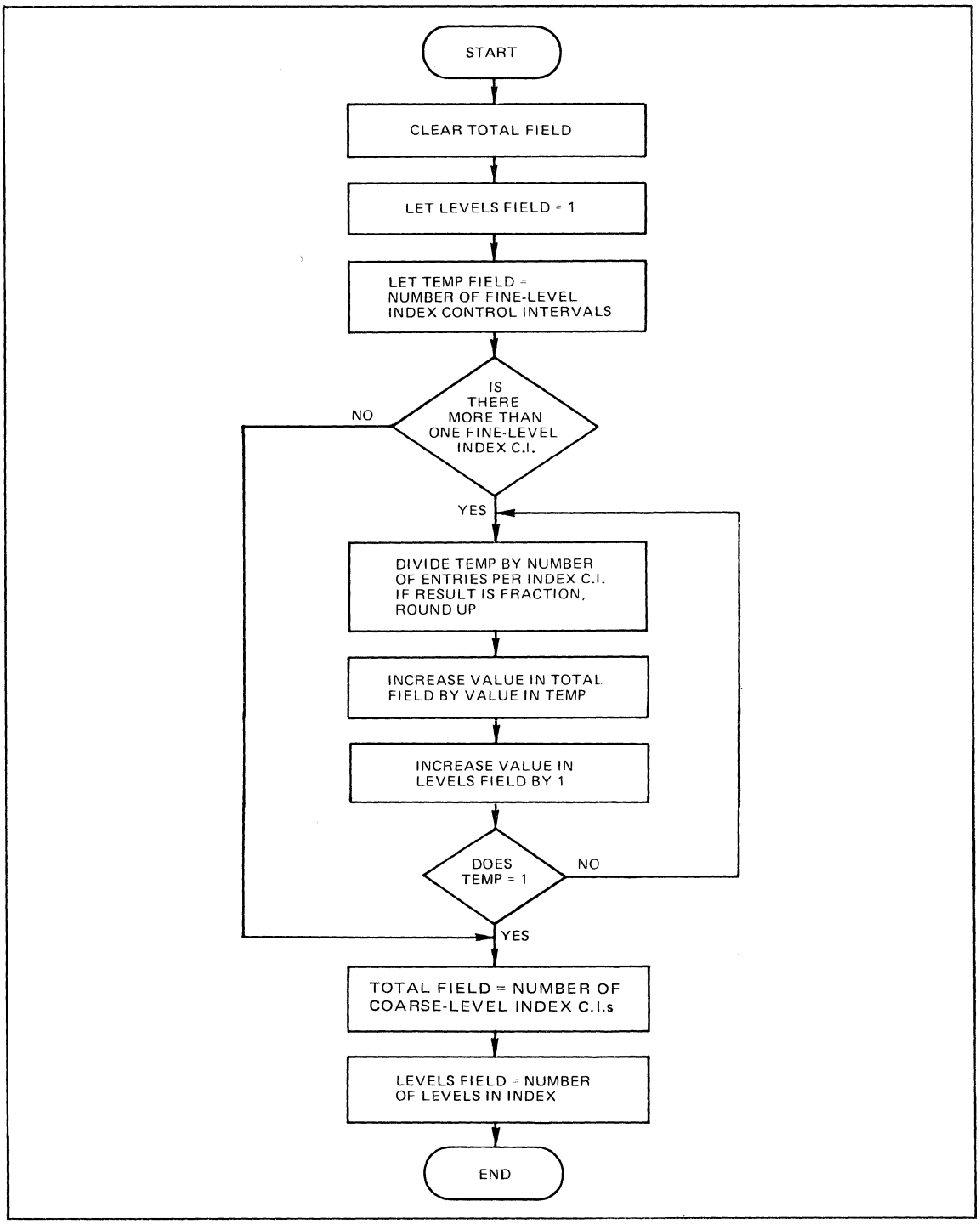


Figure 2-1. Determination of Number of Indexed Levels

TABLE 2-2. ADVANTAGES AND DISADVANTAGES OF UFAS INDEXED FILES

Advantages	Disadvantages
Easy access to records through embedded keys; system builds the index to keep track of the location of records; new records may be inserted anywhere in the file; once the file is loaded, the index is not updated again; compatible with all Series 60 indexed files; records may be fixed- or variable-length.	High overhead; direct access is relatively slow because of the time needed to search index; performance degrades as general overflow control intervals are filled (due to inserts), making it necessary to reorganize the file periodically (i.e., unload and reload) to restore performance; must be disk-resident.

3. Calculate the number of records that can be stored in each control interval as follows:

$$\text{number records per control interval} = \frac{\text{control interval size} - 8}{\text{record size}}$$

The control interval size is reduced by 8 bytes because the system builds and maintains an 8-byte control-interval header for each control interval. The record size must include the 4-byte header, and is the size of the longest record in the file. If the result is a fraction, round down to the next lower integer.

4. Calculate the number of control intervals initially required to contain the file as follows:

$$\text{number of control intervals} = \frac{\text{number of records}}{\text{number of records per control interval}}$$

If the result is a fraction, round up to the next higher integer.

You may optionally use a relative or simple key to access records stored in a relative file (see Appendix C). A relative key is a hexadecimal number that specifies the position of a record (beginning with 1) relative to the beginning of the file; on the other hand, a simple key is a hexadecimal number that specifies both the control interval number (beginning with 1) and the line number (beginning with 0) of a record. For example, as illustrated in Figure 2-2, if the eleventh record occupies the second position in the fourth control interval in the file, you could use the relative-key value Z'0000000B' or the simple-key value Z'00000401' to read the record directly.

RECORD NUMBER:	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LINE NUMBER:	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
CONTROL INTERVAL NUMBER:	1			2			3			4			5		

Figure 2-2. Sample Use of Line Numbers

Records may be written in a relative file sequentially (i.e., in the order in which they are presented) or directly (via a key). If you attempt to write a new record directly into a position occupied by an active record, the new record is not written and an error code, indicating "duplicate key," is returned. If you write new records sequentially in an existing file, they are written in the first available (i.e., unoccupied) position (i.e., in a position that has never been occupied, was occupied by a record that has been deleted, or following the last active record in the file).

In addition, you can update (i.e., rewrite) records, delete records, and read records in a relative file sequentially or directly. Finally, records can be inserted into unoccupied positions in the file as described above.

Table 2-3 lists the advantages and disadvantages of relative files. Formats of records stored in relative files are illustrated and described in Section 3.

TABLE 2-3. ADVANTAGES AND DISADVANTAGES OF RELATIVE FILES

Advantages	Disadvantages
Access to records is fast; records can be written or read directly or sequentially; records can be updated (i.e., rewritten) or deleted; overhead is relatively low; the size of a record can be altered during a rewrite record operation; compatible with all Series 60 relative files; records may be fixed- or variable-length.	Variable-length records occupy fixed-length areas (each of which equals the size of the longest record in the file), and the difference between the record size and the data portion of the record is wasted. Files must be disk-resident.

INTEGRATED I-D-S/II AREA (MOD 600 ONLY)

An integrated I-D-S/II data base area (file) is disk resident and contains fixed or variable length records. It allows many independent users access to the integrated data base via the COBOL-like DML (data manipulation language) statements or assembly calls, and provides each user with a separate subset view (subschema) of the data base structure.

The logical data structure and physical area size characteristics are defined by the DBA (data base administrator). The schema DDL (data description language) defines the logical data structure of the data base, and the schema DMCL (device/media control language) defines physical area characteristics. Area space allocation can be accomplished automatically via the -EC option in the DMCL TRANS command line (see *Data Base Administrator's Guide)*. The resulting schema-name.EC file contains all the commands necessary to create the areas of the data base.

The subschema DDL created by the DBA and used by the application programmer defines the portion of the data base available to a particular program. The DML, a series of extended COBOL verbs used within the COBOL program, permits the user access to the data base as viewed by the subschema DDL description.

An I-D-S/II area is a UFAS file that can contain one or more record types. Records can be located directly (i.e., one physical access) by a CALC key. Each CALC record has a defined CALC key that is composed of one or more fields which can be noncontiguous and of differing data types. For example, the CALC key of an employee record may be the employee's social security number. The CALC key of a product record may be a plant identification field concatenated with a product identification field.

A hash algorithm applied to a CALC key determines the location of the record in the area. The hash algorithm distributes the records uniformly throughout the area. Records that hash to the same location are chained in a CALC set. The domain of the hash (the range of values generated by the hash algorithm) is specified by the user in the DMCL (device/media control language) program.

An alternative method of storing records is to store the record via its membership in a user defined set. The member records are stored as close as possible to the owner of the set.

For each I-D-S/II area the following attributes are specified by the DBA in the DMCL:

- Page (control interval) size — a multiple of 256 bytes; the maximum page size is 2^{14} — 256 (16,128)
- Page interval — the maximum number of records that can fit on one page: 1 to 2^8 (256)
- CALC interval — the number of data base keys between each CALC header record: 0, 2 to 2^{16} (65,536)

  The maximum number of CALC sets is:

$$\frac{\text{maximum number of records} +}{\text{CLI}}$$

- Area size — the maximum number of records that can fit in one area: 1 to 2^{32} — a (4,294,967,295)
- The maximum number of pages (control intervals) in an area is 2^{24} (16,777,216).

TABLE 2-4. ADVANTAGES AND DISADVANTAGES OF INTEGRATED FILES

Advantages	Disadvantages
Eliminates data redundancy; provides data independence and data security; inter-record processing; records may be stored directly via a CALC key or near an owner of a user defined set; CALC access is very fast.	Not suitable for sequential file processing; requires a predefined data description.

NON-UFAS FIXED-RELATIVE FILES (MOD 400 AND MOD 600 ONLY)

A fixed-relative file is a disk-resident file containing fixed-length records that are logically located in positions relative to the beginning of the file; the first record in the file is record 1.

When this type of file is created, you must declare whether it contains deletable or nondeletable records. If the file contains deletable records, each record includes a 2-byte header that indicates whether or not the record is active (i.e., whether or not it has been deleted). On the other hand, nondeletable records are always active (i.e., they cannot be deleted), and they contain no headers (i.e., there is no overhead).

Since fixed-relative files are BES-compatible, the concept of control intervals does not exist; however, when the appropriate command or macro call is used to create the file, it must specify a "control interval" (i.e, unit-of-transfer) size and the number of control intervals to be allocated to the file. This can be done as follows:

1. Calculate the size of the file by using the following formula:

   file size = record size × number of records

   Sizes are specified as a number of bytes; if the file contains deletable records, the record size must include the 2-byte header. In addition, although the record size is fixed when the file is created, it may be modified when the file is opened (if the records are nondeletable).

2. Specify a control interval size that is a multiple of the physical sector size (i.e., 128, 256, 512, 768, etc.). Although space is specified in terms of control intervals, it is allocated in terms of logical sectors, each of which contains one or more physical sectors.

3. Calculate the number of control intervals required to contain the file by using the following formula:

$$\text{number of control intervals} = \frac{\text{file size}}{\text{control interval size}}$$

   If the result is a fraction, round up to the next higher integer.

You may, optionally, use a relative key (see Appendix C) to access records in a fixed-relative file. This type of key is a binary number that specifies the position of a record (beginning with 1) relative to the beginning of the file, making it possible to read or write records starting at a relative position in the file.

Table 2-5 lists the advantages and disadvantages of fixed-relative files. Formats of records stored in fixed-relative files are described in Section 3.

TABLE 2-5. ADVANTAGES AND DISADVANTAGES OF FIXED-RELATIVE FILES (NON-UFAS)

Type of Records Stored	Advantages	Disadvantages
Nondeletable	Lowest overhead; file contains only data; maximum use of available file space; size of records can be changed dynamically when the file is opened.	Records cannot be deleted.
Deletable	Low overhead (2 bytes per record); records can be deleted and new ones inserted into the positions previously occupied by a deleted record.	The size of records is fixed when the file is created, and cannot be altered when file is opened.
Either	Low overhead; maximum availability of file space for (data) records; compatible with BES files; records can be accessed directly via a relative key.	Fixed-length records only; file must reside on a disk volume.

FIXED-RELATIVE FILES WITH NONDELETABLE RECORDS

Nondeletable records are written in new fixed-relative files sequentially or in a relative position based on a key value. If it is an existing file, records can be written sequentially starting in the first available (i.e., unoccupied) position (i.e., they are appended to the file), or starting at a specified position without regard to the contents of the file (i.e., a write-with-key operation causes existing records to be overlayed by the new records).

You may update (i.e., rewrite) a nondeletable record, but you cannot delete it (i.e., a delete record command is rejected).

Finally, you may read nondeletable records stored in a fixed-relative file either sequentially from the beginning of the file, or directly via a relative key value (which specifies the position from which reading is to commence).

FIXED-RELATIVE FILES WITH DELETABLE RECORDS (MOD 400 AND MOD 600)

Deletable records are also written in fixed-relative files in order by relative record number; however, new records may be inserted into the file, in addition to being appended to it. This is possible because positions become "unoccupied" when a record is deleted; thus, records can be written in these available positions, which may be interspersed throughout the file.

You may delete, as well as update, existing records, and you may read the records sequentially, or directly via a relative key. If the system encounters a deleted record, during a sequential-read operation, the record is skipped; however, if a "read-with-key" operation directs the system to an unoccupied position (e.g., a position containing a deleted record), an error code, indicating "record not found," is returned; similarly, if a "write-with-key"operation directs the system to an occupied position (i.e., a position containing an active record), an error code, indicating "duplicate key," is returned.

Section 3
Record Formats

Standard formatting at all levels (i.e., record, file, and volume) is essential to ensure efficient storage and retrieval of data. This section describes standard record formats provided for system users.

As described in Section 1, a record is a collection of logically-related data fields. The record design process consists of deciding how many and what kind of fields the record will comprise, and the format and length of each field.

The following record formats are supported by the system:

- Fixed-length
- Variable-length
- Variable-length, spanned

Use of a particular record format depends upon the file organization selected. Table 3-1 shows the record formats supported by available file organizations.

TABLE 3-1. RECORD FORMATS SUPPORTED BY AVAILABLE FILE ORGANIZATIONS

Record Format File Organization	Fixed-Length	Variable-Length	Variable-Length, Spanned
Sequential, Disk-Resident	X[a]	X[a]	X[a]
Sequential, Tape-Resident[b]	X	X	Not Allowed
Indexed	X[c]	X[c]	Not Allowed
Relative	X[c, d]	X[c, d]	Not Allowed
Fixed-Relative	X	Not Allowed	Not Allowed
I-D-S/II (MOD 600)	X	X	Not Allowed

[a]All records stored in disk-resident sequential files are treated by the system as though they are variable-length, spanned; i.e., regardless of the actual format of the record, the system-built and maintained record header contains spanning information.

[b]MOD 400 and MOD 600 only.

[c]All records stored in this type of file are treated by the system as though they are variable-length; i.e., regardless of the actual format of the record, the system-built and maintained record header contains the same information for fixed- and variable-length records.

[d]All records stored in relative files occupy areas equal in length to the longest record in the file; e.g., if the longest record in the file is 80 bytes long, a 70-byte record also occupies an 80-byte area (i.e., 10 bytes are wasted).

FIXED-LENGTH RECORDS

Records are considered to be fixed-length when all records in a file are the same length. It is not mandatory for the same fields in the record to be the same length, nor is it mandatory that all records contain the same number of fields. However, when designing record formats, it is a good practice to define fixed-length fields in fixed locations so that your program will not have to verify the contents of each field before processing the record.

As shown in Table 3-1, records stored in fixed-relative files are fixed-length, and records stored in tape-resident sequential files may be fixed- or variable-length. These formats are described and illustrated below.

However, fixed-length records stored in the other types of files are formatted like variable length (nonspanned) or variable-length, spanned, records; refer to "Variable-Length Records" and "Variable-Length, Spanned Records," respectively, later in this section for appropriate descriptions.

FIXED-LENGTH RECORDS IN FIXED-RELATIVE FILES

The format of fixed-length records in fixed-relative files depends on whether or not the file is declared to contain deletable records. Figure 3-1 illustrates the structure of nondeletable records stored in a fixed-relative file; Figure 3-2 illustrates deletable records stored in a fixed-relative file.

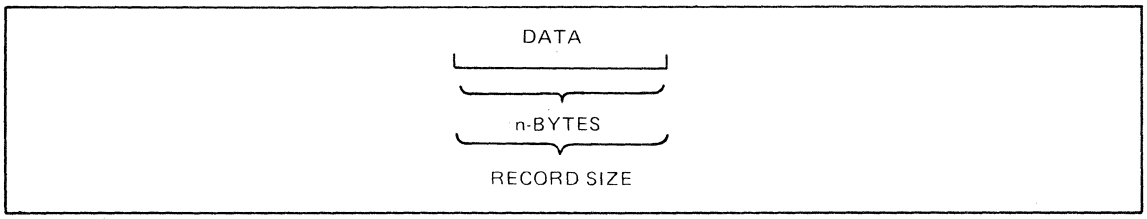


Figure 3-1. Format of Nondeletable Records in Fixed-Relative Files or Fixed-Length Records in Tape-Resident Sequential Files

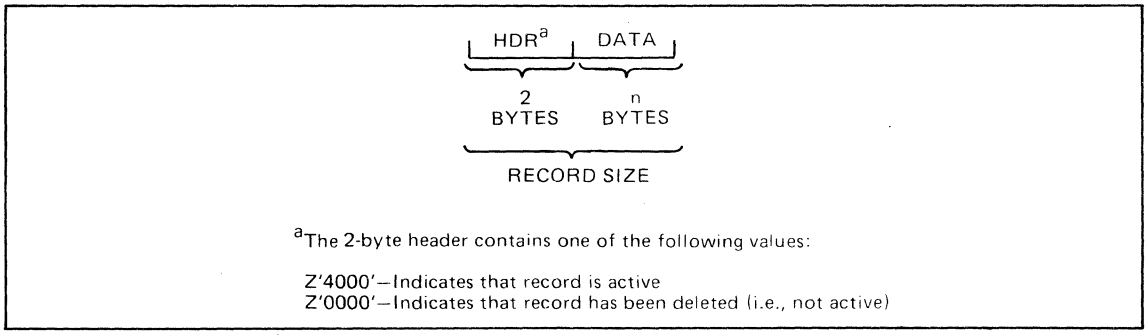


Figure 3-2. Format of Deletable Records in Fixed-Relative Files

FIXED-LENGTH RECORDS IN TAPE-RESIDENT SEQUENTIAL FILES (MOD 400 AND MOD 600 ONLY)

Fixed-length records stored in tape-resident sequential files have the same format as nondeletable records stored in fixed-relative files (i.e., the record does not have a header). Refer to Figure 3-1 for an illustration of the structure of a fixed-length record stored in a tape-resident sequential file.

Note that if block sequence number (BSNs) are included, the first record in each block is preceded by the 6-byte BSN.

VARIABLE-LENGTH RECORDS

Records are considered to be variable-length when the records in a file have varying lengths. As with fixed-length records, it is a good practice, when designing records, to ensure that fields common to all records are the same length and in the same position.

As shown in Table 3-1, variable-length records stored in tape-resident sequential files have a different format from fixed-length records stored in the same type of file. However, all records stored in relative and indexed files are treated, by the system, as variable-length; essentially, this means that regardless of the length of the data portion of the record, every record header specifies that length. For a description of the structure of variable-length records stored in disk-resident sequential files, see "Variable-Length, Spanned Record," later in this section; the following paragraphs describe the format of variable-length records stored in tape-resident sequential files, and all records stored in relative and indexed files.

RECORDS STORED IN RELATIVE FILES

As mentioned above, all records stored in relative files are treated as though they were variable-length; that is, the record header built and maintained by the system specifies, among other things, the actual length of the data portion of each record (see Appendix A). Figures 3-3 and 3-4 illustrate the structure of records stored in a relative file.

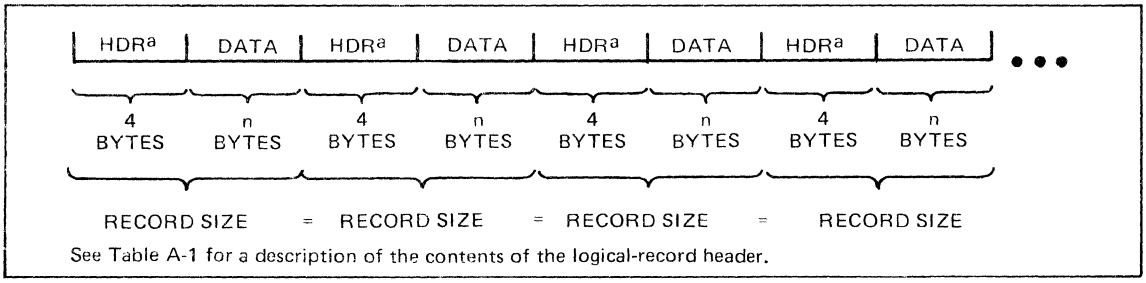


Figure 3-3. Format of Fixed-Length Records in Relative Files

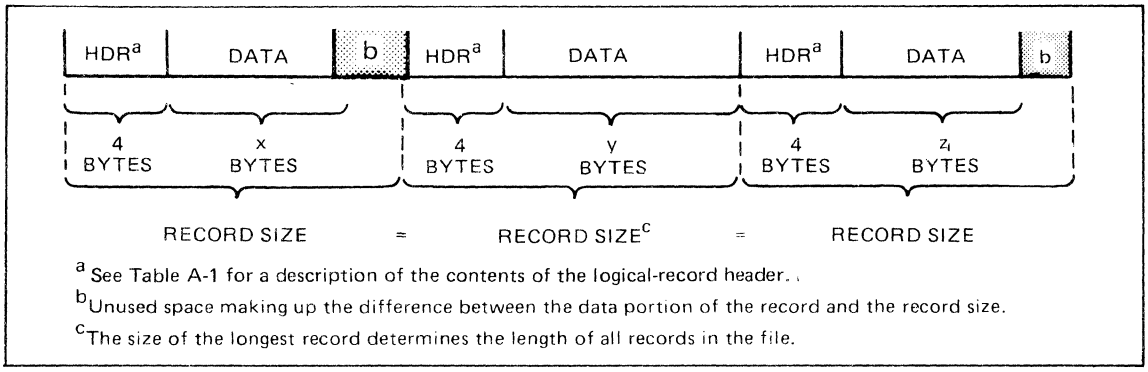


Figure 3-4. Format of Variable-Length Records in Relative Files

RECORDS STORED IN INDEXED FILES

As with relative files, all records stored in indexed files are treated as though they were variable-length; that is, the record header, built and maintained by the system, specifies, among other things, the length of the data portion of each record (see Appendix A). However, unlike records stored in other files, each record in an indexed file has associated with it a 2-byte entry that specifies the offset, in bytes, from the beginning of the control interval, in which the record is stored, to the first byte of the record (i.e., the first byte of the logical-record header); additional information about the "line offset array" is provided later in this section, under "Indexed File Format." Figure 3-5 illustrates the structure of records stored in an indexed file; note that although a line-offset-array entry is not actually part of the record, it does occupy 2 bytes perrecord.

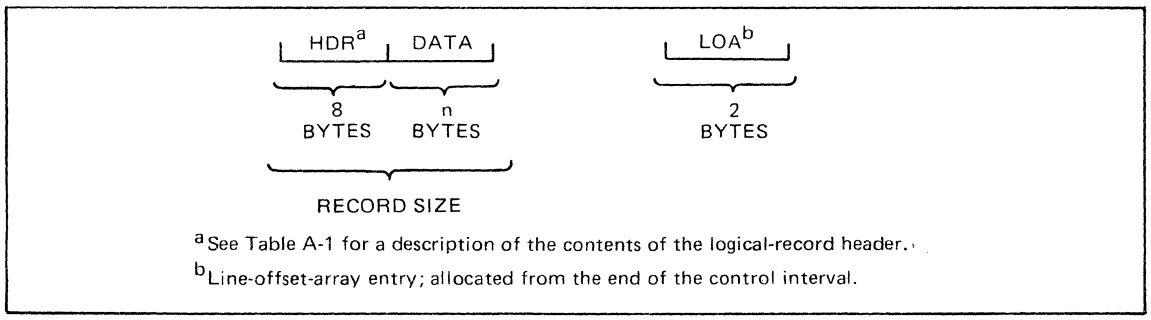


Figure 3-5. Format of Fixed- and Variable-Length Records in Indexed Files

VARIABLE-LENGTH RECORDS STORED IN TAPE-RESIDENT SEQUENTIAL FILES (MOD 400 AND MOD 600 ONLY)

Variable-length records stored in 9-track, tape-resident, sequential files have a unique format in that the first 4 bytes of each logical record must specify the length of the entire record, including that 4-byte field; the record size is expressed as a right-justified ASCII decimal-digit entry. Figure 3-6 illustrates the structure of variable-length records stored in 9-track tape resident, sequential files.

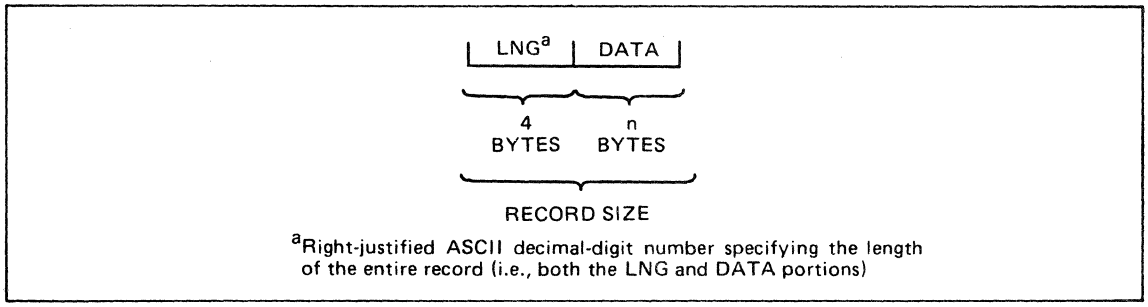


Figure 3-6. Format of Variable-Length Records in Tape-Resident Sequential Files

VARIABLE-LENGTH, SPANNED RECORDS IN DISK-RESIDENT SEQUENTIAL FILES

A record is considered to be spanned when it becomes necessary to segment it and store the segments in two or more control intervals; since the length of the data portion may vary, segmented records are referred to as variable-length, spanned records. All records stored in disk-resident sequential files are treated by the system as though they span control intervals; that is, the logical-record headers specify both the length of the record and whether the entire record is contained in one control interval or spread over two or more control intervals, in which case each segment has a header (see Appendix A). The same care should be taken as when designing any variable-length record (see "Variable-Length Records" earlier in this section). If and when it is necessary for a record to span control intervals, the system segments the record and ensures that each segment is properly identified; thus, you do not have to be concerned with segmenting a record that may not fit into the available space in a control interval.

Figure 3-7 illustrates the structure of spanned records stored in a disk-resident sequential file; note that each segment of a spanned record contains a logical-record header. Records that fit entirely into a control interval have the same format as records stored in a relative file (see Figures 3-3 and 3-4).

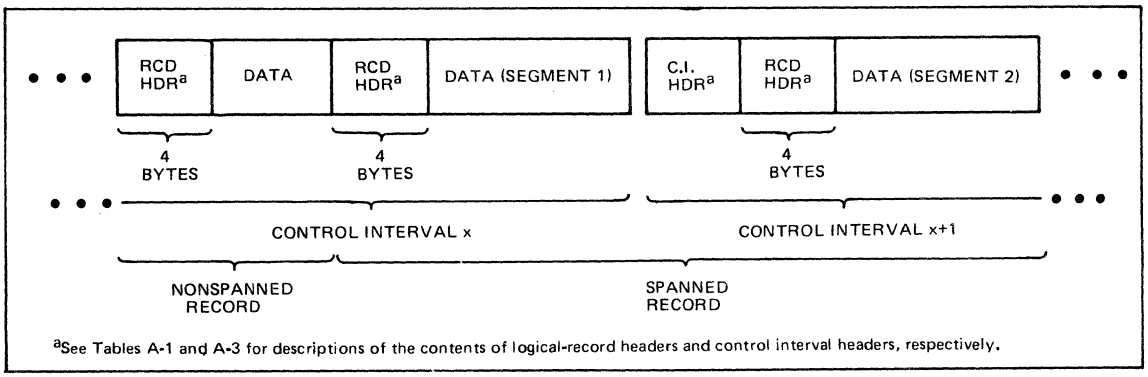


Figure 3-7. Format of Records in Disk-Resident Sequential Files

VARIABLE LENGTH RECORDS IN AN INTEGRATED I-D-S/II AREA (MOD 600 ONLY)

An integrated I-D-S/II area may contain two types of records:

- CALC records
- non-CALC records

CALC records contain a CALC key which is a user designated key field in the data portion of the record. A CALC set is zero or more records whose value has been calculated (via a hashing algorithm) to the same value. All records in a CALC set are linked by forward and backward pointers (see "Pointers" below). A CALC header record identifies the start and end of a CALC set and contains only global pointers. CALC sets may not cross area boundaries.

Non-CALC, and CALC, records are chained into I-D-S/II defined sets via local and global pointers depending on whether the records of a set cross area boundaries. If a chained record is

"owned" by another record in the set, it contains an owner pointer relating it back to the owner record. The owner record contains the forward and backward pointers for each set in which it is declared to be an owner. The pointers may be either local or global depending on whether the set crosses area boundaries.

POINTERS

Pointers are internal values established for record chaining purposes within an integrated I-D-S/II data base. A set occurrence contains one owner record and zero or more member records. The records of the set are chained using the following pointers:

- Next pointer — forms the forward chain from one record to another
- Prior pointer — forms the backward chain from one record to another
- Owner pointer — forms the chain from one record to the "owner" record in the set. Only member records contain an owner pointer.

Next and Prior pointers may be either local or global depending on whether the set is defined to cross area boundaries. Global and local pointers are defined as follows:

- Global pointer — a field that is used to link records of a set whose tenants may be in different areas. The size of the pointer field depends on the sum of all area sizes (i.e., sum of ALLOCATE clause descriptions.)
- Local pointer — a field that is used to link records of a set whose tenants are entirely contained within one area. The size of the pointer field depends on the size of the area (ALLOCATE clause).

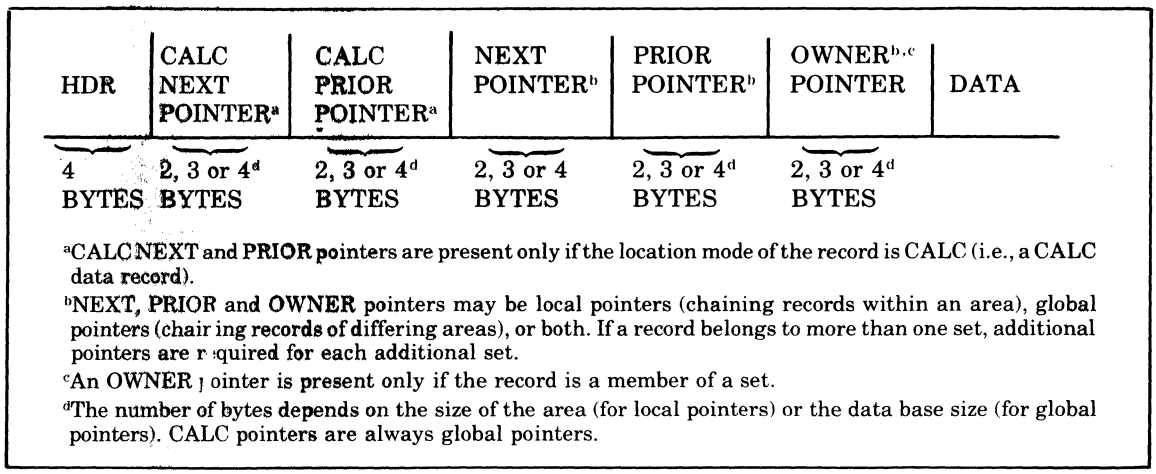
Table 3-2 can be used to determine the size of the global and local pointers.

TABLE 3-2. RELATIONSHIP BETWEEN RECORDS AND POINTER SIZE

Maximum Number of Records	Pointer Size
$2^{16} - 1$ (65,535)	2 bytes
$2^{24} - 1$ (16,777,215)	3 bytes
$2^{32} - 1$ (4,294,967,295)	4 bytes

INTEGRATED I-D-S/II RECORD FORMAT (MOD 600 ONLY)

The I-D-S/II data record contains a standard 4-byte header followed by CALC set pointers (if the location mode for this record type is CALC), intra-area and inter-area pointers (local and global pointers, respectively) and data. Figure 3-8 illustrates the format of an I-D-S/II data record.



[a]CALC NEXT and PRIOR pointers are present only if the location mode of the record is CALC (i.e., a CALC data record).

[b]NEXT, PRIOR and OWNER pointers may be local pointers (chaining records within an area), global pointers (chair ing records of differing areas), or both. If a record belongs to more than one set, additional pointers are r quired for each additional set.

[c]An OWNER j ointer is present only if the record is a member of a set.

[d]The number of bytes depends on the size of the area (for local pointers) or the data base size (for global pointers). CALC pointers are always global pointers.

Figure 3-8. Format of an Integrated I-D-S/II Data Base Record

CALC HEADER RECORD (MOD 600 ONLY)

The CALC header record, employed in the integrated I-D-S/II file organization, identifies the start and end of a CALC set. It is a special record type containing only pointers and is not directly accessible by an application program. Figure 3-9 illustrates the structure of a CALC header record.

HDR[a]	FIRST CALC GLOBAL POINTER	LAST CALC GLOBAL POINTER
4 BYTES	2, 3 or 4[b] BYTES	2, 3 or 4[b] BYTES

[a]This header is a standard UFAS record header with a special record type of 4000 decimal which identifies it as a CALC header record.

[b]The number of bytes depends on the data base size.

Figure 3-9. Format of CALC Header Record

UNIT RECORD FILE FORMATS

The following have unit record file formats:

- Card
- Printer

CARD DATA FORMAT

Data residing on cards may be read in either of two formats: verbatim or ASCII. These formats are illustrated in Figures 3-10 and 3-11, respectively. In addition, Table 3-1 shows the Hollerith and ASCII equivalence codes.

The end-of-file (EOF) indicator for a card deck consists of a card containing the ASCII characters GS punched in column 1 (the Hollerith equivalent is: 11-9-8-5); a single EOF card indicates the end of the card deck as shown in Figure 3-12.

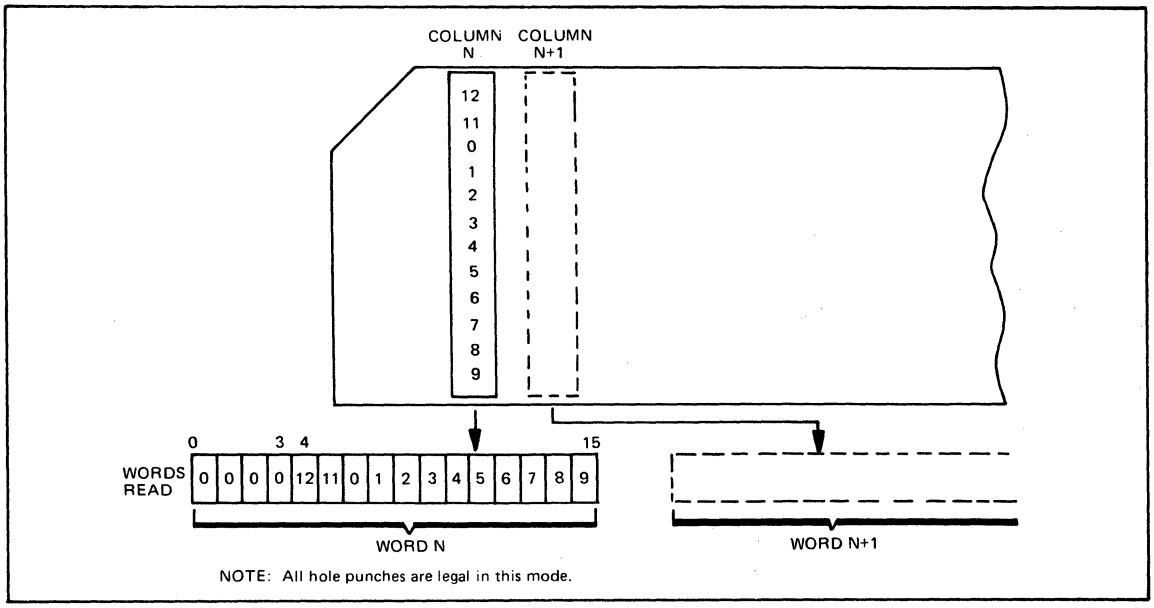


Figure 3-10. Verbatim Mode Format for Card Data

TABLE 3-3. HOLLERITH-ASCII CODE TABLE

b8 b7 b6 b5	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1	
b4b3b2b1 / COL ROW	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	COL / ROW
0000	0	NUL 12-0-9-8-1	DLE 12-11-9-8-1	SP NO PCH	0 0	∂ 8-4	P 11-7	` 8-1	p 12-11-7	11-0-9-8-1	12-11-0-9-8-1	12-0-9-1	12-11-9-8	12-11-0-9-6	12-11-8-7	12-11-0-8	12-11-9-8-4	0
0001	1	SCH 12-9-1	DC1 11-9-1	! 12-8-7	1 1	A 12-1	Q 11-8	a 12-0-1	q 12-11-8	0-9-1	9-1	12-0-9-2	11-8-1	12-11-0-9-7	11-0-8-1	12-11-0-9	12-11-9-8-5	1
0010	2	STX 12-9-2	DC2 11-9-2	" 8-7	2 2	B 12-2	R 11-9	b 12-0-2	r 12-11-9	0-9-2	11-9-8-2	12-0-9-3	11-0-9-2	12-11-0-9-8	11-0-8-2	12-11-0-8-2	12-11-9-8-6	2
0011	3	ETX 12-9-3	DC3 11-9-3	# 8-3	3 3	C 12-3	S 0-2	c 12-0-3	s 11-0-2	0-9-3	9-3	12-0-9-4	11-0-9-3	12-0-8-1	11-0-8-3	12-11-0-8-3	12-11-9-8-7	3
0100	4	ECT 9-7	DC4 9-8-4	$ 11-8-3	4 4	D 12-4	T 0-3	d 12-0-4	t 11-0-3	0-9-4	9-4	12-0-9-5	11-0-9-4	12-0-8-2	11-0-8-4	12-11-0-8-4	11-0-9-8-7	4
0101	5	ENQ 0-9-8-5	NAK 9-8-5	% 0-8-4	5 5	E 12-5	U 0-4	e 12-0-5	u 11-0-4	11-9-5	9-5	12-0-9-6	11-0-9-5	12-0-8-3	11-0-8-5	12-11-0-8-5	11-0-9-8-3	5
0110	6	ACK 0-9-8-6	SYN 9-2	& 12	6 6	F 12-6	V 0-5	f 12-0-6	v 11-0-5	12-9-6	9-6	12-0-9-7	11-0-9-6	12-0-8-4	11-0-8-6	12-11-0-8-6	11-0-9-8-4	6
0111	7	BEL 0-9-8-7	ETB 0-9-6	' 8-5	7 7	G 12-7	W 0-6	g 12-0-7	w 11-0-6	11-9-7	12-9-8	12-0-9-8	11-0-9-7	12-0-8-5	11-0-8-7	12-11-0-8-7	11-0-9-8-5	7
1000	8	BS 11-9-6	CAN 11-9-8	(12-8-5	8 8	H 12-8	X 0-7	h 12-0-8	x 11-0-7	0-9-8	9-8	12-8-1	11-0-9-8	12-0-8-6	12-11-0-8-1	12-0-9-8-2	11-0-9-8-6	8
1001	9	HT 12-9-5	EM 11-9-8-1	) 11-8-5	9 9	I 12-9	Y 0-8	i 12-0-9	y 11-0-8	0-9-8-1	9-8-1	12-11-9-1	0-8-1	12-0-8-7	12-11-0-1	12-0-9-8-3	11-0-9-8-7	9
1010	10	IF 0-9-5	SUB 9-8-7	* 11-8-4	: 8-2	J 11-1	Z 0-9	j 12-11-1	z 11-0-9	0-9-8-2	9-8-2	12-11-9-2	12-11-0	12-11-8-1	12-11-0-2	12-0-9-8-4	12-11-0-9-8-2	10
1011	11	VT 12-9-8-3	ESC 0-9-7	+ 12-8-6	; 11-8-6	K 11-2	[12-8-2	k 12-11-2	{ 12-0	0-9-8-3	9-8-3	12-11-9-3	12-11-0-9-1	12-11-8-2	12-11-0-3	12-0-9-8-5	12-11-0-9-8-3	11
1100	12	FF 12-9-8-4	FS 11-9-8-4	, 0-8-3	< 12-8-4	L 11-3	\ 0-8-2	l 12-11-3	| 12-11	0-9-8-4	12-9-4	12-11-9-4	12-11-0-9-2	12-11-8-3	12-11-0-4	12-0-9-8-6	12-11-0-9-8-4	12
1101	13	CR 12-9-8-5	GS 11-9-8-5	- 11	= 8-6	M 11-4	] 11-8-2	m 12-11-4	} 11-0	12-9-8-1	11-9-4	12-11-9-5	12-11-0-9-3	12-11-8-4	12-11-0-5	12-0-9-8-7	12-11-0-9-8-5	13
1110	14	SO 12-9-8-6	RS 11-9-8-6	. 12-8-3	> 0-8-6	N 11-5	^ 11-8-7	n 12-11-5	~ 11-0-1	12-9-8-2	9-8-6	12-11-9-6	12-11-0-9-4	12-11-8-5	12-11-0-6	12-11-9-8-2	12-11-0-9-8-6	14
1111	15	SI 12-9-8-7	US 11-9-8-7	/ 0-1	? 0-8-7	O 11-6	_ 0-8-5	o 12-11-6	DEL 12-9-7	11-9-8-3	11-0-9-1	12-11-9-7	12-11-0-9-5	12-11-8-6	12-11-0-7	12-11-9-8-3	EO 12-11-0-9-8-7	15

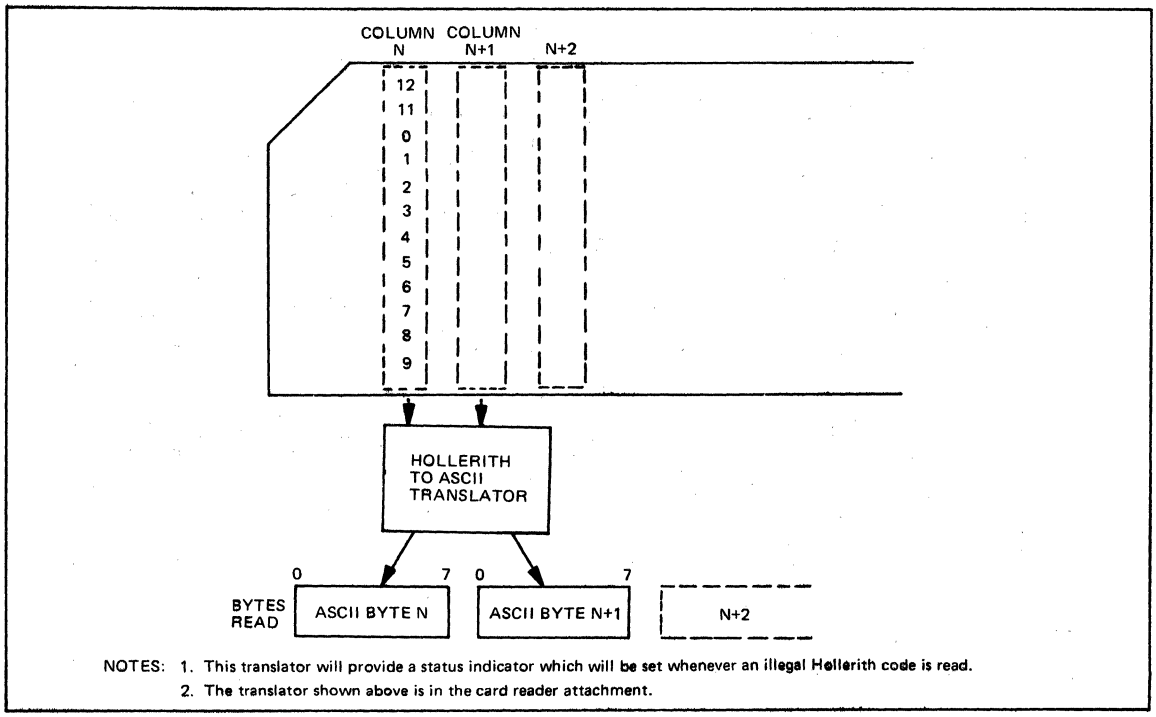


Figure 3-11. ASCII Mode Format for Card Data

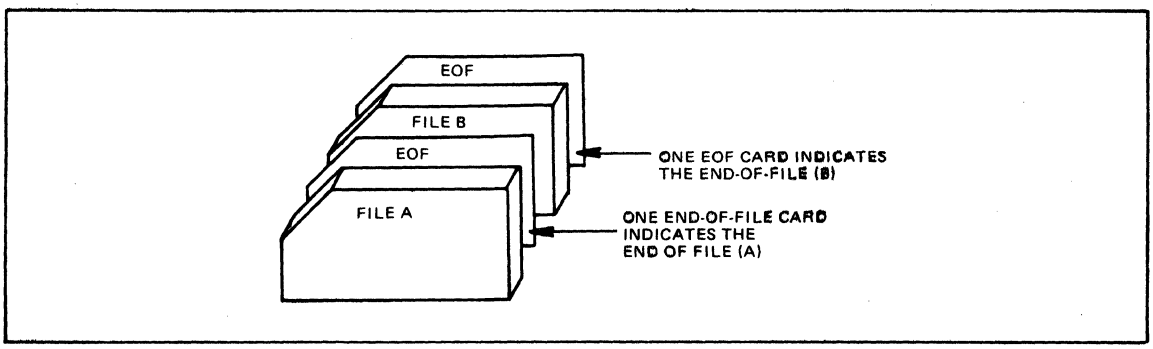


Figure 3-12. Card File Organization

PRINTER DATA FORMAT

The printer driver (refer to *System Services Macro Calls* manual) performs all data transfers to line printers or to a serial printer. Each data transfer is initiated by an individual request from a program.

The printer has two function codes: "print" and "do not print." Control of these functions is accomplished by supplying a control byte as the first entry in the data buffer to be transferred to the printer. The format of the data buffer is such that all characters except the first, i.e., the 8-bit character constituting the control byte, are interpreted as data. Figure 3-13 describes the format for constructing a printer control byte; Table 3-4 summarizes ASCII codes resulting from possible printer control byte settings and corresponding actions performed by the printer.

Note that no horizontal tab operation can be affected by tab characters (HT) embedded in the data; such characters are interpreted as nonprinting characters. To perform horizontal tab operations, the program must contain coding to effect tab control.

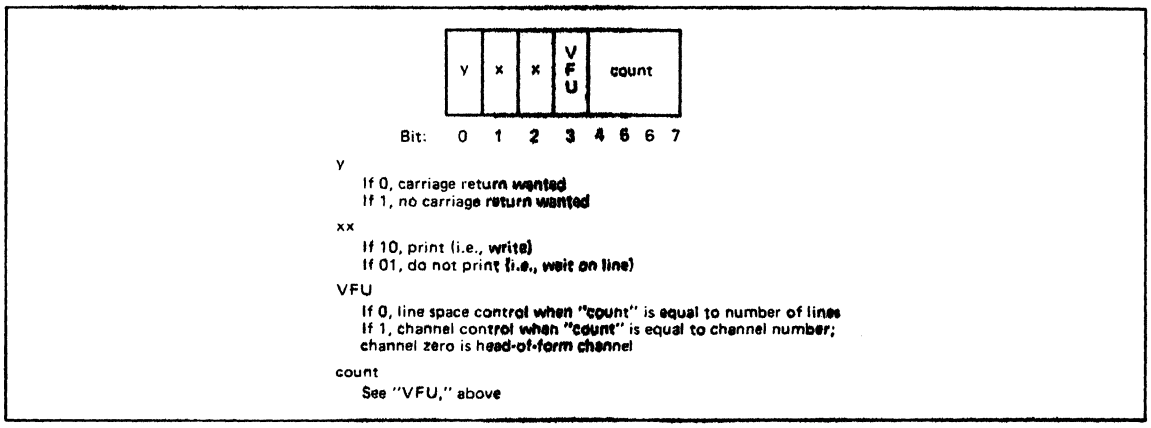


Figure 3-13. Printer Control Byte Format

TABLE 3-4. PRINTER CONTROL BYTE CODES

Hexadecimal	ASCII	Resulting Action
00 — 1F	NUL-US	Single space, then print; skip to head-of-form at end-of-form.
20 — 2F	Δ — /	Space control lines; do not print.
30 — 3F	0 — ?	Skip to VFU channel number in count; do not print.
40 — 4F	@ — O	Space count number of lines and print.
50 — 5F	P — _	Skip to VFU channel number in count and print; 50 = skip to head-of-form.
60 — 7F	/ — DEL	Reserved.

Note:

Once head-of-form is reached, the associated status bit (VFU bit 3; see Figure 3-13) is not reset to 0 until the next head-of-form request is encountered.

Section 4
File Formats

As described in Section 1, a file is a collection of one or more logically-related records. The file design process consists of selecting the file organization (as described in Section 2) that best fits your needs and available storage devices.

The following file structures are supported by the system; each is dependent upon the storage medium and file organization:

- UFAS Sequential
- UFAS Indexed
- UFAS Relative
- Fixed-Relative (BES-compatible)
- Integrated (MOD 600 only)

As described in Section 2, records are stored in logical transfer units, referred to as control intervals (on disk devices), blocks (on tape devices), and pages (in I-D-S/II). Except in the case of fixed-relative files, the system builds and maintains headers, which describe the contents of the control interval or block, based on information that you provide when creating the file and the records that you present for loading; this information is not required for fixed-relative files, because all records are fixed in length, and there are no access controls that require header information.

Following are descriptions of the various file formats, based on the file organization that you select.

UFAS SEQUENTIAL FILE FORMAT

Sequential files may be disk-resident or tape-resident (MOD 400 and MOD 600 only) and contain fixed- or variable-length records (see Figures 3-1, 3-6, and 3-7). The format of the file depends on which type of device the file resides, as described below.

DISK-RESIDENT SEQUENTIAL FILE FORMAT

As described in Section 2, a disk-resident sequential file consists of a number of control intervals, each of which contains a control-interval header built and maintained by the system, active and deleted records, and, possibly, unused (i.e., available) space at the end of the file.

Figure 4-1 illustrates the structure of a disk-resident sequential file containing a variable-length, spanned record (i.e., record 3).

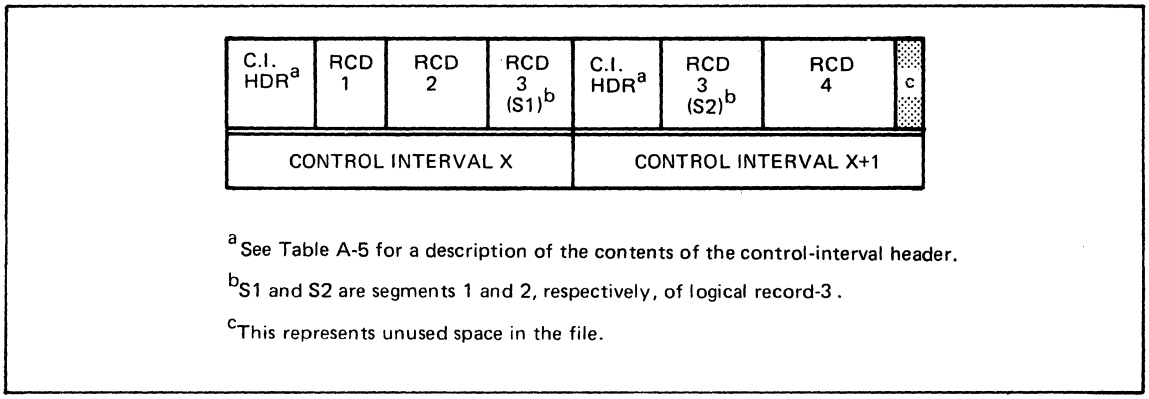


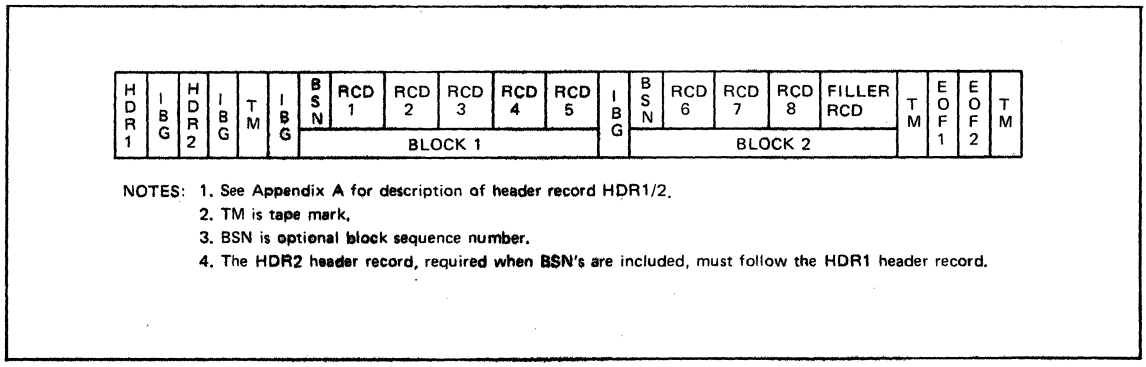
Figure 4-1. Format of Disk-Resident Sequential File

LABELED AND UNLABELED TAPE-RESIDENT SEQUENTIAL FILE FORMAT (MOD 400 AND MOD 600 ONLY)

As described in Section 2, a 9-track, tape-resident, sequential file consists of a number of blocks, each containing active records. Possibly (because of filled fixed-length records) there may be unused space at the end of the last block.

For blocked, fixed-length, records, each block will indeed contain the same number of records. Data management does load filler records into the last data block, if necessary, to accomplish this. Filler records are composed of circumflex characters (hexadecimal 5E).

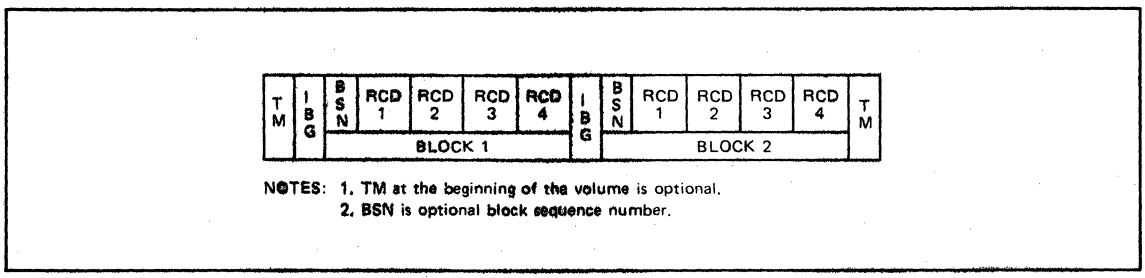
Figure 4-2 illustrates the structure of a tape-resident sequential file that occupies two blocks. Note that this file utilizes the optional block sequence number (BSN), which occupies the first 6 bytes of each block if you want to number the blocks. Also note that blocks are separated by an interblock gap (IBG).



NOTES: 1. See Appendix A for description of header record HDR1/2.
2. TM is tape mark.
3. BSN is optional block sequence number.
4. The HDR2 header record, required when BSN's are included, must follow the HDR1 header record.

Figure 4-2. Format of Tape-Resident Sequential File

Figure 4-3 illustrates the structure of unlabeled tape-resident sequential files. Note that unlabeled multivolume files are not supported.



NOTES: 1. TM at the beginning of the volume is optional.
2. BSN is optional block sequence number.

Figure 4-3. Format of Unlabeled Tape-Resident Sequential File

UFAS INDEXED FILE FORMAT

Indexed files are always disk-resident and contain fixed- or variable-length records (see Figure 3-5). As described in Section 2, indexed files consist of the following types of control intervals in a tree structure:

- Data control intervals
- Local overflow control intervals
- General overflow control intervals
- Coarse-level index control intervals
- Fine-level index control intervals
- General overflow inventory control intervals

Figure 4-4 illustrates the structure of an indexed file; the various types of control intervals are described below.

DATA CONTROL INTERVALS

Data control intervals are those control intervals that contain data records. As illustrated in Figure 4-4, they consist of a control-interval header, a link-record header, data area, line-off-setarray area, and possibly some unused (i.e., free) space.

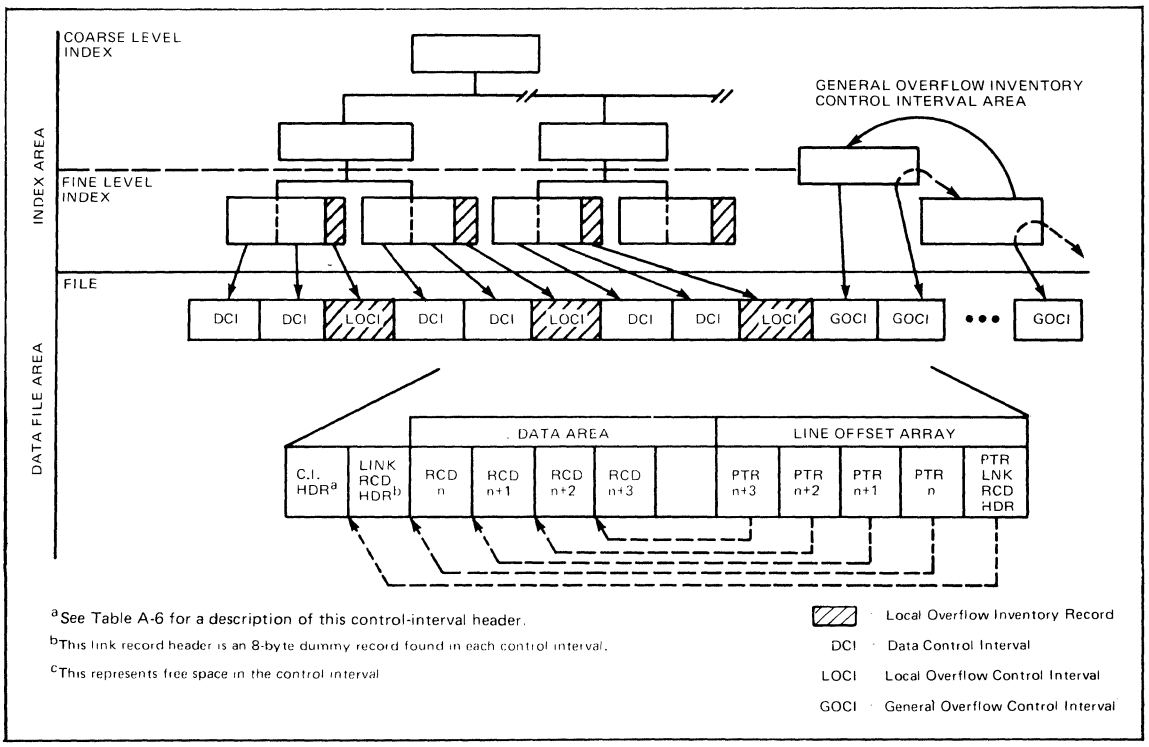


Figure 4-4. Format of UFAS Indexed File

The control-interval header, which is built and maintained by the system, is described in Table A-6 of Appendix A. The data area contains the records that you write in the file. The link-record header is an 8-byte dummy record, required and built by the system, that immediately follows the control-interval header. The line-offset-array area contains one 2-byte pointer (specifying the byte offset from the beginning of the control interval to the beginning of the record) to each record in the control interval; the pointers are built in line-number order from the end of the control interval. Space can be left in each control interval to allow for the insertion of new records.

Every local overflow control interval is normally associated with (available for overflow use by) several data control intervals. Also, a given data control interval can have more than one local overflow control interval available to it for overflow use.

LOCAL OVERFLOW CONTROL INTERVALS

Local overflow control intervals are interspersed among data control intervals at regular intervals. When the file is loaded, they are allocated at intervals specified when the file is created. Inventory entries in the fine-level index reflect the space available in local overflow control intervals.

Records are written into local overflow control intervals only after the data control interval that should contain the new record is filled.,

GENERAL OVERFLOW CONTROL INTERVALS

General overflow control intervals reside at the end of your file. They are allocated at file-load time and consist of the number of control intervals constituting the difference between the last data control interval and the maximum number of control intervals allowed for the file (as specified at file-creation time).

Records are written in them only after the data control interval that should contain the new record (based on its key value) and any available local overflow control intervals are filled. Entries in the system-built and maintained general overflow inventory control interval (described below) continuously reflect the amount of available space in each general overflow control interval.

COARSE-LEVEL INDEX CONTROL INTERVALS

Coarse-level index control intervals are system-built; they identify the highest key value contained in each immediately subordinate (fine-level or coarse-level) index control interval. (See Figure 4-4.) Each coarse-level index control interval consists of a control interval header (see Table A-7 in Appendix A), an index area (containing a 4-byte control-interval number following the highest primary key value in a record written in the associated control interval), and unused space.

FINE-LEVEL INDEX CONTROL INTERVALS

Fine-level index control intervals are also system-built control intervals that identify the highest primary key value permitted in each data control interval (i.e., the lowest key value in the next higher data control interval).

This control interval consists of a control-interval header (see Table A-8 in Appendix A), an index area, an inventory area, and unused space.

The index area contains the value of the highest primary key that a record can contain and still be written in a specific data control interval; this information is required by the system to ensure that new records are inserted in the appropriate data control interval. The inventory area contains information about available space in associated local overflow control intervals.

GENERAL OVERFLOW INVENTORY CONTROL INTERVALS

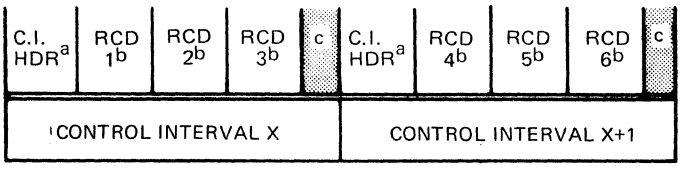
General overflow inventory control intervals are system-built and system-maintained control intervals that monitor the amount of available space in each general overflow control interval.

These control intervals consist of a control-interval header (see Table A-9 in Appendix A), an inventory area (that identifies the number of each general overflow control interval and the amount of space available in it), and unused space.

UFAS RELATIVE FILE FORMAT

Relative files are always disk-resident and contain fixed- or variable-length records (see Figures 3-3 and 3-4). As described in Section 2, the file consists of a number of control intervals each of which contains a control-interval header built and maintained by the system, active and deleted records, and possibly some unused (i.e., available) space at the end of each control interval in the file.

Figure 4-5 illustrates the structure of a relative file that occupies two control intervals.



[a] See Table A-5 for a description of the contents of the control-interval header.

[b] Although records are shown as fixed-length, they may, in fact, be variable-length (see Figure 3-4).

[c] This represents unused space in the control interval.

Figure 4-5. Format of UFAS Relative File

FIXED-RELATIVE FILE FORMAT (NON-UFAS)

Fixed-relative files are always disk-resident and contain only fixed-length records (see Figures 3-1 and 3-2). Records are stored in the same sequence in which they are presented, starting at the beginning of a physical sector (which, in this system, is coincident with a control interval).

Figure 4-6 illustrates the structure of a fixed-relative file; note that since fixed-relative files are compatible with BES files, control interval size is coincident with the physical sector size of the device on which the file is stored (i.e., 128-byte sector on diskette, and 256-byte sector on cartridge disk).

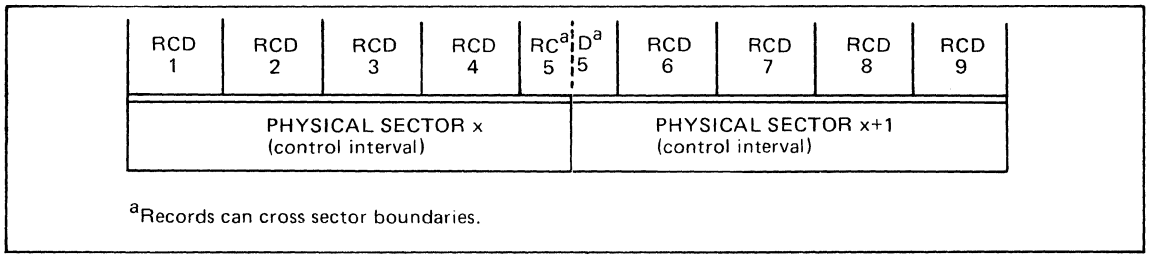


Figure 4-6. Format of Fixed-Relative File (Non-UFAS)

INTEGRATED I-D-S/II AREA FORMAT (MOD 600 ONLY)

An integrated I-D-S/II area is disk resident and employs a random storage method of related data. Data records are grouped into pages (control intervals) capable of containing 256 lines per page. An area may contain up to 16,777,216 pages. Data records may be chained into sets of related records via a pointer chaining technique.

An area may contain fixed or variable length records. Note that there is a slight deviation from the standard variable length record. An area may contain several types of records of varying lengths, but the length of each particular record is fixed. For example, an area may contain three record types such as employee name which is 24 bytes, salary which is 8 bytes, and job history which is 132 bytes.

Figure 4-7 illustrates the structure of an integrated I-D-S/II area.

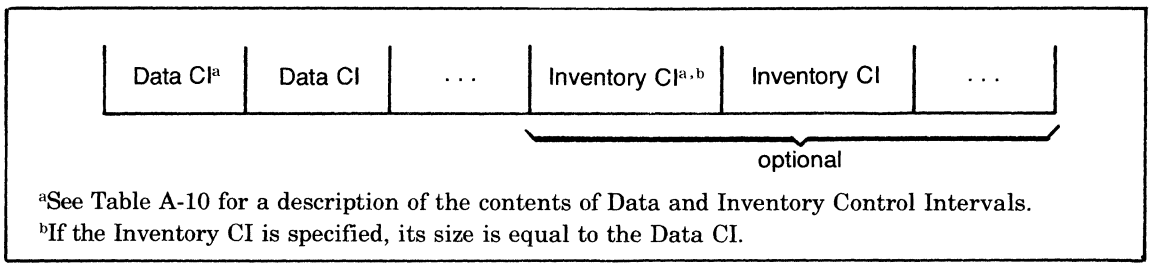


Figure 4-7. Format of Integrated I-D-S/II Area

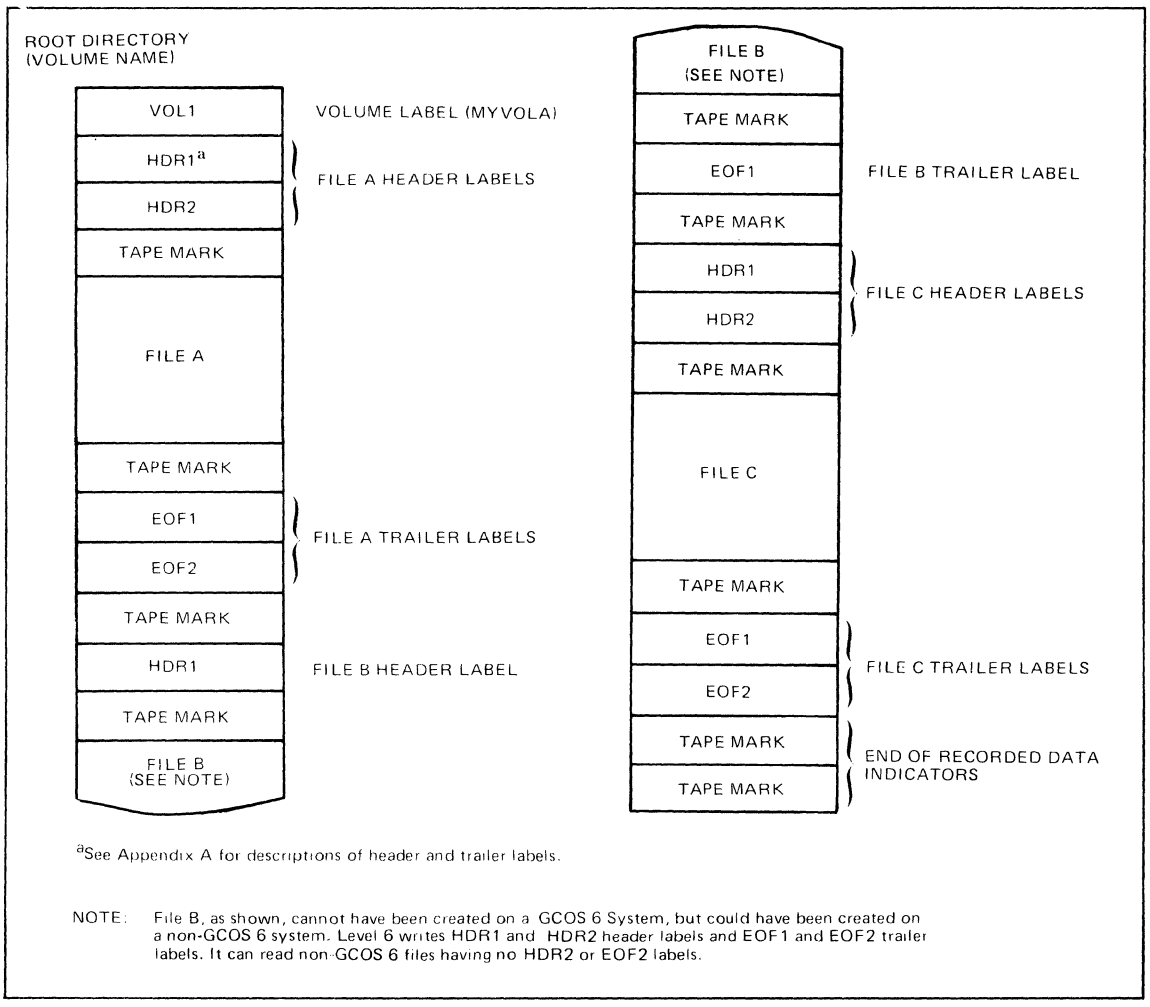
Section 5
Volume Formats

The format of a volume is dependent upon the device on which it resides. As described in Section 1, tape- and disk-resident volumes can contain one or more files.

Regardless of the device, the volume contains a main directory (i.e., root directory) which contains the location, size, attributes, etc., of each immediately subordinate directory or file; in addition, the name of the root directory is the name of the volume.

TAPE-RESIDENT VOLUME FORMAT (MOD 400 AND MOD 600 ONLY)

Nine-Track tape-resident volumes contain a root directory and one or more data files. Each file in the volume is followed by an end-of-file (EOF1) trailer label, which indicates the end-of-volume; however, if there is an HDR2 header label, an EOF2 trailer label must also be present. As illustrated in Figure 5-1, each file in a multifile volume must begin with an HDR1 (or with an HDR1 and HDR2) header label and must end with an EOF1 (or EOF1 and EOF2) trailer label. HDR2 header labels (and their corresponding EOF2 trailer labels) are required when the blocks in a file contain block sequence numbers (BSNs).



[a]See Appendix A for descriptions of header and trailer labels.

NOTE: File B, as shown, cannot have been created on a GCOS 6 System, but could have been created on a non-GCOS 6 system. Level 6 writes HDR1 and HDR2 header labels and EOF1 and EOF2 trailer labels. It can read non-GCOS 6 files having no HDR2 or EOF2 labels.

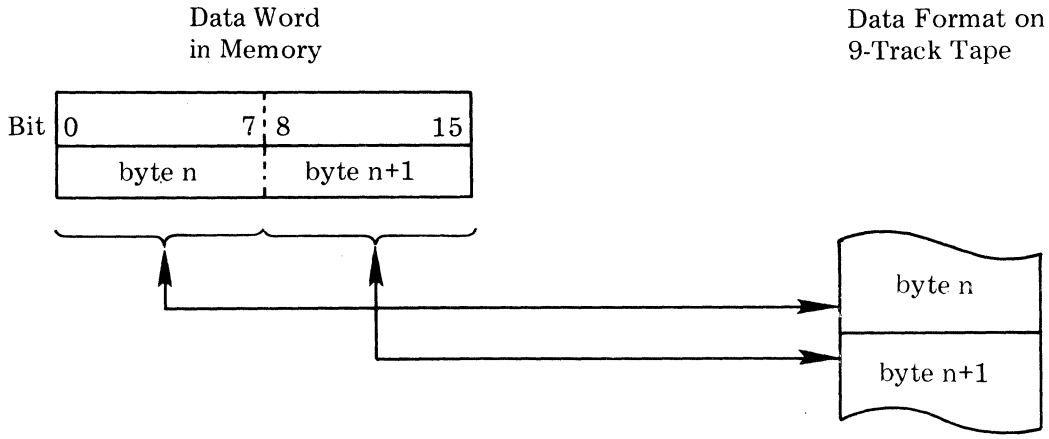
Figure 5-1. Sample Format of Tape-Resident Multifile Volume

PHYSICAL LAYOUT OF DATA ON 7- AND 9-TRACK MAGNETIC TAPE

Since magnetic tape is a well-recognized data interchange medium, it is frequently useful to know the actual physical layout of data characters on 7- and 9-track magnetic tape and how they are mapped between tape and CP memory. The following information describes both 9-track tape data transfers and 7-track tape data transfers in normal and packed modes. In 9-track tape operations, data is transferred in 8-bit bytes (equal to one tape frame) in odd parity only. For 7-track tape operations, data is transferred in one of two modes: normal mode, in which data is transferred in even or odd parity; and packed mode, in which data is transferred only in odd parity. In all cases, parity is checked on read operations.

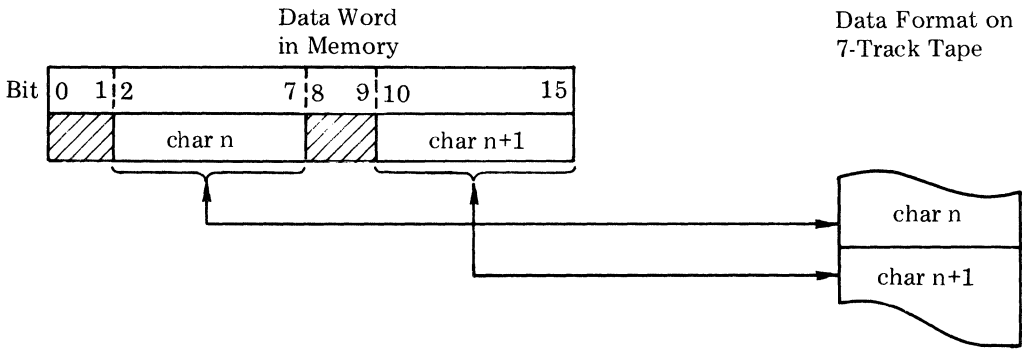
9-TRACK TAPE DATA TRANSFER

For 9-track tape, all 16 bits of a data word are transferred to or from tape in the following manner:



7-TRACK TAPE DATA TRANSFER: NORMAL MODE

For 7-track tapes written in normal mode, 12 bits of the data word are transferred to or from tape as shown in the following diagram. The four nonsignificant bits (i.e., bits 0, 1, 8 and 9) are ignored on write operations and are zero-filled on read operations.

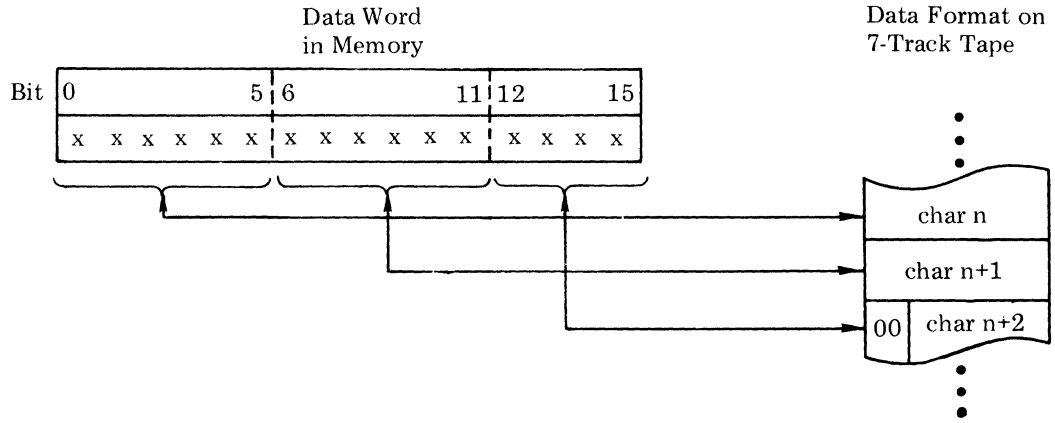


Data transfers in even parity mode include the following translation of data written to or read from tape:

- A 00_8 character from memory is written on tape as a 12_8 character with even parity.
- A 12_8 character read from tape is transferred to memory as a 00_8 character.

7-TRACK TAPE DATA TRANSFER: PACKED MODE

Packed mode is an additional mode of tape operation available only on 7-track tape devices. In this mode, data is transferred between memory and tape in 6-6-4 format, as follows:



DISK-RESIDENT VOLUME FORMAT

Each disk-resident volume contains a root directory, one or more files, and/or one or more levels of subordinate directories. Figure 5-2 illustrates the structure of a multifile volume.

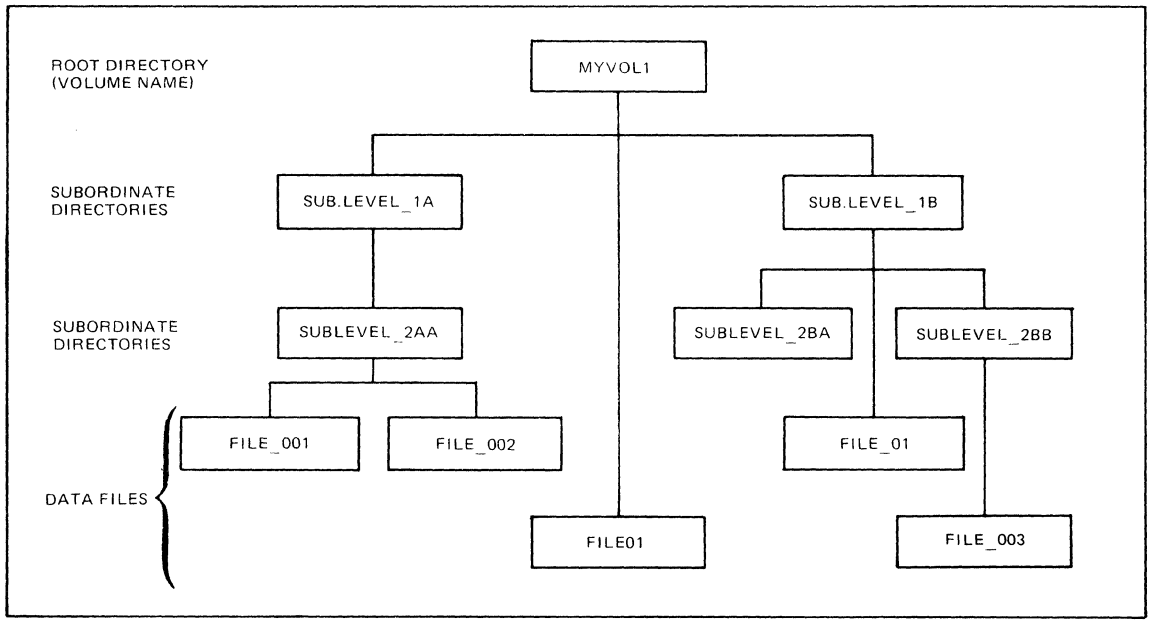


Figure 5-2. Sample Format of Disk-Resident Volume

As illustrated in this figure, each level in the hierarchy can identify and describe the subdirectories and/or data files immediately subordinate to it; there is no limit to the number of levels in a volume, but data files cannot be subordinate to other data files. Similarly, only one directory can point to a given file.

Each task group (user) is associated with a directory referred to as the working directory. It begins with a root directory name and contains zero or more intermediate directory names. It is used by the File System software to construct a full pathname whenever a task group refers to a relative or simple pathname. For example, if you were currently using MYVOL1>SUB.LEVEL_1B (see Figure 5-2) as the working directory and want to access FILE_001, use the following relative pathname:

<SUB.LEVEL_1A>SUBLEVEL_2AA>FILE_001

Similarly, you can access the file with the following absolute pathnames:

MYVOL1>SUB.LEVEL_1A>SUBLEVEL_2AA>FILE_001

The circumflex (^) at the beginning of the pathname indicates that you are starting from the root directory to locate the desired file.

The disk volume header or volume label ID (MYVOL1) resides in physical sector 7 bytes 0 through 3 of the disk.

The basic structure of a disk volume is established when the volume is initialized; after initialization is completed, you can create, add, or delete as many levels of directories and/or files as you desire, provided the pathname does not exceed 57 characters (except for the volume name, which can contain only six characters, each file and directory name can contain up to 12 characters). Additional information about the hierarchical structure of disk volumes (e.g., naming conventions, pathnames, etc.) is provided in the *Operator's Guide* manual; Table A-12 in Appendix A describes the contents of directories.

Appendix A
Disk/Tape Headers and Tape Trailers

This appendix lists and describes the contents of headers for each of the following levels:

- Logical record
- Control interval or block
- File
- Volume

and the contents of tape-volume trailers.

The system builds and maintains the information in these headers, based on information that you supply when creating and using data files. It is described here only for informational purposes (e.g., to be used for debugging).

The file and volume headers are described according to the device on which they reside (i.e., disk- or tape-resident); other headers are described according to the file organization with which they are associated, as follows:

- UFAS Relative
- UFAS Sequential
- UFAS Indexed
- UFAS Integrated (MOD 600 only)

Note that the concept of control intervals does not apply to fixed-relative file organizations. The only time a header is required is when deletable records are allowed (see Figure 3-2); therefore, fixed-relative is not included at the record or control-interval level in this appendix.

LOGICAL-RECORD HEADERS

Tables A-1 and A-2 describe the contents of logical-record headers for records stored in relative, disk-resident sequential, and indexed files; records stored in fixed-relative and tape-sequential files may or may not have headers, as described in Section 3.

TABLE A-1. LOGICAL-RECORD HEADER FOR RECORDS STORED IN RELATIVE OR DISK-RESIDENT SEQUENTIAL FILES

Field Name	Size (bytes)	Description
Status/Type Indicators	2	Bits 0-2 — Reserved; must contain zeros. Bit 3 — Record Status, as follows: 0 = Record is inactive (i.e., physically deleted) 1 = Record is active If this is a spanned record, this bit is set only in the first segment. 4-15 — Reserved; must contain zeros
Spanning/Size Information	2	Bits 0-1 — Spanning Indicator, as follows: 00 = This record is *not* a spanned record 01 = This is the first segment of a spanned record 10 = This is the last segment of a spanned record 11 = This is an intermediate segment of a spanned record Bits 2-16 — Record/Segment Size, as follows: Size, in bytes, of this record or segment, including the 4-byte header.

TABLE A-2. LOGICAL-RECORD HEADER FOR RECORDS STORED IN INDEXED FILES

Field Name	Size (bytes)	Description
Status/type Indicator	2	Reserved; must contain zeros.
Size	2	Bits 0-1 — Reserved; must contain zeros. Bits 2-15 — Record Size, as follows: Size, in bytes, of this record, including the 8-byte header.
Next Control-Interval Identifier	3	The number of the control interval containing the next logical record in primary key sequence; contains zeros for the last record in the file.
Next Line Identifier	1	Line number of the next logical record in primary key sequence; contains zeros for the last record in the file.

TABLE A-3. RECORD HEADER FOR I-D-S/II INTEGRATED AREAS (MOD 600 ONLY)

Field Name	Size (bytes)	Description
Status/Type Indicator	2	Bits 0-3 — Reserved; must contain zeros. Bits 4-15 — Indicates the type of record in the file: 0-3999 decimal for data record 4000 decimal for CALC header record 4001-4095 decimal reserved
Spanning/Size Information	2	Bits 0-1 — Spanning indicator as follows: 00 = This record is not a spanned record. Bits 2-16 — The size, in bytes, of the logical record header, set pointers, and data portion.
CALC Set Pointers:		CALC set pointers are present if location mode is VIA CALC.
CALC Next Pointer	global pointer	The data base key of the next record in the CALC set. This field in the last record in a CALC set points to the CALC header record.
CALC Prior Pointer	global pointer	The data base key of the prior record in the CALC set. This field for the first record in a CALC set points to the CALC header record.

TABLE A-4. CALC HEADER RECORD FOR I-D-S/II INTEGRATED AREAS (MOD 600 ONLY)

Field Name	Size (bytes)	Description
Status/Type Indicator	2	Bits 0-3 — Reserved; must contain zeros. Bits 4-15 — 4000 decimal indicates a CALC header record.
Spanning/Size Information	2	00 = Indicates the CALC header record starts and ends in the same CI. Bits 2-16 — Record size, in bytes, of this record header.
Address of First CALC Record	global pointer	The data base key of the first CALC record in the CALC set.
Address of Last CALC Record	global pointer	The data base key of the last CALC record in the CALC set.

CONTROL INTERVAL AND BLOCK HEADERS

Tables A-5 through A-11 describe the contents of control-interval and block headers for relative, sequential (both disk- and tape-resident) integrated I-D-S/II areas and indexed files; fixed-relative files have fixed length control intervals and, therefore, do not have control-interval headers.

TABLE A-5. CONTROL-INTERVAL HEADER FOR RELATIVE OR DISK-RESIDENT SEQUENTIAL FILES

Field Name	Size (bytes)	Description
Active Control-Interval Size	2	The size, in bytes, of the active control interval is the sum of: This 8-byte control-interval header All active and deleted records in this control interval This size does not include unused space at the end of the control interval.
Total Space Available	2	The size, in bytes, of available space in the control interval is the sum of: Unused space at the end of the control interval Space occupied by deleted records in the control interval
Control-Interval Number	4	The relative number (i.e., physical position) of this control interval in the file; the first control interval is 1.

TABLE A-6. CONTROL-INTERVAL HEADER FOR INDEXED FILES — DATA AND OVERFLOW CONTROL INTERVALS

Field Name	Size (bytes)	Description
Active Control-Interval Size	2	The size, in bytes, of the active control interval is the sum of: This 12-byte control-interval header The link-record header All active records in this control interval Unused space between the last record and the line-offset-array area The line-offset-array area
Total Space Available	2	The size, in bytes, of available space in the control interval is the sum of: Unused space between the last record and the line-offset-array area Space occupied by inactive (i.e., physically deleted) records
Control-Interval Number	4	The relative number (i.e., physical position) of this control interval in the file; the first control interval is 1.
Lowest Available Line Number	2	The numerically lowest line number in this control interval that is available for use. The line number either has never been used or belonged to a record that was physically deleted. If no line numbers are available from those currently allocated (as specified by "Current High Line Number," below), this field contains zeros. If there are no deleted records, this field contains zeros.
Current High Line Number	2	The numerically highest line number in the control interval line number that is allocated to an active record.
Data Records		
Available Space		
Record Offset	2	Contains the offset from the beginning of the control interval to a record with the line number corresponding to the relative position of the descriptor from the end of the control interval.

TABLE A-7. CONTROL-INTERVAL HEADER FOR INDEXED FILES — COARSE-LEVEL INDEX CONTROL INTERVAL

Field Name	Size (bytes)	Description
Active Control-Interval Size	2	The size, in bytes, of the active control interval is the sum of: This 12-byte control-interval header All index entries in the control interval
Total Space Available	2	The size, in bytes, of unused space in the control interval.
Control-Interval Number	4	The relative number (i.e., physical position) of this control interval in the file; the first control interval is 1.
Reserved	4	This field must contain zeros.

TABLE A-8. CONTROL-INTERVAL HEADER FOR INDEXED FILES — FINE-LEVEL INDEX CONTROL INTERVAL

Field Name	Size (bytes)	Description
Active Control-Interval Size	2	The size, in bytes, of the active control interval is the sum of: This 12-byte control-interval header All index entries in the control interval All inventory entries in the control interval Unused space between the index entries and the inventory entries
Total Space Available	2	The size, in bytes, of unused space in the control interval.
Control-Interval Number	4	The relative number (i.e., physical position) of this control interval in the file; the first control interval is 1.
End Inventory Location	2	The offset, in bytes, from the beginning of the control interval to the first entry in the inventory area (i.e., the last entry in the inventory).
Number of Index Entries	2	The number of index entries or primary key values.

TABLE A-9. CONTROL-INTERVAL HEADER FOR INDEXED FILES — GENERAL OVERFLOW INVENTORY CONTROL INTERVALS

Field Name	Size (bytes)	Description
Active Control-Interval Size	2	The size, in bytes, of the active control interval is the sum of: This 12-byte control-interval header All inventory entries in the control interval
Total Space Available	2	The size, in bytes, of unused space at the end of the control interval.
Control-Interval Number	4	The relative number (i.e., physical position) of this control interval in the file; the first control interval is 1.
Next Inventory Control-Interval Number	4	The relative number of the next control interval containing general overflow inventory information.

TABLE A-10. DATA CONTROL INTERVAL HEADER FOR I-D-S/II INTEGRATED AREAS (MOD 600 ONLY)

Field Name	Size (bytes)	Description
CI Size	2	The size of the control interval which includes the space occupied by the CI header, active records, unused space in the middle of the CI (before the line offset array), and the line offset array. Since the line offset array is located at the end of the CI, the active CI size is the same as physical CI size.
Total Space Available	2	The size of the unused space. This is the space between the last active record and the beginning of the line offset array.
CI Number	4	The relative number (physical position) of this CI in the file.
Lowest Available Line Number	2	The numerically lowest line number in this CI that is available for use. The line number has either never been used or belonged to a record that was physically deleted. If no line numbers are available from those currently allocated (as specified by 'current high line number'), this field is set to zero. If the lowest available line number is zero and line 0 is not available, no lines are available in this CI.
Current High Line Number	2	The numerically highest line number in the CI that is allocated to an active record. As records are inserted into a CI, additional line numbers may be allocated, new line offset array entries are created and the value for "current high line number" is incremented. Likewise, as the highest allocated line numbers in a CI are deleted, the value for "current high line number" is decremented so that it describes the highest active line available. At least one line number must be in the array but it may be available.
Data Records		
Available Space		
Record Offset Descriptor	2	If the line number is in use, this field contains the offset from the beginning of the control interval to a record. The line number of this record corresponds to the relative position of the descriptor from the end of the control interval. If the line number is available, this field contains zeros.
Inventory Control Interval	1	Each byte controls one data CI. It gives the number of inventory units available in the corresponding data CI.

TABLE A-11. BLOCK HEADER FOR TAPE-RESIDENT SEQUENTIAL FILES

Field Name	Size (bytes)	Description
Block-Sequence Number	6	Relative Number (i.e., physical position) of this block in the file; the first block is 1. Block sequence numbers are present if a HDR2 file header follows the HDR1 file header and the buffer offset length is specified in HDR2. If there is no file header, the blocks are not numbered.
Record Control Word	4	The record header for tape files containing variable length records. It is the decimal count of the number of bytes in the record.

FILE HEADERS

Table A-12 describes the contents of an entry in a directory. Each directory, regardless of its position in the hierarchy, contains an entry in this format for each immediately subordinate directory or file.

TABLE A-12. DIRECTORY ENTRY FOR DISK-RESIDENT FILES/DIRECTORIES

Field Name	Size (bytes)	Description
File/Directory Name	12	1- to 12-byte ASCII-character name of a file or immediately subordinate directory; the first character must be a $ or alphabetic character. The $ character should not be used in a file or directory name. This is to insure ease in transmitting files between a Level 6 and Level 66.
File Type	2	The file type of the immediately subordinate directory or file. Z'2620' = Directory Z'0702' = Static fixed-relative file without deletable records Z'0782' = Dynamic (expandable) fixed-relative file without deletable records Z'0F05' = Static fixed-relative file with deletable records Z'0F85' = Dynamic fixed-relative file with deletable records Z'4FB1' = UFAS sequential file Z'4FB2' = UFAS relative file Z'4FB3' = Data portion of a UFAS indexed file Z'4FB4' = Index portion of a UFAS indexed file Z'4FBA' = UFAS I-D-S/II integrated area (MOD 600 only)
Record Size	2	Maximum size, in bytes, of a logical record, excluding the logical-record header.
Relative End-of-Data/Logical End-of-File and Control-Interval Size	4	*For fixed-relative files,* specifies net number of bytes of valid data in the file. *For all other files* specifies logical end-of-data and control interval size, as follows: First three bytes specify relative number of last control interval in the file Last byte specifies control-interval size as a multiple of 256 (e.g., 1F = 7936 bytes)
First Extent Location	2	Specifies the number of the first sector in the extent; physical sector number if a diskette or cartridge; logical sector number if a storage module.
Volume Number/First Extent	2	Specifies the size of the first extent, as follows: Bits 0-2 — Must contain zeros Bits 3-15 — Specifies the number of sectors in the first extent; the number of physical sectors for a diskette or cartridge; the number of logical sectors for a storage module.
Second Extent Location and Size	4	Same as "First Extent Location" and "Volume Number/First Extent," above. Zeros in this field indicate that a second extent is not assigned.
Third Extent Location and Size	4	Same as "First Extent Location" and "Volume Number/First Extent," above. Zeros in this field indicate that a third extent is not assigned. The first two bytes of this entry identify the relative record number of an additional information record if this field contains Z'nnnn0000' (see Tables A-13 through A-15).

TABLE A-13. ADDITIONAL INFORMATION RECORD — REMOTE EXTENTS

Field Name	Size (bytes)	Description
ID Code	2	Must contain Z'FFFE'.
Reserved	2	Must contain zeros.
First Extent Location and Size	4	Same as "First Extent Location" and "Volume Number/First Extent fields in Table A-12.
Second Extent Location and Size	4	
Third Extent Location and Size	4	Zeros in any of these fields indicate that the extent is not assigned.
Fourth Extent Location and Size	4	
Fifth Extent Location and Size	4	
Six Extent Location and Size	4	
Seventh Extent Location and Size	4	Same as above, except for the following: The first 2 bytes of this entry identify the relative record number of another additional information record if this field contains Z'nnnn0000' (see Tables A-13 through A-15).

TABLE A-14. ADDITIONAL INFORMATION RECORD — UFAS INDEX FILE INFORMATION

Field Name	Size (bytes)	Description
ID Code	2	Must contain Z'FFFD'.
Current Size	2	Specifies the number of logical sectors currently allocated to the file.
Increment Size	2	The number of additional logical sectors to be allocated whenever it becomes necessary to do so; this is the size of an additional extent.
Maximum Size	2	Specifies the maximum number of logical sectors that can be allocated to the file (i.e., the maximum file size).
Link to ACL Record	2	Relative record number of the first Access Control List Record (ID code = 'FFFA') which contains part or all of Access Control List (ACL) for this file.
Reserved	2	Must contain zeros.
Key Type	1	The data type of the primary key. Z'00' = ASCII data 'C' Z'10' = Decimal data 'D'
Key Size	1	The size, in bytes, of the primary key.
Key Location	2	The offset, in bytes, from the beginning of each logical record in the file to the beginning of the primary key; the first record position is 0. Must contain zeros.
Start of General Overflow or Start of General Overflow Inventory	4	The relative number (starting with 1) of the first general overflow control interval (for the data portion) or the relative number of the general overflow inventory control interval (for the index portion). Must contain zeros.

TABLE A-14 (CONT). ADDITIONAL INFORMATION RECORD — UFAS INDEX FILE INFORMATION

Field Name	Size (bytes)	Description
Free Space per Data Control Interval or Number of Index Levels	2	The number of bytes to be left free at file load time so that records can be inserted without using an overflow control interval (for the data portion) or the number of levels comprising the index (for the index portion). Must contain zeros.
Local Overflow Increment	1	A local overflow control interval is allocated when the file is loaded, after this number of data control intervals, minus 1, are loaded. Must contain zeros.
Inventory Unit Size	1	A value equal to the maximum record size divided by 256 and rounded up to the next higher integer; this value is used to keep track of available space in overflow control intervals. Must contain zeros.
Top Control Interval of Index	4	The number of the control interval of the highest level of the hierarchical index structure. Must contain zeros.
Next Record Link	2	Specifies the relative record number of the next additional information record in the chain.
Reserved	2	Must contain zeros.

TABLE A-15. ADDITIONAL INFORMATION RECORD — FIXED RELATIVE, UFAS SEQUENTIAL AND UFAS RELATIVE FILES

Field Name	Size (bytes)	Description
ID Code	2	Must contain Z'FFFD'.
Current Size	2	The number of logical sectors allocated to the file.
Increment Size	2	The number of additional logical sectors to be allocated whenever it becomes necessary; this is the size of an additional extent.
Maximum Size	2	The maximum number of logical sectors that can be allocated to the file.
Link to ACL Record	2	Relative record number of the first access control list record (ID code = FFFA) that contains part or all of the access control list (ACL) for the file or directory.
Reserved	18	Must be zeros.
Next Link Record	2	The relative record number of the next additional record in the chain.
Reserved	2	Must be zeros.

TABLE A-16. ADDITIONAL INFORMATION RECORD — UFAS I-D-S/II INTEGRATED AREAS (MOD 600 ONLY)

Field Name	Size (bytes)	Description
ID Code	2	Must contain Z'FFFD'.
Current Size	2	Specifies the number of logical sectors currently allocated to the file.
Increment Size	2	The number of additional logical sectors to be allocated whenever it becomes necessary to do so; this is the size of an additional extent.
Maximum Size	2	Specifies the maximum number of logical sectors that can be allocated to the file (i.e., maximum file size).

TABLE A-16 (CONT). ADDITIONAL INFORMATION RECORD — UFAS I-D-S/II INTEGRATED AREAS (MOD 600 ONLY)

Field Name	Size (bytes)	Description
Link to ACL Record	2	Relative record number of the first Access Control List record (ID code='FFFA') which contains part or all of the Access Control List (ACL) for this file.
Reserved	2	Must contain zeros.
Global Pointer Size	1	Bit 0-7 = Global pointer size
Inventory Per Cent	1	One hundred minus the inventory threshold specified at create file time.
Global Pointer Base	4	The value added to a UFAS I-D-S/II Area relative record number to calculate a data base (global) record number. UFAS calc files must be zeros.
Number of Control Intervals	4	The number of data control intervals in the file.
CALC Interval	1	The number of DB keys between CALC header records stored as a power of 2 (i.e., a value of 0 represents the number of CALC sets in the area).
Inventory Unit	1	A value equal to: $\frac{\text{size} \times 100 - \text{inventory threshold}}{25{,}600}$ The value is rounded to the next highest integer. The values stored in the inventory unit monitor available space in the data control intervals.
Number of Record Descriptions	2	The number of record descriptions contained in the file description.
Number of Key Components	2	The total number of key components contained in the file description.
Next Record Link	2	The relative record number of the next additional information record in the chain.
Number of Records per Control Interval	2	The maximum number of data records in a control interval.

TABLE A-17. ADDITIONAL INFORMATION RECORDS — RECORD DESCRIPTORS

Field Name	Size (bytes)	Description
ID Code	2	Must contain Z'FFFC'.
Record Descriptor Information	26	Since record descriptors are of variable length and densely packed, the start of a particular descriptor cannot be found at a fixed location within an additional information record. Fields within a record descriptor are at fixed locations from the start of the descriptor. Refer to Table A-18 for details concerning the contents of a record descriptor.
Next Record Link	2	Specifies the relative record number of the next additional information record in the chain.
Record Descriptor Information	2	A continuation of the record descriptor information above.

Note:
This directory record exists for UFAS calc and I-D-S/II files. There exists as many of these records as necessary to contain all the record descriptors.

TABLE A-18. RECORD DESCRIPTOR

Field Name	Size (bytes)	Description
Record Type	2	The record type that uniquely identifies the records described by this record descriptor (must be in the range of 0 — 3999_{10}).
Number of Key Components	1	The number of components in a key of a record type containing a key. This value determines the number of key component descriptors contained in the record descriptor.
Reserved	1	Must be zero.
Number of Calc Header Records	4	If the record type contains a key, the prime number of calc header records contained in the record address range specified for this record type; zero if the record type does not contain a key.
Minimum Record Address	4	A record address expressed as relative record number that specifies the minimum range for placement of records of this record type in the file.
Maximum Record Address	4	A record address expressed as relative record number that specifies the maximum range for placement of records of this record type in the file.
Key Component Data Type	1	The data type of this key component: Z'00' = ASCII data — 'C' Z'10' = Unpacked decimal data — 'D' Z'20' = Signed binary data — 'B' Z'30' = Decimal packed data with trailing overpunched sign — 'S' Z'70' = Decimal packed unsigned data — 'U'
Key Component Size	1	For data types 'B', 'C', and 'D' the size, in bytes, of the key component. For types 'S' and 'U', the size, in half bytes, of the key component.
Key Component Location	2	The offset from the beginning of each logical record (of this record type) of the key component. The first record position is 0. The offset is in bytes for data types 'B', 'C' and 'D' and in half bytes for types 'S' and 'U'.

Note:
Key component data type, key component size and key component location constitute one key component descriptor. The number of key component descriptors present is specified by the number of key components.

Tables A-24 through A-26 describe the access control list record and common access control list records for directories and files.

TABLE A-19. ADDITIONAL INFORMATION RECORD — DIRECTORY INFORMATION

Field Name	Size (bytes)	Description
ID Code	2	Must contain Z'FFFD'.
Current Size	2	Specifies the number of logical sectors currently allocated to the directory.
Increment Size	2	Specifies the number of additional logical sectors to be allocated whenever it becomes necessary to do so; this is the size of an additional extent.
Maximum Size	2	Specifies the maximum number of logical sectors that can be allocated to the directory (i.e., the maximum directory size).
Link to ACL Record	2	Specifies the relative record number of the first Access Control List Record (ID code = 'FFFA') which contains part or all of the Access Control List (ACL) for this directory.
Reserved	2	Must contain zeros.
Link to CACL for Files	2	Specifies the relative record number of the file CACL record (ID code = 'FFF9') which contains all or part of the Common Access Control List (CACL) to be applied to all files created under this directory.

TABLE A-19 (CONT). ADDITIONAL INFORMATION RECORD — DIRECTORY INFORMATION

Field Name	Size (bytes)	Description
Link to CACL for Directories	2	Specifies the relative record number of the directory CACL record (ID code = 'FFF8') which contains all or part of the Common Access Control List (CACL) to be applied to all immediately inferior directories created under this directory.
Reserved	12	Must contain zeros.
Next Record Link	2	Specifies the relative record number of the next additional information record in the chain.
Reserved	2	Must contain zeros.

TABLE A-20. ADDITIONAL INFORMATION RECORD — TRANSACTION PROFILE (MOD 200 ONLY)

Field Name	Size (bytes)	Description
Record Type	2	Must contain 8004.
Read Access Lock	2	Contains ring number (1-16) and visibility mask code (A through L) for Read access.
Write Access Lock	2	Contains ring number (1-16) and visibility mask code (A through L) for Write access.
Execute Access Lock	2	Contains ring number (1-16) and visibility mask code (A through L) for Execute access.
Reserved	8	Must contain zeros.
Size of TCL Program	2	Specifies the size of the TCL program in words.
Reserved	1	Must be zero.
Transaction Priority	1	Contains the transaction priority code (0-15).
Reserved	12	Must contain zeros.

TABLE A-21. ADDITIONAL INFORMATION RECORD — REPORT QUEUE PROFILE (MOD 200 ONLY)

Field Name	Size (bytes)	Description
Record Type	2	Must contain 8006.
Read Access Lock	2	Contains ring number (1-16) and visibility mask code (A through L) for Read access.
Write Access Lock	2	Contains ring number (1-16) and visibility mask code (A through L) for Write access.
Execute Access Lock	2	Contains ring number (1-16) and visibility mask code (A through L) for Execute access.
Reserved	4	Must contain zeros.
Report Profile Information	12	Printer set-up and queue management information.
Reserved	8	Must contain zeros.

TABLE A-22. ADDITIONAL INFORMATION RECORD — USER PROFILE (MOD 200 ONLY)

Field Name	Size (bytes)	Description
Record Type	2	Must contain 8002.
Read Access Lock	2	Contains ring number (1-16) and visibility mask code (A through L) for Read access.
Write Access Lock	2	Contains ring number (1-16) and visibility mask code (A through L) for Write access.
Execute Access Lock	2	Contains ring number (1-16) and visibility mask code (A through L) for Execute access.
Reserved	4	Must cust contain zeros.
User Password	4	Contains the user's unique password.
Stack Size	1	User requested specification for stack size.
High LFN	1	User requested specification for number of files.
User Read Access Key	2	Contains the user's ring number and visibility mask for Read access.
User Write Access Key	2	Contains the user's ring number and visibility mask for Write access.
User Execute Access Key	2	Contains the user's ring number and visibility mask for Execute access.
Reserved	8	Must contain zeros.

TABLE A-23. ADDITIONAL INFORMATION RECORD — FORM PROFILE (MOD 200 ONLY)

Field Name	Size (bytes)	Description
Record Type	2	Must contain 8003.
Read Access Lock	2	Contains the ring number (1-16) and visibility mask code (A through L) for Read access.
Write Access Lock	2	Contains the ring number (1-16) and visibility mask code (A through L) for Write access.
Execute Access Lock	2	Contains the ring number (1-16) and visibility mask code (A through L) for Execute access.
Reserved	8	Must contain zeros.
Start of Field Descriptions	2	Contains block number of field description.
Size of Field Descriptions	2	Specifies the size of field descriptions in words.
Size of Data Entry Buffer	2	Specifies the size of the data entry buffer in words.
Reserved	10	Must contain zeros.

TABLE A-24. ACCESS CONTROL LIST RECORD

Field Name	Size (bytes)	Description
ID Code	2	Must contain 'FFFA'.
Reserved	8	Must contain zeros.
Account	8	Specifies the account name; up to 12 characters are permitted.
Person	8	Specifies the person name; up to 12 characters are permitted.
Mode	2	Specifies the mode name (three packed characters).
Next Record Link	2	Specifies the relative record number of the next additional information record in the chain.
Access	2	Access indicators, as follows: **Bit** **Value and Meaning** 0-11 Must contain zeros 12 1 = Execute (files); create (directories) 13 1 = Write (files); modify (directories) 14 1 = Read (files); list (directories) 15 1 = Null

TABLE A-25. COMMON ACCESS CONTROL LIST RECORD FOR DIRECTORIES

Field Name	Size (bytes)	Description
ID Code	2	Must contain 'FFF8'.
Reserved	8	Must contain zeros.
Account	8	Specifies a 1- to 12-character ASCII account name.
Person	8	Specifies a 1- to 12-character ASCII person identifier.
Mode	2	Specifies a 1- to 3-character mode name.
Next Record Link	2	Specifies the relative record number of the next common access control list record for a directory in the chain.
Access Indicator	2	Access indicators, as follows: **Bit** **Value and Meaning** 0-11 Must contain zeros 12 1 = Create 13 1 = Modify 14 1 = List 15 1 = Null access

Tables A-26 and A-27 describe the headers that precede each tape-resident file; the HDR1 record is always required, but the HDR2 record is required on input tapes only if block sequence numbers (BSNs) are present. HDR2 records always are written on output.

TABLE A-26. COMMON ACCESS CONTROL LIST RECORD FOR FILES

Field Name	Size (bytes)	Description
ID Code	2	Must contain 'FFF9'.
Reserved	8	Must contain zeros.
Account	8	Specifies a 1- to 12-character ASCII account name.
Person	8	Specifies a 1- to 12-character ASCII person identifier.
Mode	2	Specifies a 1- to 3-character mode name.
Next Record Link	2	Specifies the relative record number of the next common access control list record for a file in the chain.
Access Indicator	2	Access indicators, as follows: **Bit** **Value and Meaning** 0-11 Must contain zeros 12 1 = Execute 13 1 = Write 14 1 = Read 15 1 = Null access

TABLE A-27. FILE HEADER FOR TAPE-RESIDENT FILES — HDR1

Field Name	Size (bytes)	Description
Label Identifier	4	Must contain 'HDR1'.
File Identifier	17	Left-justified, blank-filled file name; all ASCII alphanumeric characters are valid. Data Management supports 12-character names (left-justified in the field).
File Set Identifier	6	1- to 6-character ASCII-alphanumeric name that specifies the name of the volume.
File Section Number	4	Generated as A'0001' for first file section; incremented by one for each succeeding file section (multivolume file).
File Sequence Number	4	Sequence number on volume, starting at A'0001'.
Generation Number	4	Generated as A'0001'.
Generation Version Number	2	Generated as A'00'.
Creation Date	6	File creation date, as follows: Δ yyddd, where yy is the year, and ddd is the day (001—366).
Expiration Date	6	The date on which the file should expire, as follows: Δ yyddd, where yy is the year, and ddd is the day.
Accessibility	1	Must contain blanks.
Block Count	6	Must contain zeros.
System Code	13	Must contain blanks.
Reserved	7	Must contain blanks.

TABLE A-28. FILE HEADER FOR TAPE-RESIDENT FILES — HDR2

Field Name	Size (bytes)	Description
Label Identifier	4	Must contain 'HDR2'.
Record Format	1	Specifies the record format, as follows: F = Fixed-length D = Variable-length U = Undefined
Block Length	5	Specifies the maximum size of the block; right-justified zero-filled field containing ASCII-numeric characters. This length includes the BSN field that occupies the first 6 bytes in the block.
Record Length	5	Specifies the size of the longest record in the block; right-justified, zero-filled field containing ASCII-numeric characters. This length includes the 4-byte record-length field at the beginning of each variable-length record.
Reserved	35	This field ignored; it can contain any ASCII-alphanumeric characters.
Buffer Offset Length	2	Contains 06 to indicate that a BSN occupies the first 6 bytes of the block, or 00 if no BSN is generated.
Reserved	28	Must contain blanks.

VOLUME HEADERS

Tables A-29 and A-30 describe the contents of the disk-volume header and the tape-volume headers, respectively.

TABLE A-29. DISK-VOLUME HEADER

Field Name	Size (bytes)	Description
Volume ID	4	Specifies the volume identifier (e.g., VOL1).
Volume Name	6	1- to 6-character ASCII name of the volume; this is the root directory name.
Reserved	27	Must contain blanks.
Owner ID	14	14-character ASCII owner identification; default is 14 blanks.
Reserved	28	Must contain blanks.
Label Version	1	Must contain Z'47' (i.e., G), which specifies standard GCOS ASCII labels.
Cylinder/ Volume	3	Specifies number of cylinders on this volume, as follows: nnnn00 = number of cylinders on this volume 0000nn = number of cylinders on a single-surface diskette
Tracks/ Cylinder	1	Specifies the number of tracks on a cylinder.
Physical Sectors/Track	1	Specifies the number of physical sectors on a track.
Physical Sectors/Logical Sector	1	Specifies the number of physical sectors in a logical sector.
Physical Sector Size	2	Specifies the number of bytes in a physical sector.
Logical/ Physical I/O Boundary	2	Specifies the lowest physical sector available to the user.

TABLE A-29 (CONT). DISK-VOLUME HEADER

Field Name	Size (bytes)	Description
Defective Sector Index Pointer	2	Specifies the physical sector number of the defective-sector index.
Defective Sector Index Size	2	Specifies the size, in bytes, of the defective-sector index.
Allocation Bit Map Pointer	2	Specifies the physical sector number of the volume allocation bit map.
Allocation Bit Map Size	2	Specifies the size, in bytes, of the volume allocation bit map.
Root Directory Pointer	2	Specifies the physical sector number of the root directory, see Table A-12 for a description of the contents of an entry in the root (or any other directory.)
Root Directory Size	2	Specifies the initial size, in bytes, of the root directory.
Reserved	26	Must contain zeros.

TABLE A-30. TAPE-VOLUME HEADER

Field Name	Size (bytes)	Description
Volume ID	4	Specifies the volume identifier 'VOL1'.
Volume Name	6	1- to 6-character ASCII name of the volume; this name is left-justified and blank-filled.
Reserved	27	Must contain blanks.
Owner ID	13	13-character ASCII owner identification.
Tape Density	1	Must contain zeros.
Reserved	28	Must contain blanks.
Label Version	1	Contains one of the following values: 1, 2, or 3 = Honeywell/American National Standard/ASCII tape volume; contents conform to American National Standard/ASCII tape interchange standard. H = Honeywell/American National Standard/ASCII extended tape volume; contents conform to American National Standard/ASCII tape interchange standard, and allow additional Honeywell data types.

VOLUME TRAILERS

Tables A-31 through A-32 describe the tape-volume trailers, which terminate every tape volume.

TABLE A-31. TAPE-VOLUME TRAILER — EOF1

Field Name	Size (bytes)	Description
Label Identifier	4	Specifies end-of-file identifier 'EOF1'.
File Identifier	17	Left-justified, blank filled file name; all ASCII alphanumeric characters are valid. Data management supports 12-character names (left-justified in the field).
File Set Identifier	6	1- to 6-character ASCII-alphanumeric name that specifies the name of the volume.

TABLE A-31 (CONT). TAPE-VOLUME TRAILER — EOF1

Field Name	Size (bytes)	Description
File Section Number	4	Generated as A'0001'.
File Sequence Number	4	A'0001'; file sequence number on the volume.
Generation Number	4	Generated as A'0001'.
Generation Version Number	2	Must contain A'00'.
Creation Date	6	File creation date, as follows: Δ yyddd, where yy is the year, and ddd is the day (001-366).
Expiration Date	6	The date on which the file should expire, as follows: Δ yyddd, where yy is the year, and ddd is the day.
Accessibility	1	Must contain blank.
Block Count	6	Specifies the number of blocks in the file stored on this volume.
System Code	13	13-character ASCII code entered by the system.
Reserved	7	Must contain blank.

TABLE A-32. TAPE-VOLUME TRAILER — EOF2

Field Name	Size (bytes)	Description
Label Identifier	4	Specifies end-of-file identifier 'EOF2'.
Record Format	1	Specifies the record format, as follows: F = Fixed-length U = Undefined D = Variable length
Block Length	5	Specifies the maximum size of the block; right-justified, zero-filled field containing ASCII-numeric characters. This length includes the BSN field that occupies the first 6 bytes in the block.
Record Length	5	Specifies the size of the longest record in the block; right-justified, zero-filled field containing ASCII-numeric characters. This length includes the 4-byte record-length field at the beginning of each variable-length record.
Reserved	35	This field is ignored; it can contain any ASCII-alphanumeric characters.
Buffer Offset	2	Contains 06 to indicate that a BSN occupies the first 6 bytes of the block.
Reserved	28	Must contain blanks.

TABLE A-33. TAPE-VOLUME TRAILER — EOV1

Field Name	Size (bytes)	Description
Label Identifier	4	Specifies end-of-volume identifier 'EOV1'.
File Identifier	17	Left-justified, blank-filled file name. All ASCII alphanumeric characters are valid. Data Management supports 12-character names (left-justified in the field).
File Set Identifier	6	1- to 6-character ASCII-alphanumeric specifying name of the volume.

TABLE A-33 (CONT). TAPE-VOLUME TRAILER — EOV1

Field Name	Size (bytes)	Description
File Section Number	4	Generated as 'A0001'.
File Sequence Number	4	Sequence number on volume; starting at 'A0001'.
Generation Number	4	Generated as A'0001'.
Generation Version Number	2	Generated as A'00'.
Creation Date	6	File creation date, shown as follows: Δ yyddd, where yy is the year, and ddd is the day (001-366).
Expiration Date	6	Date of file's expiration shown as follows: Δ yyddd, where yy is the year and ddd is the day.
Accessibility	1	Must be blank.
Block Count	6	Specifies the number of data blocks in the preceding file on this volume.
System Code	13	Must contain blanks.
Reserved	7	Must contain blanks.

TABLE A-34. TAPE-VOLUME TRAILER — EOV2

Field Name	Size (bytes)	Description
Label Identifier	4	Specifies end-of-volume identifier 'EOV2'.
Record Format	1	Specifies the record format, as follows: F = Fixed-length D = Variable-length U = Undefined
Block Length	5	Specifies the maximum size of the block; right-justified, zero-filled field containing ASCII-numeric characters. This length includes the BSN field that occupies the first 6 bytes in the block.
Record Length	5	Specifies the size of the longest record in the block; right-justified, zero-filled field containing ASCII-numeric characters. This length includes the 4-byte record-length field at the beginning of each variable-length record.
Reserved	35	This field is ignored; it can contain any ASCII-alphanumeric characters.
Buffer Offset Length	2	Contains 06 to indicate that a BSN occupies the first 6 bytes of the block, or 00 if no BSN is generated.
Reserved	28	Must contain blanks.

Appendix B

ASCII and EBCDIC Character Sets

Tables B-1 and B-2 illustrate the ASCII and EBCDIC character sets, respectively. In addition to the ASCII characters, Table B-1 shows the hexadecimal equivalents; Table B-2 shows the binary and hexadecimal equivalents of the EBCDIC character set.

Following are lists of the control characters and special graphic characters that appear in the two tables:

CONTROL CHARACTERS

ACK	Acknowledge	ETB	End of Transmission Block	NUL	Null
BEL	Bell			PF	Punch Off
BS	Backspace	ETX	End of Text	PN	Punch On
BYP	Bypass	FF	Form Feed	RES	Restore
CAN	Cancel	FS	File Separator	RLF	Reverse Line Feed
CC	Cursor Control	GE	Graphic Escape	RS	Record Separator
CR	Carriage Return	GS	Group Separator	SI	Shift In
CU1	Customer Use 1	HT	Horizontal Tab	SM	Set Mode
CU2	Customer Use 2	IFS	Interchange File Separator	SMM	Start of Manual Message
CU3	Customer Use 3				
DC1	Device Control 1	IGS	Interchange Group Separator	SO	Shift Out
DC2	Device Control 2			SOH	Start of Heading
DC3	Device Control 3	IL	Idle	SOS	Start of Significance
DC4	Device Control 4	IRS	Interchange Record Separator		
DEL	Delete			SP	Space
DLE	Data Link Escape	IUS	Interchange Unit Separator	STX	Start of Text
DS	Digit Select			SUB	Substitute
EM	End of Medium	LC	Lowercase	SYN	Synchronous Idle
ENQ	Enquiry	LF	Line Feed	TM	Tape Mark
EO	Eight Ones	NAK	Negative Acknowledgement	UC	Uppercase
EOT	End of Transmission			US	Unit Separator
ESC	Escape	NL	New Line	VT	Vertical Tab

SPECIAL GRAPHIC CHARACTERS

¢	Cent Sign	)	Right Parenthesis
.	Period, Decimal Point	;	Semicolon
<	Less-than Sign	¬	Logical NOT
(	Left Parenthesis	-	Minus Sign, Hyphen
+	Plus Sign	/	Slash
¦	Logical OR	|	Vertical Line
&	Ampersand	,	Comma, Cedilla
!	Exclamation Point	%	Percent
$	Dollar Sign	_	Underscore
*	Asterisk	>	Greater-than Sign

? Question Mark
‘ Grave Accent, Opening Single Quotation Mark
: Colon
Number Sign
@ At Sign
’ Prime, Apostrophe, Acute Accent, Closing Single Quotation Mark
= Equal Sign
“ Quotation Mark

≈ Tilde
{ Opening Brace
Hook
Fork
} Closing Brace
\ Reverse Slant
Chair
| Long Vertical Mark
[Opening Bracket
] Closing Bracket
^ Circumflex

TABLE B-1. ASCII/HEXADECIMAL EQUIVALENTS

	H1							
H2	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

TABLE B-2. EBCDIC/HEXADECIMAL/BINARY EQUIVALENTS

Bit Positions 4, 5, 6, 7	Second Hexadecimal Digit	00				01				10				11				Bit Positions 0,1
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	Bit Positions 2,3
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	First Hexadecimal Digit
0000	0	NUL	DLE	DS		SP	&	-						{[a]	}[a]	\[a]	0	
0001	1	SOH	DC1	SOS				/		a	j	~[a]		A	J		1	
0010	2	STX	DC2	FS	SYN					b	k	s		B	K	S	2	
0011	3	ETX	TM							c	l	t		C	L	T	3	
0100	4	PF	RES	BYP	PN					d	m	u		D	M	U	4	
0101	5	HT	NL	LF	RS					e	n	v		E	N	V	5	
0110	6	LC	BS	ETB	UC					f	o	w		F	O	W	6	
0111	7	DEL	IL	ESC	EOT					g	p	x		G	P	X	7	
1000	8	GE[a]	CAN							h	q	y		H	Q	Y	8	
1001	9	RLF[a]	EM						\[a]	i	r	z		I	R	Z	9	
1010	A	SMM	CC	SM		¢	!	:[a]	:								][a]	
1011	B	VT	CU1[a]	CU2[a]	CU3[a]	.	$	,	#									
1100	C	FF	IFS		DC4	<	*	%	@					⌠[a]		⌡[a]		
1101	D	CR	IGS	ENQ	NAK	(	)		'									
1110	E	SO	IRS	ACK		+	;	>	=					⌡[a]				
1111	F	SI	IUS	BEL	SUB		¬	?	"								EO[a]	

[a]This character is not supported in the 2780 character set.

Appendix C

Keys

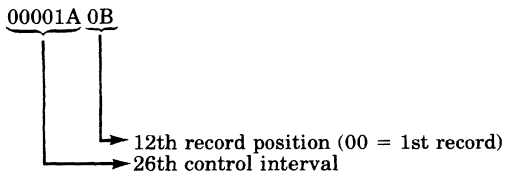
Table C-1 identifies and describes the keys available for use with the various file organizations.

TABLE C-1. TYPES OF KEYS

Type of Key	Description	Associated File Organization(s)
Relative	A 32-bit binary integer[a] (in main memory) that specifies the relative position of a record in a file; the first record is record 1.	Fixed-relative Relative
Simple	A 32-bit binary integer[a] (in main memory) that specifies the control interval (bits 0 through 23) in which a record is stored, and the line number (bits 24 through 31) of that record; the first control interval is control interval 1, and the first record in each control interval is record 0.	Disk-resident sequential Relative
Primary (required)	A user-defined field (of a fixed length) in a fixed location in each logical record stored in the file; the first position in a record is position 1; a maximum of 256 ASCII characters are allowed.	Indexed
Data Base Key (MOD 600 only)	A unique internal identifier assigned by I-D-S/II software indicating a record's position in the data base.	Integrated (I-D-S/II)
CALC Key (MOD 600 only)	A user defined field (of fixed length) in a fixed location in each CALC record type stored in the files; used together with a hash algorithm to locate the desired record.	Integrated (I-D-S/II)

[a] A 32-bit binary integer is expressed as an eight-digit hexadecimal number in the range 00000000-FFFFFFFF (i.e., each 4-bit hexadecimal integer has an equivalent decimal in the range 0 through 15).

Following is an example of a simple key to specify the 12th record in the 26th control interval in a relative file; the key would be coded as:



CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS

Technical Publications Remarks Form

TITLE	SERIES 60 (LEVEL 6) GCOS 6 DATA FILE ORGANIZATIONS AND FORMATS	**ORDER NO.**	CB05, REV. 2
		DATED	NOVEMBER 1978

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by appropriate technical personnel and action will be taken as required. If you require a written reply, check here and furnish complete mailing address below. ☐

FROM: NAME _____ **DATE** _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

CUT ALONG LINE

FOLD ALONG LINE

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

FIRST CLASS
PERMIT NO. 39531
WALTHAM, MA
02154

Business Reply Mail
Postage Stamp Not Necessary if Mailed in the United States

Postage Will Be Paid By:

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTENTION: PUBLICATIONS, MS 486

FOLD ALONG LINE

Honeywell

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

22120, 2.51178, Printed in U.S.A. CB05, Rev. 2