IBM Data Processing Techniques

Data File Handbook

Design

This manual is a basic primer that provides fundamental information about the use, composition, organization, and design of data files for all types of IBM equipment including System/360. Reference material is included on the characteristics of the various processors, I/O devices, and IOCS programs, since these factors affect the design of data files.

Minor Revision (March 1966)

This edition, C20-1638-1, is a reprint of C20-1638-0 and incorporates changes released in Technical Newsletter N20-0055. The original publication and Newsletter are not obsoleted.

Minor clarifications and additional information have been incorporated into the text. Changes are designated in three ways:

1. A vertical line appears at the left of affected text where only part of a page is changed.

2. A dot (•) appears at the left or right of the page number where a complete page should be reviewed.

3. A dot (•) appears at the left of the title of each figure that has been changed.

The affected pages are: 4, 5, 8, 11, 12, 18, 25, 31, 33, 35, 39, 40, 51-62, 66, 67, 70, 71

Copies of this and other IBM publications can be obtained through IBM branch offices. A form has been provided at the back of this publication for readers' comments. If the form has been detached, comments may be directed to:
IBM, Technical Publications Department, 112 East Post Road, White Plains, N.Y. 10601

CONTENTS

INTRODUCTION

To broaden the scope of this manual and to facilitate its use by individuals of divergent backgrounds and experience, the material is presented under the following three captions:

1. Fundamentals of Data Files

This section is designed primarily to acquaint the reader with the definition, functions, composition, and processing of data files. Those more experienced with data processing may prefer merely to scan this portion of the manual as refresher material or to skip it entirely.

2. Design of Data Files

The purpose of this section is twofold: to provide checkpoint type of information for the more experienced and to furnish those new to data processing with pointers on file design, beginning with the determination of data, field size, and file organization through record format and file capacity.

3. Reference Material for Data Files

The aim of this section is to draw together under one cover quick, easy-to-use data file reference materials that cross over computer and I/O device lines so that the user of this manual, no matter what his experience or needs may be, will not have to seek data file information from a battery of separate sources.

Although it is assumed that the reader has a general knowledge of the media involved, the reference material in section 3 may be read first to provide sufficient background.

SECTION 1: FUNDAMENTALS OF DATA FILES

DATA FILES

Definition

A data file (or data set) is a collection of related records that provides specific information about a fixed area of activity. Data files may be stored in such media as paper, cards, magnetic tape, paper tape, or direct access storage devices (DASD). (See section 3 for reference information concerning media storage and data recording.)

Types of Data Files and Their Functions

Master File

A master file contains the current status of a given list of items. A relatively fixed number of items are in the file over a long period of time, and the number of insertions and the number of deletions tend to be fairly well balanced despite temporary seasonal or cyclic fluctuations. Each record is subject to updating. The file is a major source of information for facilitating decisions in a particular area of operations, both internally with computer programming and externally with management review of printed reports of data contained in the file.

Example:

An inventory master file for a hardware concern might contain one record for each item of stock. Although about 20,000 different items can be represented on the file at any given time, some items, such as lawn sprinklers, may be stocked only in the summer and others, such as snow shovels, only in the winter. The quantity on hand for a certain item must be updated to reflect any change in stock caused by such transactions as sales, returns, receipts, etc. Based on a minimum balance on hand, the computer, through programming, can determine the time to reorder. Periodically, the information on the file can be used to print out reports indicating sales trend, slow moving items, low-profit sellers, etc., which can be reviewed by management.

Transaction File

The primary purpose of a transaction file is to contain activity or inquiry records that will be used to examine and/or update a master file. Each activity record contains data about an occurrence that will affect the master file in some way.

Example:

Activity, such as receipts, sales, returns, etc., could be contained in the transaction file that is used to update a master inventory file.

History File

A history file can be an obsolete master file or a compilation of transaction records that have affected a master file within a particular period. It is maintained primarily to gather statistical data or to capture sufficient detail of past processing to facilitate reconstruction of a master file.

Summary Files

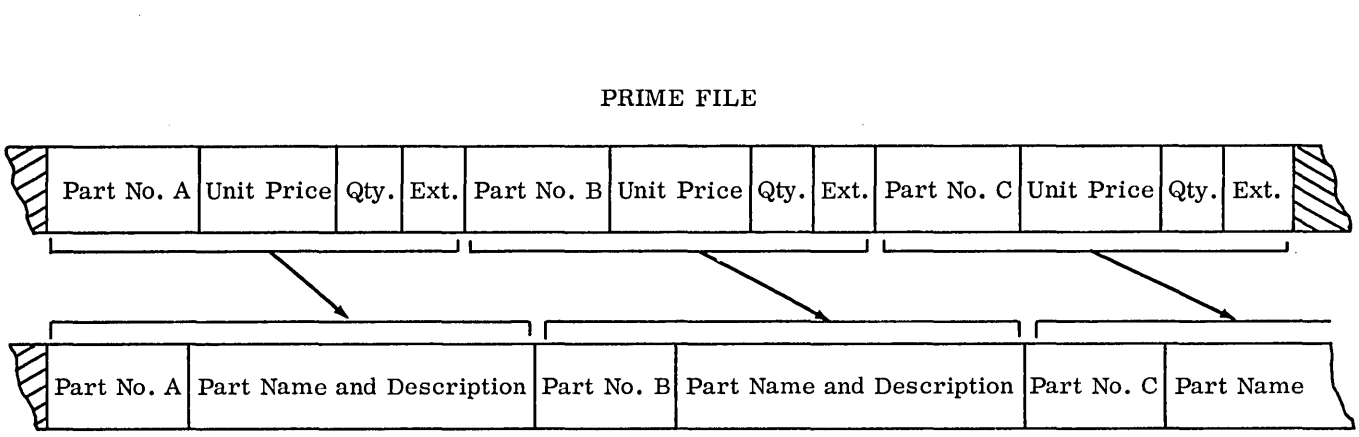
A summary file represents data from another file reduced to a more concise form. The information from several records in the original file can be shown in aggregate form on one record of the summary file by using a broader criteria for record uniqueness. For example, employee earnings records in a payroll master file may be summarized into fewer records showing total earnings by department. Depending upon its use, a summary file also may be considered as a master file, a transaction file, or a history file.

Trailer File

A trailer file contains detail records associated with particular records in another file. The latter often are called prime records and constitute a prime file. The records in the trailer file provide additional information to augment the data found in the associated prime records. Trailer files may be processed individually or together with their prime file. The trailer files here are not to be confused with trailer or overflow records, which are discussed later under "Types of Records and Their Functions".

Example: Inventory Parts File

1. For daily processing of inventory updating only the prime file is used.
2. For preparation of cross reference part number and name lists only the trailer file is handled.
3. For periodic inventory status reports requiring part name as well as part number and the associated quantitive data, both the prime file and trailer files are processed simultaneously.



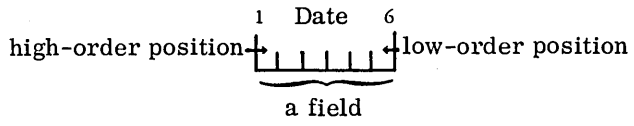
COMPOSITION OF A DATA FILE

Fields and Subfields: Definition

Fields and subfields, the smallest elements of a data file, are composed of adjacent positions or characters that describe a unit of information. The leftmost position of a field is known as the high-order position; the rightmost is known as the low-order or units position.

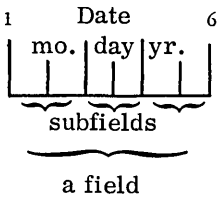
Example:

In the field labeled Date, which is six positions long, position 1 is called the high-order position and position 6, the low-order position.



Subfields are meaningful subdivisions of a field to facilitate data identification and manipulation.

Example:



In the field called Date, the left two positions contain the subfield month, the middle two positions the subfield day, and the right two positions the subfield year.

Types of Fields and Their Functions

The data recorded in a field may be classified by the function it serves.

Control fields permit the proper identification and handling of a given record or section of a record.

A record control field or key establishes the uniqueness of the record within the file.

Example:

Each data record in an employee payroll master file contains the employee number, which is the record control field. Therefore, the employee number can control the sequence of the file and the application of transaction data to the correct employee.

A coding control field identifies the record or section.

Example:

Each data record in a transaction file may contain a single character field that controls the kind of information contained in the record and that indicates what effect this record will have on the master file. A digit 1 in the position may indicate that a new record is to be created in the master file with the data in the record; a 2 that this record contains data that will change data in a master file record; a 3 that the master file record is to be deleted, etc.

Indicative fields usually are nondynamic and contain miscellaneous data pertinent to the record identified by the record control field. Some of these fields have a more specific nature:

A statistical field provides additional information that normally is used for the gathering of statistical information.

Example:

Each employee payroll master record may have a field containing a code to indicate the sex of the employee. An M may indicate male and an F, female.

A constant field contains fixed data that otherwise might have to be developed each time the record is processed.

Example:

Each employee payroll master record in a data file may have a field containing the dollar amount of federal tax exemption allowable for that employee each week. The amount is stored in the record to avoid recalculating it each time the payroll is processed.

A reference field provides data identifying the transaction with the original source document from which it was created. Reference fields are essential in providing adequate audit trail.

Example:

In a sales application the transaction record contains an invoice number. Should a question arise regarding the transaction, the invoice number relates the record to its source document.

Quantitative fields contain amounts and may contain sign indication. Frequently, these amounts are used in calculations, or they may be the results of computations.

Example:

Each weekly employee payroll record has a field containing hours worked. This amount field is used in calculating gross weekly earnings, which also becomes a quantitative field.

Field Characteristics

Length

Fields usually contain a fixed number of positions designed to hold the maximum amount of data that can occur.

Class

The data in a field is alphabetic, numeric, or alphameric.

Significance

The significant digits of a numeric field are those digits which are necessary to make the number meaningful and which, when specified, include a fixed number of decimal positions. Embedded blanks (blanks in the midst of significant characters) normally are unacceptable in numeric fields.

The significant characters of an alphabetic or alphameric field include all positions from the leftmost through the rightmost nonblank character. Embedded blanks are acceptable.

Right and Left Adjustment (or Justification)

The number of significant characters for a field can vary. A right-adjusted field contains the rightmost significant character in the low-order position of the field; a left-adjusted field contains the leftmost significant character in the high-order position of the field.

An alphabetic or alphameric field is normally left-adjusted, with the nonsignificant portion of the field filled with blanks. A numeric field usually is right-adjusted, with the nonsignificant portion of the field filled with leading zeros, although blanks may be used. Zeros are preferred to blanks because they are positive proof of the value required; blanks may represent omissions. Since blanks normally are considered of lesser numeric value than are zeros, the use of blanks instead of zeros may affect the sorting sequence. When a numeric field is left-adjusted, the nonsignificant portion of the field usually is filled with blanks.

Examples:

1. 0 0 0 1 2 3 0 6 nonsignificant zeros; significant digits	This is an 8-position right-adjusted numeric field with leading zeros.
2. 1 2 3 0 6 b b b significant digits; nonsignificant blanks	This is an 8-position left-adjusted numeric field.
3. J o h n b J o n e s b b b b b significant characters; embedded blank; nonsignificant blanks	This is a 15-position left-adjusted alphabetic field.

Special Characteristics of Numeric Fields

Signs. When a numeric field may be either positive or negative, sign coding must be present.

Decimal Positions. Usually decimal points are assumed; this means that no space is reserved in a field for the decimal point. This conserves media storage and permits arithmetic manipulation of the field by equipment that normally does not recognize the point of a number. (See "Field Compaction Techniques" for a discussion of decimal scaling and floating decimal-point numbers.)

Recording Mode. To facilitate computer functions or to reduce storage data requirements, it may be preferable to work with numeric data in some form other than the normal decimal notation. Translation to and from decimal format may be accomplished by computer subroutines or by hardware capabilities of the equipment; or the data may remain in the modified mode. (See "Field Compaction Techniques" for a discussion of binary, hexadecimal, and packed format.)

Records: Definition

A record is a collection of fields arranged in a defined format and related to a common identifier.

Types of Records and Their Functions

Data records (which represent the bulk of the records in any file) are those records which contain specific information about a given data processing application. Each individual data record is made up of coding and record control fields, which establish the uniqueness of each record, as well as fields containing pertinent information about the particular record.

Checkpoint records are created at specific intervals during the running of a lengthy program to retain the contents of main storage and other data required for restart from an intermediate point rather than from the beginning, in the event the program has to be interrupted for some reason, such as an uncorrectable error or a job with higher priority, which takes precedence.

Label records are used for file identification and for checking purposes. Normally, header labels containing such information as file name, file number, creation date, and retention cycle are processed before any of the data records to verify that the file is the proper file for use. A trailer label indicates the end of a file (EOF), or the end of a physical subdivision of a file (EOR or EOV), such as the end of a reel of tape, disk pack, etc. Trailer labels also may contain a cumulative count of the number of records and blocks (groups of records) in the data file and, sometimes, one or more control totals, each of which is the sum of the contents of a particular field in the records.

These internal label records are in addition to the external labels used for visual identification of the file.

Trailer or overflow records contain additional data, pertaining to a given record, which cannot be recorded in the prime record for some reason. Often, this is information that occurs only occasionally, so instead of designing a long record with wasted space, a subsidiary is created. Normally, the prime record contains coding that can be used to reference the trailer record, whereas the latter has the same record control field to assure that the proper trailer has been located.

Record Characteristics

Length

The length of a record is the sum of the lengths of its fields. However, because of differences in hardware characteristics, the length of a record in medium storage may vary from its length in core storage.

Examples:

Load mode operation (1400 series), compressed tape reading (7070/7074), and alphameric and numeric mode recording differences (tape and 1301/1302 with 7070/7074). (See section 3, "Differences between Core and Media Storage Requirements".)

Form

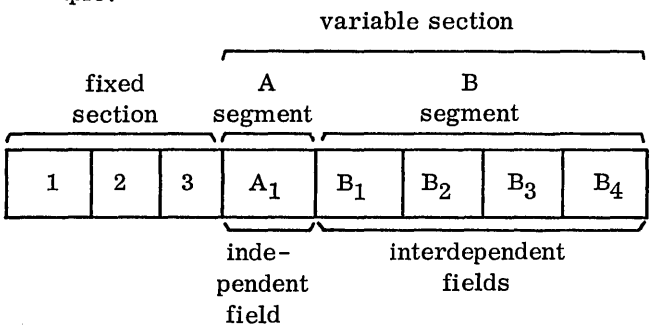
There are three basic sections in a record: the fixed section, the variable section, and the control section.

The fixed section, which is normally present and which is fixed in length, is composed of those fields that provide the coding and record control fields, as well as all required indicative and quantitative fields. There is only one fixed section per record; it may be the entire record or the first section of a multisection record.

The variable section is a collection of a variable number of segments, which, in turn, are composed of either an independent field or a group of interdependent fields. An independent field is a single field that requires no supporting field, whereas interdependent fields are two or more related fields treated as a whole. Thus, when any of the interdependent fields contains significant data, the entire segment must exist. Each segment is fixed in

length and is retained in the variable section only when data exists.

Example:



The control section is a section that may or may not be used when a record contains a variable section. It provides information about the presence or absence as well as length of the segments in the variable section of the record and is used to facilitate the location and manipulation of the data. Occasionally, in lieu of the control section, special codes are used to signify omission of a segment.

Example:

Fixed	Control Section				Variable Section		
Section	A	B	C	D	A	C	D

Segment B is missing in this record, but the control field for B is present. Normally, the control section follows the fixed section and is fixed in length.

Fixed	Variable Section		
Section	A	#	C

The # is a special code to indicate the absence of segment B; no control section is used.

A fixed-length record contains only a fixed section; the length and relative location of each field is constant. A variable-length record consists of a fixed section, where length and location of the fields are known, and a variable section, where the relative location and length of each field is determined by programmed use of the control section or by special codes included in each record for that purpose.

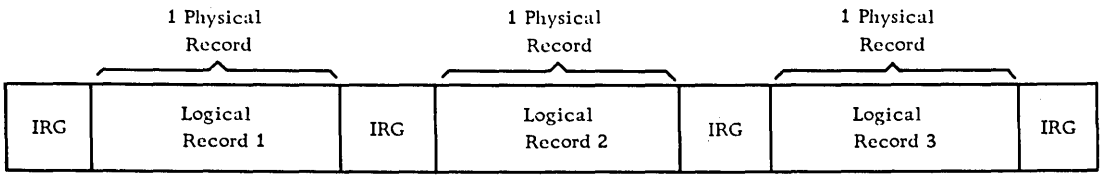
Blocking

Blocked records are two or more records grouped together and treated as a whole for reading, writing, and storage purposes. This grouping is referred to as a block of data records. Each record within the block is processed individually by the program and is known as a logical record. All of the records read or written as a block of data are referred to as a physical record, since the entire block is transferred physically between the I/O device and main storage. Records that are read, written, and stored individually are known as unblocked records.

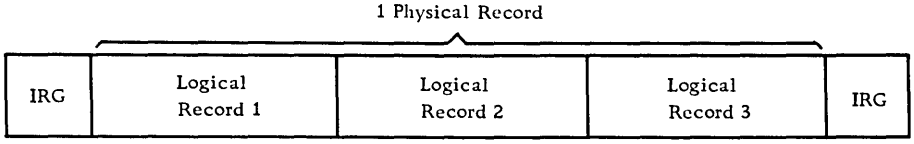
The blocking factor is the number of records that are contained within a block. Blocking may be utilized to make more efficient use of medium storage and to conserve input/output time. The concept of blocking is applicable to all types of medium storage. If the number of positions per record is 40 or less, even card records can be blocked for computer operations.

Example:

Tape records are separated by blank tape, which is called an interrecord gap (IRG). IRG's, created automatically during tape-write operations, signal the end of a tape record when the tape is read. Whenever data records are blocked, fewer IRG's are created. This conserves tape storage and computer time. (See section 3, "Magnetic Tape Device Characteristics", for the length of IRG's and for tape-processing speeds.)



Unblocked data record



Blocked data records

Figure 1. Examples of unblocked and blocked data records

FILE ORGANIZATION

Definition

File organization deals with the relationship of the control fields of a file record to the physical location of that record in the storage medium.

Types of File Organization

Although a file of records can be arranged in a storage medium in many different ways, all of the ways can be classified by either of two basic techniques — sequential or random.

Sequential

Sequential order implies that there is a certain numeric or alphameric sequence, either ascending or descending, of the adjacent records in the file. Particular fields, located in the same relative positions within the fixed section of all data records of the file, are selected as sort control fields for a specific file sequence.

Random (Non-sequential)

A random file organization contains records stored without regard to the sequence of their record control fields.

Sequential and random file organization should not be confused with sequential and random processing. The terms random and sequential, when used with the term processing, usually refer to the order of the input transaction records or to the order of reference to records in the master file. Figure 2 shows the relationship between file organization and data processing, as follows:

Case 1	Sequential processing of sequentially organized data
Case 2	Random processing of sequentially organized data
Case 3	Sequential processing of randomly organized data
Case 4	Random processing of randomly organized data

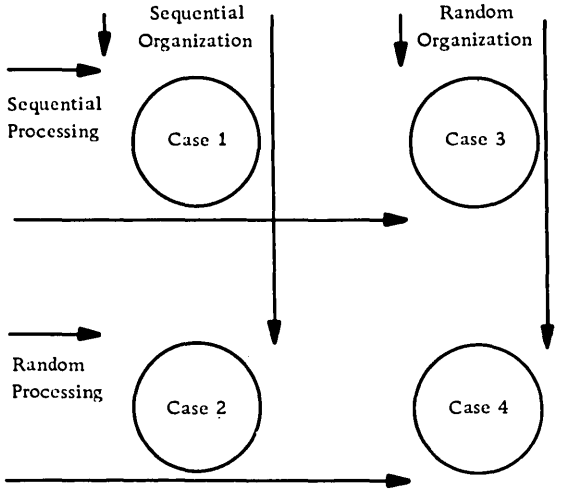


Figure 2. File organization and processing approaches

Tape and Card

Sequential

Records on tape and card files must be processed as they are encountered because of the physical nature of the storage medium. Therefore, tape and card master files, and their associated transaction files (regardless of the medium), tend to be sequential.

Random

A random file organization for tape master files is impractical, since the desired record can fall any where within the file limits, and each search for a specific record must begin with the first record of the file. A random organization of transaction data is common when the master file is stored on a direct access storage device (DASD), since such units are capable of retrieving records randomly. In those cases where it is desirable to maintain information in the same sequence as the source documentation, to facilitate control and validation procedures, a transaction file may be created in the order of occurrence, with sorting as an intermediate step, before processing against a sequentially organized file.

Direct Access Storage Device (DASD)

Sequential

In a sequentially organized DASD file, the records are stored in record control number sequence, so that records with successively higher control numbers have successively higher address numbers. Normally, the record control number is not the same as the address where it is stored; the only requirement is that the control numbers be in sequence and in sequential (not necessarily consecutive) disk storage locations.

Additions and deletions to the file present the greatest design challenge. Additions which cannot be inserted in sequence in the original (prime) area are known as overflow records. Overflow areas are set aside on specific tracks of the same cylinder, or, since it is difficult to predict the overflow pattern, a single cylinder or group of cylinders may hold all overflow records for the entire file. As the number of overflow records increases, the processing time also rises. Therefore, it is customary to reorganize the file periodically, incorporating all overflow records into the prime area.

Although DASD sequential files can be processed strictly sequentially in the same fashion as tape, such a method does not take full advantage of the ability of the DASD to locate a specific record directly and thereby eliminate the time required to read inactive records. An index system is used frequently to narrow the search for a particular record.

Such a system may be likened to the index system used in locating an item in a multiple-volume standard dictionary. The index on the cover gives the last item in a volume. In a DASD the master index performs this function. The thumb index is similar to a cylinder index, while the upper-page index is analagous to the track index. The specific item is located by searching the page or track.

Each of these indices may be considered as a level. The number of levels and the size of the index is dependent upon the total number of items in the file. The indices usually are contained in the same or another DASD, and, if possible, the master and cylinder indices are read once into core and retained to minimize the retrieval time. The index itself is a table normally composed of at least two elements per entry: the record control key for the last record entry contained in the specific logical group (cylinder group, track group), and the DASD track address. Figure 3 illustrates the general scheme of an index system.

Because inactive records in a file may be skipped, the term skip sequential is used frequently to refer to the indexed sequential file organization methods. Specific sequential file organization techniques and their related overflow record and index systems are discussed in section 2, under "Determination of File Organization".

Random

In a random file, records are stored at an address that is obtained by applying a mathematical formula to the record control field. No indices are required to locate a specific record, since the storage address can be found by using the same conversion routine.

Example:

Assume that three types of loans are to be stored on DASD as follows:

Type of loan	DASD address	Number of available locations
A	000600-034099	33500
B	034100-047999	13900
C	048000-059999	12000

Apply the following formula to the loan account number to obtain the DASD address.

1. Multiply account number by the number of available locations. Assume Type C account number 99999.

```
99999 x 11999 = 1199888001
```

2. Add the lowest location of the allotted block to the five high-order positions of the product obtained in step 1.

```
11998 + 048000 = 059998
```

3. Use the result from step 2 as the DASD address.

In the transformation of two different record control numbers, it is possible to obtain the same DASD address. Such duplicate addresses are called synonyms. If only one record can be stored at a given address, or if multiple records may be stored, and all available areas are used, the synonyms become overflow records. Address conversion routines and overflow techniques for randomly organized files are discussed in section 2 under "File Organization Techniques".

PROCESSING OF DATA FILES — INPUT/OUTPUT CONTROL SYSTEM (IOCS)

Definition

IOCS (or Data Management) is a set of programmed subroutines designed to relieve the programmer of the necessity for writing input/output routines by automatically handling the preparation and checking of labels, the blocking and

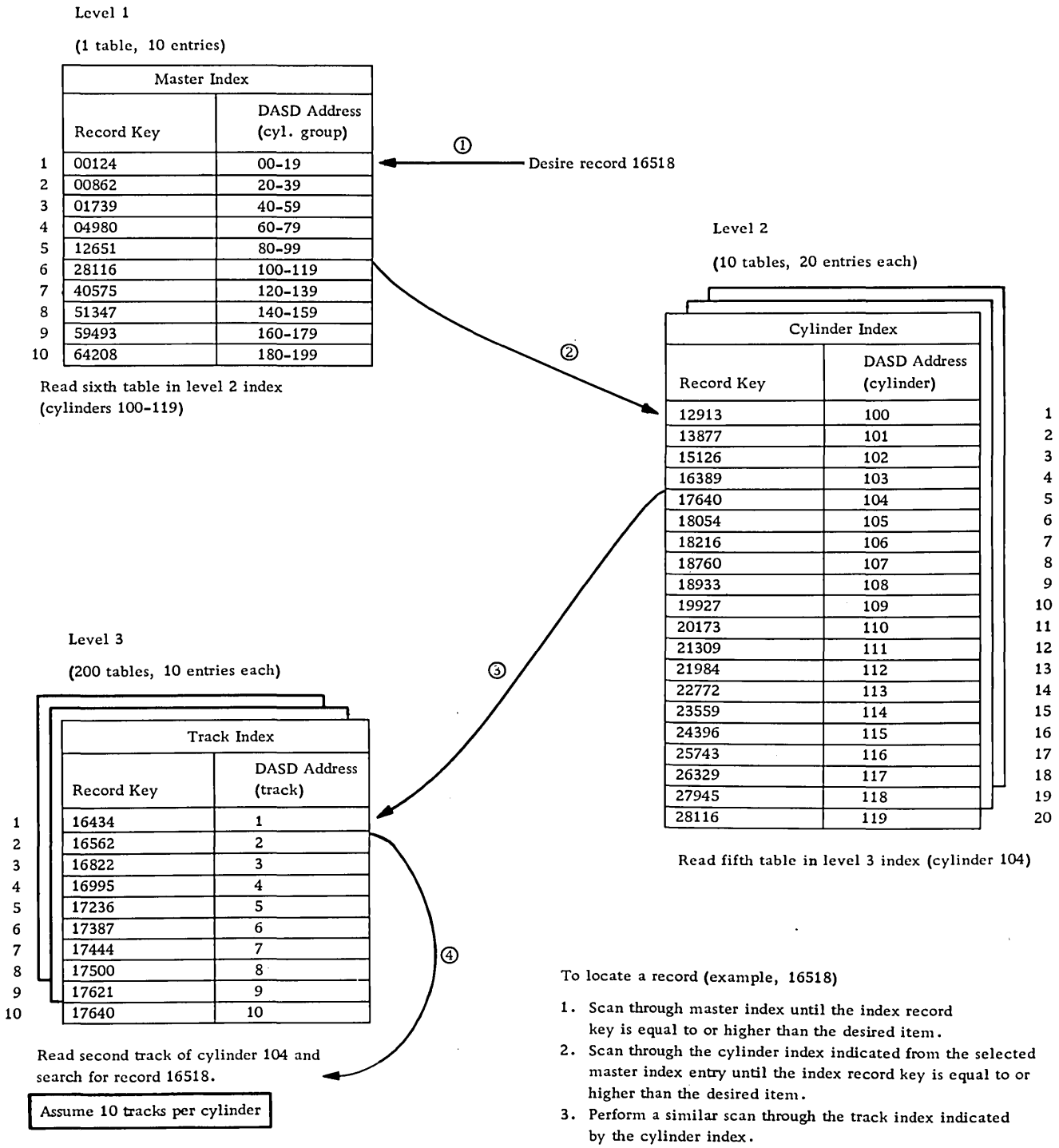


Figure 3. Generalized index system for DASD

deblocking of data records, detection and error recovery procedures, and the overlapping of processing with input and output functions. Through the use of pretested, error-free I/O routines, the programmer has more time to concentrate on the data manipulation requirements of his specific program.

Functional Concepts

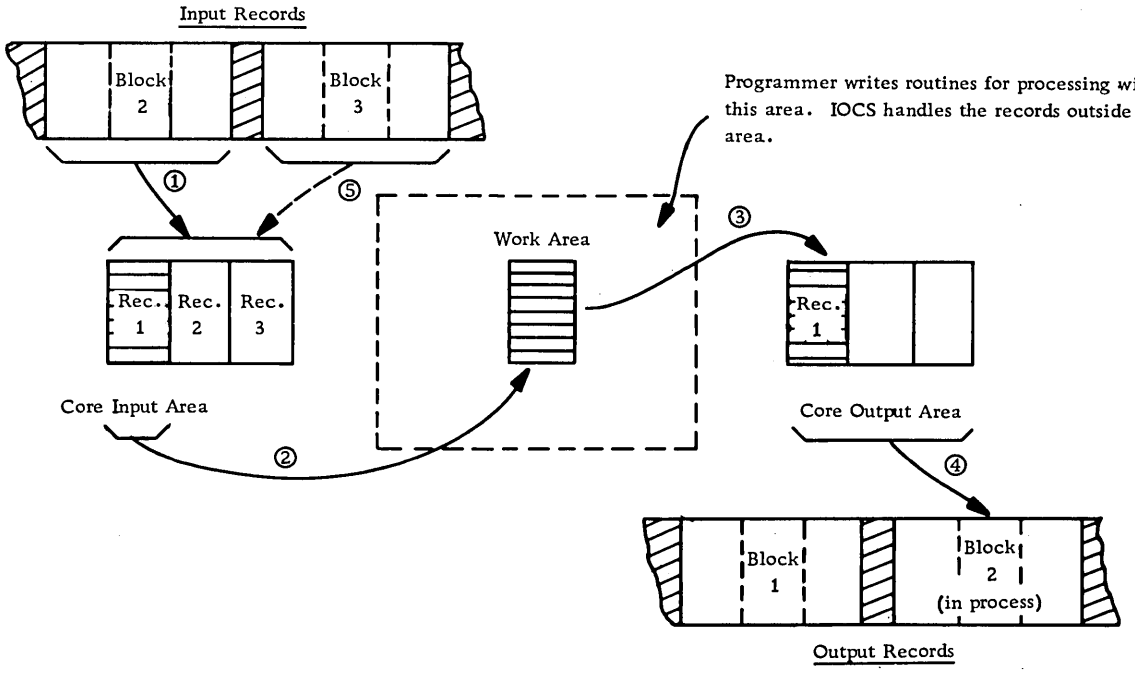
The general flow of data records from a blocked data file through a program controlled by IOCS is illustrated in Figure 4.

Many modifications of this basic flow of data are possible. In some cases records are processed in the input area and transferred from there directly to the output area. In other cases records are read and written in the same area. Records also can be processed in the output area.

Usually, a unique device-error recovery routine exists for each class of I/O device. Upon detection of an error, such as a misread, misseek, etc., the appropriate error routine is entered, and an attempt is made to recover from the error (for example, reread tape, reseek, etc.). If recovery is not possible, various choices may be provided, such as bypass of the record or return of control to a user error procedure.

I/O devices may be attached to channels instead of directly to the central processing unit (CPU). Channels provide paths for data transfer between the CPU and the I/O device. This allows I/O operations to be overlapped with CPU operations so that instructions can be executed simultaneously with data movement in the channels. For example, one channel may be reading data from an input file, another channel may be writing data on an output file, while a record that was read previously is being processed. This is often called read/write/compute overlap.

The amount of overlapping actually achieved (effective overlap) is governed through the assignment of I/O areas and work areas. An I/O area (or buffer) is that area of main storage to which, or from which, a block of data will be transferred physically. A work area is an area used for processing an individual record from the block of data. Overlap is most effective, usually, when at least two I/O areas are



1. A data block is read into an input area in core storage.
2. Records are moved individually to a work area for processing.
3. After processing is completed, the records go to an output assembly area in core.
4. When the data block in the output assembly area is complete, the output area is transferred to the output medium.
5. When all records in the input area have been processed, another block of records is read into the input area.

Figure 4. Flow of data records from a blocked data file through a program controlled by IOCS

assigned to each data file used by a program.

When a request is received by IOCS for an I/O operation, the requested operation is started, and control passes back to the problem (or user's) program, if the affected channel and device are not busy. If the channel or device is busy, the request is placed in a list of I/O requests (separate list or queue for each channel), and the operation is performed as soon as previous requests have been handled.

On some computers the channels have the ability to interrupt processing at the completion of an I/O operation. The interrupt transfers program control to IOCS, which examines the queue for the affected channel. If the queue has no pending I/O request, control is returned to the problem program at the point of interruption. If, instead, a request is pending, IOCS starts the I/O operation and then returns to the problem program. If IOCS fails to have an available area or record when one is required by the problem program, a force situation results. IOCS is forced to suspend processing of the user's program temporarily until it can service the I/O demands. All operations are suspended, except channel operations in progress, until the needed channel interrupts the system, and IOCS is able to resolve the situation by servicing the data file.

Programming Concepts

IOCS is specified by the programmer at the symbolic programming level. It is inserted automatically into the user's program at compilation time and becomes an integral part of the user's program. In some cases standardized IOCS packages may be used to avoid compilation time. For certain computer systems, some of the IOCS routines may be located in the operating system, which is supervising and controlling the tasks that a computer is to perform. When I/O operations are required, the problem program turns control over to the operating system, which performs the necessary I/O functions and returns control to the problem program.

To enable specific coding to be generated to perform the IOCS functions needed by a particular program, the programmer issues two types of statements (declarative and imperative) in his source program.

Declarative Statements

These statements, known also as DTF (Define the File) statements, or DD (Data Definitions), provide information that:

- Describes the characteristics of the logical file, such as blocking factor, record size, record format, type of labels, file name, etc.
- Describes the physical device on which the file resides, such as channel, type of device, etc.
- Identifies options to be taken under predefined conditions, such as uncorrectable read errors.
- Contains addresses of user-written routines, such as end-of-file routines.

Imperative Statements

Four basic verbs cause various IOCS functions to occur when specified in a user's program — OPEN, GET, PUT, and CLOSE.

OPEN does the following:

- Makes a data file available to a program.
- Checks header labels on input files.
- Performs control functions as specified (rewind, etc.).
- Writes header labels on output files.

GET causes a data record to be made available to the program, either in the input area or in a work area. Issuing a GET provides linkage to routines that can perform various necessary functions for input files, such as:

- Initiate read of data blocks into main storage as they are needed.
- Deblock data input records.
- Count data blocks and records read into main storage for comparison to equivalent count fields in the trailer label.
- Recognize and handle errors originating as a result of the I/O operation.
- Recognize an end of reel (EOR) condition; check the trailer label and, for tape, rewind and unload; read and check the header label on the next sequential storage medium unit (tape, pack, etc.).
- Recognize an end of file (EOF).

PUT causes a data record to be moved to the output area from the input area or from a work area, or it may place the core limits of the record in a list containing the addresses of all records ready for output. Issuing a PUT provides linkage to routines that can perform various necessary functions for output files, such as:

- Block data output records.
- Initiate write of data blocks when they are assembled.
- Count data records and blocks for placement in count fields of the trailer label.
- Recognize the physical end of the storage medium; write the trailer label and, for tape, rewind or rewind and unload; write the header label on the next storage medium unit.

- Recognize and handle errors originating as a result of the I/O operation.

CLOSE does the following:

- Causes a data file to become unavailable to a program.
- Writes any output records that may still be in the output areas.
- Writes trailer labels.
- Performs control functions as required (write tape marks, rewind, etc.).

Record Formats

In many files different record formats may have identical lengths, while other record formats may vary in length. However, when records of different lengths are grouped together to create a data file, only one format can be specified to IOCS for the file. Unless every record (excluding header, trailer, and checkpoint records) in the file is of identical length, the file is considered to be composed of variable-length records. Unless all of the records in the file are unblocked, the file is considered to be composed of blocked records.

Various file formats can be specified to IOCS (see Figure 5):

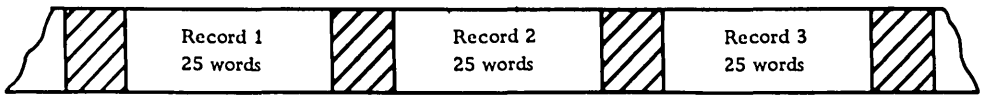
- Unblocked fixed-length records.
- Blocked fixed-length records with a fixed blocking factor. Padding may be used when insufficient data records are available to complete the last block of a data file. A padding character, such as 9, is inserted, usually in each position of any unused records in the last block. (Some IOCS routines process a short block instead of using padding.)
- Unblocked variable-length records. The length of an individual record must fall within a specified maximum record length established for each file.
- Blocked variable-length records with a fixed blocking factor.
- Blocked variable-length records with a variable blocking factor. Instead of using a fixed blocking factor, the records are assembled within a block so that their combined length does not exceed a specified block length. Records are not split between blocks.
- Undefined records. This format permits handling of records that do not conform to the other formats.

Special fields may be required by IOCS on some data records to aid in the deblocking process:

- A special character, usually a record mark, located in the last position of each record and considered as part of the record.
- A record length indicator, containing the word or character count of the data record, located in the same relative position within each data record.
- A block length indicator, showing the number of words or characters in the block, used to ascertain that the correct block length has been brought into core.

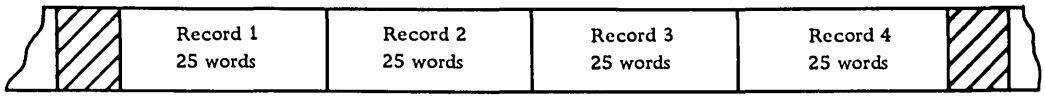
Since variances exist in the IOCS packages developed for different computers, the specific IOCS must be checked to determine the exact record formats available, any specialized field requirements, and the processing characteristics.

Unblocked Fixed-Length Records



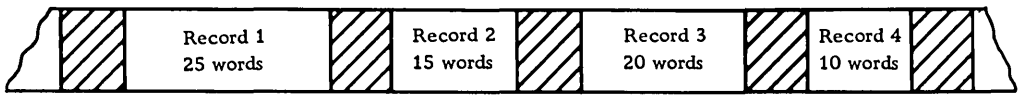
Blocking Factor = 1
Record Length = 25 words

Blocked Fixed-Length Records with a Fixed Blocking Factor



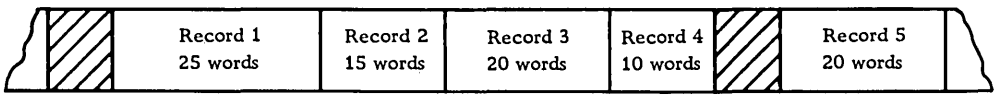
Blocking Factor = 4
Record Length = 25 words

Unblocked Variable-Length Records



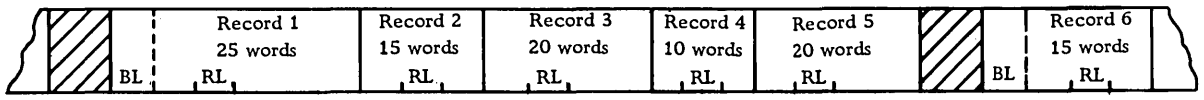
Blocking Factor = 1
Maximum Record Length = 25 words

Blocked Variable-Length Records with a Fixed Blocking Factor



Blocking Factor = 4
Maximum Record Length = 25 words
Maximum Block Size = 100 words

Blocked Variable-Length Records with a Variable Blocking Factor



Blocking Factor = Variable
Maximum Record Size = 25 words
Maximum Block Size = 100 words
Block Length (BL) + /record length (RL) may be required dependent upon specific IOCS.

Figure 5. File formats that can be specified to IOCS

Though the factors in determining the design of data files are presented separately in this manual, no one factor can be considered independently, since all factors are interrelated and must be weighed one against the other to select the best approach. Consideration must be given to the demands of the job, as well as to the hardware requirements.

DETERMINATION OF DATA

The first step in file design requires a study of all procedures that utilize the file. On the basis of the findings, record each necessary item on a worksheet similar to the one illustrated in Figure 6. Indicate type of information, frequency of occurrence, and sequence in source document, if applicable. The following should be done:

- Check that the necessary reference data is included, if this is a source file.
- Weigh the effects of media storage costs vs program execution time for constant-type data, such as tax exempt dollars in payroll.
- Include fields obtained by processing, if the results must be recaptured later.
- Examine all applications that utilize the file to prevent omission of necessary data.
- Explore future requirements of the current procedures. For example, it might be judicious to include an additional deduction field in a payroll application.
- Determine any additional information needed for planned applications. It may be more practical to include an extra field now than to reorganize the files later.
- Study the feasibility of consolidating existing data files into a single data file to eliminate duplication of common information, if such a combined record would not too adversely affect the running time of the volume application.
- Ascertain that material needed in the new application, for which the data file is to be designed, is not available already in an existing data file.
- Verify that the data file, when set up, will contain all the basic information to meet the

FILE DESIGN WORKSHEET

Date Started _____ Completed _____

File Name _____ Designer _____

Process Cycle	Record Characteristics		File Dynamics					File Media Requirements	
DA MO WK YR	Type: Fixed Var.	Character Size MIN MAX AV	NO. REC. A	YRLY % ADD B	YRLY % DROP C	5 YR % GROWTH D	TOT NO. REC E	TYPE	AMOUNT

5(B-C)=D · A+AD=E

Information Required for Processing and Reporting	Type of Information Required	Field Size				Sequence			REMARKS
		TRIAL	TRIAL	TRIAL	FINAL	IN SOURCE DOC	IN RECORD	IN RELATED FILES	

Figure 6. File design worksheet

requirements of all persons who will be using the end products resulting from the file processing.

- Add fields required for technical reasons, such as IOCS requirements for variable-length records.
- Consider file maintenance and audit control.

DETERMINATION OF FIELD SIZE

The number of positions required to record each item of information should be determined and entered on a form similar to that shown in Figure 6.

Determining Factors For Field Size (And Subsequent Record Length)

Type of Field

Control and indicative data field size should equal the total number of digits in the largest single item to be recorded in the particular field. Occasionally, to conserve storage, the high-order digits may be disregarded for a field, such as order number.

Quantitative data field size may equal the total number of digits in the largest amount to be recorded, or the number of digits that will occur with reasonable frequency. Procedures can be developed to handle the rare exceptions. See Omission of High-Order Digit under "Field Compaction Techniques".

Recording Medium

Since some media, such as cards and disks, contain a fixed number of positions per unit of storage (disk sector or track, etc.), it is essential to consider this overall limit to design efficient and practical records.

Example:

> Assume a DASD composed of 100-character disk sectors that can be read consecutively in blocks of 200. If a disk record plans to 82 positions, it would be better to reduce the record to 80 positions, since five such records would fit into four sectors with no unused positions. By beginning the first record of the file in an address divisible by four, no extra programming for hardware capability is required.

Processor Characteristics

The length of a field must be communicated to the equipment for proper handling of data.

Unit record or character machines, in which every position of core storage is addressable, impose few restrictions, since hardware characteristics, such as word marks or next non-number character, are utilized to control the movement of data.

In fixed-word-length machines, in which only a group of positions of core storage is addressable, size becomes critical. It is common practice to pack two or more fields into a single word, but if the fields require different signing, special consideration is necessary, since normally provision is made for only one sign per word. Also, it may be more costly in time to extract portions of words or bridge words (one field to two words). Depending upon the computer, alphabetic data may require two core storage positions. Normally, alphabetic and numeric data cannot be stored in the same word unless both are treated alphabetically. The tradeoffs between time, core storage, and media storage must be kept in mind. Some computers possess the characteristics of both character and fixed-word-length machines, thereby making possible complete flexibility of design.

File Size (Total Number of Records)

Since the field size affects the total record size, all unnecessary positions should be eliminated to decrease I/O time and storage media requirements.

Future Requirements

If the demands to be placed on the information indicate that the need for another position is impending, it would be easier to incorporate the additional character in the design phase so as to avoid rewiring or reprogramming and a patched-up record layout.

Field Compaction Techniques

Because a reduction in the length of a record produces such positive results as an increase in DASD packing and a decrease in time to read and/or write, field compaction techniques should be investigated and the cost of the technique evaluated as each file is designed. Some methods to consider for reducing the number of positions are:

- Decimal Scaling

Whole numbers, or decimal numbers with a fixed number of decimal positions are called fixed-point numbers. In dealing with very large whole numbers or very small decimal numbers, it is necessary to store a great many zeros solely for decimal positioning. In decimal scaling, only the significant digits are retained; the fixed number of zeros omitted from either the right or the left of the

significant digits is known as the decimal scaling factor. This factor must be known and considered for manipulation of the data. It is negative if the new decimal point is moved to the right; it is positive if the decimal point is shifted to the left.

Example:

Variable	Unscaled Number	Scaled Number	Scaling Factor
X	.00123	.123	-2
Y	100.00	.100	3
Z	.987	.987	0

All factors of X would have a -2 scaling factor.

All factors of Y would have a 3 scaling factor.

All factors of Z would have a 0 scaling factor.

If X and Y were multiplied, the product would have a scaling factor of 1 (3-2).

- Floating-Point Numbers

The difference between decimal scaling and floating point is that in the former the number of zeros omitted is fixed for all numbers represented in a given field. In floating point, each individual number is transformed into two parts: the significant digits, known as the mantissa, and the variable number of zeros needed to position the significant digits, known as the characteristic. Both mantissa and characteristic are carried in the record. In fixed-word-length machines, a constant is often added to the characteristic so that it can be treated as a positive value. This leaves the sign position free for the mantissa. Either program subroutines or floating-point hardware are capable of handling such numbers.

Example:

Assume an 8-position mantissa and a 2-position characteristic in a 10-position fixed word.

Actual number	Mantissa	No. of pos. dec. moved	+ Constant	= Char.	Floating-point number
0000.0004583	.45830000	-3	50	47	+4745830000
-1396.430000	.13964300	4	50	54	-5413964300
0000.321659	.32165900	0	50	50	+5032165900

- Omission of High-Order Digit

In quantitative fields, when the maximum number of digits rarely occurs, the field length can be shortened. When significant digits do occur, an overflow condition results. Computers will turn on an overflow indicator, which can be tested, and an alternate subroutine can be executed when applicable. In media storage, the overflow digit can be represented by some type of zoning, such as x overpunch in cards or A/B bits over the high-order position, as in 1400-series computers. This is typical of the flagging compaction technique, where the use of a character replacement indicates a given following condition within a field.

- Variable-Length Fields

Only significant digits are recorded, and fields are separated by a special symbol, such as plus or minus. The number of fields for a given record is constant, and each field is identified by its position within the sequence. A small routine in the computer expands each field to the required size. Savings are greatest when the total of the average number of significant digits plus one for each field is less than the sum of the maximum field sizes. This technique is of greatest value for input preparation, where transmission line cost and transcription time can be reduced by its use. This is an example of the marking compaction technique, which uses a special mark or symbol to indicate the beginning or the end of a given condition.

- Bitting

Multiple items can be stored in a character or group of characters by partitioning into bit notation, where each bit has a specific value or control function. Thus, bit 1 may represent active or inactive; bits 2 and 3, one of four credit ratings; and bits 4 and 5, one of four age classifications. Care must be taken not to develop combinations that are invalid to a particular hardware system.

- Coding

Coding may be used to replace a larger field. During processing, the codes can be interpreted or translated to the constants they represent. For example: a one-position code may represent the unit of measure of inventory items--a 1 for dozens; a 2 for ounces; a 3 for pounds; etc.

- Heading

Heading takes advantage of redundant information. Thus, one header containing information common to a series of items can precede its respective detail items. For example: spread input records, such as might be used in a billing operation--
date/order no./cust. no./item 1/item 2/----/item n/

- Substituting

Substituting makes use of the number of free bits that appear in a given character set or, in the case of EBCDIC, some of the less used combinations, such as the lower case alpha representations. One of these characters replaces more than one other character, primarily numeric pairs. For example, to reduce month from a two-digit numeric code, the letters A and B can be substituted for November (11) and December (12).

• Table Lookup

Use of table-lookup techniques--whereby such factors as rates, constants, or other types of information are stored in core in table form--permits a given field to be reduced to a code. This differs from straight coding methods that substitute codes permanently for given values, because it allows the shortening of a data field without precluding retrieval of the longer values that the codes represent. The size of the table is limited only by the storage capacity of the system involved.

Example:

Use of code for a management budget report.

	Table	
Search Argument (coded input field)	Table Argument (same coded data normally stored in ascending or descending sequence)	Function (expanded field information)
101 (dept. code) →	100	Accounting
	101	Accts. Receivable
	102	Accts. Payable
	103	Internal Audit

Coupled with an error routine for unlocated search arguments, table lookup also serves as an excellent validating tool.

• Binary

The binary system uses only two symbols (0 and 1) as opposed to ten symbols (0-9) for the decimal system. Since the position value of these digits is based on the powers of 2 rather than of 10, the units position of a binary number has the value of 1; the next position, a value of 2; the next, 4; the next, 8; the next, 16; and so on.

8	7	6	5	4	3	2	1	0
256	128	64	32	16	8	4	2	1

Place value of binary numbers

4	3	2	1	0
10,000	1000	100	10	1

Place value of decimal numbers

The decimal number 20 expressed in binary would be 10100 or $(1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0)$ or $(1 \times 16) + (0 \times 8) + (1 \times 4) + (0 \times 2) + (0 \times 1)$.

Since binary notation requires only two symbols, one bit can be used to represent each place value of a number. In standard decimal format (BCD or EBCDIC), six or eight bits are required for each place value of the number. (Refer to Figure 27.) Thus, in binary the eight bits 11111111 represent the decimal number 255. Relating this to media or core storage capable of handling eight bits per unit of storage (one column on tape, one byte in core, etc.), only one unit is required for binary, whereas three are needed for decimal. Binary notation is a very effective compaction technique for numeric data, providing the computer characteristics lend themselves to efficient handling of such data.

• Hexadecimal

Hexadecimal uses a base 16 (as opposed to 2 and 10 for binary and decimal respectively) and 16 symbols (0-9, A-F). It is used mainly as a compact notation for binary to facilitate man-machine communication. Any four digits of binary have a maximum value of fifteen. Therefore, one hexadecimal digit can be used to represent four binary digits. To illustrate:

2F	=	0010 1111	=	47
hex.		binary		decimal

• Packed Numeric Format

Numeric data represented in BCD or EBCDIC requires only four bits (1, 2, 4, or 8) to represent its value. In storage units capable of handling eight bits per unit of storage, the packed numeric format takes advantage of this characteristic by placing two numeric characters into one unit of storage. To illustrate:

8-bit code	Packed format	Example of numeric representation in packed format	
C	C	0	1 } check bit
0	8	1	0
1	4	0 } numeric	0 } numeric
2	2	0 } value of 9	1 } value of 3
3	1	1	1
4	8	0	0
5	4	1 } numeric	1 } numeric
6	2	1 } value of 7	0 } value of 5
7	1	1	1

Packed Format

To operate most efficiently in packed format, the computer should contain in its instruction set codes to pack and unpack, as well as to operate arithmetically when in the packed mode.

Evaluation of Compaction Techniques

A given compaction technique must be evaluated for:

1. Amount of memory (core) required to hold the endode-decode instructions
2. Encode-decode subroutine timing requirements
3. Compaction percentage acheived
4. Compatibility with programming systems
5. Retention of collating sequence
6. Retention of fixed field length
7. Effect on the overall system, including related clerical functions

For a discussion in depth of compaction techniques, see Record Compaction Techniques (E20-8252).

DETERMINATION OF DATA SEQUENCE

Data sequence is most critical for those files that work with source documents. Card punching, terminal operation, etc., being manual operations, are subject to the greatest variation in rate of production. Anything that simplifies these functions tends to ensure a faster and more accurate operation. The following are points to bear in mind:

- Recording of data in the same order as that in which it is normally read. If the data sequence is considerably different from that on the source document, it may be necessary to redesign the source document and retrain personnel. If the file is to be used as input to a serial I/O unit, such as tape to card, the sequence is dictated mainly by the sequence desired on the output unit.
- Location of like fields in the same relative record positions in files that work together. This assures that sorting and controlling can be accomplished if the file is contained in cards; it also facilitates programming.
- Placement of sorting fields adjacent to one another, with the minor code on the right and each progressively higher code to the left. Although sort programs can operate on multiple-control fields, time is used to extract and combine fields into a single key.
- Availability of results supplied by machine to serial output unit.
- Compatibility with computer characteristics so that data sequence does not affect processing speed. For example, with the 1400 series, field sequence, size, and grouping determine whether instruction chaining can be used.
- Arrangement of alphabetic/alphameric data in one area of the record. This facilitates handling of data, particularly in fixed-word machines, and permits minimum core and media requirements.
- Frequency of occurrence of each field. If it is decided to use variable-length records because some fields are not present in all records, the variably occurring fields should be last in the record to keep the fixed fields in the same relative location on each record.
- Adherence to requirements of programming systems. For example, the block-length field specified for variable-length records normally must be the first field in the block.

DETERMINATION OF FILE ORGANIZATION

For strictly card- and/or tape-oriented systems, file organization normally is sequential. Therefore, the following discussion is oriented mainly toward the design of DASD data files.

Sequential vs Random Organization

Sequential Advantages

- Both sequential and random transactions can be handled effectively in most cases.
- Reports arranged in data file sequence can be obtained without sorting.
- Control over both the processing and the stored file can be more positive.
- Less medium storage space is required.
- Frequently the entire file need not be on line simultaneously.

Sequential Disadvantages

- More core storage may be required because of index handling routines.
- Process time is greater for random input because of index file seeking and processing.

Random Advantages

- Less core storage is required normally.
- Process time is less for random input.

Random Disadvantages

- To maintain access requirements, frequent reorganization may be necessary if the file is dynamic.
- Extensive key analysis and development of address conversion routines probably are required for implementation.

File Organization Techniques

Sequential Techniques

- Control Sequential

Records are written initially on the DASD file in a sequential manner in the primary file area. As new records are inserted into the file, they are written into available record space in a separate overflow file area. Every record in a control sequential file must have a sequence linkage field, which contains (when required) the actual DASD address of the next sequential record. This field is utilized in a record in the primary area only when the next sequential record is stored in the additions area. All records in the additions area utilize the sequence linkage field, which shows whether the location of the next sequential record is in the additions or the primary area. A dummy record is the first record in the data file to permit additions with lower record keys. As the file is loaded, a distribution index is created, which makes it possible to locate records at random. The index is a single-level index that contains the record control key and the DASD address, from selected records located in the file. The frequency of the entries can be chosen to satisfy the control and retrieval needs of the application. The first and the last entry in the file are always given. The addition of overflow records does not affect the index unless a record higher than the last record in the file is to be added. (See Figure 7.)

Control sequential files are reorganized periodically to reduce access time for all records in the file. Reorganization results in the placement of all records sequentially in the primary area, with the entire additions area and all sequence linkage fields cleared.

This method should be considered where processing is primarily sequential, with some random input where access time is not critical. It is supported by IBM Programming Systems (load, add, tag for deletion, delete, and unload programs) for the 1401, 1440, and 1460 Data Processing Systems with the IBM 1311 or 1301 Disk Storage Units (except for the 1301 on the 1401). For further information refer to 1401/1440/1460 File Organization Specifications (C24-3185).

- File Organization System

The File Organization System (FOS) is supported by the IBM 1410/7010 Operating System, which provides programs to load and maintain files for the IBM 1301 and 1302 Disk Storage Units. This technique uses a two-level index (cylinder and track), although a third level (master) is generated if the cylinder index grows too large. All overflow records cause an entry into the track index. Therefore, no sequence link field is required to retrieve overflow records. (See Figure 3.) For further details refer to 1410/7010 File Organization System (C28-0405).

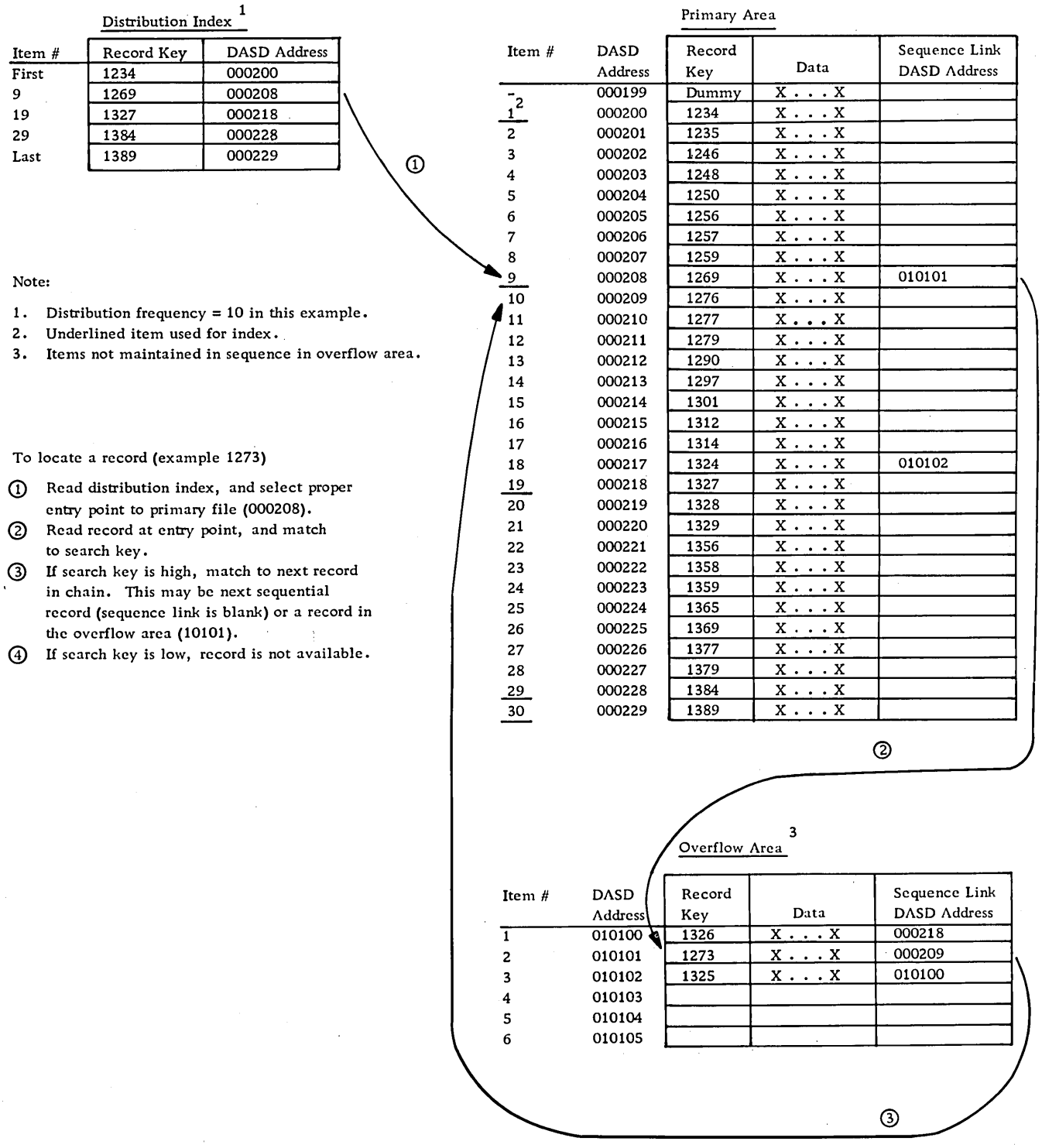
Index Sequential -- 1. In this method, additions are inserted in the data file in their correct sequential location by moving the records with higher keys down the track. This may result in the shifting of a record from the end of the track into an additions area. The most practical location for the additions area is in a single overflow area, although it is possible to leave the tracks unpacked or to create an additions area within each file cylinder, provided that an even distribution of additions is anticipated.

Normally, at least two levels of index files (cylinder and track) are created, although with larger files, a master cylinder index may be warranted also. A new entry must be made into the track index whenever a record is moved to the additions area. This method is effective for the storage of sequenced files of fixed-length unblocked records with the IBM 1301 or 2302-1 and 2 DASD units. The technique is not program-supported. For an example of the use of Index Sequential --1, see Figure 8.

Index Sequential -- 2. Records are stored in sequence by their record key, and their processing and location are controlled by a two- or three-level index.

1. Level 3, or master index, is not required but reduces search time when a number of cylinder index tracks occur. Each entry consists of a specific cylinder index track address and the highest key present in that track.
2. Level 2, or cylinder index, must exist. Maintained in record key sequence, each entry contains the highest record present in each data file cylinder, as well as the track address of that cylinder's level 1 (track) index.
3. Level 1, or track index, must exist in each cylinder used to store data file records. There are two entries for each track in the primary area:
   a. Normal entry, which contains the key of the last record and the address of the first record on the track.
   b. Overflow entry, which contains the highest key of an overflow record from a prime data track, along with the address of the overflow record with the lowest key.

The index sequential -- 2 technique combines the insertion technique of the index sequential -- 1 method with the concept of chaining for the records that are shifted to the additions area. An addition is inserted into the data file in its proper sequential location (through replacement of some record), and the remainder of the track is shifted to the right.



Note:

1. Distribution frequency = 10 in this example.
2. Underlined item used for index.
3. Items not maintained in sequence in overflow area.

To locate a record (example 1273)

① Read distribution index, and select proper entry point to primary file (000208).
② Read record at entry point, and match to search key.
③ If search key is high, match to next record in chain. This may be next sequential record (sequence link is blank) or a record in the overflow area (10101).
④ If search key is low, record is not available.

Figure 7. Control sequential organization

This causes the last record on the track to be moved to the additions area, along with a linkage field. The overflow entry of the track index is updated. Thus, the records in the primary area tracks are in sequence with keys lower than any that have overflowed that track. The overflow area contains records in sequence by time of arrival but retrieved in record key sequence for any given track by use of the sequence linkage field and overflow entry in the track index. (See Figure 9.)

Designed for use with the 2302-3 and 4, 2311, 2314 and 2321-1 DASD units, the index sequential

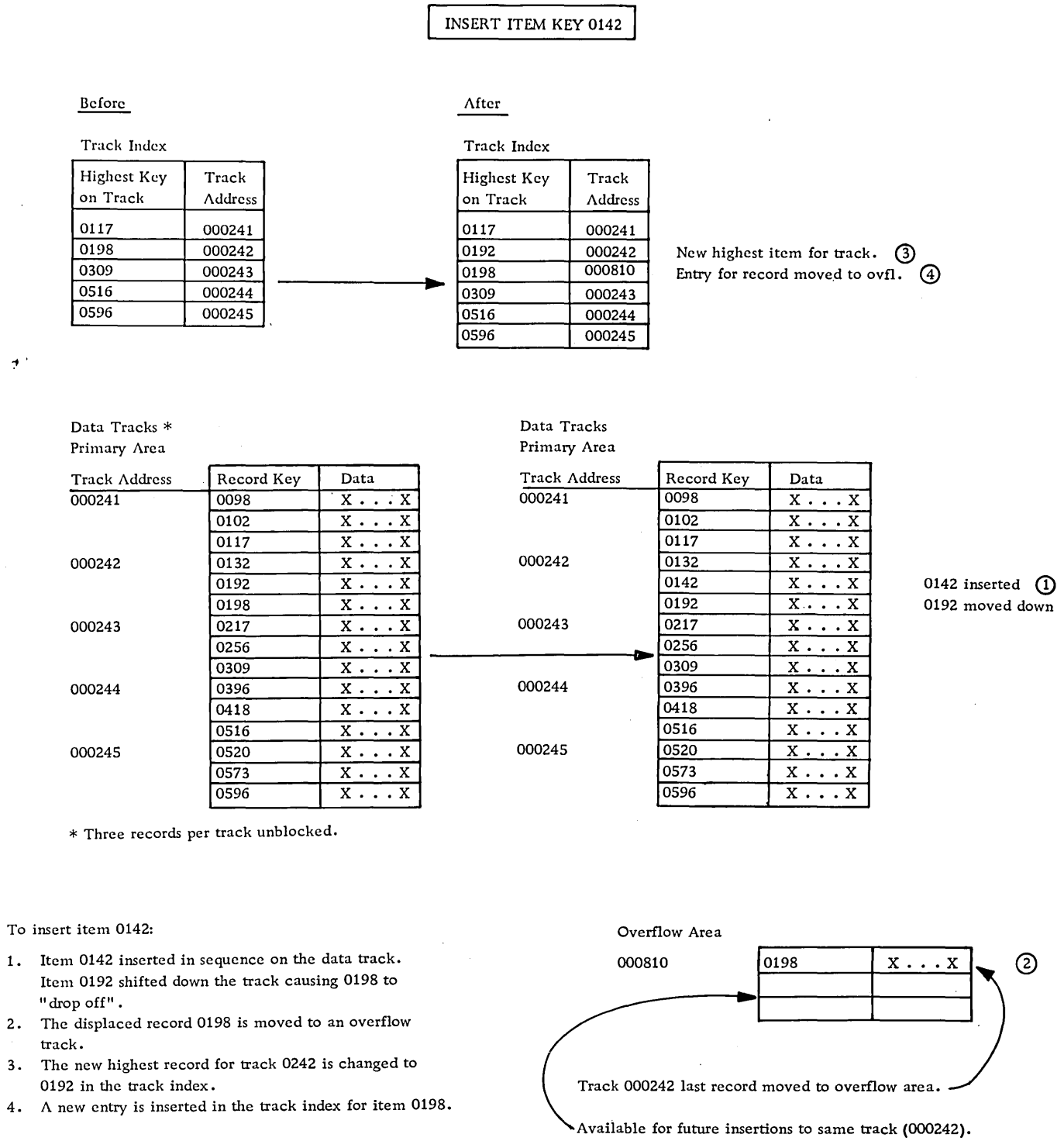


Figure 8. Index sequential — 1 organization

-- 2 organization technique is compatible also with the 2301 and 7320 Drum Storage unit. Fixed-length records in either blocked or unblocked mode, as well as variable-length records, are handled. No track index entry repositioning is required to handle an inserted record, nor is rearrangement of the overflow area required. During processing, the records can be flagged for deletion. Periodic file reorganization is required.

- Direct Addressing

In direct addressing, the record key is equal numerically to the DASD storage address of the storage

Before

Track Index

Normal Entry | Overflow Entry*

High Key in Track	Low Track Address	High Key in Chain	Low Address
0117	000241	0117	000241
0198	000242	0198	000242
0309	000243	0309	000243
0516	000244	0516	000244
0596	000245	0596	000245

Data Tracks - Primary Area**

Track Address	Record Key	Data
000241	0098	X...X
	0102	X...X
	0117	X...X
000242	0132	X...X
	0192	X...X
	0198	X...X
000243	0217	X...X
	0256	X...X
	0309	X...X
000244	0396	X...X
	0418	X...X
	0516	X...X
000245	0520	X...X
	0573	X...X
	0596	X...X

Data Tracks - Overflow Area***

Track	Link	Key	Data

After 1st Insertions

Item Key: 0142; 0450

Track Index

Normal Entry | Overflow Entry

High Key in Track	Low Track Address	High Key in Chain	Low Address
0117	000241	0117	000241
0192	000242	0198	000810-1#
0309	000243	0309	000243
0450	000244	0516	000810-2
0596	000245	0596	000245

Data Tracks - Primary Area

Track Address	Record Key	Data
000241	0098	X...X
	0102	X...X
	0117	X...X
000242	0132	X...X
	0142	X...X
	0192	X...X
000243	0217	X...X
	0256	X...X
	0309	X...X
000244	0396	X...X
	0418	X...X
	0450	X...X
000245	0520	X...X
	0573	X...X
	0596	X...X

Data Tracks - Overflow Area

Track	Link	Key	Data
000810	000242	0198	X...X
	000244	0516	X...X

After 4th Insertion

Item Keys: 0196; 0199

Track Index

Normal Entry | Overflow Entry

High Key in Track	Low Track Address	High Key in Chain	Low Address
0117	000241	0117	000241
0192	000242	0198	000810-3
0256	000243	0309	000811-1
0450	000244	0516	000810-2
0596	000245	0596	000245

Data Tracks - Primary Area

Track Address	Record Key	Data
000241	0098	X...X
	0102	X...X
	0117	X...X
000242	0132	X...X
	0142	X...X
	0192	X...X
000243	0199	X...X
	0217	X...X
	0256	X...X
000244	0396	X...X
	0418	X...X
	0450	X...X
000245	0520	X...X
	0573	X...X
	0596	X...X

Data Tracks - Overflow Area

Track	Link	Key	Data
000810	000242	0198	X...X
	000244	0516	X...X
	000810-1	0196	X...X
000811	000243	0309	X...X

* Overflow entries are same as normal entries until overflow occurs.

\# Suffix indicates location of record on track.

** Records blocked 3 to a track.

*** All overflow records are unblocked.

Figure 9. Index sequential — 2 organization

location containing the record. This results in a sequential file that requires no indices, no additions area, and no special programming. The record keys must fit into the range of addresses for the particular DASD unit. To use this method, it may be practical, on occasion, to consider assignment of a new set of codes to the file. The entire range of addresses covered by the record keys must be reserved for the file. Any unused record key causes the corresponding storage location to be empty. Consequently, effective utilization of a storage unit may be affected.

Figure 10 illustrates the relationship between the record control key and the DASD address.

Some manipulation of the record key is possible to achieve compatibility between the storage medium and the record keys, as:

1. Multiplication of the key by a factor to create an effective storage address where the record length does not equal the length of the addressable unit.

Record Key	DASD Address
21320	021320
21321	021321
	021322*
21323	021323
	021324*
	021325*
21326	021326
21327	021327
etc.	etc.

*Empty locations, since no records exist with matching keys.

Figure 10. Example of direct addressing storage

Example: If 200 character records are to be stored in a 100-character addressable unit, the key should be doubled.

2. Omission of the units position of the key; this results in the assignment of ten records to the given storage address.

Random Techniques

Record Key Conversion. In a random file organization the records are stored at an address that has been derived from the record control field or key. To select the proper address conversion routine, an evaluation of the calculated addresses must be made in terms of unique addresses, synonyms, packing factor (percentage of the allocated file actually used for records), and the time required to retrieve records. Programs, such as AUTOPAC II (1401-1.4.172), are available to assist in the analysis and evaluation of proposed address conversion routines.

Prime number division (divide/remainder) is a flexible, simple record conversion method. The following example illustrates its use:

Assume a data file that is to occupy 5000 storage address locations, with record keys ranging from 11111 to 99999.

1. Select a divisor. This should be a prime number (a number divisible only by itself and 1) or a number ending in 1, 3, 7, or 9. A number slightly less than the number of storage tracks or sectors assigned to the given file is a common choice. If the control number is greater than six digits, 999983 may be selected. Use 4999 for this example.

2. Divide the record control key by the selected prime number, and use the remainder to generate the track address.

Example:

If the record control key is 19800,

$$19800 \div 4999 = 3 \text{ with a remainder of } 4803$$

Adjust the remainder to bring it into the range of selected DASD addresses. If the addresses range from 012000 to 016999,

$$4803 + 12000 = 16803$$

Therefore, 16803 is the converted address for 19800. By applying the same formula, the numbers 29798, 44795, and 94703 create duplicate addresses or synonyms.

Overflow Organization Techniques. Because record key conversion routines create duplicate addresses, a method must be found to store and to retrieve the synonym records. Chaining is the technique used most often.

The first synonym is stored in the original address (home record) developed by the record key conversion routine. To reduce seek time, the additional synonyms are stored in overflow locations (unused home addresses) located near the original home record. The DASD address of the first overflow record is placed in the home record; the address of the second overflow record is stored in the first overflow record; etc. A field for the overflow address is reserved on every record but is used only when a link must be added to the chain of synonyms. This is illustrated in Figure 11.

To locate a record, the record address is created by the record conversion routine. The home record is read and its key compared to that of the record being sought. If the keys are unequal, the next record in the chain (as determined by the overflow record address) is read, and the keys are compared. These steps are repeated until the record is located. Each link in the chain requires a read and, in some cases, a seek. Therefore, the length of time needed to find a given record depends upon its location in the chain. Those records which are most active should be stored first in the chain to

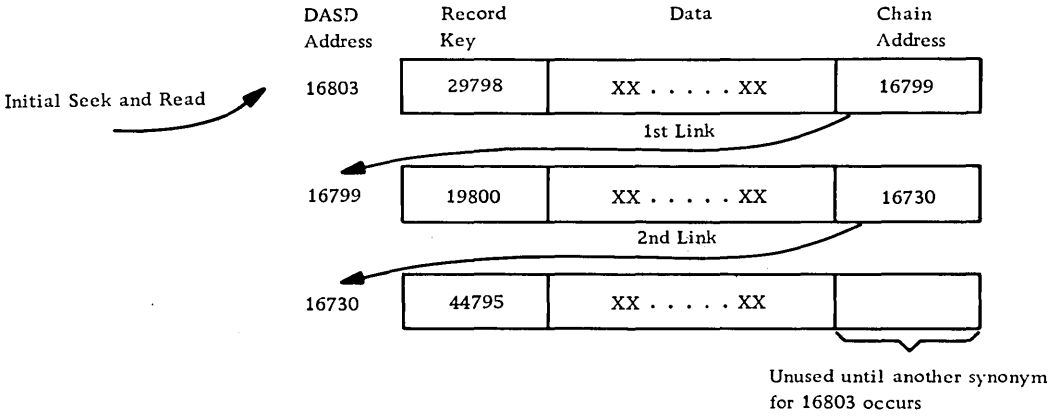


Figure 11. Example of chaining, unblocked records

minimize retrieval time. To assist in determining activity, a count that is updated by one for each usage, may be maintained as part of each record data. If records are blocked, each seek and read makes available a number of synonyms equal to the blocking factor. Refer to Figure 12 for an example of chaining where blocked records are used.

The handling of trailer records can also be accomplished by use of the chaining method. The trailers are loaded into a separate area of DASD storage. Their addresses are stored in the overflow address field of the corresponding master.

The random file organization routines for 1401/40/60, with 1311 or 1301 (except 1401), use the chaining random file organization technique. Included are routines to random load, to add records, to tag for deletions, to delete tagged records, and to unload. For further details refer to IBM 1401/1440/1460 Disk File Organization Routines--Specifications, (C24-3185).

DETERMINATION OF RECORD FORMAT AND BLOCKING

To select the record format and blocking, each of the following factors must be considered:

File Boundaries

Paper and magnetic tape are continuous recording media and, hence, present no file boundary problems. Cards are limited to 80 columns of punched data, while each DASD has a specific maximum number of positions that may be recorded on a given track or sector. Comb access arms must be moved between cylinders, and different read/write heads may have to be selected between tracks. Although these conditions can be handled, extra programming steps, which cost both in processing time and in core storage requirements, are necessary. When designing a file, consider the following:

- For fixed-length records, select a record length that, when blocked, fits well into a track.
- For variable-length records, limit the total block to a track.
- If the DASD operates in sector mode, multiple sectors may be read or may be written with one command. This allows track-to-track operation but not cylinder-to-cylinder. Therefore, for most efficient operation (assuming blocked fixed-length records which are to be stored in tracks which bridge cylinders), the total number of sectors read or written by a single operation code should be divisible evenly into the total number of cylinders for the DASD. The starting address for the file plus the number of sectors per read or write -1 must the divisible by the same factor.

Core Storage Requirements

Since IOCS handles physical records for input/output operations, a core storage area large enough to accommodate the physical record is required. In addition, for efficient operation, multiple I/O areas may be required for the I/O devices. A work area the size of a logical record may be desirable.

File Capacity

The storage of records in an unblocked form normally lowers the gross data capacity of a given storage medium. This results from the presence of interrecord gaps, or unused storage positions, in an addressable unit of storage on a DASD. However, unblocked records are the easiest to process and require a minimum of core storage (no deblock/block routines). For further details refer to the discussion of magnetic tape timing and capacity formulas and DASD capacities in section 3.

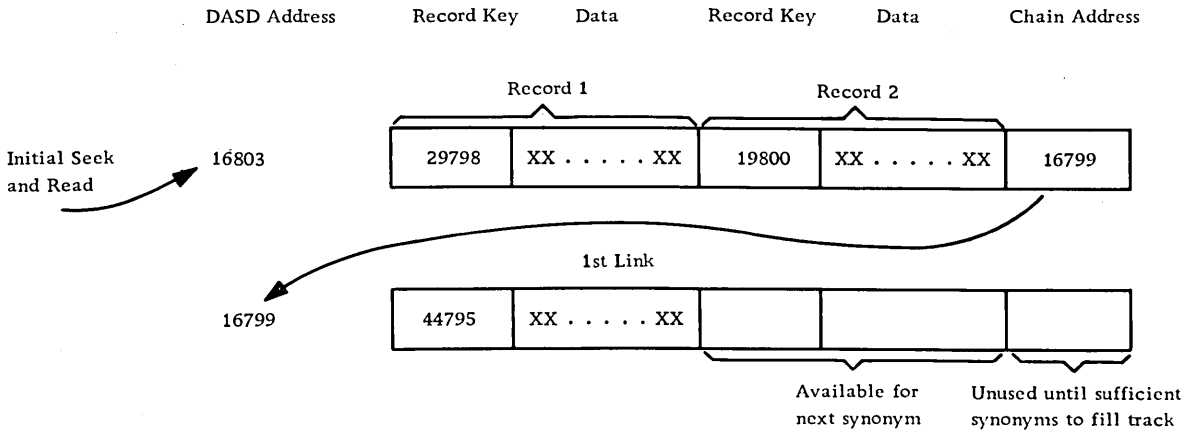
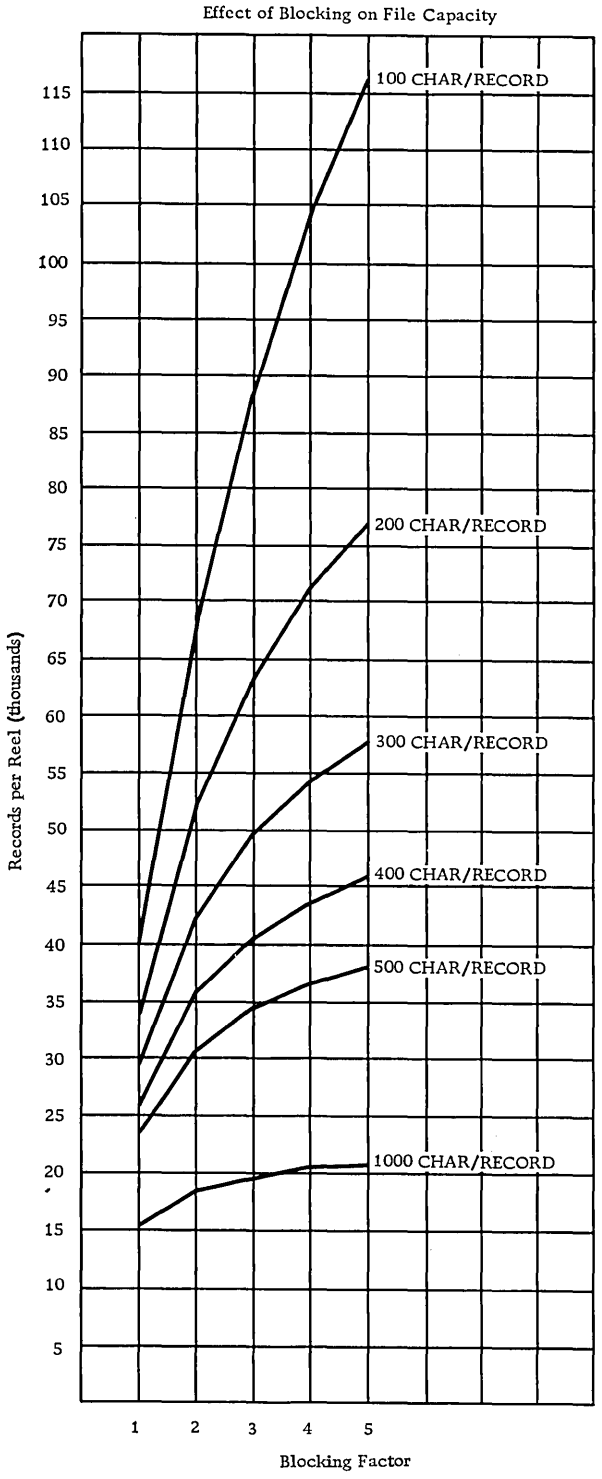


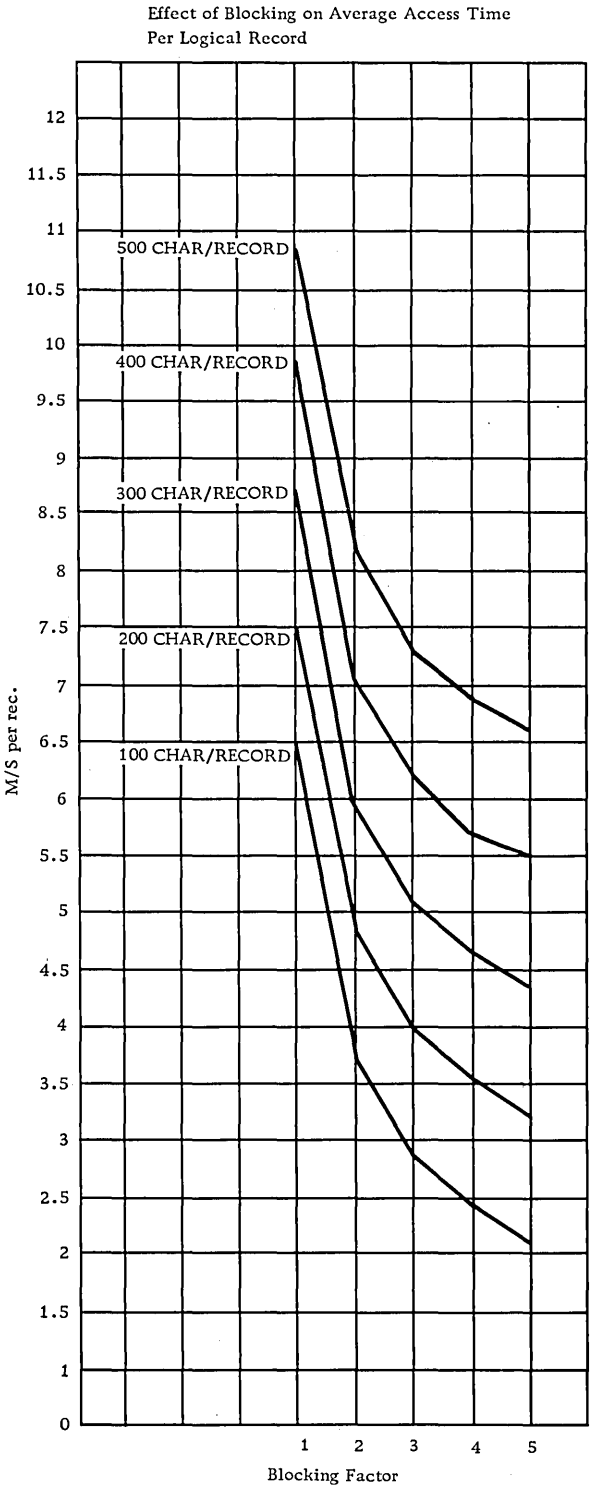
Figure 12. Example of chaining, blocked records

The tables in Figures 13 and 14 indicate the relationship between blocking and file capacity or access time. A 2400 model 3 tape drive with 800 density has been used for comparative purposes.

Note that the gains are in proportion to the size of the logical records.



● Figure 13. Relationship between blocking and file capacity



● Figure 14. Relationship between blocking and average access time

Program Support

Just as certain file organizations are IBM program-supported, certain record formats are supported. These will vary with the IOCS, the computer system, and the file organization. A chart of IOCS characteristics appears in section 3.

Other IBM utility programs, such as sorts, must be considered when establishing record formats, since not all formats allowed by IOCS may be handled by the other programs. Sometimes files are prepared on one computer system for use on another. Therefore, the characteristics of both systems must be considered.

Sample record and label formats are given in section 3.

FILE PROCESSING

Before the file design is finally determined, the run time and associated costs should be calculated for the entire system. The results must be evaluated to determine whether the original design objectives have been met. If the system is input/output limited (I/O time exceeds process time), the following approaches may be considered:

- Create a second master file splitting away from the main master file those fields not required on the primary runs. For example, name and address records could be kept in a separate name and address file. This new file would be used perhaps only as output documents are printed.
- Extract from the master file the active records for processing. This method is useful for tape-oriented systems, where the ratio of active master records to total master records is very low. During the extraction run, the updated active records of the previous day are filed back into the master tape. Usually a high blocking factor can be used for the master, since the extract program is simple and requires little core.
- Use a change-tape approach if the skew distribution against the master is great (for example, if 85% of the activity occurs against 15% of the file). Applicable to tape, this system is a variation of the extract procedure, since it divides the master file into active (change-tape) and inactive segments. As activity occurs against a master record, it is moved to the change-tape. Periodically, an extra run is needed to return the updated records from the active file to the inactive file, since the system loses its effectiveness when the size of the change-tape equals one-half of the inactive file.
- Split the master file into two or more sections, and organize the transaction records in the same fashion. (Use separate drives if necessary.) By processing all sections of the file concurrently, portions of the file may be searched for the next active item while an active record is being processed. The split-file technique is useful in tape approaches, where the activity rate is low, but process time per active record is high. The split-file technique requires buffering and an increase in channels and tape units to be effective.
- Increase the number of input buffers. If the activity rate is low, and processing time per hit is high, more process time can be overlapped if the input is queued in additional buffers. If process time requires 250 milliseconds, while an input area can be filled in 50 milliseconds, there would be 200 milliseconds of unoverlapped process time per hit, with two input areas. If the number of input areas were increased to four, only 100 milliseconds would not be overlapped.

FILE CONTROL

The design of a data file cannot be divorced from the environment in which the file must function. Some of the considerations of file control and maintenance are now discussed.

Data Validation

The entry of incorrect data into a file should be prevented. The following techniques are suggested methods that may be used to control the accuracy of input data:

- Precoded forms, or standardized and simplified forms, which reduce the possibility of error at the point of origin of the data.
- Batch controls that establish totals for a given group of records to detect the loss or distortion of data during intermediate handling. A batch may consist of a fixed number of items or the transactions that occurred in a given period of time. Typical batch totals are record counts, dollar or quantity amounts, or "hash" totals of significant data, such as wage rates. Frequently, batch totals are recorded in a trailer record to provide automatic zero-balance checking.
- Turn around documents, such as prepunched remittance forms, which require little or no extra recording and a minimum of handling.
- Character checking, which determines whether the data in given positions of the record contain permissible characters. This type of check can be used to ensure that the proper algebraic sign is present for the type of transaction or that alphabetic data is not included in numeric fields, and vice versa, or that data is present where required (not blank).
- Field checks that examine the contents of a field for certain characteristics. These include:

1. Limit checks, which determine whether data is within a prescribed range. Such checks can apply to such fields as employee's wage rate, amount of gross pay, etc.

2. Historical checks, which use prior experience as a basis of validation. The public utility industry often compares, for reasonableness, prior consumption for a year or more against the current usage.

3. Validity checks, which compare the contents of a field against a list of existing "good" numbers. This prevents posting to nonexistent account numbers. Matching by control key against a master file indicates duplicate and missing numbers.

4. Logical relationships, which determine whether the items of input data have a logical relationship to one another or to the file they affect. For example, if an employee adds a bond deduction, a bond denomination is also required.

5. Self-checking numbers, which detect incorrect identification numbers (such as account number, employee number, etc.) by performing certain mathematical calculations on the base number and comparing the resulting digit against a check digit appended to the base number.

Operating Controls

The following controls are common methods used to detect errors caused by poor operator performance, equipment failure, or malfunctioning programs:

- Label checking, which verifies that the proper file is online before any processing can take place by checking such data as file identification, creation date and serial number, which are contained in the label record of the file.
- Record counts, which check that the numbers of records before and after processing are the same, in order to guard against accidental loss of a record.
- File totals, which ensure that the file is in balance in light of the transactions just processed. For example, the previous file total for a given field plus the net change represented in the transactions should be equal to the sum of the individual record fields after the transactions are processed.
- Intervention logging, which records through a console typewriter any intervention by the operator.
- File restrictions, which, through programming or hardware, prohibit access to records stored within specified areas.

Error Analysis

The file control techniques suggested above indicate the wide variety of methods available. Selection of the specific control procedures depends on such factors as the frequency of possible occurrence, the results were the error allowed to enter the system, and the chance that the error might remain undetected even through later operations. All errors should be logged indicating the nature and the cause. A review of these error logs can serve as a guide to management for the increase or decrease of error control.

When errors are detected, any of the following procedures can be used:

- Programmed halts, where the computer is halted by detection of certain conditions, and the operator follows prescribed steps dependent upon the nature of the halt. The trend is away from programmed halts to eliminate operator intervention.
- Bypass procedures, where the error condition is recorded on some output medium, as tape or printer, for later analysis, and the computer continues without stopping.
- Suspense accounts, where totals are posted for invalid records to keep all items requiring analysis in a single account.

Audit Trail

An audit trail may be defined as the means whereby the source transaction and its corresponding supporting documentation can be related to processed data. Although the audit trail may be a by-product of normal processing, it may sometimes be additional. The requirements of the auditor should be discussed to provide the necessary historical information trail.

Reconstruction

If the information on a file is mutilated, the need for reconstruction arises. The method selected depends upon such factors as job priority, the time and cost required to provide reconstruction data, and the time and cost required to perform the reconstruction. Listed below are several approaches:

1. Tape

- If a dynamic file is maintained on tape, several generations should be kept. Referred to as the "grandfather, father, son" system, the retention of the two previous files, coupled with access to the transactions that have occurred since the last update, provides a means of reconstruction should anything happen to the current-generation tape.
- If a static file is maintained on tape, a copy should be made. Should the need arise to use the copy, a new copy should be made first. In lieu of a tape copy, it may be practical to keep the information in cards and to reconstruct the tape when required.

2. DASD

- Periodically, a dynamic DASD file should be copied (dumped) on tape, on cards, or on another DASD. Often, the copy can be made as a by-product

of a periodic run. All transactions since the last dump must be retained to update the copy to current status.

- To avoid reprocessing of all transactions since the last dump, write the updated records on tape as the transactions are processed against the file. In sequential processing, only one tape record per active disk record is written. In case of reconstruction, the record with the most recent status can be used to replace the corresponding record on the dumped file.

- If no output unit is available to record the updated records, as suggested above, the master can be flagged, and on a later run the flag can signal a copy operation for a given record. This technique requires a rewrite to the file for removal of the flag.

- The contents of a static file should be available either by copy to another DASD, or by dumping onto tape or cards that may be used later to reload the mutilated file.

SECTION 3: REFERENCE MATERIAL FOR DATA FILES

STORAGE MEDIA AND RECORDING CHARACTERISTICS

Cards

IBM cards provide 80 vertical columns with twelve punching positions in each column. The twelve punching positions form twelve horizontal rows across the card. One or more punches in a single column represent a character. Each card is read or punched as a unit of information, but the actual data on a card may consist of a part of a record, one record, or more than one record. If more than 80 columns are needed to contain the data of a record, two or more cards may be used. Continuity in the cards of one record is obtained by punching identifying data into each card.

The standard IBM card code divides the twelve punching positions into two areas: the numeric, which is made up of the first nine punching positions from the bottom of the card and which represent the values 9 through 1, respectively; and the zone, which consists of the remaining positions known as the 0, 11, and 12 positions. The 0 position doubles as both a numeric and a zone code. Numbers 0 through 9 are represented by a single punch (0 through 9) in a vertical column. Alphabetic characters use one of the zone punches with one of the numeric punches. Special characters are combinations of one, two, or three punches in a column and consist of punch configurations not used for the numbers or letters. (See Figure 15.) The standard IBM card code is consistent with the binary coded decimal interchange code (BCD or BCDIC).

Not all of the possible combinations of punches are utilized by the standard IBM card code. The extended binary coded decimal interchange code (EBCDIC) makes use of the free punches to provide representation for more special characters and for lower case letters. Figure 27 shows the punched card codes used by EBCDIC.

To provide direct input or output without conversion from decimal for those computers operating in binary, two methods of recording binary information are available.

In row binary, data are arranged serially across each row of the card, from left to right, starting at the 9 row and continuing through the 12 row. Each of the twelve rows is divided into two parts; the left half consists of columns 1-36, and the right half of columns 37-72. One card can contain twenty-four 36-bit binary words. (See Figure 16.)

In column binary, information is placed in parallel, with each column of the card containing 12 information bits. One 36-bit word requires three columns, and the entire card can contain twenty-six 36-bit binary words. (See Figure 17.)

In both systems a punched hole indicates a binary 1. No punch represents a binary 0. A computer using the EBCDIC system does not require special binary cards, since the code provides for punching in one column all possible binary combinations for eight bits. Thus, four columns of the card can contain one 32-bit word. (See Figure 27.)

Magnetic Tape

IBM magnetic tape is a continuous recording medium similar to the tape used in home recorders. Data is recorded in magnetized spots or bits, is permanent, and can be retained for an indefinite period.

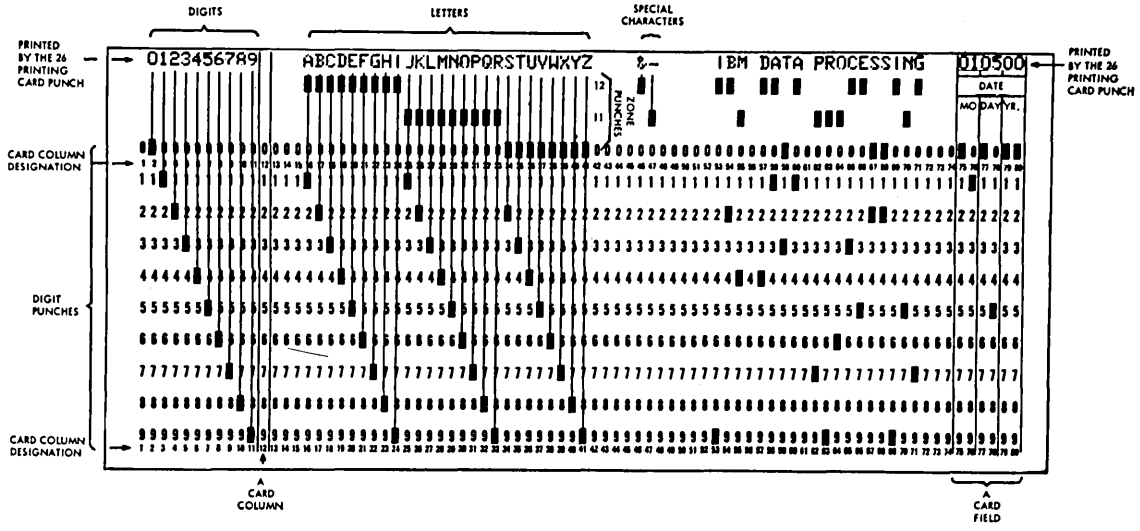


Figure 15. IBM card — standard card code

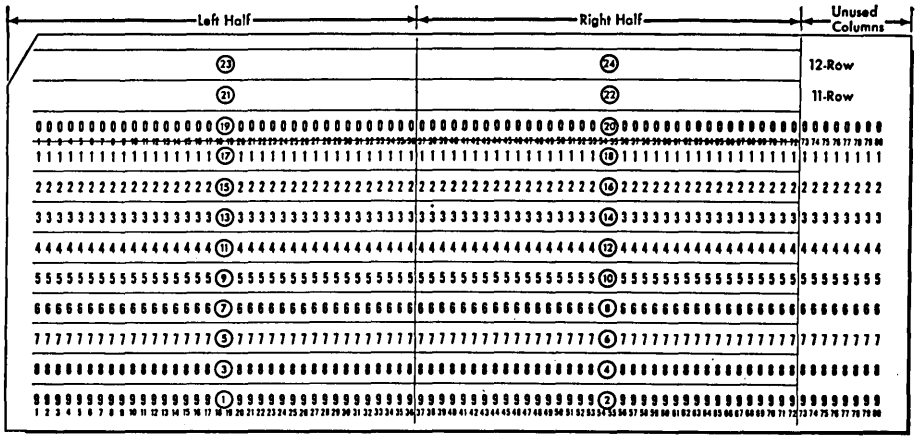


Figure 16. IBM card — row binary

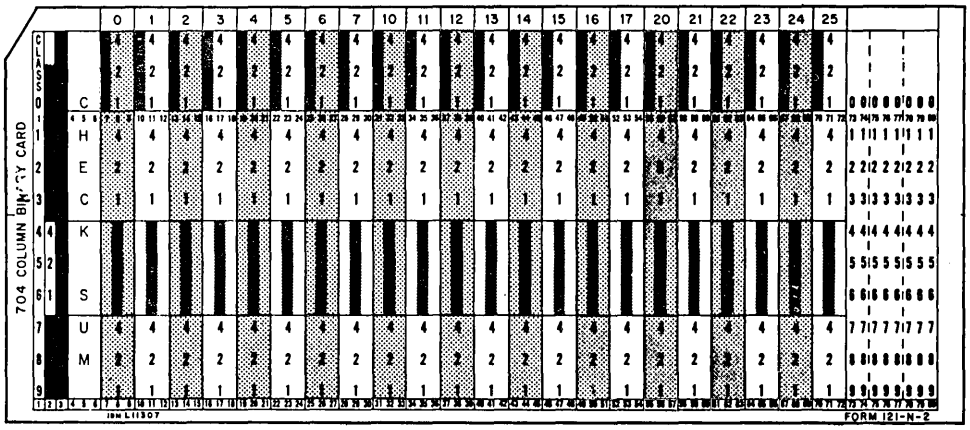


Figure 17. IBM card — column binary

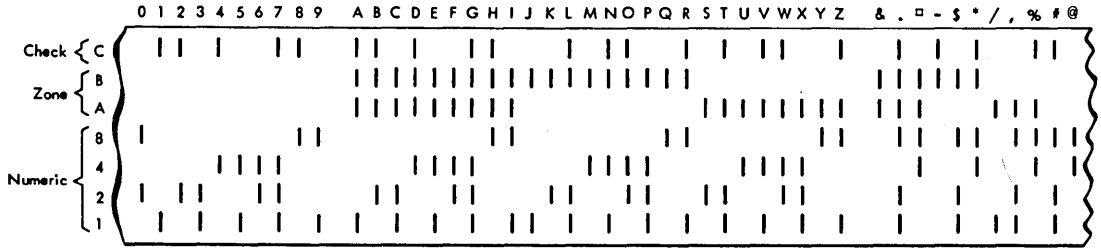


Figure 18. Magnetic tape — six-bit alphameric code

As data is recorded, the previous information is erased, thus permitting repetitive use of the tape. However, the fluctuations in tape speed prohibit data record overlays. Consequently, whenever a data file that is recorded on magnetic tape is to be updated, the entire file is rewritten onto a new tape. The distance between characters recorded on the tape is known as density and ranges from 200 to 3022 characters per inch, depending upon the tape unit used and the method of recording.

The size of tape records may vary from a single character to several thousand and is limited by the length of the tape and the units that process the data. The end of the information recorded for a given file is recognized by a single character record known as a tape mark. The physical end of the tape is determined by a piece of aluminum foil (reflective spot) affixed to the edge of the tape near the end of the tape. This spot is sensed by the tape unit when data is recorded over the area and alerts the computer so that the proper steps may be taken.

Each record is separated from the adjoining record by a blank section of tape known as an inter-record gap. The data records may be blocked or unblocked. (See Figure 1.) The recording on tape varies with the tape unit and the computer system.

Seven-Track Tape

The tape is divided into seven parallel rows along the length of the tape. A combination of the seven possible recording positions in one column across the width of the tape is used to represent a single digit or character. (See Figure 18.)

The four lower positions are assigned the values of 1, 2, 4, and 8, respectively. Thus, to record a given number, the positions are used whose sum equals the value of the number to be recorded, for example, a 1 and an 8 represent a 9. The A and B positions serve the function of the zone punches in IBM cards. The A represents the 0 zone; the B the 11 zone; both the A and B the 12 zone. Since the addition or the loss of a recording position can change the value a given column represents, an additional channel, known as the check (C) channel, is provided so that all characters are made up of either an even or an odd number of bits. This type of check is called a redundancy or parity check, and a given device uses either even or odd parity. The sample in Figure 18 shows even parity.

Tape written in six-bit alphameric code (BCD) can be used by several data processing systems. However, there are instances where special characters peculiar to one system are written on tape. (Refer to "Differences Between Core and Media Storage Requirements" in this section of the manual.) Therefore, consideration must be given to the characters used when tape written on one system may be used on another.

To record binary information on seven-track tape, the six recording positions (excluding the check channel) represent six positions of a binary word. If a computer uses a 24-bit binary word, four columns on the tape are required to record the entire word. (See Figure 19.)

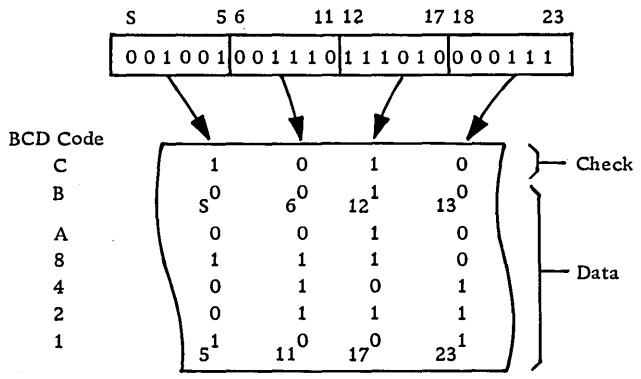


Figure 19. Magnetic tape — seven-track binary system

Nine-Track Tape

The nine-track alphameric code is used to record the extended binary decimal interchange code (EBCDIC) representation.

As can be seen in Figure 20, the 4-7 recording positions of nine-track tape parallel the function of the 8, 4, 2, 1 bit positions of seven-track tape. The 2 and 3 recording positions are the exact reverse of the A and B bit positions of seven-track. Positions 0 and 1 are the two additional recording channels that group the characters into one of four classifications: upper case alpha and numeric, lower case alpha, special characters, and no assigned character. Figure 21 shows the format of seven- and nine-track tape. Note that the channels on nine-track tape do not run in 0-7 sequence.

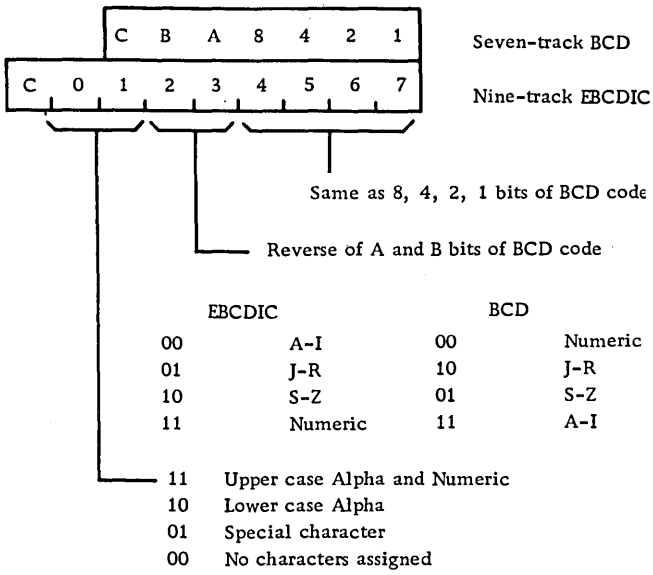


Figure 20. Comparison of seven-track and nine-track alphabetic code

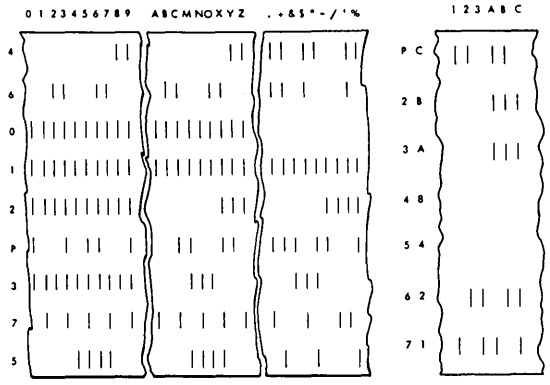
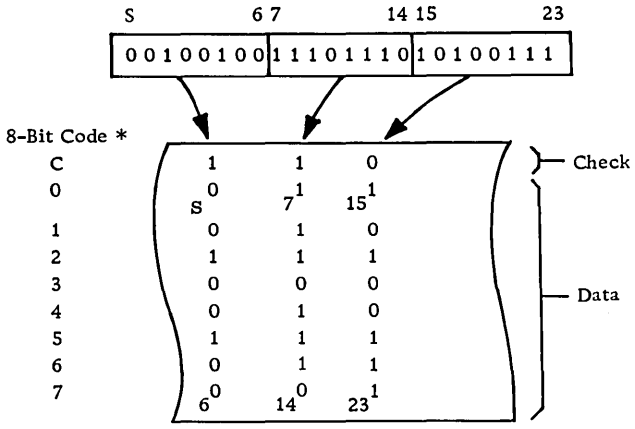


Figure 21. Nine-track and seven-track tape data format

Binary data is recorded by placing eight bits of a binary word in a column of the tape. Thus, a 24-bit binary word requires three columns. The check channel establishes parity for each column recorded.

Since numeric data requires only four channels (1, 2, 4, and 8) to represent its value, it is possible, when dealing with numeric data, to pack two numeric characters into one column of a nine-track tape. This method of recording is known as numeric packed mode. (See Figure 23.)



* Tracks are shown in sequence for visual purposes. See Figure 21 for true location.

Figure 22. Magnetic tape — nine-track binary system

Packed Format	8-Bit Code	Example of Numeric representation in packed format
C	* C	0 1 — Check Bit
8	0	
4	1	9 (1 0 0 1) 3 (0 0 1 1) — Numeric Value
2	2	
1	3	
8	4	
4	5	7 (0 1 1 1) 9 (1 0 0 1) — Numeric Value
2	6	
1	7	

* Tracks are shown in sequence for visual purposes. Refer to Figure 21

Figure 23. Magnetic tape — nine-track packed format

Magnetic Tape Device Characteristics

The following chart presents in tabular form some of the characteristics of various IBM tape devices. The following information is given to aid in use of the table:

1. Recording Mode

The method by which characters are represented on tape. (See the discussions of seven-track and nine-track tape.)

2. Data Rate

The number of characters that can be read or written per second. If the device has a variable density rate, multiple data rates are given. Note that for the nine-track tape of the 2400 unit, the figures are in parentheses. Example: A nine-track 2400, model 3, in unpacked mode (800 density), can operate at 90,000 characters per second. In packed mode (1600 density), it can operate at 180,000 numeric digits per second. A seven-track 2400, model 3, at 556 density, can operate at 62,500 characters per second.

3. Nominal Interrecord Gap (IRG)

The amount of space between physical records.

4. Average Access Time

The amount of time required to pass over the interrecord gap. This is added once for each block of records in computing tape passing time for specific records. (See the second chart below for magnetic tape timing and capacity.)

5. Character Transfer Rate

The length of time required to transfer one character to or from magnetic tape. This figure times the number of characters in a physical record, plus the nominal IRG time, completes the formula for computing tape passing time. (See reference in item 4.)

6. Rewind

The length of time required to rewind a full reel of tape.

7. Rewind and Unload

The length of time required to rewind and unload a full reel of tape.

8. Tape Length

The maximum length of tape on a reel. Two sizes are available in most cases — the 2400-foot reel and the 1200-foot reel.

MAGNETIC TAPE DEVICE CHARACTERISTICS

Magnetic Tape Unit	727	7330	729				7340[1]		
Model			2	4	5	6	1	2	3
Systems on which available	650 705 7080	1400[2] 7072 7010 7040/44	1400 except 1440 7000 except 7072				7074 7080 7090/94	1400 except 1440	S/360
Recording Mode	binary or BCD	binary or BCD	binary or BCD				BCD zoned or packed, binary	BCD zoned or packed	EBCDIC zoned or packed format, binary
Density (char[4] per inch)	200	200/556	200/556	200/556	200/556/800	200/556/800	1511[3]	1511[3]	1511/3022[3]
Data Rate (thousands of char/sec[4])	15	7.2/20	15/41.7	22.5/62.5	15/41.7/60	22.5/62.5/90	170	34	170/340
Tape Speed (in/sec)	75	36	75	112.5	75	112.5	112.5	22.5	112.5
Nominal Interrecord Gap (inches)	.75	.75	.75	.75	.75	.75	.45	.45	.38
Average Access Time (ms)	10.8	20.8	10.8	7.3	10.8	7.3	4.2	18.5	3.5
Character Transfer Rate (ms)	.067	.139/.050	.067/.024	.044/.016	.067/.024/.0167	.044/.016/.011	.00588	.029	.00588/.00294
Rewind (min) Rewind and Unload (min)	1.4 1.5	36. in/sec. 2.2	1.4 1.5	1.0 1.1	1.4 1.5	1.0 1.1	1.5	3.75	1.5
Reference Manuals	A22-6589	A22-6589	A22-6589				A24-1470 A22-6616	A22-6671	A22-6828

Magnetic Tape Unit	2401-4 [6]						2415	
Model	1	2	3	4	5	6	1,2,3	4,5,6
Systems on which available	S/360 except 20						S/360 except Models 44 & 67	
Recording Mode	BCD or binary (7-track) EBCDIC-Zoned or packed format, binary (9-track)						same as 2401-4	
Density (char[4] per inch)	200/556/800	200/556/800	200/556/800				200/566/800	200/556/800
(9-track)	(800)[3]	(800)[3]	(800)[3]	(800/1600)[3]	(800/1600)[3]	(800/1600)[3]	(800)[3]	(800/1600)[3]
Data Rate (thousands of char[4]/sec.)	7.5/20.9/30	15/41.7/60	22.5/62.5/90				3.75/10.4/15	3.75/10.4/15
(9-track)	(30)	(60)	(90)	(30/60)	(60/120)	(90/180)	(15)	(15/30)
Tape Speed (in/sec)	37.5	75	112.5	37.5	75	112.5	18.75	18.75
Nominal Interrecord Gap (inches)	.75	.75	.75				.75	.75
(9-track)	(.6)	(.6)	(.6)	(.6)	(.6)	(.6)	(.6)	(.6)
Average Access Time (ms)	20.0	10.0	6.6				40.0	40.0
(9-track)	(16.0)	(8.0)	(5.3)	(16.0)	(8.0)	(5.3)	(32.0)	(32.0)
Charater Transfer Rate (ms)	.133/.048/.033	.067/.024/.0167	.044/.016/.011				.266/.095/.066	.266/.095/.066
(9-track)	(.033)	(.0167)	(.011)	(.033/.0167)	(.0167/.008)	(.011/.005)	(.066)	(.066/.033
Rewind (min) Rewind and Unload (min)	3. 2.2	1.4 1.5	1.0 1.1	3. 2.2	1.4 1.5	1.0 1.1	4.0[5] 4.0	4.0[5] 4.0
Reference Manuals	A22-6866			A22-6866			A22-6866	

For definitions and interpretation of this chart, refer to the immediately preceding discussion.

1) 10-track tape.
2) 7335 for 1440, A22-6789
3) Packing feature allows two numeric digits to be translated to one 8-bit character.
4) A character is considered the data recorded in one vertical column on the tape.
5) Includes reload.
6) No 2404 Model 4, 5 or 6; 7-track read/write available only on Models 1, 2 or 3; see A22-6866 for compatibility requirements.

Magnetic Tape Timing and Capacity Formulas — Use of Chart

Calculation of Tape Passing Time

1. Select the formula under the heading "Milliseconds per Record" for the desired tape unit and density. For 2400 model 3, 7-track tape with 556 density, this would be 6.6 + .016N.

2. This formula is interpreted as follows:

6.6	=	average access time in milliseconds
.016	=	character time in milliseconds
N	=	number of characters per physical record, where a character is equivalent to the recording in one vertical column of the tape

3. The extension of this formula provides the time to pass one record. If each logical record is 200 characters, and the records are blocked 5, then, 6.6 + .016 (1000) = 22.6 milliseconds.

4. Total time to pass a file of 10,000 logical records is:

$$\frac{\text{total number of records}}{\text{blocking factor}} \times \text{time per physical record, or}$$

$$\frac{10,000}{5} \times 22.6 = 45,200 \text{ ms} = 45.2 \text{ seconds}$$

Calculation of Records per Reel

1. Select the formula under the heading "Records per Reel" for the desired tape unit and density. For 2400, model 3, 7-track tape with 556 density, this would be:

$$\frac{28440 \text{ or } 14040}{.75 + .0018N}$$

2. This formula is interpreted as follows:

28440	=	inches per 2400-foot reel minus 30 feet of combined header and trailer leader
14040	=	inches per 1200-foot reel minus 30 feet of combined header and trailer leader
.75	=	length of interrecord gap
.0018	=	inches per character
N	=	number of characters per physical record, where a character is equivalent to the recording in one vertical column of the tape

3. The extension of this formula provides the number of blocks that can be written on one reel. Using 200 character records, blocked 5, this is:

$$\frac{28440}{.75 + .0018\ (1000)} = 11,152 \text{ blocks}$$

Since the records are blocked by 5, this means 11,152 x 5, or 55,760 logical records.

Extension tables are available for common record lengths. Refer to X22-6785 for 729 and 7330 tapes, X22-6840 for 7340-S/360 tapes and X22-6837 for 2400 tapes.

MAGNETIC TAPE TIMING AND CAPACITY

Tape Unit		Milliseconds per Record				Records per Reel			
		200	556 [1]	800 [2]	1600	200	556 [1]	800 [2]	1600
727		10.8 + .067N [3]				$\frac{28440 \text{ or } 14040}{.75 + .005N}$			
7330		20.8 + .139N	20.8 + .050N			$\frac{28440 \text{ or } 14040}{.75 + .005N}$	$\frac{28440 \text{ or } 14040}{.75 + .0018N}$		
729	II	10.8 + .067N	10.8 + .024N			$\frac{28440 \text{ or } 14040}{.75 + .005N}$	$\frac{28440 \text{ or } 14040}{.75 + .0018N}$		
	IV	7.3 + .044N	7.3 + .016N			$\frac{28440 \text{ or } 14040}{.75 + .005N}$	$\frac{28440 \text{ or } 14040}{.75 + .0018N}$		
	V	10.8 + .067N	10.8 + .024N	10.8 + .017N		$\frac{28440 \text{ or } 14040}{.75 + .005N}$	$\frac{28440 \text{ or } 14040}{.75 + .0018N}$	$\frac{28440 \text{ or } 14040}{.75 + .00125N}$	
	VI	7.3 + .044N	7.3 + .016N	7.3 + .011N		$\frac{28440 \text{ or } 14040}{.75 + .005N}$	$\frac{28440 \text{ or } 14040}{.75 + .0018N}$	$\frac{28440 \text{ or } 14040}{.75 + .00125N}$	
2401-4 [4]	Model 1 - 7-track	20.0 + .133N	20.0 + .048N	20.0 + .033N		$\frac{28440 \text{ or } 14040}{.75 + .005N}$	$\frac{28440 \text{ or } 14040}{.75 + .0018N}$	$\frac{28440 \text{ or } 14040}{.75 + .00125N}$	
	(9-track)			(16.0 + .033N)				$\left(\frac{28440 \text{ or } 14040}{.6 + .00125N}\right)$	
	Model 2 - 7-track	10.0 + .067N	10.0 + .024N	10.0 + .017N		$\frac{28440 \text{ or } 14040}{.75 + .005N}$	$\frac{28440 \text{ or } 14040}{.75 + .0018N}$	$\frac{28440 \text{ or } 14040}{.75 + .00125N}$	
	(9-track)			(8.0 + .017N)				$\left(\frac{28440 \text{ or } 14040}{.6 + .00125N}\right)$	
	Model 3 - 7-track	6.6 + .044N	6.6 + .016N	6.6 + .011N		$\frac{28440 \text{ or } 14040}{.75 + .005N}$	$\frac{28440 \text{ or } 14040}{.75 + .0018N}$	$\frac{28440 \text{ or } 14040}{.75 + .00125N}$	
	(9-track) [5]			(5.3 + .011N)				$\left(\frac{28440 \text{ or } 14040}{.6 + .00125N}\right)$	
	Model 4 - (9-track) [5]			(16.0 + .033N)	(16.0 + .017N)			$\left(\frac{28440 \text{ or } 14040}{.6 + .00125N}\right)$	$\left(\frac{28440 \text{ or } 14040}{.6 + .000625N}\right)$
	Model 5 - (9-track) [5]			(8.0 + .017N)	(8.0 + .008N)			$\left(\frac{28440 \text{ or } 14040}{.6 + .00125N}\right)$	$\left(\frac{28440 \text{ or } 14040}{.6 + .000625N}\right)$
	Model 6 - (9-track)			(5.3 + .011N)	(5.3 + .005N)			$\left(\frac{28440 \text{ or } 14040}{.6 + .00125N}\right)$	$\left(\frac{28440 \text{ or } 14040}{.6 + .000625N}\right)$
2415 [4]	all models - 7-track	40.0 + .266N	40.0 + .095N	40.0 + .066N		$\frac{28440 \text{ or } 14040}{.75 + .005N}$	$\frac{28440 \text{ or } 14040}{.75 + .0018N}$	$\frac{28440 \text{ or } 14040}{.75 + .00125N}$	
	Models 1, 2 or 3 - (9-track)			(32.0 + .066N)				$\left(\frac{28440 \text{ or } 14040}{.6 + .00125N}\right)$	
	Models 4, 5 or 6 - (9-track)			(32.0 + .066N)	(32.0 + .033N)			$\left(\frac{28440 \text{ or } 14040}{.6 + .00125N}\right)$	$\left(\frac{28440 \text{ or } 14040}{.6 + .000625N}\right)$
7340 [4]	Model 1		4.2 + .00588N				$\frac{21240}{.45 + .00066N}$		
	Model 2		18.5 + .029N				$\frac{21240}{.45 + .00066N}$		
	Model 3		3.5 + .00588N	3.5 + .00294N			$\frac{21240}{.38 + .00066N}$	$\frac{21240}{.38 + .00033N}$	

1) Use 1511 for Hypertape 7340.
2) Use 3022 for Hypertape 7340.
3) N = number of characters per block.
4) If the packed mode is used, the basic formulas may be modified by suffixing $\left(\frac{1 + PF}{2}\right)$ behind each N. For 7340, model 1, the formula becomes $4.2 + .00588N\left(\frac{1 + PF}{2}\right)$.
PF = packing factor or the number of nonpacked characters in a record divided by the total number of characters in the record.
5) No 2404 Model 4, 5, or 6.

Paper Tape

Paper tape is a continuous recording medium. Thus, paper tape can be used to record data in records of any length, limited only by the capacity of the equipment that processes the paper tape. Depending upon the specific device, the recording on paper tape may be either in eight-channel code or in five-channel code.

Eight-Channel Code

Data is coded in a fashion similar to that described for magnetic tape — six-bit alphameric code. However, holes are punched into the paper tape. (See Figures 18 and 24.) The O and X channels serve the function of the A and B channels in magnetic tape.

Five-Channel Code

This system is similar to the eight-channel code, except only five rows extend the length of the tape. Since the possible combination of holes with only five channels is not sufficient to provide for all of the numbers and letters of the alphabet, certain combinations serve the dual function of representing both a letter and a number. To distinguish between the two, letters are preceded by a special combination of punches known as letters shift, while the number combinations are preceded by a figures shift character. Since there are ten numbers and 26 letters, the other figures shift characters are used to represent special characters and functions. The five-channel code is suitable for transmission over telegraph or telephone wires.

Direct Access Storage Devices (DASD)

Magnetic Disk

A disk device is composed of magnetically coated disks, which are stacked on a rotating spindle. A movable access arm, containing two read-write heads, passes between the physical disks. One head services the bottom surface of the disk directly above it, while the other handles the upper surface of the disk directly beneath it. The surface of each disk is divided into concentric tracks. Depending upon the particular DASD, the tracks may be subdivided into sectors. The tracks on each disk surface are located physically one above the other and may be thought of as forming concentric cylinders. (See Figure 26.)

The tracks or sectors (where present) are numbered sequentially and thus form the basis for the addressing scheme of disk devices.

Although some disk devices employ a single-arm access mechanism, which moves both horizontally and vertically to access any track within the disk file, the majority of the IBM disk devices are equipped with a comb-type access mechanism in which the arms are arranged like the teeth on a comb and move horizontally between the disks. The heads are aligned vertically, and all are moved together so that for each position of the access mechanism one entire cylinder surface is accessible to the read-write heads.

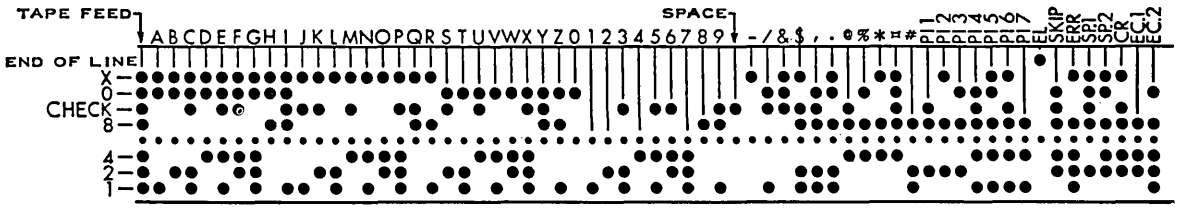


Figure 24. Paper tape — eight-channel code

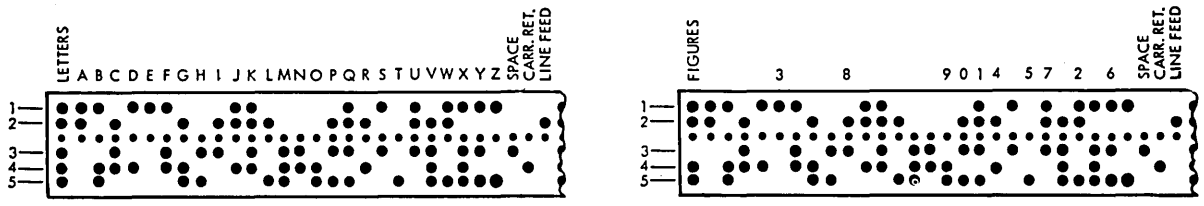


Figure 25. Paper tape — five-channel code

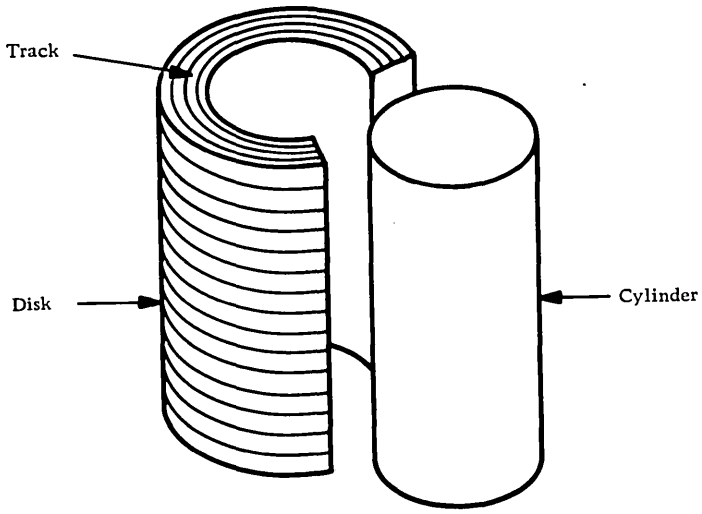


Figure 26. Disk storage cylinder concept

Data Cell Drive

In the IBM 2321 Data Cell Drive, data is stored in magnetically coated strips rather than in disks. The strips are contained in a removable cell assembly, and ten cell assemblies can be mounted on a data cell drive. To access data on the strips, the cell assemblies are rotated until the cell containing the proper strip is under a strip-picking device that pulls the desired strip from the data cell and wraps it around a rotating drum. The read-write mechanism can access 20 tracks on a strip without moving, thereby forming a cylinder of data tracks.

DASD Device Characteristics

Unlike tape, DASD files can replace old data with new data through record overlay. Thus, inactive records do not require rewrite on a DASD unit. The chart on the next page presents in tabular form some of the characteristics of various IBM DASD units. The following information is given to aid in use of the table:

1. Recording Mode

The 6-bit mode refers to BCD; the 8-bit mode refers to EBCDIC on devices attached to S/360; on devices attached to computers in the 1400 series, the 8-bit mode is used whenever data is recorded in the load mode (the extra bits make possible the storage of word marks, along with the character). Refer to IBM Disk Storage Drive, (A24-1472), for a more detailed explanation of load mode; 8-bit also may be used with packed numeric data.

2. Character Transfer Rate

The length of time required to transfer one character to or from a DASD.

3. Rotational Period

The length of time required for a complete revolution of the DASD. The figure in parentheses gives the average time usually used in timing a DASD.

4. Scan Time

The length of time required to search a cylinder for a specific identifier. This may be an optional feature.

5. Access Time

The length of time required to move the access arm from one location to another.

DASD Capacity Formulas

The record storage capacity of a DASD is a function of the length of the records to be stored, the addressing scheme of the DASD, and the maximum character capacity of the recording track. The full track mode provides maximum capacity but increases the main storage requirements.

The symbols used in the formulas shown in the chart on page 39 have the following significance:

DL	=	data length or total number of bytes in data area
HA_2	=	home address identifier (minimum of two characters)
KL	=	key length or total number of bytes in key area (KL = O if records are stored without keys)
RA	=	length of record address field (minimum of six characters)

For non-System/360 devices, the formulas pertain to the IBM 1410 and 7010.

DIRECT ACCESS STORAGE DEVICE CHARACTERISTICS

Direct Access Storage Device Model	1311 -	1301 1	1301 2	1405 1	1405 2	2302 1	2302 2	2302 3	2302 4	2311 -	2314 1	2321 1
Systems on which available	1400 1600 7000	1400 7000		1401 1410		1410 7000 (except 7070)		S/360		S/360	S/360-	S/360
Type of storage media	Disk	Disk		Disk		Disk		Disk		Disk	Disk	Magnetic Strip
Removable Media	Yes	No		No		No		No		Yes	Yes	Yes
No. of units[1] per control	5	5		5		5		4	2	8	(see note 5)	8
No. of access mechanisms per unit	1	1	2	Single arm mechanism		2	4	2	4	1	1	1
				1	1[10]							
No. of arrays[2] per unit	1	1	2	1	1	1	2	1	2	1	(see note 5)	1
No. of cylinder per unit	100	250	500	no cylinder concept: 400 tracks/per disk		500	1000	500	1000	200[3]	200[3]	10,000
No. of tracks per cylinder	10	40		25 disks	50 disks	40	40		45[4]	10	18	20
No. of tracks per unit	1000	10,000	20,000	10,000	20,000	20,000	40,000	22,500	45,000	2,000[6]	3600	200,000[7]
Maximum data capacity per unit (thousands)[11] 6-bit mode	2980	28,000	56,000	10,000	20,000	117,000	234,000	-		-		-
(8-bit mode)	(2682)	(21,650)	(43,300)	(8,800)	(17,600)	(90,660)	(181,320)	(112,140)	(224,280)	(7,250)	25,876	(400,000[8])
Maximum data capacity per cylinder[11] 6-bit mode	29,800	112,000		-		234,000		-		-		-
(8-bit mode)	(26,820)	(86,600)		-		(181,320)		(224,280)		(36,250)	129,384	(40,000)
Maximum data capacity per track[11] 6-bit mode	2980	2800		1000		5850		-		-		-
(8-bit mode)	(2682)	(2165)		(880)		(4533)		(4984)		(3625)	(7,188)	(2000)
Track Storage characteristics	20 hardware addr. sectors of 100 chars. each with full track, R/W special feature available.	Changeable record format for all tracks in cylinder plus full track R/W as normal.		5 hardware addr. sectors of 200 chars. each.		Same as 1301		Same as 2311		Hardware designed so that each stored record defines its own format, every record can be different, no track orientation.	same as 2311	same as 2311
Character transfer rate -6-bit mode	77KC	90.1KC		25KC		184KC		-		-		-
(8-bit mode)	(69KC)	(70.1KC)		(22KC)		(143KC)		(156KC)		(156KC)	(312KC)	(55KC)
Rotational period (ms)	40	34		50		34		34		25	25	50
(average delay)	(20)[12]	(17)		(25)		(17)		(17)		(12.5)	(12.5)	(25)
Scan time per cylinder[9] (seconds)	.4	1.33		-		1.33		1.5		.25	.25	.9
Access time (ms) Min.	100 55	50		100		50		50		25	25	175
Aver.	250 150	165		450		165		165		75	75	-
Max.	400 250 without direct seek with direct seek	180		800		180		180		135	135	600

1) 1 unit is the minimum orderable element.
2) An array is a stack of disks except on 2321 where it is 10 data cells.
3) 203 cylinders are addressable.
4) 46 tracks are addressable.
5) Each 2314 includes 8 single array units and necessary controls.
6) 2030 tracks are addressable.
7) Represents 10 removable and interchangeable data cells of 20,000 tracks each.
8) Represents 10 removable and interchangeable data cells of 40,000,000 bytes each.
9) Refers to use of hardware scan feature.
10) Up to 3 arms available, 1 standard.
11) Capacities listed are for IBM 1410/7010 on all non-System/360 devices.
12) Does not include 2 ms. head select time.

DASD CAPACITY FORMULAS

DASD	Max. Track Capacity	Formula for No. of Records in Usable Characters	Example of Computation of No. of Records in Usable Characters
1301 - 6-bit (8-bit)	2800 (2165)	$\frac{2840 - HA_2}{DL + RA + 32}$ (substitute 2205 for 2840)	$\frac{2840 - 2}{164 + 6 + 32} = \frac{2838}{202} =$ 14 records, 10 unused positions, where $HA_2 = 2$, DL = 164, RA = 6
1311 - 6-bit (8-bit)	2980 (2682)	Sector mode --No special formula-- Twenty 100-character sectors (Twenty 90-character sectors)	Track Mode
2302-1/2 - 6-bit (8-bit)	5850 (4533)	$\frac{5902 - HA_2}{DL + RA + 44}$ (substitute 4585 for 5902)	$\frac{5902 - 2}{256 + 10 + 44} = \frac{5900}{310} =$ 19 records, 10 unused positions, where $HA_2 = 2$, DL = 256, RA = 10
2301	20483	$1 + \frac{20483 - (53 - C + KL + DL)}{186 - C + (KL + DL)}$ C = 0 when KL ≠ 0 C = 53 when KL = 0	$1 + \frac{20483 - (53 - 0 + 10 + 150)}{186 - 0 + (10 + 150)} =$ $\frac{20270}{346} =$ 59 records, with 202 unused byte positions, where KL = 10, DL = 150
2302-3/4	4984	$1 + \frac{4984 - (20 - C + KL + DL)}{81 - C + 1.049 (KL + DL)}$ C = 0 when KL ≠ 0 C - 20 when KL = 0	$1 + \frac{4984 - (20 - 0 + 10 + 150)}{81 - 0 + 1.049 (10 + 150)} =$ $1 + \frac{4804}{249} =$ 20 records, with 73 unused byte positions, where KL = 10, DL = 150
2303	4892	$1 + \frac{4892 - (38 - C + KL + DL)}{146 - C + (KL + DL)}$ C = 0 when KL = 0 C = 38 when KL = 0	$1 + \frac{4892 - (38-0 + 10 + 150)}{146 - 0 + (10 + 150)} =$ $1 + \frac{4694}{306} =$ 16 records, with 104 unused byte positions, where KL = 10, DL = 150
2311	3625	$1 + \frac{3625 - (20 - C + KL + DL)}{81 - C + 1.049 (KL + DL)}$ C = 0 when KL ≠ 0 C = 20 when KL = 0	$1 + \frac{3625 - (20 - 0 + 10 + 150)}{81 - 0 + 1.049 (10 + 150)} =$ $1 + \frac{3445}{249} =$ 14 records, with 208 unused byte positions, where KL = 10, DL = 150
2314	7188	$1 + \frac{7188 - (41 - C + KL + DL)}{166 - C + 1.043 (KL + DL)}$ C = 0 when KL ≠ 0 C = 41 when KL = 0	$1 + \frac{7188 - (41 - 0 + 10 + 150)}{166 - 0 + 1.043 (10 + 150)} =$ $1 + \frac{6987}{333} =$ 21 records, with 327 unused byte positions, where KL = 10, DL = 150
2321-1	2000	$1 + \frac{2000 - (16 - C + KL + DL)}{100 - C + 1.049 (KL + DL)}$ C = 0 when KL ≠ 0 C = 16 when KL = 0	$1 + \frac{2000 - (16 - 0 + 10 + 150)}{100 - 0 + 1.049 (10 + 150)} =$ $1 + \frac{1824}{268} =$ 7 records, with 216 unused byte positions, where KL = 10, DL = 150

Capacity and Transmission Time--Reference cards available as follows:
2302--X20-1706; 2311--X20-1705; 2321--X20-1704; 2303--X20-1718;
2314--X20-1710; 2301--X20-1717.

EXTENDED BINARY CODED DECIMAL INTERCHANGE CODE

Instructions for use of Figure 27*.

Conversion from graphic to punched-card code

1. Locate the graphic in one of the four charts.
2. Trace to the right for the digit punches.
3. Trace down for the zone punches.
4. All punches indicated in 2 and 3 must be punched in the same column.

Conversion from punched-card code to graphic

1. Locate the two charts that contain the zone punches of the punched code to be deciphered.
2. Select from the two charts the one that contains the digit punches for the character to be decoded.
3. The graphic desired is at the intersection of the lines traced up from the zone and left from the digits chosen in steps 1 and 2.

Conversion from graphic to eight-bit code

1. Locate the graphic in one of the four charts.
2. Trace to the left for bit positions 4-7.
3. Trace up for bit positions 0-3.

Conversion from eight-bit code to graphic

1. Locate the two charts that contain the proper pattern for bits 4-7.
2. Select from the two charts the one that contains the zone bit pattern for the character to be decoded.
3. The graphic desired is at the intersection of the lines traced to the right from the 4-7 bits and down from the 0-3 bits chosen in steps 1 and 2.

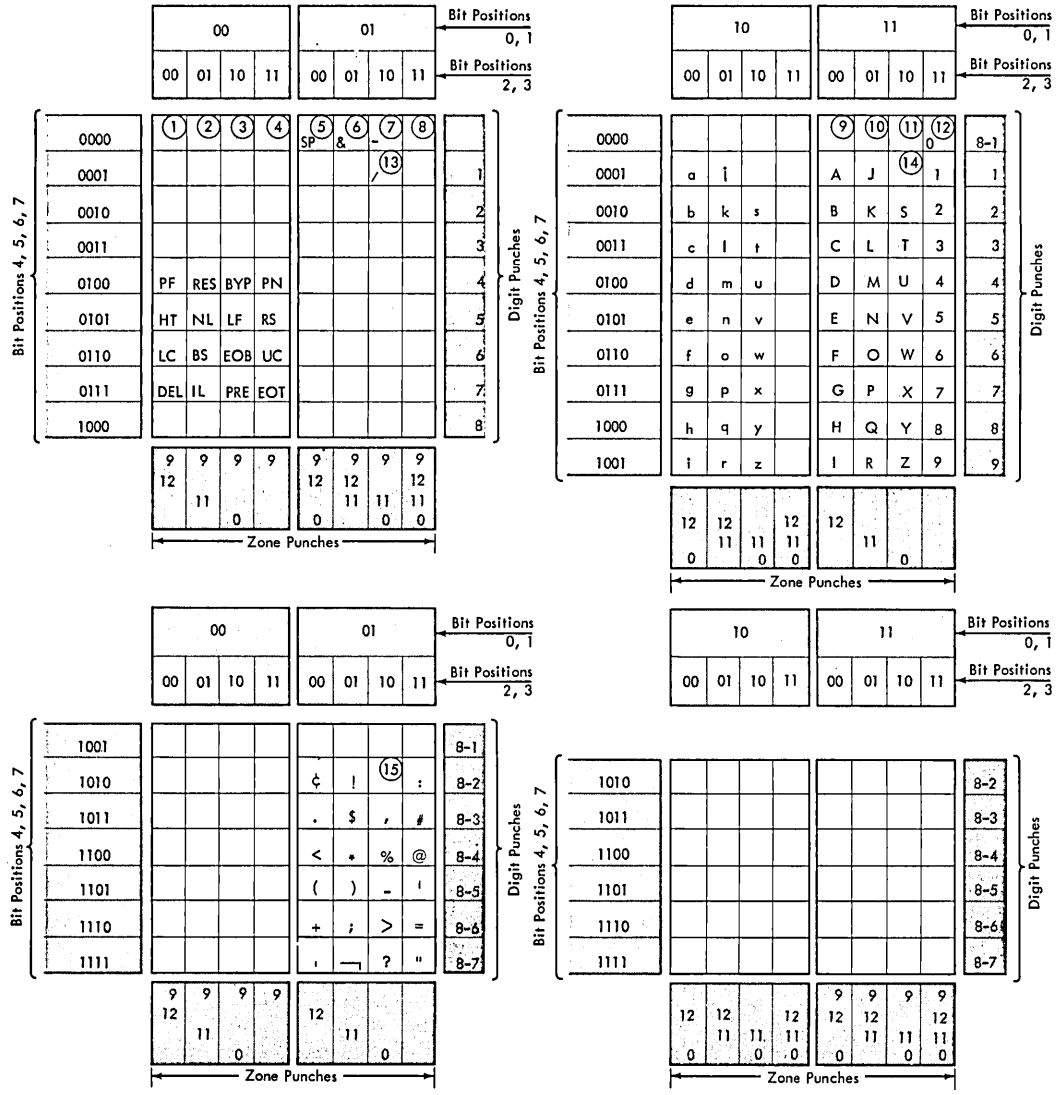
Conversion from eight-bit code to punched-card code (to punch binary data in EBCDIC)

1. Locate the two charts that contain the proper pattern for bits 4-7.
2. Select from the two charts the one that contains the 0-3 bit pattern for the character to be coded.
3. Trace to the extreme right from bits 4-7 for the digit punches.
4. Trace to the bottom from bits 0-3 for the zone punches.

Conversion from punched-card code to eight-bit code (to decode binary data in EBCDIC)

1. Locate the two charts that contain the zone punches of the punched code to be deciphered.
2. Select from the two charts the one that contains the digit punches for the value to be decoded.
3. Trace to the extreme left from the digit punches for bits 4-7.
4. Trace to the top from the zone punches for bits 0-3.

* There are 15 exceptions to the punching equated to bit positions. These exceptions are shown in the chart by circled numbers 1 through 15, and the substituted punching is shown below the chart under "Exceptions".



Refer to instructions above for use of this chart.

Figure 27. Extended binary coded decimal interchange code (EBCDIC)

DIFFERENCES BETWEEN CORE AND MEDIA STORAGE REQUIREMENTS

System/360

Structure	Media Storage	Core Storage
8-bit unpacked recording mode (EBCDIC or zoned format)	1 position for each alphameric or numeric character; each character uses 8 data bits; sign should be over units position of numeric data.	1 position contains a nonbinary data character. For proper conversion to packed decimal, numeric data must contain sign in 0-3 data bits of units position.
8-bit packed numeric recording mode (packed decimal)	2 numeric digits are stored in one position (0-3 and 4-7 data bits); sign control requires 4-7 data bits of units positions.	2 numeric digits are stored in one position (0-3 and 4-7 data bits); sign control requires 4-7 data bits of units positions.
8-bit recording mode (binary)	4 positions contain the contents of one binary word.	1 word contains 32 data bits.
6-bit recording mode (BCD)	1 position for each alphameric or numeric character; each character uses 6 data bits - BCD format (7-track tape).	With 7-track feature and translator on: each 6-bit character is translated to the EBCDIC 8-bit code and vice versa; therefore, each position on tape uses 1 position in core. With 7-track feature and translator off: each 6-bit character is placed in 2-7 data bits of one position; the 0-1 bits are ignored (on write), or zero-filled (on read).
6-bit recording mode (binary)	6 positions contain the contents of one 36-bit binary word (7-track tape).	With data conversion and 7-track feature: each group of 24 bits from tape (4 positions) is converted to three 8-bit bytes in core and vice versa. See 2400 Principles of Operation (A22-6866), for details when data is fewer than 4 positions.

DIFFERENCES BETWEEN CORE AND MEDIA STORAGE REQUIREMENTS

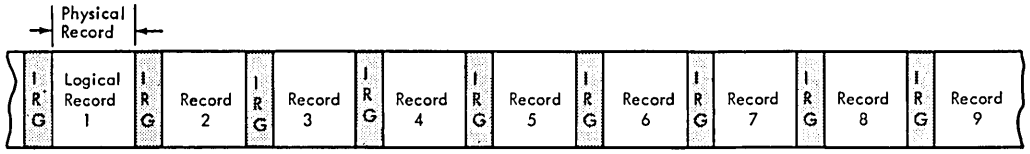
Non-System/360

Structure	Media Storage	Core Storage
6-bit recording mode BCD move mode (without wordmark)	1 position for each alphameric or numeric character. Each character uses 6 data bits.	1 position contains a nonbinary data character, except for the 7070/7074, where each alpha character requires 2 positions.
8-bit unpacked recording mode BCD load mode (with wordmarks)	DASD — each character uses 8 data bits, thus reducing the number of storage positions available as compared to 6-bit mode. The wordmark requires the extra space. Tape — the wordmark becomes a word separator character and is stored 1 storage position ahead of the associated character, thus increasing the media storage requirements.	1 position contains a nonbinary data character with or without wordmarks. 7070/7074 requires 2 positions for each alpha character.
8-bit packed numeric recording mode	2 numeric digits are stored in one position.	2 numeric digits are stored in one position.
Binary recording mode	6 positions contain the contents of one binary word.	1 word contains 36 data bits in the binary computers.
Special 7070/74 Compressed tape recording mode	Up to 5 leading zeros of each numeric word are eliminated; alpha words use 5 tape positions.	Complete 10-position words, numeric and alpha.
Delta (Δ) mode change	Requires 1 position between alpha and numeric words.	The delta does not appear in core; it is only in media storage and is used to transfer and sign words between core and media storage.
Theta (θ) sign control used with 8-bit packed recording mode	Theta character precedes and follows the contents of each negative word, thus requiring 2 extra positions. For use with 1301 DASD and hypertape (7340).	This special code does not appear in core; it controls the negative sign designation as words are transferred between core and media storage.

Processor for	1400 7010	7040/44	7070/74	705/7080	7090/94	S/360
Type	Character - numeric/alphabetic	Word - 36 data bits & 1 validity bit	Word - 10-pos. numeric 5-pos. alphabetic	Character - numeric/alphabetic	Word-36 data bits	Character-1 to n bytes (8 bits-byte) Halfword - 2 bytes Word - 4 bytes, Doubleword - 8 bytes
Core	BCD-8 bits (6 data, 1 check, 1 wordmark)	binary; alpha in BCD 6-bit char., 6 char/word; numeric may be in BCD	2 out of 5[1]	BCD-7 bits (6 data and 1 check)	binary; alpha in BCD as 6-bit char., 6 char/word; numeric may be in BCD	binary; EBCDIC or ASC II[2] (zoned); numeric data may be in packed format.
Instruction Type	Two-address, variable	Single address	Single address, 5 positions	Single address	Single address	Single or two-address variable length, located on integral boundary.
Field Definition	High-order word-mark		Length specified in instruction	Zone bit to left of field		Length specified in instruction for char. oper.; implied on fixed-length operations.
Signing	Units position-blank/+ = plus	First data bit	1 position/word - alpha, plus or minus	Units position	First data bit	High-order bit - binary; units position in packed format.
Remarks	Chaining of data and instructions	Subroutines required to convert from binary to decimal and vice versa		Some operations more efficient with length of 5 or multiples thereof	Subroutines required to convert from binary to decimal and vice versa	Fixed-length data must be located on integral boundary; numeric data must be in packed format for decimal operations; instructions available for packing, unpacking, conversion from decimal to binary and vice versa.

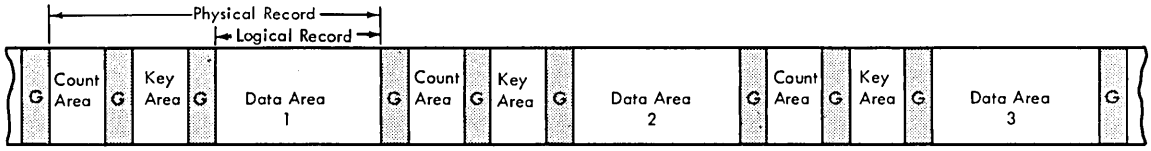
1. 5 bits per position have values of 0, 1, 2, 3, 6. Use sum of 2 bits to represent a numeric value, for example, 0 + 3 = 3, 3 + 2 = 5, (1 + 2 used for 0). Alpha requires 2 positions per letter, where the left position is a numeric representation of the zone, and the right position is the numeric portion of the character, for example, 61 = A, 62 = B, 71 = J, 82 = S, etc.

2. ASC II - See appendix in S/360 Principles of Operation, (A22-6821). Choice of EBCDIC or ASC II under program control.

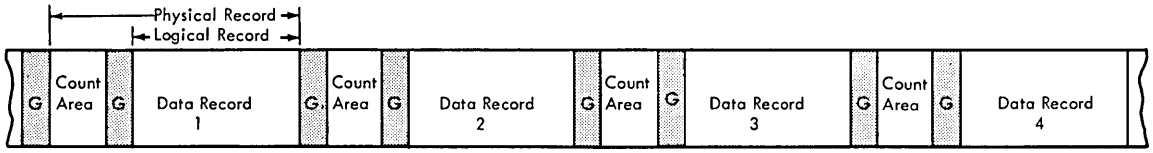


IRG = Inter-Record Gap

TAPE RECORDS



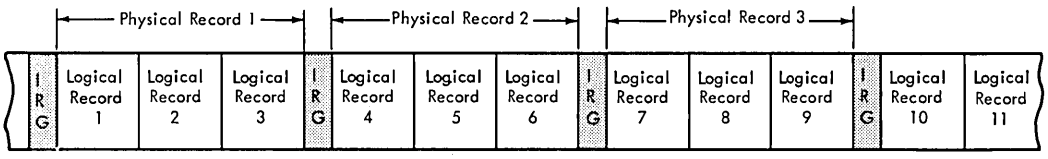
DASD RECORD - WITH KEY AREA



DASD RECORD - WITHOUT KEY AREA

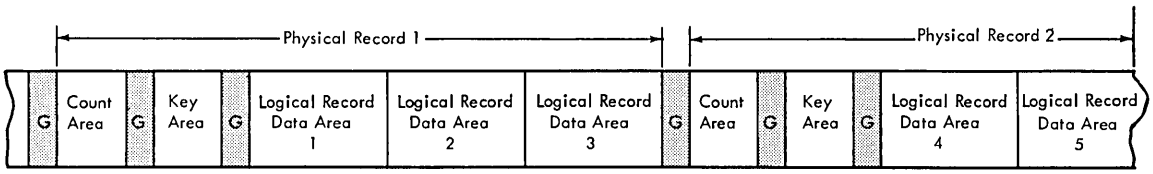
G = Gap, Area Separator

Fixed-Length Unblocked Record Format

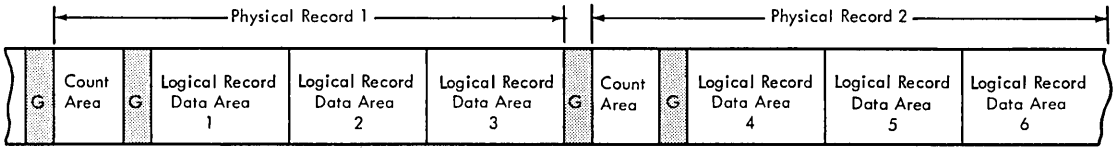


IRG = Inter-Record Gap

TAPE RECORDS



DASD RECORDS - WITH KEY AREAS

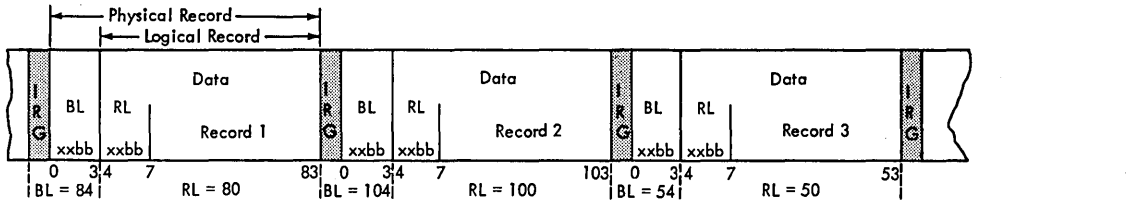


DASD RECORDS - WITHOUT KEY AREAS

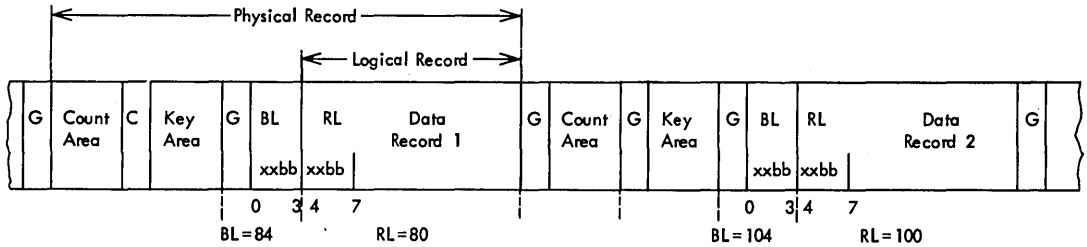
G = Gap, Area Separator

Fixed-Length Blocked Record Format

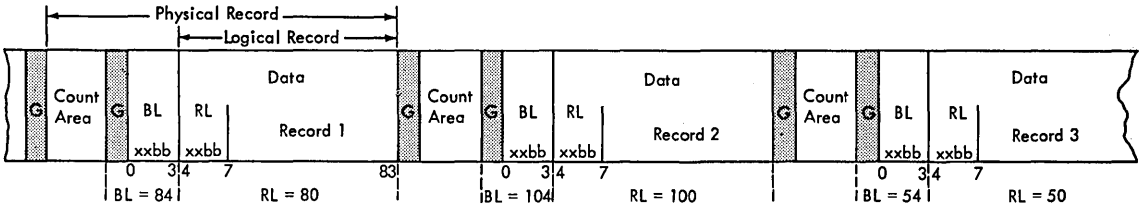
RECORD FORMATS - S/360



TAPE RECORDS



DASD Records — With Keys

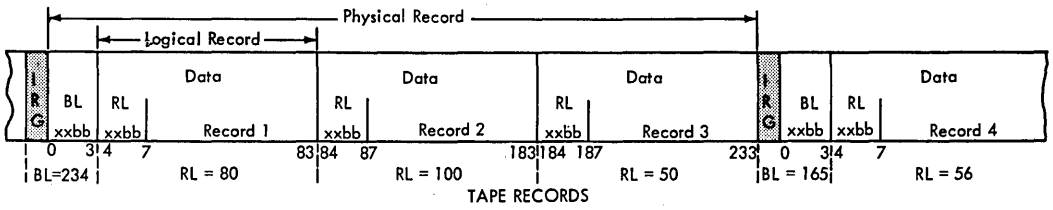


DASD Records — Without Keys

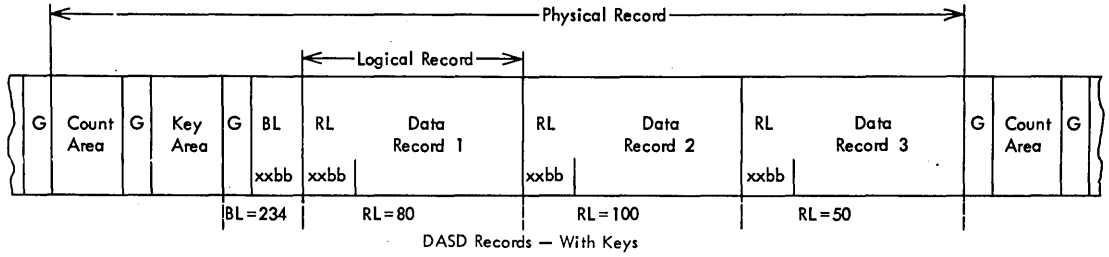
BL = Block Length
RL = Record Length
} in binary half - word (16 bit) format, plus two blank bytes.

IRG = Inter-record Gap
G = Gap

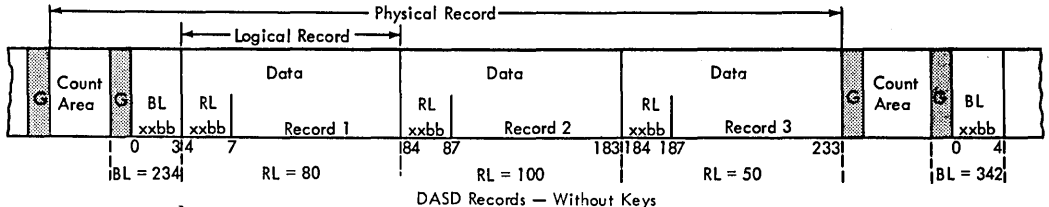
Variable-Length Unblocked Record Format



TAPE RECORDS



DASD Records — With Keys

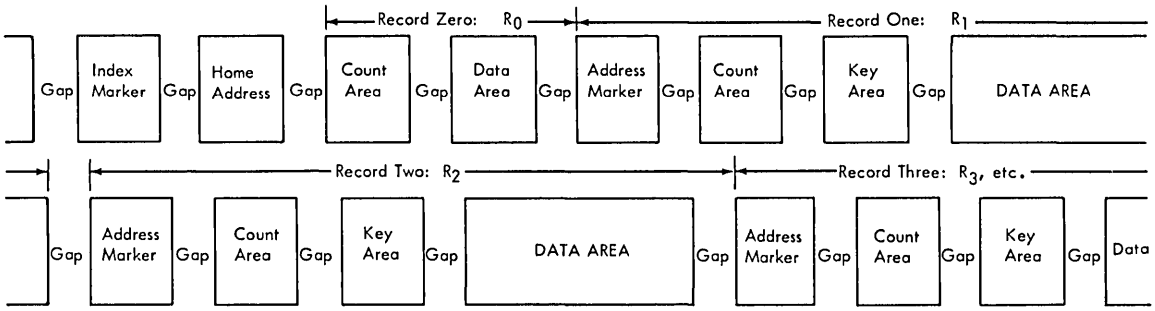


DASD Records — Without Keys

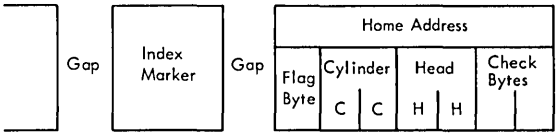
BL = Block Length
RL = Record Length
} in binary half - word (16 bit) format, plus two blank bytes.

IRG = Inter-record gap
G = Gap

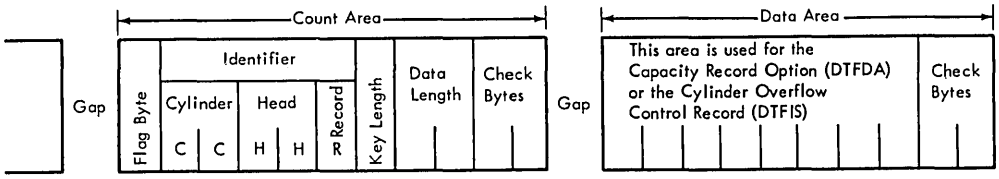
Variable-Length Blocked Record Format



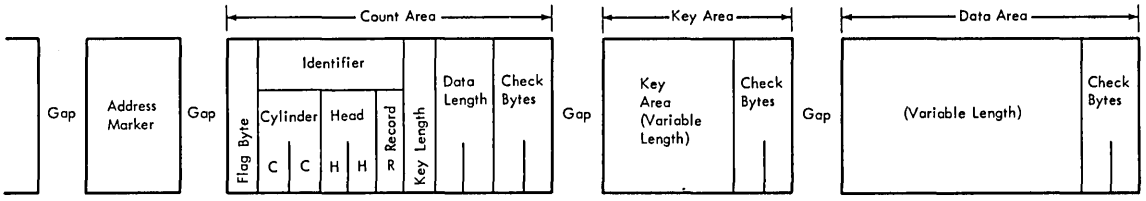
Schematic Representation of DASD (with Key Area)



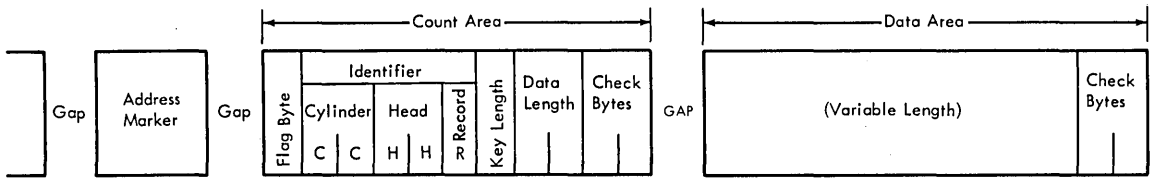
Schematic Representation of the Home Address



Schematic Representation of Record Zero



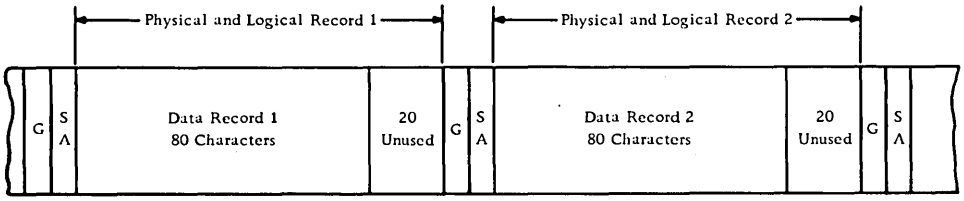
Schematic Representation of DASD Record with a Key Area



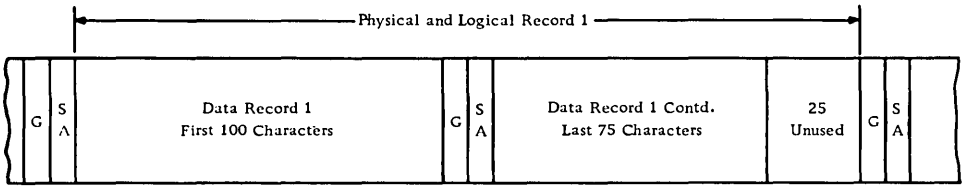
Schematic Representation of DASD Record without a Key Area

NON-SYSTEM/360 SAMPLE RECORD FORMATS

1311 Sector (Move Mode) *



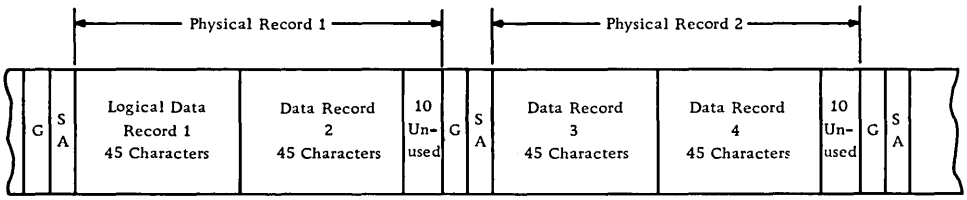
80-Character Record



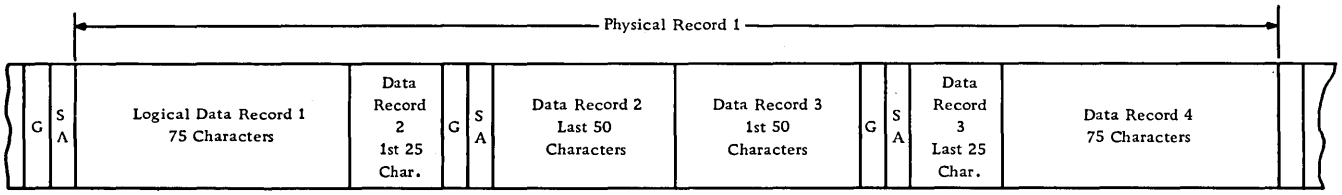
175-Character Record

G = Gap
SA = Sector Address
100 Character Sectors
20 Sectors Per Track

Fixed-Length Unblocked Record Format



45-Character Record



75-Character Record

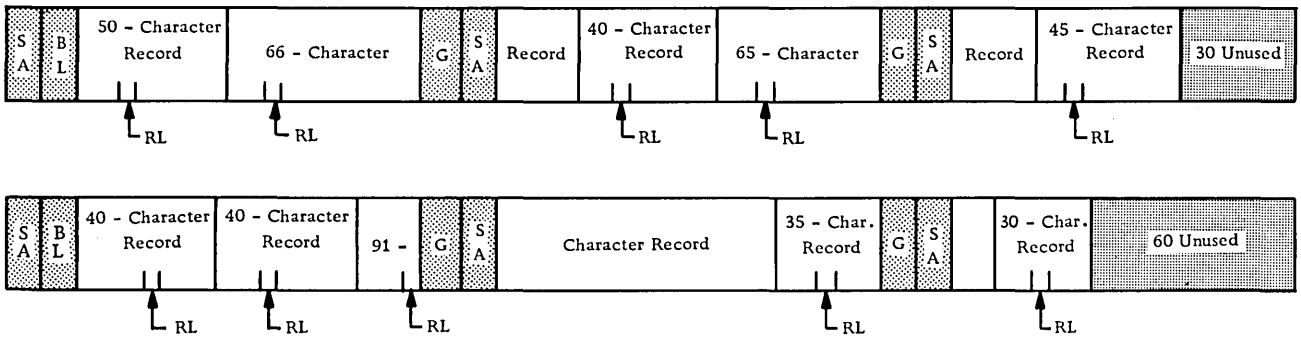
G = Gap
SA = Sector Address
100 Character Sectors
20 Sectors Per Track

Fixed-Length Blocked Record Format

* Reduce Number of Characters Per Sector from 100 to 90 for Load Mode.

NON-SYSTEM/360 SAMPLE RECORD FORMATS

1311 Sector Mode (Move) *



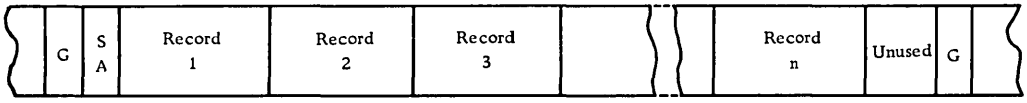
Largest Block = 300 Characters; Largest Record = 296 Characters

G = Gap
SA = Sector Address
BL = Block Length
RL = Record Length
100 Character Sectors
20 Sectors Per Track

Variable-Length Blocked Record Format

* Reduce Number of Characters Per Sector from 100 to 90 for Load Mode.

1311 Track Mode (Move) **



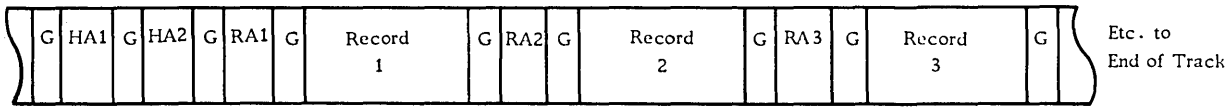
2980 Characters Per Track

G = Gap
SA = Sector Address

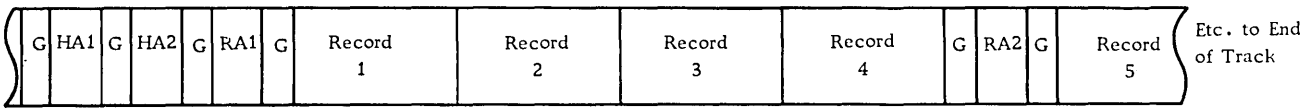
Fixed-Length Blocked Record Format

* Reduce Number of Characters Per Track from 2980 to 2682 for Load Mode.

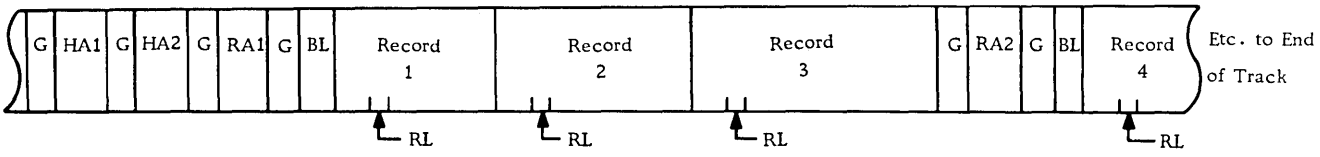
1301, 2302-1/2,



Fixed-Length Unblocked Record Format



Fixed-Length Blocked Record Format



Variable-Length Blocked Record Format

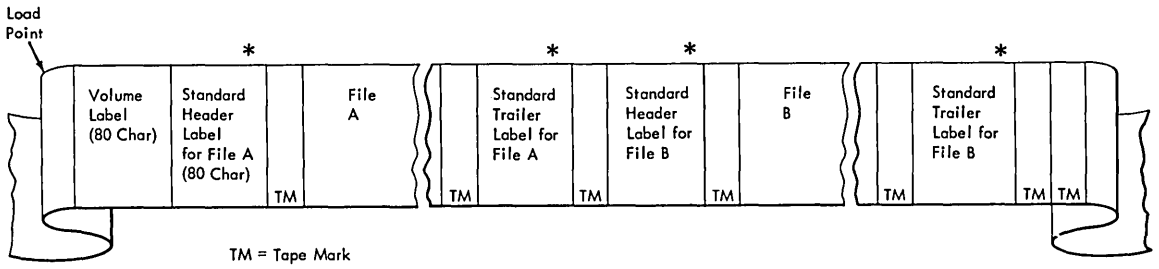
G = Gap Area Separator
HA1 = Home Address of Actual Physical Address. It is Prerecorded and Cannot be Written by User.
HA2 = Home Address Identifier Written by the User.
RA = Record Address Usually Generated from a Key.
RL = Record Length
BL = Block Length

See DASD Capacity Chart for Track Length of Specific Device.

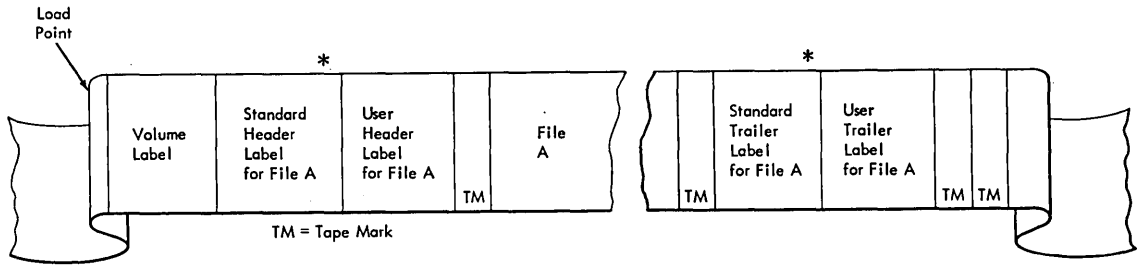
See S/360 Sample Record Formats for Sample Tape Formats.

S/360 File Label Formats

Tape File with Standard Labels

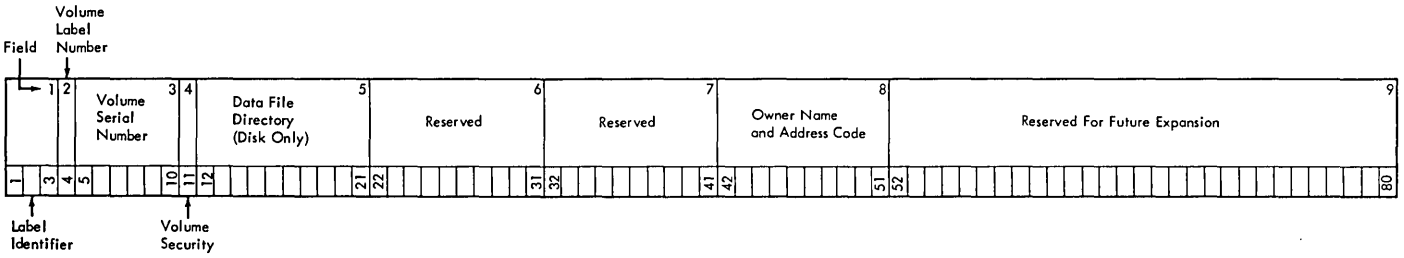


Tape File with Standard and User Labels



*one header and one trailer for Basic Operating System;
two headers and two trailers for Operating System

Standard Volume Label, Tape or DASD

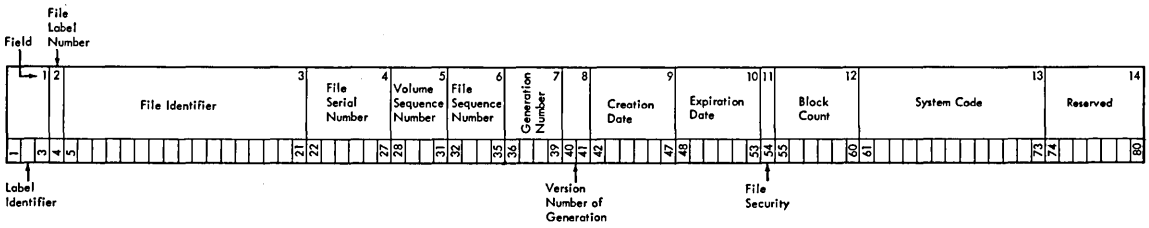


Volume Label Format (80 bytes) for Tape or DASD

FIELD	NAME AND LENGTH	DESCRIPTION
1.	LABEL IDENTIFIER 3 bytes	Must contain VOL to indicate that this is a Volume Label.
2.	VOLUME LABEL NUMBER 1 byte	Indicates the relative position (1-8) of a volume label within a group of volume labels.
3.	VOLUME SERIAL NUMBER 6 bytes	A unique identification code which is assigned to a volume when it enters an installation. This code may also appear on the external surface of the volume for visual identification. It is normally a numeric field 000001 to 999999, however any or all of the 6 bytes may be alphameric.
4.	VOLUME SECURITY 1 byte	Indicates security status of the volume: 0 = no further identification for each file of the volume is required. 1 = further identification for each file of the volume is required before processing.
5.	DATA FILE DIRECTORY 10 bytes	For DASD only. The first 5 bytes contain the starting address (CCHHR) of the VTOC. The last 5 bytes are blank. For tape files, this field is not used and should be recorded as blanks.
6.	RESERVED 10 bytes	Reserved.
7.	RESERVED 10 bytes	Reserved.
8.	OWNER NAME AND ADDRESS CODE 10 bytes	Indicates a specific customer, installation and/or system to which the volume belongs. This field may be a standardized code, name, address, etc.
9.	RESERVED 29 bytes	Reserved.

Note: All reserved fields should contain blanks to facilitate their use in the future. Any information appearing in these fields at the present time will be ignored by the Basic Operating System/360 programs as well as the the Operating System/360 programs.

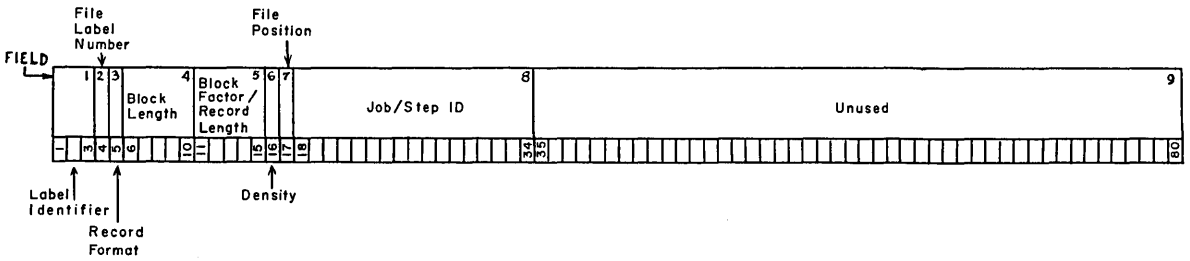
S/360 Tape File Label #1



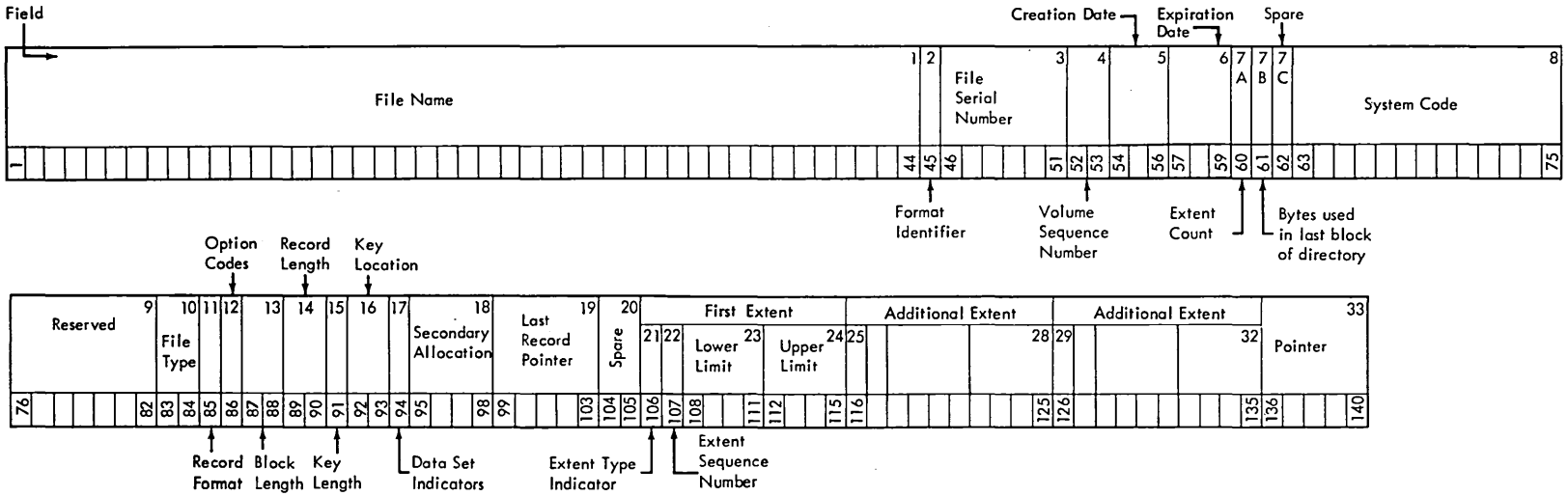
The standard tape file label format and contents are as follows:

FIELD	NAME AND LENGTH	DESCRIPTION
1.	LABEL IDENTIFIER 3 bytes, EBCDIC	identifies the type of label HDR = Header -- beginning of a data file EOF = End of File -- end of a set of data EOV = End of Volume -- end of the physical reel
2.	FILE LABEL NUMBER 1 byte, EBCDIC	always a 1
3.	FILE IDENTIFIER 17 bytes, EBCDIC	uniquely identifies the entire file, may contain only printable characters.
4.	FILE SERIAL NUMBER 6 bytes, EBCDIC	uniquely identifies a file/volume relationship. This field is identical to the Volume Serial Number in the volume label of the first or only volume of a multi-volume file or a multi-file set. This field will normally be numeric (000001 to 999999) but may contain any six alphameric characters.
5.	VOLUME SEQUENCE NUMBER 4 bytes	indicates the order of a volume in a given file or multi-file set. The first must be numbered 0001 and subsequent numbers must be in proper numeric sequence.
6.	FILE SEQUENCE NUMBER 4 bytes	assigns numeric sequence to a file within a multi-file set. The first must be numbered 0001.
7.	GENERATION NUMBER 4 bytes	uniquely identifies the various editions of the file. May be from 0001 to 9999 in proper numeric sequence.
8.	VERSION NUMBER OF GENERATION 2 bytes	indicates the version of a generation of a file.
9.	CREATION DATE 6 bytes	indicates the year and the day of the year that the file was created: Position — Code — Meaning 1 — blank — none 2-3 — 00-99 — Year 4-6 — 001-366 — Day of Year (e.g., January 31, 1965, would be entered as 65031).
10.	EXPIRATION DATE 6 bytes	indicates the year and the day of the year when the file may become a scratch tape. The format of this field is identical to Field 9. On a multi-file reel, processed sequentially, all files are considered to expire on the same day.
11.	FILE SECURITY 1 byte	indicates security status of the file. 0 = no security protection 1 = security protection. Additional identification of the file is required before it can be processed.
12.	BLOCK COUNT 6 bytes	indicates the number of data blocks written on the file from the last header label to the first trailer label, exclusive of tape marks. Count does not include checkpoint records. This field is used in trailer labels.
13.	SYSTEM CODE 13 bytes	uniquely identifies the programming system.
14.	RESERVED 7 bytes	Reserved. Should be recorded as blanks.

System/360 Tape File Label #2 (Operating System only)



FIELD	NAME AND LENGTH	DESCRIPTION
1.	LABEL IDENTIFIER 3 bytes, EBCDIC	identifies the type of label HDR - Header — beginning of a data file EOF - End of File — end of a set of data EOV - End of Volume — end of the physical reel
2.	FILE LABEL NUMBER 1 byte, EBCDIC	always a 2
3.	RECORD FORMAT 1 byte, EBCDIC	indicates the record type F - Fixed V - Variable U - Undefined
4.	BLOCK LENGTH 5 bytes	indicates the length of a block; use maximum length for variable or undefined records
5.	BLOCKING FACTOR/RECORD LENGTH 5 bytes	indicates blocking factor for fixed length records; maximum record length for variable length records; zero for undefined
6.	DENSITY 1 byte	indicates the recording density 0 - 200 1 - 556 2 - 800 - 1600 (not defined yet)
7.	FILE POSITION 1 byte	identifies the conditions that caused creation of the label 0 - if HDR & OPEN; if trailer & CLOSE 1 - if created because of EOV
8.	JOB/STEP ID 17 bytes	identifies the job or step
9.	FUTURE USE 46 bytes	must be recorded as blanks



Format 1: This format is common to all data files on Direct Access Storage Devices.

FIELD	NAME AND LENGTH	DESCRIPTION
1.	FILE NAME 44 bytes, alphameric EBCDIC	This field serves as the key portion of the file label. Each file must have a unique file name. Duplication of file names will cause retrieval errors. The file name can consist of three sections: 1. File ID is an alphameric name assigned by the user and identifies the file. Can be 1-35 bytes if generation and version numbers are used, or 1-44 bytes if they are not used. 2. Generation Number. If used, this field is separated from File ID by a period. It has the format Gnnnn, where G identifies the field as the generation number and nnnn (in decimal) identifies the generation of the file. 3. Version Number of Generation. If used, this section immediately follows the generation number and has the format Vnn, where V identifies the field as the version of generation number and nn (in decimal) identifies the version of generation of the file.

FIELD	NAME AND LENGTH	DESCRIPTION
		Note: Basic Operating System/360 compares the entire FILENAME field against the file name given in the DLAB card. The generation and version numbers are treated differently by Operating System/360.
	The remaining fields comprise the DATA portion of the file label:	
2.	FORMAT IDENTIFIER 1 byte, EBCDIC numeric	1 = Format 1
3.	FILE SERIAL NUMBER 6 bytes, alphameric EBCDIC	Uniquely identifies a file/volume relationship. It is identical to the Volume Serial Number of the first or only volume of a multi-volume file.
4.	VOLUME SEQUENCE NUMBER 2 bytes, binary	Indicates the order of a volume relativ to the first volume on which the data file resides.
5.	CREATION DATE 3 bytes, discontinuous binary	Indicates the year and the day of the year the file was created. It is of the form YDD, where Y signifies the year (0-99) and DD the day of the year (1-366).

FIELD	NAME AND LENGTH	DESCRIPTION
6.	EXPIRATION DATE 3 bytes, discontinuous binary	Indicates the year and the day of the year the file may be deleted. The form of this field is identical to that of Field 5.
7A	EXTENT COUNT 1 byte, binary	Contains a count of the number of extents for this file on this volume. If user labels are used, the count does not include the user label track. This field is maintained by the Basic Operating System/360 programs.
7B*	BYTES USED IN LAST BLOCK OF DIRECTORY 1 byte, binary	Used by Operating System/360 only for partitioned (library Structure) data sets. Not used by Basic Operating System/360.
7C	SPARE 1 byte	Reserved.
8	SYSTEM CODE 13 bytes	Uniquely identifies the programming system. The character codes that can be used in this field are limited to 0-9, A-Z, or blanks.
9*	RESERVED 7 bytes	Reserved.
10*	FILE TYPE 2 bytes	The contents of this field uniquely identify the type of data file: Hex 4000 = Consecutive organization Hex 2000 = Direct-access organization Hex 8000 = Indexed-sequential organization Hex 0200 = Library organization Hex 0000 = Organization not defined in the file label.
11.*	RECORD FORMAT 1 byte	The contents of this field indicate the type of records contained in the file:

Bit Position	Content	Meaning
0 and 1	01	Variable length records
	10	Fixed length records
	11	Undefined format
2	0	No track overflow
	1	File is organized using track overflow (Operating System/360 only)
3	0	Unblocked records
	1	Blocked records
4	0	No truncated records
	1	Truncated records in file
5 and 6	01	Control character ASA code
	10	Control Character machine code
	00	Control Character not stated
7	0	Records have no keys
	1	Records are written with keys.

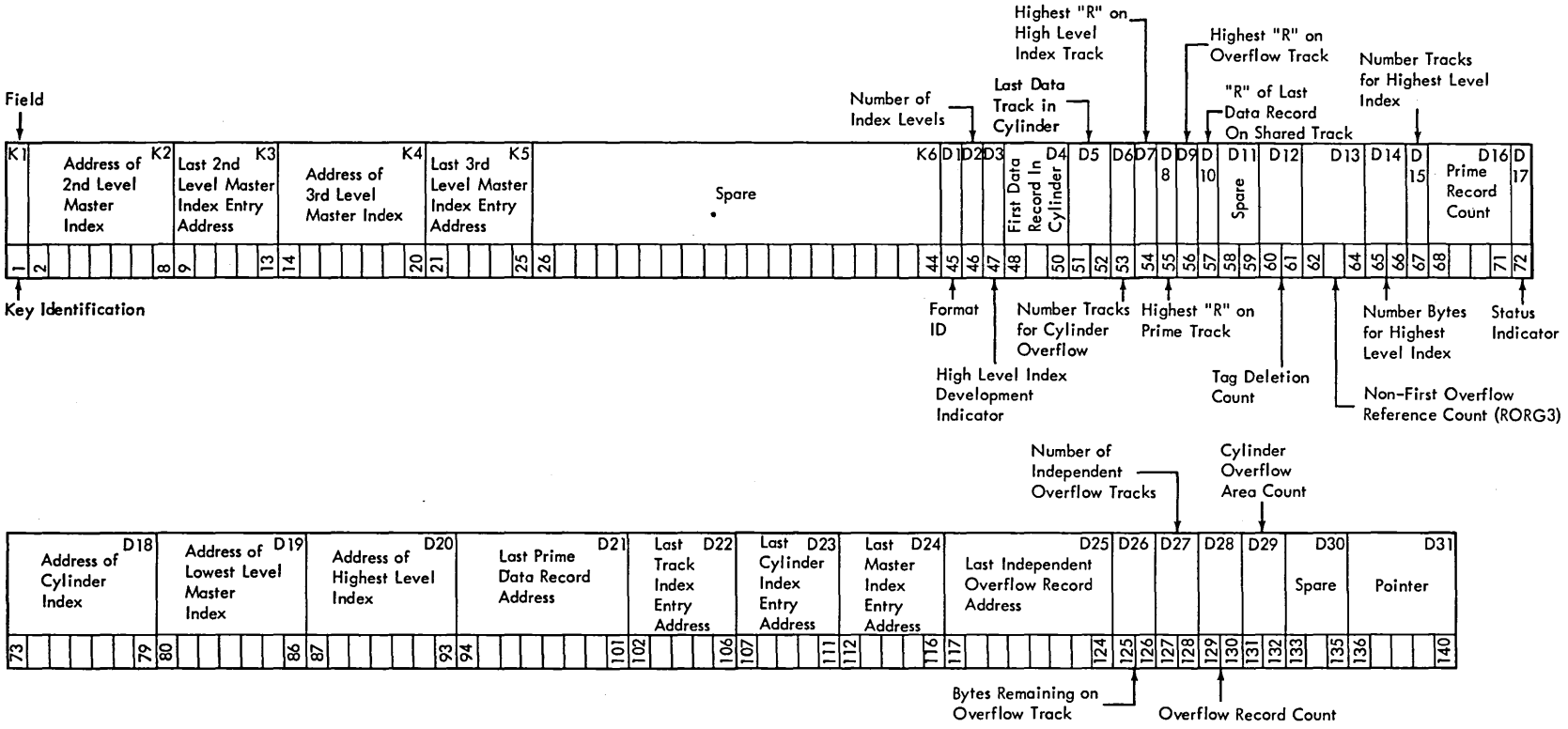
FIELD	NAME AND LENGTH	DESCRIPTION
12.*	OPTION CODES 1 byte	Bits within this field are used to indicate various options used in building the file. Bit 0 = If on, indicates data file was created using Write Validity Check. 1-7 = unused
13.**	BLOCK LENGTH 2 bytes, binary	indicates the block length for fixed length records or maximum block size for variable length blocks.
14.**	RECORD LENGTH 2 bytes, binary	indicates the record length for fixed length records or the maximum record length for variable length records.
15.**	KEY LENGTH 1 byte, binary	indicates the length of the key portion of the data records in the file.
16.**	KEY LOCATION 2 bytes, binary	indicates the high order postion of the data record.
17.	DATA SET INDICATORS 1 byte	Bits within this field are used to indicate the following: BIT 0 If on, indicates that this is the last volume on which this file normally resides. This bit is used by the Basic Operating System/360 DTFSR routine only. None of the other bits in this byte are used by BOS.

FIELD	NAME AND LENGTH	DESCRIPTION
		BIT 1 If on, indicates that the data set described by this file must remain in the same absolute location on the direct access device. 2 If on, indicates that Block Length must always be a multiple of 8 bytes. 3 If on, indicates that this data file is security protected; a password must be provided in order to access it. 4-7 Spare. Reserved for future use.
18.*	SECONDARY ALLOCATION 4 bytes, binary	indicates the amount of storage to be requested for this data file at End of Extent. This field is used by Operating System/360 only. It is not used by Basic Operating System/360 routines. The first byte of this field is an indication of the type of allocation request. Hex code "C2" (EBCDIC "B" indicates blocks (physical records), hex code "E3" (EBCDIC "T") indicates tracks, and hex code "C3" (EBCDIC "C") indicates cylinders. The next three bytes of this field is a binary number indicating how many bytes, tracks or cylinders are requested.
19.*	LAST RECORD POINTER 5 bytes discontinuous binary	points to the last record written in a sequential or partition-organization data set. The format is TTRLL, where TT is the relative address of the track containing the last record, R is the ID of the last record, and LL is the number of bytes remaining on the track following the last record. If the entire field contains binary zeros, the last record pointer does not apply.
20.	SPARE 2 bytes	Reserved.

FIELD	NAME AND LENGTH	DESCRIPTION
21.	EXTENT TYPE INDICATOR 1 byte	indicates the type of extent with which the following fields are associated: HEX CODE 00 Next three fields do not indicate any extent. 01 Prime area (Indexed Sequential); or Consecutive area, etc., (i.e., the extent containing the user's data records.) 02 Overflow area of an Indexed Sequential file. 04 Cylinder Index or master Index area of an Indexed Sequential file. 40 User label track area. 8n Shared cylinder indicator, where n = 1, 2, or 4.
22.	EXTENT SEQUENCE NUMBER 1 byte, binary	indicates the extent sequence in a multi-extent file.
23.	LOWER LIMIT 4 bytes, discontinuous binary	the cylinder and the track address specifying the starting point (lower limit) of this extent component. This field has the format CCHH.
24.	UPPER LIMIT 4 bytes	the cylinder and the track address specifying the ending point (upper limit) of this extent component. This field has the format CCHH.
25-28.	ADDITIONAL EXTENT 10 bytes	These fields have the same format as the fields 21-24 above.
29-32.	ADDITIONAL EXTENT 10 bytes	These fields have the same format as the fields 21-24 above.
33.	POINTER TO NEXT FILE LABEL WITHIN THIS LABEL SET 5 bytes, discontinuous binary	the address (format CCHHR) of a continuation label if needed to further describe the file. If field 10 indicates Indexed Sequential organization, this field will point to a Format 2 file label within this label set. Otherwise, it points to a Format 3 file label, and then only if the file contains more than three extent segments. This field contains all binary zeros if no additional file label is pointed to.

* These fields are not supported by 8K/16K Basic Operating System/360

** 8K/16K BOS supports fields 13-16 for ISFMS only



Format 2: This format is applicable only to Indexed Sequential data files. It is always pointed to by a Format 1 label.

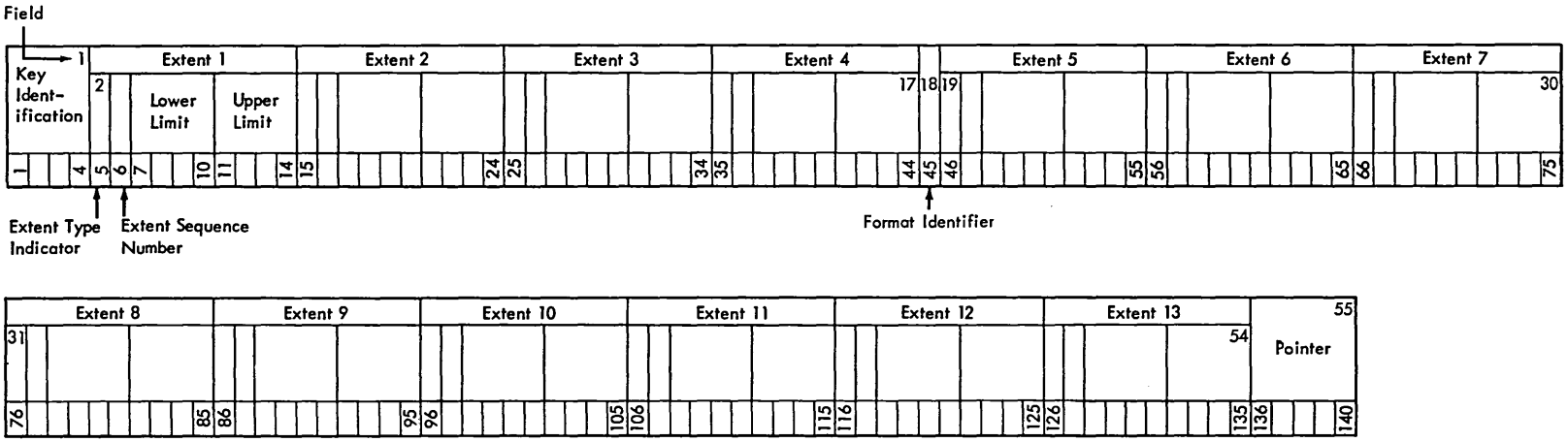
FIELD	NAME AND LENGTH	DESCRIPTION
K1	KEY IDENTIFICATION 1 byte	This byte contains the Hex Code 02 in order to avoid conflict with a file name.
K2*	ADDRESS OF 2ND LEVEL MASTER INDEX 7 bytes, discontinuous binary	This field contains the address of the first track of the second level of the master index, in the form MBBCCHH.
K3*	LAST 2ND LEVEL MASTER INDEX ENTRY 5 bytes, discontinuous binary	This field contains the address of the last index entry in the second level of the master index, in the form CCHHR.
K4*	ADDRESS OF 3RD LEVEL MASTER INDEX 7 bytes, discontinuous binary	This field contains the address of the first track of the third level of the master index, in the form MBBCCHH.
K5*	LAST 3RD LEVEL MASTER INDEX ENTRY 5 bytes, discontinuous binary	This field contains the address of the last entry in the third level of the master index, in the form CCHHR.
K6*	SPARE 19 bytes	Reserved.
D1	FORMAT IDENTIFIER 1 byte, EBCDIC numeric	'2' = Format 2
D2	NUMBER OF INDEX LEVELS 1 byte, binary	The contents of this field indicate how many levels of index are present with an Indexed Sequential file.

FIELD	NAME AND LENGTH	DESCRIPTION
D3*	HIGH LEVEL INDEX DEVEOPMENT INDICATOR 1 byte, binary	This field contains the number of tracks determining development of Master Index.
D4	FIRST DATA RECORD IN CYLINDER 3 bytes	This field contains the address of the first data record on each cylinder in the form HHR.
D5	LAST DATA TRACK IN CYLINDERS 2 bytes	This field contains the address of the last data track on each cylinder, in the form HH.
D6	NUMBER OF TRACKS FOR CYLINDER OVERFLOW 1 byte, binary	This field contains the number of tracks in cylinder overflow area.
D7	HIGHEST "R" ON HIGH-LEVEL INDEX TRACK 1 byte	This field contains the highest possible R on track containing high-level index entries.
D8	HIGHEST "R" ON PRIME TRACK 1 byte	This field contains the highest possible R on prime data tracks for form F records.
D9	HIGHEST "R" ON OVERFLOW TRACK 1 byte	This field contains the highest possible R on overflow data tracks for form F records.
D10	"R" OF LAST DATA RECORD ON SHARED TRACK 1 byte	This field contains the R of the last data record on a shared track.
D11*	SPARE 2 bytes	Reserved.
D12	TAG DELETION COUNT 2 bytes, binary	This field contains the number of records that have been tagged for deletion.
D13	NON-FIRST OVERFLOW REFERENCE COUNT (RORG3) 3 bytes, binary	This field contains a count of the number of random references to a nor first overflow record.
D14	NUMBER OF BYTES FOR HIGHEST-LEVEL INDEX 2 bytes, binary	The contents of this field indicate how many bytes are needed to hold the highest-level index in main storage.
D15*	NUMBER OF TRACKS FOR HIGHEST-LEVEL INDEX 1 byte, binary	This field contains a count of the number of tracks occupied by the highest-level index.
D16	PRIME RECORD COUNT 4 bytes, binary	This field contains a count of the number of records in the prime data area.
D17	STATUS INDICATOR 1 byte	The eight bits of this byte are used for the following indications:

bit	description
0	last block full
1	last track full
2-7	must remain off

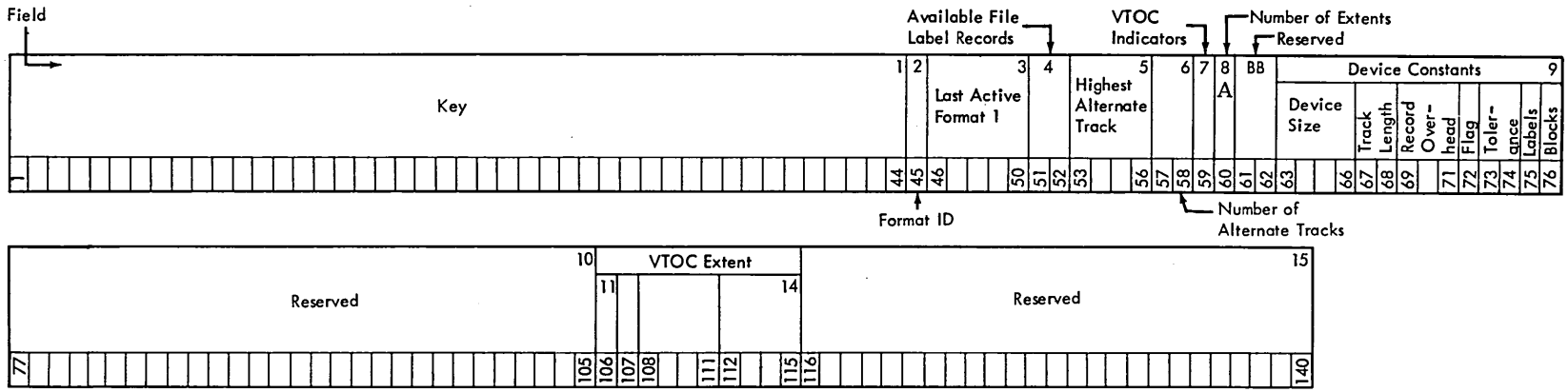
FIELD	NAME AND LENGTH	DESCRIPTION
D18	ADDRESS OF CYLINDER INDEX 7 bytes	This field contains the address of the first track of the cylinder index, in the form MBBCCHH.
D19	ADDRESS OF LOWEST-LEVEL MASTER INDEX 7 bytes	This field contains the address of the first track of the lowest-level index of the high level indexes, in the form MBBCCHH.
D20*	ADDRESS OF HIGHEST-LEVEL INDEX 7 bytes	This field contains the address of the first track of the highest level master index, in the form MBBCCHH.
D21	LAST PRIME DATA RECORD ADDRESS 8 bytes	This field contains the address of the last data record in the prime data area, in the form MBBCCHHR.
D22	LAST TRACK INDEX ENTRY ADDRESS 5 bytes	This field contains the address of the last normal entry in the track index on the last cylinder in the form CCHHR.
D23	LAST CYLINDER INDEX ENTRY ADDRESS 5 bytes	This field contains the address of the last index entry in the cylinder index in the form CCHHR.
D24	LAST MASTER INDEX ENTRY ADDRESS 5 bytes	This field contains the address of the last index entry in the master index in the form CCHHR.
D25	LAST INDEPENDENT OVERFLOW RECORD ADDRESS 8 bytes	This field contains the address of the last record written in the current independent overflow area, in the form MBBCCHHR.
D26*	BYTES REMAINING ON OVERFLOW TRACK 2 bytes, binary	This field contains the number of bytes remaining on current independent overflow track.
D27	NUMBER OF INDEPENDENT OVERFLOW TRACKS (RORG2) 2 bytes, binary	This field contains the number of tracks remaining in independent overflow area.
D28	OVERFLOW RECORD COUNT 2 bytes, binary	This field contains a count of the number of records in the overflow area both indep. and dep.
D29	CYLINDER OVERFLOW AREA COUNT (RORG1) 2 bytes, binary	This field contains the number of cylinder overflow areas full.
D30	SPARE 3 bytes	Reserved.
D31*	POINTER TO FORMAT 3 FILE LABEL 5 bytes	This field contains the address (in the form CCHHR) of a Format 3 file label if more than 3 extent segments exist for the data file within this volume. Otherwise it contains binary zeros.

* These fields are not supported by 8K/16K Basic Operating System/360.



Format 3: This format is used to describe extra extent segments on the volume if there are more than can be described in the Format 1 (and Format 2 if it exists) file label. This file label is pointed to by a Format 1, Format 2, or another Format 3 file label.

FIELD	NAME AND LENGTH	DESCRIPTION
1.	KEY IDENTIFICATION 4 bytes	Each byte of this field contains the Hex Code 03 in order to avoid conflict with a data file name.
2-17	EXTENTS (in KEY) 40 bytes	Four groups of fields identical in format to fields 21-24 in the Format 1 label are contained here.
18.	FORMAT IDENTIFIER 1 byte, EBCDIC numeric	3 = Format 3
19-54	ADDITIONAL EXTENTS 90 bytes	Nine groups of fields identical in format to fields 21-24 in the Format 1 label are contained here.
55.	POINTER TO NEXT FILE LABEL 5 bytes	This field contains the address (in the form CCHHR) of another Format 3 label if additional extents must be described. Otherwise, it is all binary zeros.



Format 4: This format is used to describe the Volume Table of Contents and is always the first file label in the VTOC. There must be one and only one of these Format 4 file labels per volume.

FIELD	NAME AND LENGTH	DESCRIPTION
1.	KEY FIELD 44 bytes, binary	Each byte of this field contains the Hex Code 04 in order to provide a unique key.
2.	FORMAT ID 1 byte, EBCDIC numeric	4 = Format 4
3.	LAST ACTIVE FORMAT 1 5 bytes	Contains the address (in the form CCHHR) of the last active Format 1 file label. It is used to stop a search on a file name.
4.	AVAILABLE FILE LABEL RECORDS 2 bytes, binary	Contains a count of the number of unused records in the VTOC.
5.	HIGHEST ALTERNATE TRACK 4 bytes	Contains the highest address (in the form CCHH) of a block of tracks set aside as alternates for bad tracks.
6.	NUMBER OF ALTERNATE TRACKS 2 bytes, binary	Contains the number of alternate tracks available.
7.	VTOC INDICATORS 1 byte	Bit 0, if on, indicates no DADSM (format 5) label, or DADSM label does not reflect true status of volume. Bit 1-7 not used.
8A.	NUMBER OF EXTENTS 1 byte	Contains the hexadecimal constant 01, to indicate one extent in the VTOC.
8B.	RESERVED 2 bytes	Reserved.
9.	DEVICE CONSTANTS 14 bytes	This field contains constants describing the device on which the volume was mounted when the VTOC was created. The following describes each of the subfields.

Device Size (4 bytes) - The number of cylinders (CC) and tracks per cylinder (HH).

Track Length (2 bytes) - The number of available bytes on a track exclusive of home address and record zero (record zero is assumed to be a non-keyed record with an eight byte data field).

Record Overhead (3 bytes) - The number of bytes required for gaps, check bits, and count field for each record. This value varies according to the record characteristics and thus is broken down into three subfields.

- I - Overhead required for a keyed record other than the last record on the track.
- L - Overhead required for a keyed record that is the last record on the track.
- K - Overhead bytes to be subtracted from I or L if the record does not have a key field.

Flag (1 byte) - Further defines unique characteristics of the device.

bits	meaning
0-5	reserved
6	CC and HH must be used as 1-byte values, as in the case of the 2321.
7	A tolerance factor must be applied to all but the last record on the track.

FIELD	NAME AND LENGTH	DESCRIPTION

Tolerance (2 bytes) - A value that is to be used to determine the effective length of the record on the track. The effective length of a record is calculated in the following manner:

1. Add the key length to the data length of the record.
2. Test bit 7 in the flag byte:
   a. if 0 go to 3
   b. multiply value from 1 by the tolerance factor
   c. shift result 9 bits to the right
3. Add overhead bytes to the result.

NOTE: Step 2 is not required if the calculation is for the last record on the track.

Labels/Track (1 byte) - A count of the number of labels that can be written on each track in the VTOC. (Number of full records of 44-byte key and 96-byte data lengths that can be contained on one track of this device).

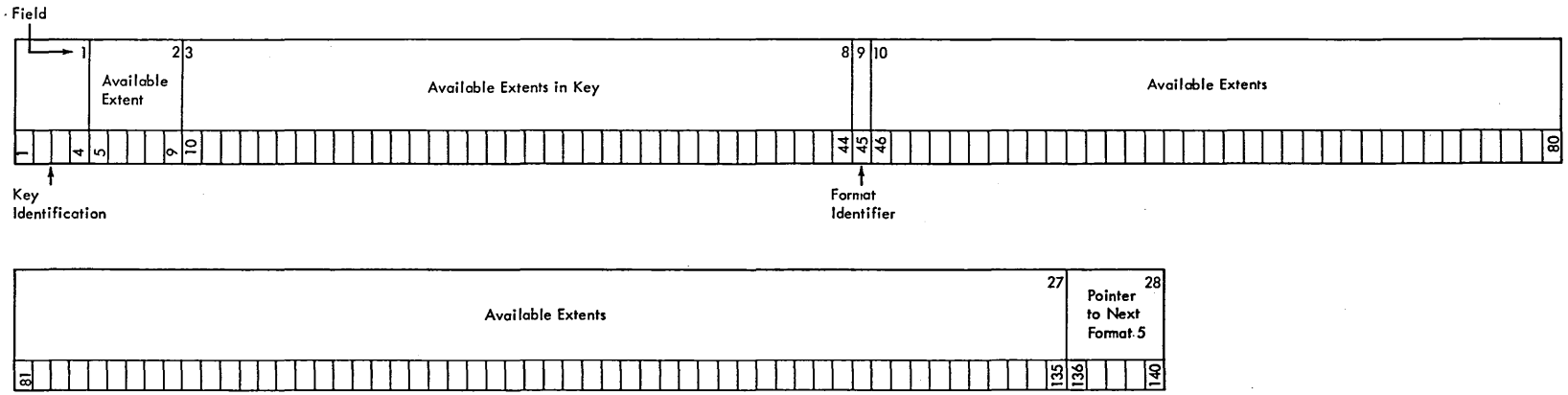
Directory Blocks/Track (1 byte) - A count of the number of directory blocks that can be written on each track for an Operating System/360 partitioned data set. (Number of full records of 8-byte key and 256-byte data lengths that can be contained on one track of this device.)

The following illustrates the device constants field for the various direct access devices:

Device	CC	HH	Track Length	I	L	K	Flag	Tolerance	Labels/ Track	Dir Blk/ Track
2311	203	10	3656	82	55	20	1	537	16	10
2321	20 10	5 20	2027	101	47	16	3	537	8	5
2301	0	200	20616	186	186	53	0	512	63	45
2302	250	46	5070	82	55	20	1	537	22	14
7320	0	400	2129	111	43	14	1	537	8	5

NOTE: CCHH for the 2321 above are separate 1 byte quantities.

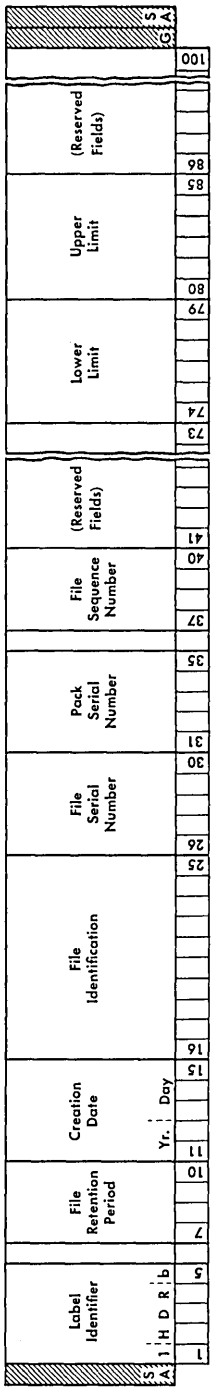
10.	RESERVED 29 bytes	Reserved.
11-14.	VTOC EXTENT 10 bytes	These fields describe the extent of the VTOC, and are identical in format to fields 21-24 of the Format 1 file label. Extent type 01 (prime data area).
15.	RESERVED 25 bytes	Reserved.



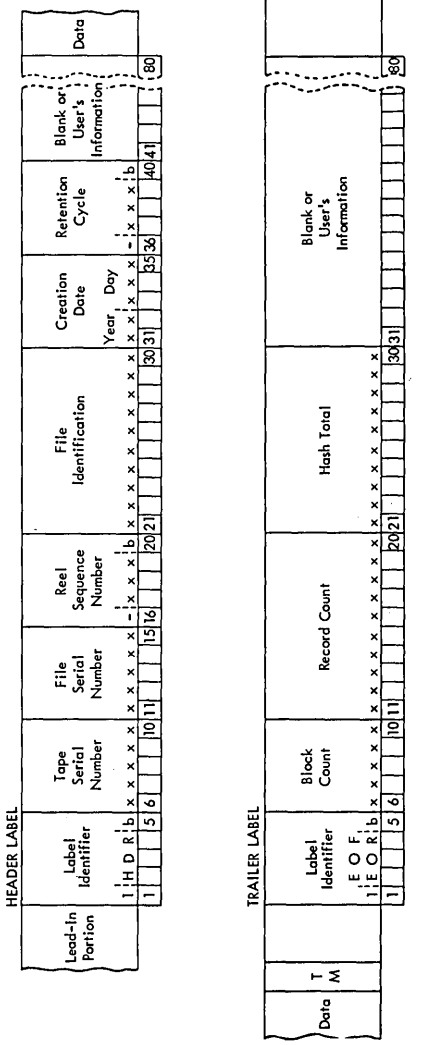
Format 5: This format is used for Direct Access Device Space Management (DADSM) only.

FIELD	NAME AND LENGTH	DESCRIPTION
1.	KEY IDENTIFICATION 4 bytes	Each of these four bytes is a hex 05.
2.	AVAILABLE EXTENT 5 bytes	This field indicates an extent of space available for allocation to a data file. The first two bytes are relative track address. The next two are the number of full cylinders included in the extent. The last byte is the number of tracks in addition to the cylinders in the extent.
3-8	AVAILABLE EXTENTS IN KEY 35 bytes	These fields are identical to field 2. They are in relative track address sequence.
9.	FORMAT IDENTIFIER 1 byte EBCDIC numeric	5 = Format 5
10-27	AVAILABLE EXTENTS 90 bytes	These fields are the same as Field 2. There are 26 available extent fields in the Format 5 label.
28.	POINTER TO NEXT FORMAT 5 5 bytes	Contains the address (in the form CCHHR) of the next Format 5 file label if one exists.

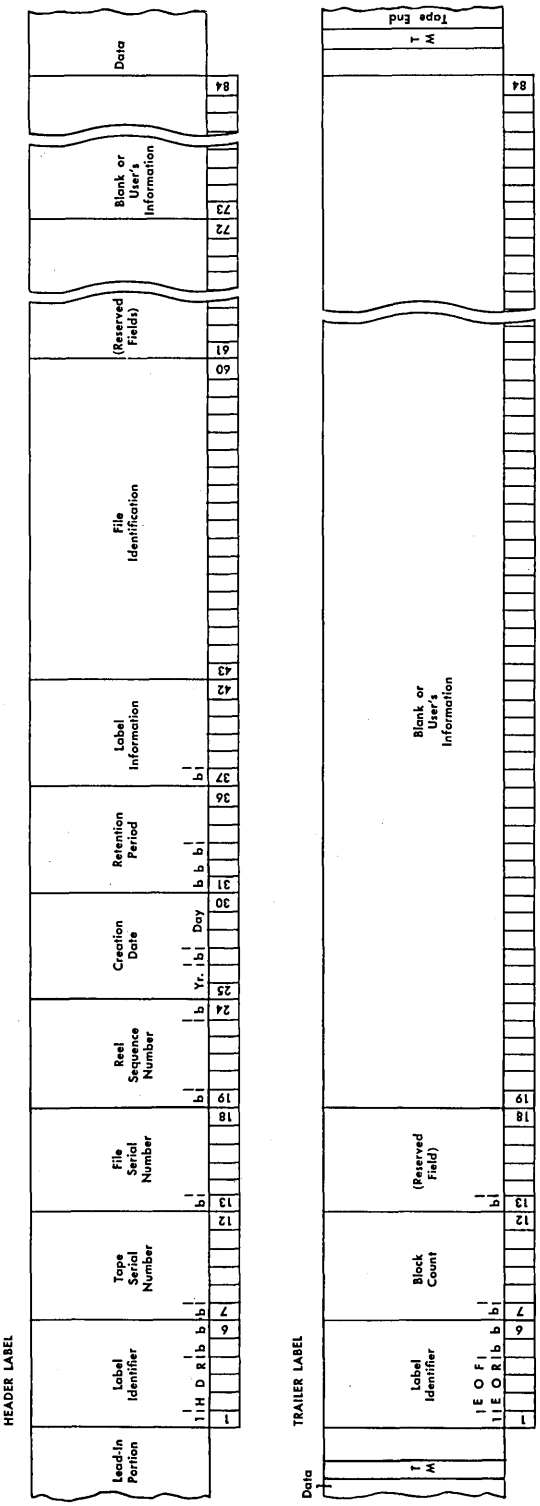
Non-System 360



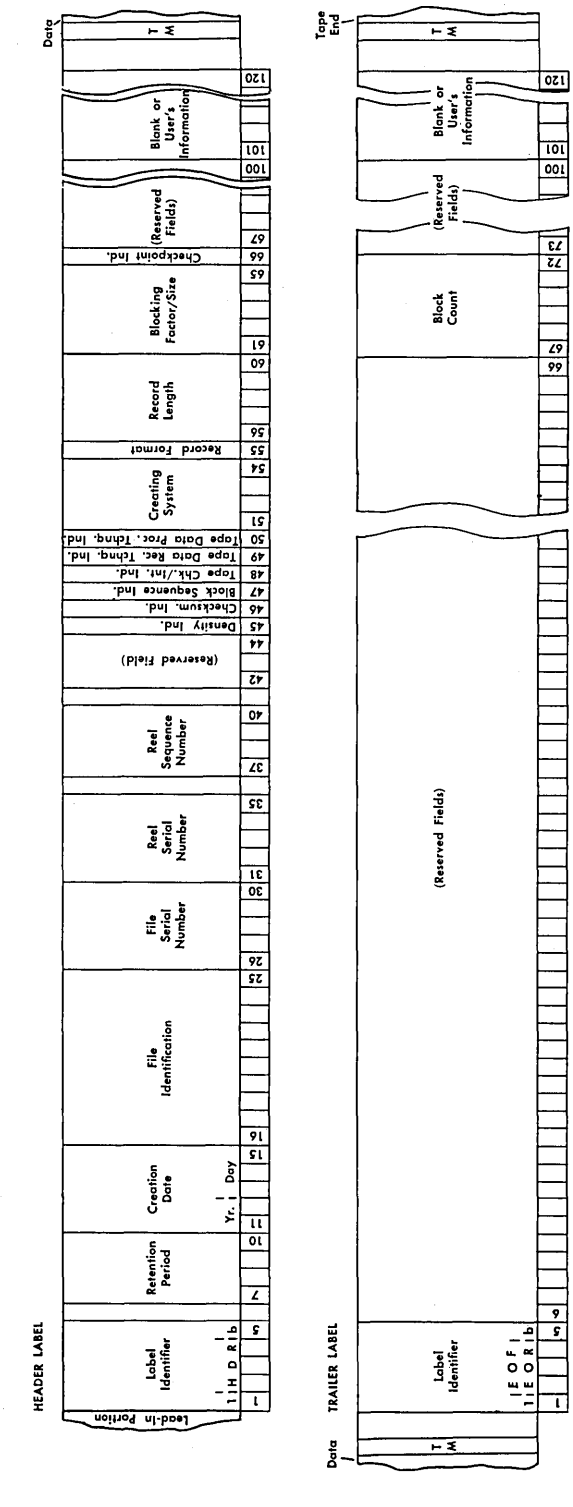
Schematic of Standard Disk Label



Schematic of Standard 80-Character Tape Label

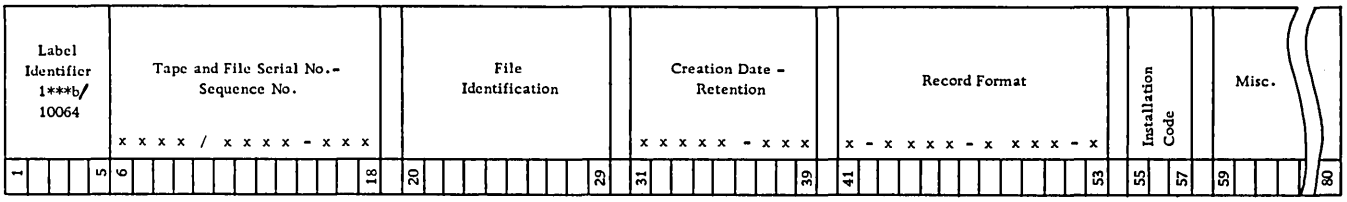


Schematic of Standard 84-Character Tape Label

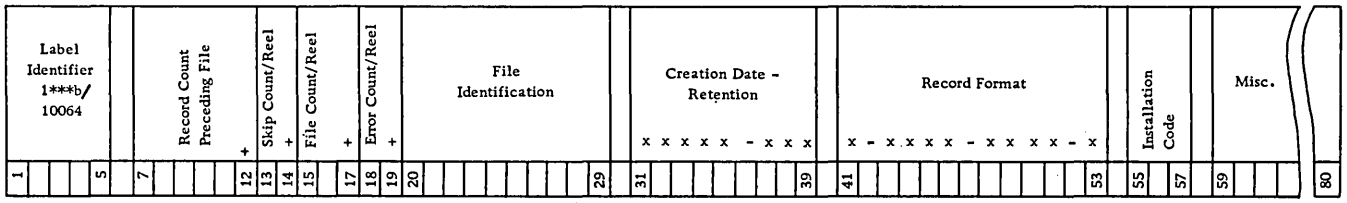


Schematic of Standard 120-Character Tape Label Fields

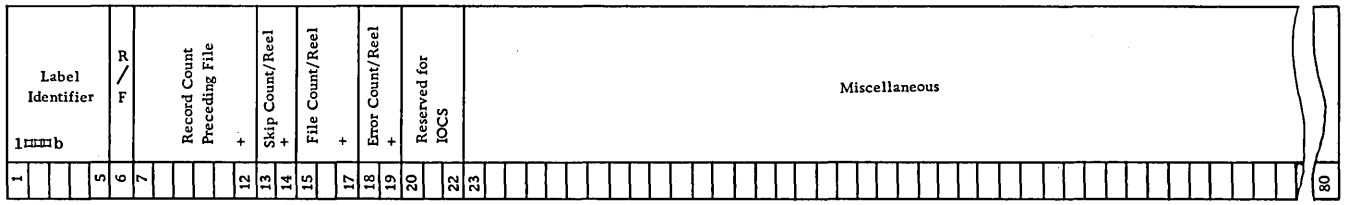
Non-System 360



Standard Header Label



Standard Intermediate Header Label



Standard Trailer Label

Schematic of 7080 Tape Labels

IOCS Characteristics

S/360	Basic Programming Support	Basic Operating System					
		8K Disk	16K Tape/16K Disk		8K Disk/16K Disk		
File Organization	Sequential	Sequential	Sequential		Direct (DAM)	Index Sequential (ISFMS)	
Primary Macros	GET/PUT DTFSR	GET/PUT DTFSR	GET/PUT, READ/WRITE DTFCN, DTFCD, DTFPR, DTFMT, DTFSD (Disk), DTFSR		READ/WRITE DTFDA	READ/WRITE (Random) GET/PUT (Sequential) DTFIS	
Record Formats			With GET/PUT	With READ/WRITE			
Fixed Unblocked	X	X	X	X	X	X	
Fixed Blocked	X	X	X		IOCS handles as unblocked. User must deblock and block.	X	
Variable Unblocked	X	X	X				
Variable Blocked	X	X	X				
Undefined	X	X	X	X	X		
I/O and Work Areas	1 I/O 1 I/O & work 2 I/O 2 I/O & work	1 I/O 1 I/O & work 2 I/O 2 I/O & work	1 I/O 1 I/O & work 2 I/O 2 I/O & work	1 I/O	1 I/O & no work	1 I/O 1 I/O & work Work required to load or add records	
DASD							
Key Requirements (External)		Without keys	Without Keys		With or Without Keys	Keys Required	
Indices		None	None		None. Uses track & record references (key or physical location). Multi track search to end of cylinder if using key.	Master - optional-immediately precedes cylinder index Cylinder - located anywhere except cylinder on which data is stored. If more than one cylinder is required, all must be in same disk pack. Track - begins on 1st track of cylinder it indexes; may require full track, partial or more than one track; data follows	
Additions and/or overflows	NO DASD SUPPORT	Rewrite the file	Rewrite the file		With key reference - IOCS can indicate no room & user provides OF routine With location reference - entirely user responsibility; traditional chaining is common method	Cylinder OF area, independent OF area or both may be specified. If no room in specified area, branch to user's routine.	
Data Location		1 or more sets of limits. Within each set records must be adjacent; within 1 volume, the sets need not be adjacent and more than one volume is permissible. (Except on work files). Only one volume need be in line at one time.			Entire file on line; address checked to see if it is within extent limits only on 16K	1 extent for prime area on 1 pack. If area exceeds 1 pack the additional extents must be adjacent and on line. Volume table of contents (VTOC) can interrupt prime data only on cylinder zero.	
Devices Supported	1442 1403 1015 (inquiry) 2501 1404* 1052 2520 1443 2400 2540 1445 2671 1285 *Continuous form only	1442 1403 2671 2260** 2501 1404* 2701/2/3 with ** 2400 2520 1443 1030 1060 2311 (8K/16K disk) 2540 1445 1050 2740 2321** 1285 1015 (inquiry) 1052 ** 16K disk					
Reference Manuals	C24-3354 C24-3555	C24-3361 (8K) C24-3372 (8K)	C24-3430 (16K tape) C24-3427 (16K disk)				

Tape File Labels - BPS & BOS

Standard

Required
- 1 standard volume label per reel
- 1 standard file header & 1 standard file trailer per logical file
  These labels are fixed in length & format & processed by IOCS

Additional
- Up to 7 additional volume labels
- Up to 7 additional file headers and up to 7 additional file trailers
  The volume labels are fixed in length but not format. The file labels are fixed in length, identifier & format. All are bypassed by IOCS unless user desires to provide physical IOCS routines.
- Up to 8 user header labels and up to 8 user trailer labels
  These are fixed in length and identifier but not in format. IOCS reads and writes and passes to user processing routine.

Non-standard - Unrestricted in size, format or identifier
User supplies all routines (physical IOCS). If tape mark follows labels & no checking is required, IOCS can position to first data record. On output OPEN puts tape mark after header label unless DTF specifies none. CLOSE puts tape mark after last data record before trailer & after trailer.

Unlabelled
- Input: If no tape mark, first record is processed as data
  Output: Assumes output tape is unlabelled and writes tape mark unless DTF specifies none
  Unlabelled tapes can be read backwards only if written on S/360, not in data conversion mode and have a tape mark as the first record

Disk Labels - see System/360 Label Formats

S/360 DATA MANAGEMENT (IOCS) - OPERATING SYSTEM

Organization	Sequential		Partitioned	Indexed Sequential			Direct	Telecommunications	
Access Method	QSAM	BSAM	BPAM	QISAM		BISAM	BDAM	QTAM	BTAM
				LOAD	SCAN				
Primary Macro Instructions	GET, PUT PUTX	READ WRITE	READ, WRITE FIND;STOW	PUT	SETL GET, PUTX	READ WRITE	READ WRITE	GET PUT	READ WRITE
Synchronization of Program with Input-Output Device	Automatic	CHECK	CHECK	Automatic	Automatic	WAIT	WAIT	Automatic	WAIT
Record Format	Logical F, V Block U	Block F, V, U	Block (part of a member F, V, U)	Logical F, V	Logical F, V	Logical F, V	Block F, V, U	Message segment of Block, U	Block U
Buffer Creation and Construction	BUILD GETPOOL Automatic	BUILD GETPOOL Automatic	BUILD GETPOOL Automatic	BUILD GETPOOL Automatic	BUILD GETPOOL Automatic	BUILD GETPOOL Automatic	BUILD GETPOOL Automatic	BUFFER	BUILD GETPOOL Automatic
Buffer Technique	Automatic; Simple Exchange	GETBUF FREEBUF	GETBUF FREEBUF	Automatic: Simple	Automatic: Simple	GETBUF FREEBUF Dynamic FREEDBUF	GETBUF FREEBUF Dynamic FREEDBUF	Automatic: Chained Segment	GETBUF FREEBUF Dynamic FREEDBUF
Transmittal Modes (work area/buffer)	Move Locate Substitute	-	-	Move Locate	Move Locate	-	-	Move	-
Devices Supported	*continuous form only		1442 2501 2520 2540 1285	1403 1404* 1443 1445		2671 2701/2/3 with 1030 1060 1050 2740	1015 1052 2400 7340	2301 2302 2303 2311 2314	2321 2250 2260 2280/1/2
Reference Manuals	C28-6535		C28-6537			C28-6553			

F = fixed V = variable U = undefined

NON-SYSTEM/360
IOCS TAPE

	7040/7044	7090/7094	7070/7074	7080	1440/1311 1460/1301	1401	1460	1410	1410/7010 Operating System	
Minimum Record Size	3 words	3 words	3 words	15	5	13	13	13	13	
Maximum Record Size	2000 words	2000 words	9999 words	9995	999	999				
Maximum Block Size	2000 words	2000 words	9999 words	39995		9999				
Fixed-Length Unblocked Records	Type 1	X		X	Form 1	Form 1	Form 1	Form 1	Form 1	Form 3
Block Character Count										Pos. 1-4 in block
Record Mark Required				Option				GET, PUT		GET, PUT
Fixed-Length Blocked Records	Type 1	X	Form 1	X	Form 2	Form 2	Form 2	Form 2	Form 2	
Record Mark Required				X	X	X	X	X	X	
Padding Required		Option			X	X	X	X		
Variable-Length Unblocked Records	Type 2	X		X	Form 3	Form 3	Form 3	Form 3	Form 1	Form 3
Record Mark Required				X	GET, PUT	Option	GET, PUT	GET, PUT	GET, PUT	GET, PUT
Record Length Indicator	BCD Tape Only - in first word in record			Pos. 1-5 in record						
Block Length Indicator				Pos. 1-3 in block						Pos. 1-4 in block
Block Factor				Pos. 4-5 in block						
Variable-Length, Variable Block Factor, Maximum Block Size	Type 2	X	Form 3	X	Form 4	Form 4	Form 4	Form 4	Form 4	
Record Mark Required				X	X	X	X	X	X	
Record Length Indicator	BCD Tape Only - in first word in record		X	Pos. 2-5 in rec.	3 pos. in rec.	3 pos. in rec.	3 pos. in rec.	4 pos. in rec.	Pos. 1-4 rec.	
Block Length Indicator				Pos. 1-3 in block	Pos. 1-4 in block	Pos. 1-4 in block	Pos. 1-4 in block	Pos. 1-4 in block	Pos. 1-4 in block	
Block Factor				Pos. 4-5 in block						
Variable-Length Records, Fixed Blocking Factor			Form 2							
Maximum Record Size Specified Record Mark Required			X Short Rec.							
Variable-Length Sections, Variable-Length Records, Fixed Block Factor			Form 4							
Maximum Section Length Specified Record Mark Required			X Short Section							
Recording Mode										
6-bit (move)	X	X	X	X	Move	Move	Move	Move	Move	
8-bit unpacked (load)	X		X	X						
8-bit packed			Disk Hypertape	Disk Hypertape						

NON-SYSTEM/360
IOCS DISK

	1405	1301					1311		
	1410	7040/7044	7090/7094	7070/7074	7080	1460	1410	1401,1440	1410
Fixed-Length Unblocked Records	Form 1	Type 1	X		X	Form 1	Form 1	Form 1	Form 1
Record Mark Required	GET, PUT				Option		Option		GET, PUT
Fixed-Length Blocked Records	Form 2	Type 1	X	Form 1	X	Form 2	Form 2	Form 2	Form 2
Record Mark Required	X				X	GET, PUT	X	GET, PUT	X
Padding Required	X				Option		X		X
Block Length Indicator									Pos. 1-4 in block
Variable-Length Unblocked Records		Type 2	X		X		Form 3*		Form 3
Record Mark Required					X		Option		GET, PUT
Record Length Indicator		In first word of record			Pos. 1-5 in rec.				4 pos. in rec.
Block Factor					Pos. 4-5 in blk.				
Variable-Length, Variable Block Factor	Form 4	Type 2	X	Form 3	X	Form 4	Form 4	Form 4	Form 4
Record Mark Required	X				X	X	X	X	X
Record Length Indicator	1 to 4 pos. in record	In first word of record		X	Pos. 1-5 in rec.	3 pos. in rec.	4 pos. in rec.	3 pos. in rec.	1 to 4 pos. in rec.
Block Length Indicator	Pos. 1-4 in block				Pos. 1-3 in blk.	Pos. 1-4 in blk.	Pos. 1-4 in blk.	Pos. 1-4 blk.	Pos. 1-4 blk.
Block Factor					Pos. 4-5 in blk.				
Recording Modes Used									
6-bit (move)	Move	X	X	X	X	Move	Move	Move	Move
8-bit unpacked (load)		X		X	X				Load
8-bit packed				X	X				
Available positions per track									
6-bit	1000	2796	2796	2780	2800	2800	2800	2000	2000
8-bit unpacked	880	2160	2160	2150	2165	2165	2165	1800	1800
8-bit packed				4300	4330				
Full track mode (special)								2980	2980

* 1410 form 3 record treated as a form 1 record.

IBM System/360 Character Sets

These charts present the character sets used by IBM System/360 programming languages, chain and train printers, and typewriter-printers. They also include the character sets used by some current-system equipment for cases in which such equipment is used with, or as part of, a transition to System/360.

SYSTEM/360 PROGRAMMING LANGUAGE CHARACTER SETS

SYSTEM/360 PROGRAMMING LANGUAGE CHARACTER SETS

<table>
<tr><th rowspan="2">PROGRAM LANGUAGE</th><th colspan="4">ACCEPTABLE SOURCE LANGUAGE CHARACTERS</th><th rowspan="2">COMMENTS (1) CHARACTERS</th></tr>
<tr><th>TOTAL</th><th>ALPHA</th><th>NUMERIC</th><th>SPECIAL</th></tr>
<tr><td>Assembler(2)</td><td>51</td><td>A-Z</td><td>0-9</td><td>$ # @ + - , = . * ( ) ' / & ƀ</td><td>Any EBCDIC character</td></tr>
<tr><td>Basic Assembler(2)</td><td>45</td><td>A-Z</td><td>0-9</td><td>+ - , . * ( ) ' ƀ</td><td>Any EBCDIC character</td></tr>
<tr><td>COBOL(3)</td><td>51</td><td>A-Z</td><td>0-9</td><td>$ + - , = . * ( ) ' / > < ; ƀ</td><td>Any EBCDIC character</td></tr>
<tr><td>FORTRAN (E)</td><td>48</td><td>A-Z, $ (4)</td><td>0-9</td><td>+ - , = . * ( ) / : ƀ</td><td>Any EBCDIC character</td></tr>
<tr><td>FORTRAN (H)</td><td>49</td><td>A-Z, $ (4)</td><td>0-9</td><td>+ - , = . * ( ) ' / & ƀ</td><td>Any EBCDIC character</td></tr>
<tr><td>PL/I (60-char.)(5)</td><td>60</td><td>A-Z, $ @ #</td><td>0-9</td><td>+ - , = . * ( ) ' / & > < ; : % ¬ | _ ? ƀ</td><td>Any EBCDIC character</td></tr>
<tr><td>PL/I (48-char.)(5)</td><td>48</td><td>A-Z, $</td><td>0-9</td><td>+ - , = . * ( ) ' / ƀ</td><td>Any EBCDIC character</td></tr>
</table>

NOTES:

(1) Acceptable characters in comments are those in the source language plus those in this column, provided the I/O device is capable of handling these characters.

(2) Character constants and character self-defining terms may be represented by any EBCDIC character.

(3) In a non-numeric literal in System/360 COBOL, any EBCDIC character except the quotation mark may be used. (However, such characters may not be acceptable to other COBOL Compilers). The EBCDIC codes for @ $ and # may be used for alphabetic characters in foreign-language COBOL for overseas use.

(4) In both levels of System/360 FORTRAN, $ is an optional alphabetic character.

(5) The following character(s) and/or operator(s) available in the 60-character set of PL/I are replaced as follows in the 48-character set:

<table>
<tr><td>PL/I (60-character set)→</td><td>:</td><td>;</td><td>%</td><td>></td><td>>=</td><td>¬=</td><td><=</td><td><</td><td>¬</td><td>|</td><td>&</td><td>||</td></tr>
<tr><td>PL/I (48-character set)→</td><td>..</td><td>.,</td><td>//</td><td>GT</td><td>GE</td><td>NE</td><td>LE</td><td>LT</td><td>NOT</td><td>OR</td><td>AND</td><td>CAT</td></tr>
</table>

CHARACTER SETS FOR IBM 29 CARD PUNCH AND IBM 59 VERIFIER

EXPANDED KEYBOARD MODELS

The IBM 29 Card Punch, Models A12, A22, B12 and B22, and the IBM 59 Verifier, Model 2, have an expanded character set. Each character can be punched or verified with one key depression. For printing versions of the IBM 29, each character is printed as well as punched with the single key depression. These expanded keyboard models have 64 characters consisting of 26 alphabetic (A-Z), 10 numeric (0-9), a space bar (no punches), and the following 27 special characters:

<table>
<tr><td>Card Code →</td><td>12</td><td>12-8-3</td><td>12-8-4</td><td>11</td><td>11-8-3</td><td>11-8-4</td><td>0-8-3</td><td>0-8-4</td><td>0-1</td><td>8-3</td><td>8-4</td><td>12-8-2</td><td>12-8-5</td><td>12-8-6</td><td>12-8-7</td><td>11-8-2</td><td>11-8-5</td><td>11-8-6</td><td>11-8-7</td><td>0-8-2</td><td>0-8-5</td><td>0-8-6</td><td>0-8-7</td><td>8-2</td><td>8-5</td><td>8-6</td><td>8-7</td></tr>
<tr><td>Character →</td><td>&</td><td>.</td><td><</td><td>-</td><td>$</td><td>*</td><td>,</td><td>%</td><td>/</td><td>#</td><td>@</td><td>¢</td><td>(</td><td>+</td><td>|</td><td>!</td><td>)</td><td>;</td><td>¬</td><td>(blank space)</td><td>_</td><td>></td><td>?</td><td>:</td><td>'</td><td>=</td><td>"</td></tr>
</table>

The 64-character set of expanded keyboard models may be replaced by any one of the 48-character sets A-K shown on reverse side. The keyboard is still a 64-character keyboard, but only 48 characters have identified keytops (and only 48 characters can be printed). When duplicating, all 64 characters can be punched, and only these 64 characters should be duplicated to avoid damage to the code plate.

NUMERIC KEYBOARD MODELS.

The IBM 29 Card Punch, Models A11, A21, B11 and B21, and the IBM 59 Verifier, Model 1, have a numeric keyboard with 12 keys. The numerics 0-9 and only those special characters with "12" or "11" punch coding can be punched (and printed on the printing version) or verified with a single keystroke. Other characters require depression of the multipunch key. All 64-characters mentioned above can be punched when duplicating.

CHARACTER ARRANGEMENTS for System/360 Printers

<table>
<tr><th rowspan="2">ARRANGEMENT</th><th colspan="2" rowspan="2">NOMINAL SPEED (LPM)</th><th colspan="4">SOURCE CHARACTERS</th><th rowspan="2">APPLICABLE MACHINES</th></tr>
<tr><th>TOTAL</th><th>ALPHA</th><th>NUMERIC</th><th>SPECIAL CHARACTERS(6)</th></tr>
<tr><td>Standard Chains/Trains</td><td>1403-2,7 1404-2</td><td>1403-3,N1</td><td></td><td></td><td></td><td></td><td>CHAINS
1403 Models 2,7; All S/360 Models
1404 Model 2; S/360 Models 30, 40 and 50 Only</td></tr>
<tr><td>AN</td><td>600</td><td>1100</td><td>48</td><td>A-Z</td><td>0-9</td><td>/ @ # ⌑ . + * $ - % , &</td><td></td></tr>
<tr><td>HN</td><td>600</td><td>1100</td><td>48</td><td>A-Z</td><td>0-9</td><td>/ ' =) . + * $ - (, &</td><td></td></tr>
<tr><td>Universal Character Set (7) Chains/Trains</td><td>1403-2</td><td>1403-3,N1</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>PCS-AN</td><td>600</td><td>1100</td><td>48</td><td>A-Z</td><td>0-9</td><td>/ @ # ⌑ . + * $ - % , &</td><td></td></tr>
<tr><td>PCS-HN</td><td>600</td><td>1100</td><td>48</td><td>A-Z</td><td>0-9</td><td>/ ' =) . + * $ - (, &</td><td></td></tr>
<tr><td>PN and QN (8)</td><td>500(8)</td><td>950</td><td>60</td><td>A-Z</td><td>0-9</td><td>/ ' =) . + * $ - (, & " — : | > ? # % @ < ; ¬</td><td>TRAINS
1403 Model 3; All S/360 Models except 20 and 67</td></tr>
<tr><td>RN (8)</td><td>500(8)</td><td>950</td><td>52</td><td>A-Z</td><td>0-9</td><td>/ ' =) . + * $ - (, & ⌑ # % @</td><td></td></tr>
<tr><td>SN (8)</td><td>392</td><td>770</td><td>84</td><td>A-Z a-z</td><td>0-9</td><td>/ ') . + * $ - (, & " — : | ⌑ ? ! % @ ¢ ;</td><td></td></tr>
<tr><td>TN</td><td>273</td><td>570</td><td>120</td><td>A-Z a-z</td><td>0-9</td><td>/ ' =) . + * $ - (, & " — : | ⌑ ? ! % @ ¢ ; ¬
- • ┐ ┌ ┘└ { } [ ] ■ ≥ ≤ > < ≠ # °) (+ - 0 9
8 7 6 5 4 3 2 1 ± +</td><td>1403 Model N1; All S/360 Models</td></tr>
<tr><td>XN (Chain Only)</td><td>692</td><td>—</td><td>40</td><td>A-Z</td><td>0-9</td><td>. * $,</td><td></td></tr>
<tr><td>YN (Train Only) (8)</td><td>—</td><td>1250</td><td>42</td><td>A-Z</td><td>0-9</td><td># . * $ - ,</td><td></td></tr>
<tr><td>IBM 1443 Sets</td><td colspan="2">Model N1</td><td></td><td></td><td></td><td></td><td>1443 Model N1; All S/360 Models except 20 and 67</td></tr>
<tr><td>13-Char. Set</td><td colspan="2">600</td><td>13</td><td>—</td><td>0-9</td><td>. * -</td><td></td></tr>
<tr><td>39-Char. Set</td><td colspan="2">300</td><td>39</td><td>A-Z</td><td>0-9</td><td>. $,</td><td></td></tr>
<tr><td>52-Char. Set (Std.)</td><td colspan="2">240</td><td>52</td><td>A-Z</td><td>0-9</td><td>/ @ # ⌑ . & * $ - % , + ( ) ' =</td><td></td></tr>
<tr><td>63-Char. Set</td><td colspan="2">200</td><td>63</td><td>A-Z</td><td>0-9</td><td>/ @ # ¢ . & * $ - % , + ( ) ' = ! ; : > < _ | ¬ " ? ƀ</td><td></td></tr>
<tr><td>IBM 1445 Sets</td><td colspan="2">Model N1</td><td></td><td></td><td></td><td></td><td>1445 Model N1; S/360 Model 30 Only</td></tr>
<tr><td>14-Char. Set</td><td colspan="2">525</td><td>14</td><td>—</td><td>0-9</td><td>. * $ -</td><td></td></tr>
<tr><td>42-Char. Set</td><td colspan="2">240</td><td>42</td><td>A-Z</td><td>0-9</td><td>/ . * $ - ,</td><td></td></tr>
<tr><td>56-Char. Set (Std.)</td><td colspan="2">190</td><td>56</td><td>A-Z</td><td>0-9</td><td>/ . * $ - , 1 2 3 4 5 6 7 8 9 0 [illegible]</td><td></td></tr>
<tr><td>IBM 2203-A1 Sets</td><td colspan="2"></td><td></td><td></td><td></td><td></td><td>2203 Model A1; S/360 Model 20 Only</td></tr>
<tr><td>13-Char. Set</td><td colspan="2">750</td><td>13</td><td>—</td><td>0-9</td><td>. * -</td><td></td></tr>
<tr><td>39-Char. Set</td><td colspan="2">425</td><td>39</td><td>A-Z</td><td>0-9</td><td>. $,</td><td></td></tr>
<tr><td>52-Char. Set</td><td colspan="2">350</td><td>52</td><td>A-Z</td><td>0-9</td><td>/ @ # ⌑ . & * $ - % , + ( ) ' =</td><td></td></tr>
<tr><td>63-Char. Set</td><td colspan="2">300</td><td>63</td><td>A-Z</td><td>0-9</td><td>/ @ # ¢ . & * $ - % , + ( ) ' = ! ; : > < _ | ¬ " ? ƀ</td><td></td></tr>
<tr><td colspan="8">Note: In this table ƀ is not a character; rather it indicates a blank space</td></tr>
</table>

NOTES:

(6) The codes that cause these characters to print are the EBCDIC codes of System/360; each character has its own internal and card codes; that is, there are no dual characters. For example, in non-System/360 the dual characters % and (have the same card code (0-8-4) and the same internal code (A84); in System/360 % is represented by card code 0-8-4 and by the internal code HEX 6C, whereas (is represented by card code 12-8-5 and by the internal code HEX 4D.

(7) The Universal Character Set (UCS) feature for the 1403 Models 2, 3, and N1 provides printing of any set of graphics up to a maximum of 240. The 1403 must be attached to the IBM 2821 Control Unit (or a 2020 processing unit) to use this feature. The chain/train arrangements shown above are the standard UCS chains and trains. In addition, the user can design chains or trains tailored to his own needs.
The printing speeds attainable with the UCS feature are dependent on the chain/train design, format, spacing, etc. The speeds shown above for the standard arrangements are nominal speeds for the various character sets, with continuous printing and single-line spacing. The speeds are for full representation of the character sets. Speed for preferred arrangements (see Note 8) will be somewhat higher. Also, the UCS feature uses an "End-of-Print Line" concept that permits intermediate carriage movement when a printed line is complete. Thus, printing speed is variable and can be in excess of the nominal speeds given in the table. The SRL manual for the 2821 Control Unit (A24-3312-1) provides more information on the UCS feature.

(8) The QN Chain and Train (for use with PL/I) consists of 60 graphics with 45 preferred.
The RN Chain and Train (for use with COBOL) consists of 52 graphics with 47 preferred.
The SN Chain and Train (for text printing) consists of 84 graphics with 78 preferred.
The YN Train (high-speed alphameric printing) consists of 42 graphics with 39 preferred.

Printing speeds for preferred arrangements will approach those that would be possible if only the preferred set were on the chain or train. For example, the speed for the RN Train on the 1403 Models 3 and N1 will approach 1100 lpm, which is the speed of the 48-character PCS-AN Train.

CHARACTER ARRANGEMENTS for System/360 Typewriter-Printers (PTTC/EBCDIC)

ARRANGEMENT	SOURCE CHARACTERS TOTAL	ALPHA	NUMERIC	SPECIAL	APPLICABLE TYPEWRITER-PRINTERS
PTTC/EBCDIC (9) for 1052 Models 1-6 and 1053, 2740/2741					1052 Models 1-6 1053 2740/2741
Upshift	44	A-Z		= ; % > *) < : ' (" ? ¢ | - ! + ¬	
Downshift	44	a-z	0-9	# / @ , - $ & .	
PTTC/EBCDIC (10) for 1052 Model 7 only					1052 Model 7
Upshift	44	A-Z		(<) * ' ¢ + ¬ ; " = - ? : > ! % |	
Downshift	44	a-z	0-9	# / - , & $ @	

NOTES:
(9) For 1052 Models 1-6, 1053, 2740/2741 — Non-System/360 PTTC/BCD codes are compatible with System/360 PTTC/EBCDIC for BCD assignments except for characters > < " | ¬
(10) This element provides 20 graphic changes from the Standard Dual Case Element used in a non-System/360 typewriter. It is for the 1052 Printer-Keyboard, Model 7, only as a reading board console typewriter (input/output) in a System/360 Model 40, 50, or 65 --- or as a stand-alone console typewriter (input/output) in a System/360 Model 65 or 75. No 1051 is required for these attachments.

CHARACTER ARRANGEMENTS for Non-System/360 Typewriter-Printers (PTTC/BCD)

ARRANGEMENT	SOURCE CHARACTERS TOTAL	ALPHA	NUMERIC	SPECIAL	APPLICABLE TYPEWRITER-PRINTERS Dual Case 1052/1053 2740/2741	Mono Case 1052/1053 1033	Mono Case 1052/1053 2740/2741
Std. Dual Case Element							
Upshift	44	A-Z	-	= ; % " *) ␣ : ' (± ? ¢ , - ! + .	X		
Downshift	44	a-z	0-9	# / @ , - $ & .			
Std. Mono Case Element							
Upshift	44	A-Z	-	= ; % " *) ␣ : ' (± ? ¢ , - ! + .		X	X
Downshift	44	A-Z	0-9	# / @ , - $ & .			
Arrangement A							
Upshift	44	A-Z	-	> ; % " *] ␣ ↄ : [≠ ? Δ , \ ! < .	X		X
Downshift	44	(11)	0-9	# / @ , - $ & .			
Arrangement E							
Upshift	44	A-Z	-	= ; % " *) < : ' (± ? ¢ , - ! + .			X
Downshift	44	a-z	0-9	# & > , / ° - .			
Arrangement H							
Upshift	44	A-Z	-	> ; (" *]) ↄ : [≠ ? Δ , \ ! < .	X	X	X
Downshift	44	(11)	0-9	= / ' , - $ + .			
Typewriter Option							
Upshift	44	A-Z	-	± # % & *) @ $ ¢ (" ? ¼ , - ! + .	X		X
Downshift	44	(11)	0-9	' / ½ , - ; & .			
Slashed Zero (12)							
Upshift	44	A-Z	-	= ; % " *) ␣ : ' (± ? ¢ , - ! + .	X		X
Downshift	44	(11)	Ø-9	Ø # / @ , - & .			

Note: In this table ↄ is not a character; rather it indicates a blank space

NOTES:
(11) The downshift alpha characters depend upon the case of the element. For Mono case, downshift characters are capitals A-Z; for dual case, downshift characters are lowercase a-z.
(12) This arrangement is identical to the standard arrangements except that zero (0) is replaced by slashed zero(Ø).

IBM EQUIPMENT (Non-System/360) CHARACTER SETS

SPECIAL CHARACTER ARRANGEMENTS (48-character, including A-Z, 0-9 and the following special characters)

Card Code → / Arrangement ↓	12	12-8-3	12-8-4	11	11-8-3	11-8-4	0-8-2	0-8-3	0-8-4	0-1	8-3	3-4	
A	&	.	␣	-	$	*	blank	,	%	/	#	@	
B	/	.	␣	-	$	*	blank	,	%	&	#	@	
C	&	.	␣	-	$	*	blank	,	%	0	#	@	
D	-	.	␣	-	$	*	blank	,	%	/	#	@	
E	-	.	<	/	°	*	blank	,	%	&	#	>	
F	+	.	)	-	$	*	blank	,	(	/	=	-	
G	+	.	␣	-	$	*	blank	,	%	/	+	-	FORTRAN
H	+	.	)	-	$	*	blank	,	(	/	=	'	COBOL
J	+	.	␣	-	$	*	blank	,	%	/	#	@	
K	+	.	)	-	$	*	blank	,	(	/	=	@	

The blank space (0-8-2) uses the multipunch key and results in a * character on the 1403 printer.

Note: Numerical chain for 1403, Models 1 and 2 only, consists of the numerics 0-9 and the six special characters ␣ . * $ - ,
Note: The 1403, Models 2 and 3, and 1404, Model 2, may also be used with S/360, but with different arrangements (shown on reverse side)

APPLICABLE EQUIPMENT
24, 26, 29 (Expanded Keyboard Model)
56, 59 (Expanded Keyboard Model)
370, 380, 381 — Only arrangements A, B, and D.
407, 408, 409, 716, 7400
557
824, 826 — Only arrangements A, B, and D.
838, 7900 — Only arrangements A, B, and D.
1058 — Has its own PTTC/BCD code print plate. A-K arrangements are available as substitutes for part of 1058 arrangement; however, only 55 characters (total) may then be used. Also, card codes 8-1, 0-8-1, 12-11-2, 12-11-3, 12-11-4, 12-11-5, 12-11-6, 12-11-7, and 12-8-1 must not be used to avoid plate damage due to overdrive.
1403 Models 1, 2, 4, 5&6
1404 Model 2
1403 Model 3 (1416) — Only arrangements A & H

See X20-1719 for reference card with this information.

READER'S COMMENT FORM

Data File Handbook C20-1638-1

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. All comments will be handled on a non-confidential basis.

	Yes	No
• Does this publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

- What is your occupation? _____
- How do you use this publication?

As an introduction to the subject?	☐	As an instructor in a class?	☐
For advanced knowledge of the subject?	☐	As a student in a class?	☐
For information about operating procedures?	☐	As a reference manual?	☐

Other _____

- Please give specific page and line references with your comments when appropriate.

COMMENTS:

- Thank you for your cooperation. Space is available on the other side of this page for additional comments.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Technical Publications

fold fold

Printed in U.S.A. C20-1638-1

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601

Printed in U.S.A. C20-1638-1

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601