File Number S360-30
Form C24-3427-3

DOS

IBM Systems Reference Library

IBM System/360
Disk Operating System
Data Management Concepts

This reference publication contains a discussion of the data management facilities inherent in the Disk Operating System. It describes the file formats, labeling procedures, and access methods available with the system. There is also a general section describing the design of the direct-access storage devices supported.

The following publications are recommended as prerequisite readings:

1. IBM System/360 Principles of Operation, Form A22-6821.

2. IBM System/360 Disk Operating System, System Control and System Service Programs, Form C24-5036.

Other related publications are:

1. IBM System/360 Disk Operating System Supervisor and Input/Output Macros, Form C24-5037.

2. IBM System/360 Disk and Tape Operating Systems Assembler Specifications, Form C24-3414.

3. Glossary for Information Processing, Form C20-8089.

For a list of associated System/360 publications, refer to IBM System/360 Bibliography, Form A22-6822.

Major Revision (February 1968)

This edition, Form C24-3427-3, is a major revision of, and obsoletes Form C24-3427-1 and Form C24-3427-2. It also obsoletes the following Technical Newsletters: N24-5122, N24-5169, N24-5197, N24-5276, and N24-5291. Changes are designated in three ways:

1. A vertical line appears to the left of affected text where only part of the page has been changed.

2. A dot (•) appears to the left of the page number where the complete page should be reviewed.

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

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or Technical Newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for readers' comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Endicott, New York 13760.

CONTENTS

The data management facilities of the Disk Operating System are provided by a group of routines that are collectively referred to as the input/output control system (IOCS). A distinction is made between two types of routines:

1. Physical IOCS - The physical I/O routines included in the supervisor.

2. Logical IOCS - The logical I/O routines linked with the user's problem program.

Physical IOCS is used by all programs run within the system. Its function is to supervise the execution of channel programs supplied by the problem program without regard to the logical content, format, or organization of the data being read or written. It includes facilities for:

1. Scheduling and queuing I/O operations.

2. Checking for and handling error conditions and other exceptional conditions related to input/output devices.

3. Handling I/O interruptions to maintain maximum I/O speeds without burdening the user's problem program.

The logical IOCS routines are linked with and executed as a part of the user's problem program. They provide an interface between the user's file-processing routines and the physical IOCS routines. Unlike physical IOCS, the logical IOCS routines handle logical data files. Following are the two options the user has of including these routines in his program.

1. Assembling the routines with his source programs, using the appropriate LIOCS macro instructions.

2. Keeping a selected group of LIOCS object modules (modules and DTF's) in the Relocatable Library from where they will be called by the Linkage Editor when unresolved EXTRN statements are encountered when the user's program is edited.

Depending on the defined attributes of the file, the logical IOCS routines perform the following functions where applicable.

1. Requesting (of physical IOCS) execution of appropriate channel programs.

2. Handling end-of-file and end-of-volume conditions.

3. Blocking and deblocking of records.

4. Switching between I/O areas when two areas are specified for a file (except when combined files are specified).

PHYSICAL IOCS

All I/O operations are performed by the physical IOCS routines included within the disk operating system supervisor. The main element of physical IOCS is the channel scheduler. This routine is entered through a supervisor call (SVC) instruction. The SVC may be issued by the user's program (assembled from an EXCP macro instruction), by a logical IOCS routine for the user's program, or by any of the other routines of the control program. In all cases, the routine issuing the SVC supplies a command control block (CCB) which indicates the I/O device to be operated (see Symbolic Device Addressing) and the location of a channel program (one or more channel command words - CCW's) to be executed. The CCB is also used to indicate the error options selected by the user and, at the completion of the operation, to indicate any exceptional ending conditions.

Physical IOCS determines the channel for which the request was made and places the CCB on a queue for that channel. If the channel and device are not busy, the I/O operation is started and control is returned to the next instruction in the problem program. If the channel or device is busy, the I/O request is placed in the channel queue and control is returned to the next instruction in the problem program. When the channel and device become free, as the result of I/O interruption processing, the operation is started. Channel queuing is discussed further under Symbolic Device Addressing.

The problem program will be interrupted when the I/O operation is complete. This is indicated either by the completion of the channel operation (channel end) or the completion of the device operation (device end). These occur simultaneously for many I/O operations. Physical IOCS checks for exceptional ending conditions and posts, in the CCB, an indication of any detected. If I/O errors are detected, physical IOCS will

retry if the error is of a type that can possibly be corrected. If errors cannot be corrected in this manner, the requesting routine is notified by a flag in the CCB, or the operator is notified by a message.

SYMBOLIC DEVICE ADDRESSING

The physical IOCS routines are designed to allow a program to refer to I/O devices by symbolic names. This allows data files to be processed from one physical device during one run and from another device during the next without modifying the program. The symbolic name used in the CCB refers to a logical-unit block (LUB). The LUB in turn points to a physical-unit block (PUB) that contains the physical address of the device and other information needed by physical IOCS.

The physical unit to which a symbolic name points can be changed between job steps by the machine operator or the problem programmer via job-control statements. The LUB table is arranged in a fixed logical sequence, allowing the language translators to equate the symbolic device names to displacement factors pointing to the proper LUB. The PUB table, on the other hand, is arranged in an order determined by the channel to which each device is attached. This sequencing allows the channel scheduler to determine quickly whether an I/O operation can be started when requested, or whether the request must be queued. A separate queue is maintained for each physical unit.

On the multiplexor channel, the queuing operation can take account of whether a device operates in burst or multiplex mode and, if in multiplex mode, whether it is likely to lose information in an overrun condition. This is done by a specification made at the time of system generation. This specification allows burst-mode devices to be present on the channel. For example, the 1285, 1287, 1442-N1, 2501, and 2520-B1 are overrunable devices. If there are no burst-mode devices specified on the multiplexor channel, each device can be considered to have a separate channel. Regardless of the number of operations already started on the channel, each new request is started as soon as the specific device is available. If burst-mode devices are specified present, I/O requests for burst-mode and overrunable devices are executed one at a time. Only those devices that operate in the multiplex mode with no danger of overrun are operated concurrently (without concern for other channel operations).

PHYSICAL IOCS MACRO INSTRUCTIONS

A user's problem program normally uses logical IOCS for file processing. Logical IOCS, in turn, uses physical IOCS to perform actual data transfers. There are occasions, however, when a user may need to bypass the logical IOCS routines to perform a particular I/O operation. Three macro instructions are provided to allow the user to communicate directly with physical IOCS.

CCB — This macro instruction can be used to create a command control block.

EXCP — This macro instruction is converted to the proper SVC instruction to request execution of a channel program. It supplies the location of the corresponding CCB.

WAIT — This macro instruction is used to test a bit in the CCB to determine when an I/O operation has been completed. If the operation is not completed, the program yields control to the Supervisor until physical IOCS sets the bit to indicate completion of the operation.

The EXCP macro instruction gives the programmer more freedom in controlling devices than the logical IOCS macros yet EXCP retains many of the operational advantages of an operating system. The system provides for scheduling and queuing of I/O requests, efficient use of channels and devices, data protection, interruption procedures, and error recognition and retry. To use physical IOCS, however, the programmer needs detailed knowledge of device control and system functions. He must supply his own channel programs, using the CCW (define channel command word) assembler instruction statement.

When the EXCP macro instruction is used, physical IOCS knows nothing of the logical content or structure of the files being accessed. It is possible, however, for the user to make use of the protection features of the system available through label processing.

LABEL PROCESSING

All DASD and tape label processing is performed by routines included as a permanent part of the disk-resident system. Label processing is performed in response to OPEN and CLOSE macro instructions issued in the problem program. These macros are assembled into linkages to logical IOCS routines

that fetch the actual label-processing routines into the transient area, from where they are executed. A special logical IOCS routine is available to allow labels to be processed on files that the user handles directly with physical I/O macros. The user includes a DTFPH (Define The File for PHysical IOCS) macro instruction in his program. The label processing performed by the routine is explained in the last part of this publication.

LOGICAL IOCS

Logical IOCS consists of a number of logical-file accessing routines that provide an interface between the user's processing function and physical IOCS. These routines, called IOCS logic modules, are executed in response to imperative macro instructions in the problem program. A number of such logic modules, each designed to perform I/O operations on a particular type of data file, are provided. Each logic module is generalized, using file-description information included in the user's program to process any number of specific files.

If a selected group of LIOCS object modules has previously been assembled and cataloged to the Relocatable Library, the proper logic modules are retrieved from the relocatable library and linked with the user's program by the linkage editor. This eliminates the need for lengthy macro generation each time a program is assembled. In certain cases, the logic modules can be assembled along with the user's problem program and included in the same output object module.

Logical IOCS operates on logical data files that are defined as such in the user's program. It differs from physical IOCS in that physical IOCS knows only the location of data being accessed and other information about the physical I/O device being used. Logical IOCS, on the other hand, uses some or all of the following information, depending on the specific logic modules used.

1. File name
2. I/O device type
3. Organization structure
4. Access method
5. Record format and size
6. I/O areas (number and location)
7. Location and size of record identification (control) fields
8. Labeling procedures
9. Error options
10. Other optional information

The programmer includes some or all of this information as parameters in DTF (define the file) macro instructions. His first choice is among the following DTF macro instructions*:

DTFCN	Define The File for CoNsole
DTFCD	Define The File for CarD.
DTFPR	Define The File for PRinter.
DTFMT	Define The File for Magnetic Tape.
DTFPT	Define The File for Paper Tape
DTFOR	Define The File for Optical Reader
DTFMR	Define The File for Magnetic Reader
DTFDI	Define The File for Device Independence
DTFSD	Define The File for Sequential DASD.
DTFDA	Define The File for Direct Access file.
DTFIS	Define The File for Indexed Sequential file.

*Note: An additional macro instruction, DTFSR, is available to provide the ability to assemble programs written for the disk-resident version of the Basic Operating System. In the Disk Operating System, this macro instruction is used to call the specific macro instruction corresponding to any of the first six DTF's listed above. The tables and logic modules generated are the same ones used for files defined with the individual DTF's listed above. DTFSR requires more time during program assembly and should be used only to avoid recoding programs written for the Basic Operating System or Basic Programming Support. This macro is described along with the others in the Supervisor and Input/Output Macros publication listed on the cover of this publication.

The basis of choice for the first seven of

the preceding is obvious. DTFCN is used for the 1052 Printer Keyboard. DTFCD is used for all card files, including input, output, and combined files (where a card is read and then punched). DTFPR is used for all printer output files, DTFMT for all magnetic tape files, DTFPT for all paper tape files, DTFOR for all optical reader files, and DTFMR for all magnetic reader files.

Three of the preceding macros: DTFSD, DTFDA, and DTFIS, are all used for files on direct-access storage devices. The choice of which of these should be used for a specific file is determined by the organization structure of the file and the processing sequence to be used. The combination of these two factors defines the access method that should be used on the file.

There are three access methods available within the Disk Operating System:

1. Sequential-Access Method

2. Direct-Access Method

3. Indexed-Sequential-Access Method

The sequential-access method is used for files defined by the following DTF's listed: DTFCD,DTFCN,DTFPR,DTFMT, DTFOR, DTFMR, DTFDI, DTFPT and DTFSD. The direct-access method is used for files defined by DTFDA. The indexed-sequential access method is used for files defined by DTFIS. Logical IOCS includes a number of routines to load indexed-sequential files, add records, and provide sequential and/or random retrieval. These routines are much more than an access method, and are referred to as the indexed-sequential-file management system (ISFMS).

The DTF macro instructions create constants called DTF tables. These tables contain all of the information necessary for the generalized logic modules to process the specific file defined. They include the CCB's (command control blocks, discussed under Physical IOCS) and either include or reference the channel programs to be used. Also produced during assembly of these tables are references to the proper logic modules. The linkage editor uses these references to resolve the linkages between the tables and the modules.

The macro definitions, when received from IBM, include complete capability for generating all options that might be selected. The user can assemble a number of individual modules, each with only those capabilities needed for specific applications. Note, however, that the assembled routines are still generalized and can be used for a large number of different files.

DEVICE INDEPENDENT SEQUENTIAL FILE PROCESSING FOR SYSTEM UNITS

Define The File For Device Independence (DTFDI) system units and the Device Independent MODule (DIMOD) are IOCS file definition macros that provide device-independent sequential file processing for system units SYSRDR, SYSIPT, SYSPCH, and SYSLST.

The physical device can be assigned at execution time, allowing data files to be processed by the physical device the user prefers at that time. For example, when the device-independent macro is used in the program, at execution time the system logical unit SYSRDR could be assigned to a card reader, magnetic tape, or a disk extent. This can be particularly advantageous to the user when one physical device is temporarily inoperative, enabling him to process on another device. The user's program need not be modified. The user's program, however, must be reassembled with the new DTFDI/DIMOD macros to obtain device independent capabilities for system units.

These macros are described in the Supervisor and Input/Output Macros publication listed on the front cover of this publication.

DATA FILES

Many types of data files are used in data processing applications. Theoretically, there is no restriction on the logical content of information that can be processed, on the relationship of various units of information in the file, on the organization, or format. Physical IOCS knows nothing of these aspects of files processed with the EXCP level of macro instruction. As long as the user's program includes the necessary channel programs and processing capability, the only restrictions imposed are those made by the physical device itself. Logical IOCS, on the other hand, is concerned with more than the physical device and the physical unit of recorded information. Files defined to logical IOCS are called logical files. They are named, organized collections of logically-related data. Depending on the facilities to be provided by logical IOCS, files may be restricted only to a maximum size or they may be required to be completely fixed with regard to size, format, logical sequence, and, to a limited degree, logical content.

The following sections discuss logical record formats that can be processed with various parts of logical IOCS. Then follow three major sections that describe the three access methods available and the specific file configurations handled by each.

Logical Records

A data file is made up of a collection of logical records that have some relation to one another. The logical record is the basic unit of information for a data processing program. A logical record might be, for example, one employee's record in a master payroll file, or one part-number record in an inventory file. Much data processing consists of reading, processing, and writing individual logical records.

Record Blocking

Blocking of records is the process of grouping a number of logical records before writing them on an external storage device. Such a grouping of logical records is called a block. Blocking improves effective data rate and conserves storage space on the device by reducing the number of interrecord gaps in the file. Blocking usually increases processing efficiency by reducing the number of input/output operations required to process a file.

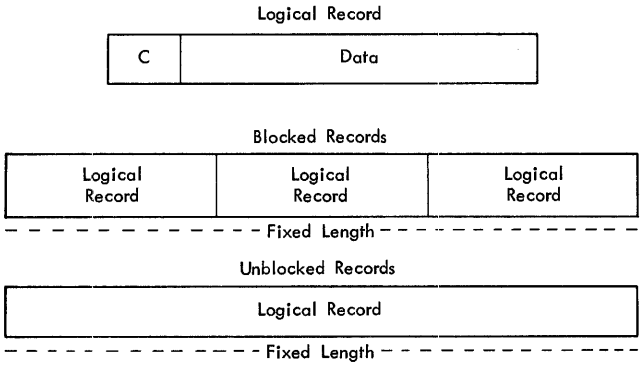


Figure 1. Format F Records

Record Formats

Logical records may be in one of three formats: fixed-length (format F), variable-length (format V), or undefined (format U). The record format and whether or not the file is blocked are specified in the DTF macro instruction used to define the file.

The prime consideration in the selection of a record format is the nature of the file itself. The programmer knows the type of input his program will receive and the type of output it will produce. His selection of a record format is based on this knowledge, as well as an understanding of the type of input/output devices on which the file is written and of the access method used to read or write the file.

FIXED-LENGTH (FORMAT F): Format F records are fixed-length records. The number of logical records within a block (blocking factor) is normally constant for every block in the file unless the block is truncated (short block).

In unblocked format F, the logical record constitutes the block.

The system performs physical length checking on blocked format F records, automatically making allowance for truncated blocks. Because the channel and interruption system can be used to accommodate length checking, and the blocking/deblocking is based on a constant record length, format F records can be processed by logical IOCS faster than format V.

Format F records are shown in Figure 1. The optional control character , represented by C in Figure 1, is used for stacker selection and carriage control. It may be included in each logical record.

VARIABLE-LENGTH (FORMAT V): Format V provides for variable-length records, each of which describes its own length, and for variable-length records in variable-length blocks, each of which includes a block length. The system performs length checking of the block and makes use of the record length information in deblocking and blocking. Format V records are shown in Figure 2. The first four characters of the logical record contain control information; "ll" represents the length of the logical record and 'bb' represents two characters reserved for system use. These characters must be provided by the user when he is creating the record. An optional control character , represented by C in Figure 2, may be specified as the fifth character of each logical record.

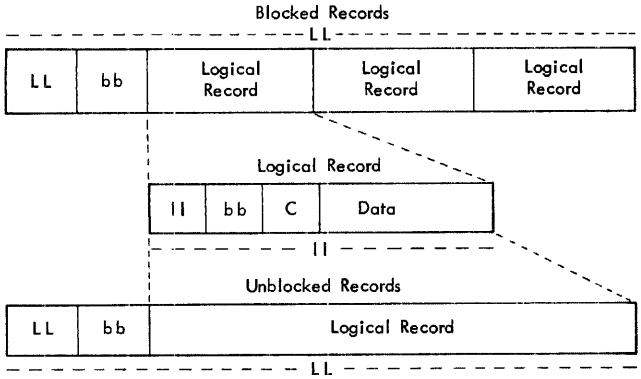


Figure 2. Format V Records

In blocked format V, 'LL' represents the block length and 'bb' represents the two characters reserved for system use. These characters are automatically provided when the file is written. Although these four characters do not appear in the logical record furnished the user, input and output areas must be large enough to accommodate them.

In unblocked format V, the logical record and the block control information constitute the block.

The initial four characters (five if the optional control character is specified) of the block are not printed or punched.

UNDEFINED (FORMAT U): Format U is provided to permit the processing of any blocks that do not conform to the F or V formats. Format U records are shown in Figure 3. The optional control character may be used in each logical record.

Since each block is treated as a logical record (unblocked), any deblocking must be performed by the problem program.

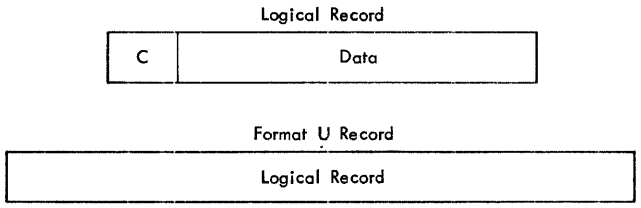


Figure 3. Format U Records

Control Character

The programmer may optionally specify, in the DTF macro-instruction, that a control character precedes each logical record in a file. This character specifies carriage control when the file is printed or stacker selection when the file is punched. The character itself is never printed or punched but is a part of the record in storage. I/O areas must be large enough to accommodate this character. If the immediate destination of the record is a device that does not recognize this control character, e.g., disk, the system assumes that the control character is the first character of the data.

If the destination of a record is a printer or a punch and the user has not specified that the first character of the record is to be used as a control character, this character is simply treated as the first character of the data.

SEQUENTIAL-ACCESS METHOD

The sequential-access method allows the programmer to store and retrieve the records of a sequential file. The method can be used for card, printer, printer-keyboard, magnetic tape, optical reader, magnetic reader, paper tape, and DASD files. The logical IOCS routines used are linked with the user's program as the result of his use of one of the following file-description macro instructions:

DTFCN - Console (Printer Keyboard)

DTFCD - Card

DTFPR - Printer

DTFMT - Magnetic Tape

DTFPT - Paper Tape

DTFOR - Optical Reader

DTFMR - Magnetic Reader

DTFDI - Device Independence

DTFSD - Sequential DASD

DTFSR - Serial

The sequential-access method provides two levels of I/O macro instruction language. The most common is the GET-PUT level, for normal input/output files. The READ-WRITE level is a more restricted language, used in applications where records are to be alternately read and written from and to DASD or tape, used as a temporary extension of main storage. The READ-WRITE macro instructions are used for files defined as work files.

Data Format-Device Type Relationships

The following text discusses data format considerations that apply to specific input/output device types supported by the sequential-access method. Included are descriptions of acceptable data formats for printer keyboard, card readers, punches, printers, magnetic tape devices, paper tape devices, optical readers, magnetic readers, and direct-access devices.

CONSOLE: Records can be read from or written on the IBM 1052 Printer-Keyboard in either format F or U. No control characters are recognized. All records read or written must be 256 characters or less. When keying information in from the printer keyboard, the end of the record is indicated by the end-of-block (Ⓑ) character.

CARD READERS AND PUNCHES: All card input must consist of fixed-length unblocked (80 characters or less) records. Card output can be any unblocked record format (F, V, or U). When format V records are punched, the four record length characters in the output area are not punched. The control character, if specified, is used for stacker selection purposes only; it is not punched.

Note: Logical IOCS accommodates only 8-bit character codes. Column binary cards cannot be processed.

OPTICAL READERS: Records can be read from both the IBM 1285 and 1287 Optical Reader in either format F or U. Format F is used when reading journal tapes containing an equal number of characters per line. When the line length is variable, format U is used. When processing documents, each field to be read by the 1287 Optical Reader can be treated as either format F or U. This is possible because the user provides CCW's to read the various fields of a document, and can set the SLI flag bit to ON or OFF at his discretion. If the user wishes to avoid certain IOCS register generations (RECSIZE and IOREG), records can be defined as format F.

MAGNETIC CHARACTER READERS: Undefined records can be read from both the IBM 1412 and 1419 Magnetic Character Readers. Record definition is determined by the settings of the field selection switches on the reader.

PRINTERS: The printer accepts data of any standard unblocked record format (F, V, or U). When format V records are printed, the record length characters in the output area are not printed. The control character, if specified, is used for carriage control purposes only.

Records to be printed must not exceed the length of a print line.

MAGNETIC TAPE: All standard blocked or unblocked record formats (F, V, and U) are acceptable to magnetic tape, when using the GET-PUT macro level. The READ-WRITE level allows only unblocked formats F and U. All control bytes are transmitted. 7-Track tapes not using the data-conversion feature do not handle format V.

Multivolume Tape Processing: When more than one reel of tape is used during the processing of a single file (multivolume file), IOCS will automatically switch to the alternate drive if one or more is specified in the ASSGN cards. The switching will be in the same order as the sequence of the ASSGN cards; not by physical drive number.

If more switching is needed and no more assignments are available, IOCS will switch back to the original drive and begin going through the original sequence again.

If no alternate drive is assigned and the end of volume is reached, a message will be written and the user may mount a new reel on the existing drive and continue processing.

PAPER TAPE READER: Undefined records terminated by an end-of-record character, or fixed-length unblocked records are acceptable as input from paper tape when using the GET macro instruction.

DIRECT-ACCESS DEVICES: All standard blocked or unblocked record formats (F, V, and U) are acceptable to direct-access devices when using GET-PUT macro instructions. READ-WRITE allows only unblocked formats F and U. All control bytes are transmitted.

All direct-access devices have the same track format. Each track consists of control information, a track descriptor record (R0), and the records (R_1 - R_n). Record reading and writing always begins with R_1 (R0 is ignored).

Note: R_1 has a special meaning to IBM System/360 Operating System programs, where it may be either the first complete data record on a track, or the overflow portion of a record from a preceding track. The Disk Operating System does not support record overflow for direct access devices.

When writing output files, IOCS generates the successive disk addresses (count fields) and writes the user's record as the data field of the record. Files with key fields cannot be read or written using the sequential-access method.

When OPENing a DASD file for consecutive processing, the user may define that the file use only a specified portion of each cylinder. This portion must be within the head limits for the cylinder and within the range of the defined extent limits. For example, three cylinders might be allocated for two files; one file to occupy the first two tracks of each cylinder and the other file to occupy the remaining tracks. This allocation of split cylinders reduces access time in some specialized applications. Split cylinder specification is accomplished through the use of the Type 8 EXTENT card (explained in the System Control and System Service Programs publication referenced on the front cover).

GET-PUT LEVEL SEQUENTIAL ACCESS

GET-PUT macro instructions permit the programmer to store and retrieve records of a sequential file without coding blocking/deblocking routines. The programmer can, therefore, concentrate all his efforts on processing the data he reads and writes. Another major feature of these macro instructions is the ability to use one or two I/O areas and to process records either in a work area or in the I/O area. These factors are discussed in the following section as they relate to achieving maximum overlap of processing with I/O.

Storage Areas and Effective I/O Overlap

These routines are designed to provide for overlapping the physical transfer of data with processing. The amount of overlapping actually achieved (effective overlap) is governed by the problem program through the assignment of I/O areas and work areas. An I/O area is that area of main storage to or from which a block of data is physically transferred by the channel scheduler and physical IOCS routines. A work area is an area used for processing an individual logical record from the block of data.

There are certain combinations of I/O areas and work areas that are possible. These are:

1. One I/O area with no work area.
2. One I/O area with a work area.
3. Two I/O areas with no work area.
4. Two I/O areas with a work area.

In some cases, a larger blocking factor may improve processing speed more than the use of either two I/O areas or a work area.

Also, certain devices are buffered, increasing the potential amount of overlap. Illustrations of these combinations for buffered devices, unbuffered devices, blocked tape, and blocked DASD records follow. These illustrations reflect the principle of overlap processing and are not intended to indicate the exact amount of overlap possible with any specific I/O device.

The maximum achievable overlap in Figure 4 is the device time only. The transfer time between I/O area and buffer is not overlapped. If the next GET (or PUT) is issued prior to device-end, the data transfer between I/O area and buffer does not take place until device-end is reached.

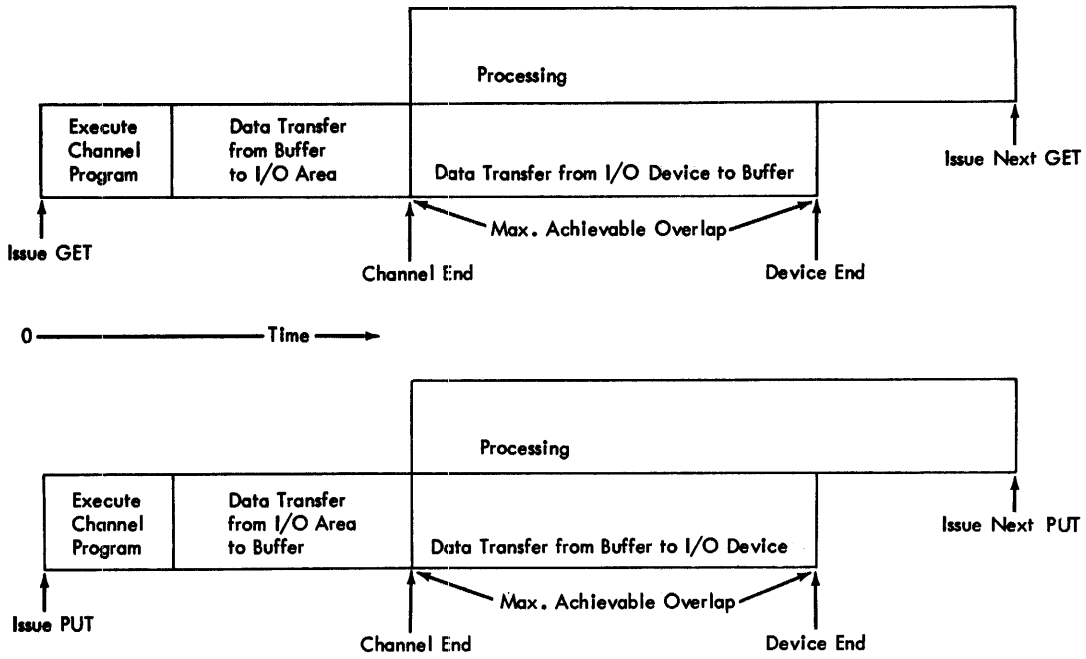


Figure 4. Overlap of Processing and I/O: One I/O Area and No Work Area (Buffered I/O Device)

The maximum achievable overlap in Figure 5 is the total data transfer time (the device time plus the time for data transfer between I/O area and buffer). If the next GET (or PUT) is issued after channel-end but before device-end, the transfer of data between the work area and the I/O area can take place (even though physical IOCS cannot start the data transfer between the device and the buffer until device-end is reached). Control transfers to the problem program.

If the next GET (or PUT) is issued before channel-end, logical IOCS must wait until channel-end to transfer data between the work area and the I/O area.

The maximum achievable overlap in Figure 6 is the total data transfer time (the device time plus the time for data transfer between I/O area and buffer). If the next GET (or PUT) is issued after channel-end and before device-end, only I/O area switching occurs. Control transfers to the problem program but physical IOCS does not start the device/buffer transfer until device-end is reached.

If the next GET (or PUT) is issued before channel-end, logical IOCS must wait for channel-end before performing any action.

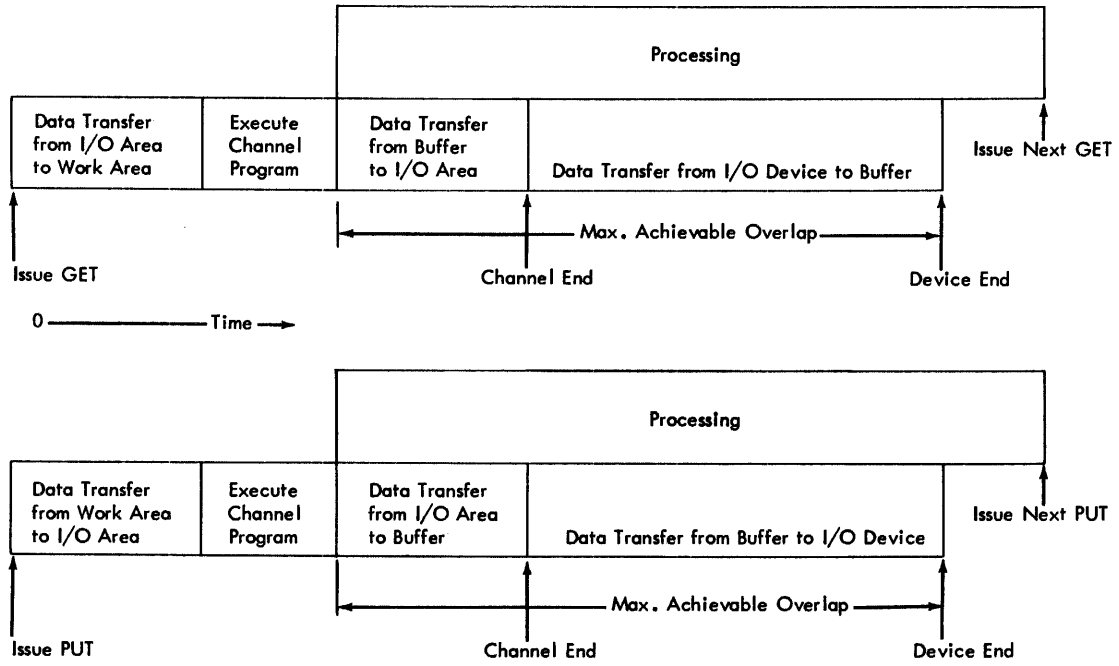


Figure 5. Overlap of Processing and I/O: One I/O Area and One Work Area (Buffered I/O Device)

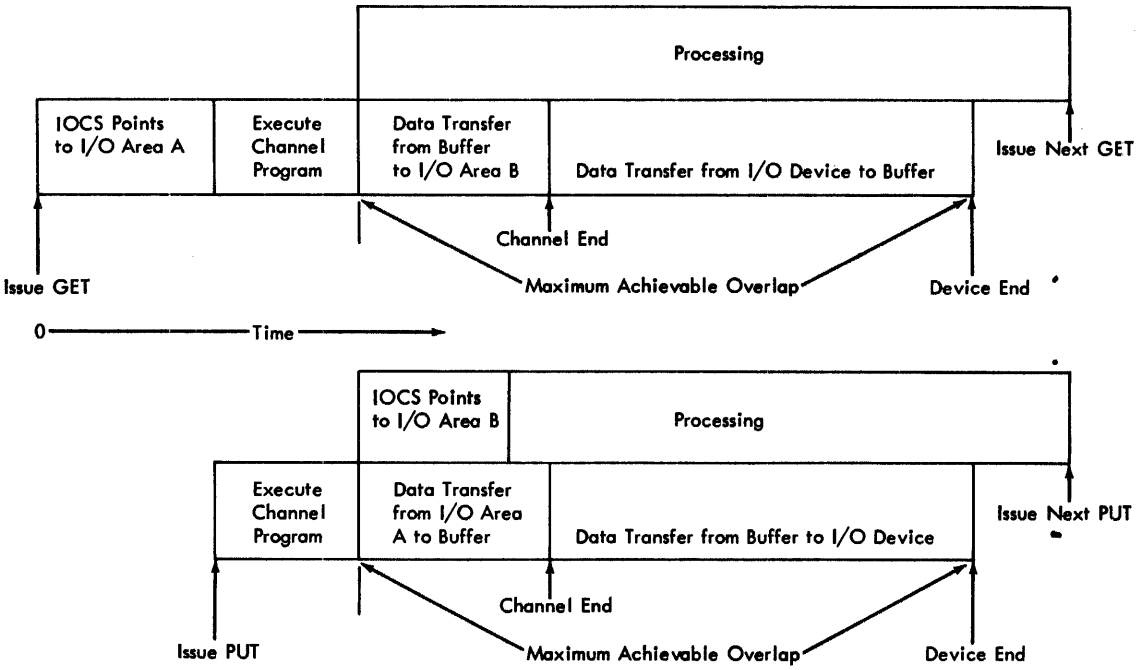


Figure 6. Overlap of Processing and I/O: Two I/O Areas and No Work Area (Buffered I/O Device)

The maximum achievable overlap in Figure 7 (as in Figures 5 and 6) is the total data transfer time (the device time plus the time for data transfer between I/O area and buffer). However, there is a disadvantage to the combination illustrated (in comparison with Figures 5 and 6), because it requires extra main storage.

If the next GET (or PUT) is issued after channel-end but before device-end, the data transfer between the I/O area and the work area can take place. Control returns to the problem program (even though physical IOCS cannot start the device until device-end is reached).

If the next GET (or PUT) is issued before channel-end, logical IOCS must wait.

There is no overlap possible in the illustration in Figure 8.

The maximum achievable overlap in Figure 9 is the data transfer time between device and I/O area. If the next GET (or PUT) is issued prior to channel-end, logical IOCS must wait before performing any action.

The maximum achievable overlap in Figure 11 (as in Figures 9 and 10) is the data transfer time between device and I/O area. However, there is a disadvantage to the combination illustrated (in comparison with Figures 9 and 10), because it requires extra main storage. If the next GET (or PUT) is issued before channel-end, logical IOCS must wait before any action is performed.

The combination illustrated in Figure 12 has no overlap of processing with input/output. The input/output time per record depends on the blocking factor. Therefore, with this combination, the I/O time can be reduced if the blocking factor is increased.

The maximum overlap achievable in Figure 13 is the time for data transfer between device and I/O area. The GET (or PUT) for all records, except the last in a block, involves only a transfer between work area and I/O area. For the last record in a block, the data transfer is followed by an overlap of device time and processing (control returns to the problem program). Channel-end must occur before logical IOCS can process the first record of the next block.

The maximum overlap achievable in Figure 14 is the time for data transfer between device and I/O area. The GET for all but the first record of a block takes time only for pointing to the next record. The GET for the first record must wait for channel-end of the data transfer to the alternate area. Then, pointing to the first record and returning control to the program is overlapped with the next device transfer. The PUT is the same as GET, except that the wait occurs with the last record of a block.

There is a disadvantage to the combination shown in Figure 15 over those in Figures 13 and 14 because it requires extra main storage.

A summary of the overlap of processing and I/O is shown in Figure 16.

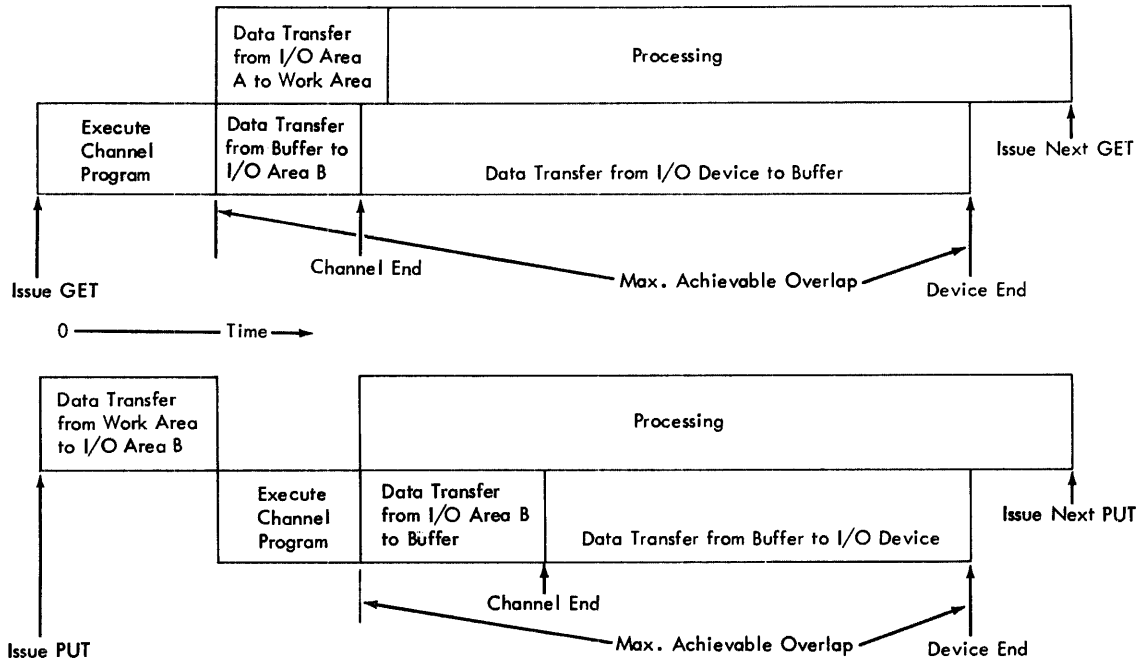


Figure 7. Overlap of Processing and I/O: Two I/O Areas and a Work Area (Buffered I/O Device)

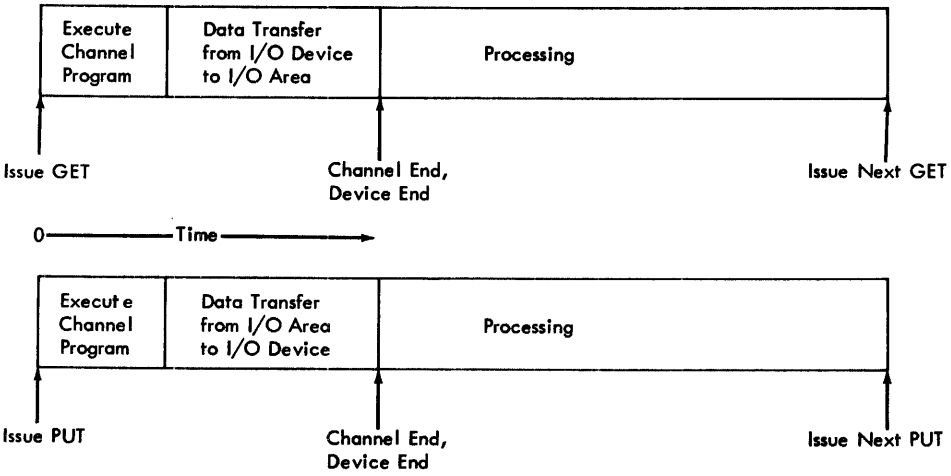


Figure 8. Overlap of Processing and I/O: One I/O Area and No Work Area (Unbuffered I/O Device)

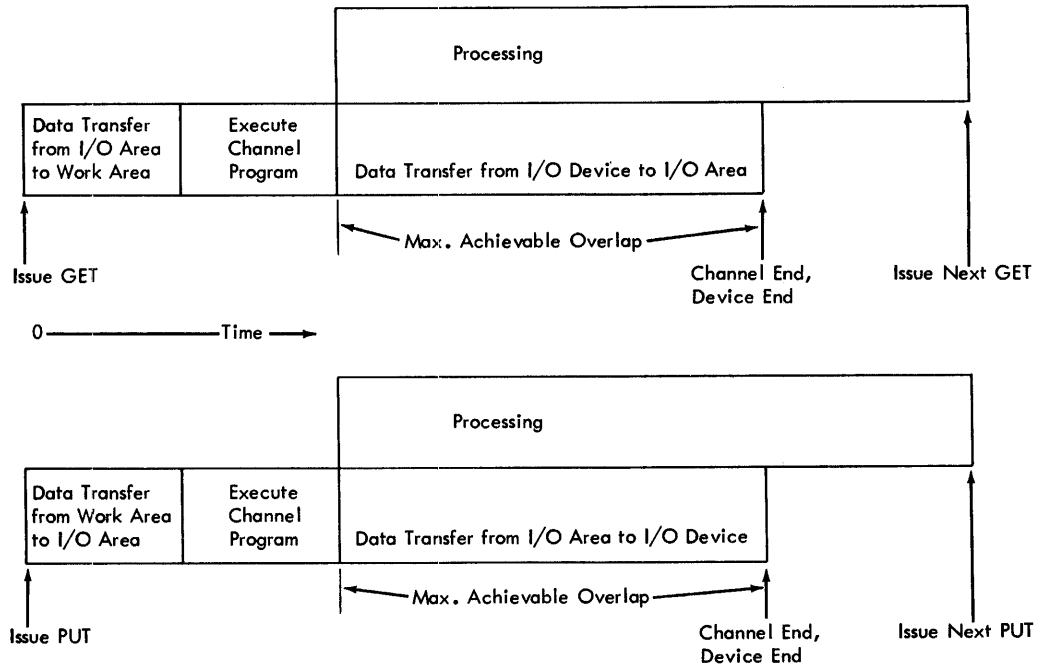


Figure 9. Overlap of Processing and I/O: One I/O Area and a Work Area (Unbuffered I/O Device)

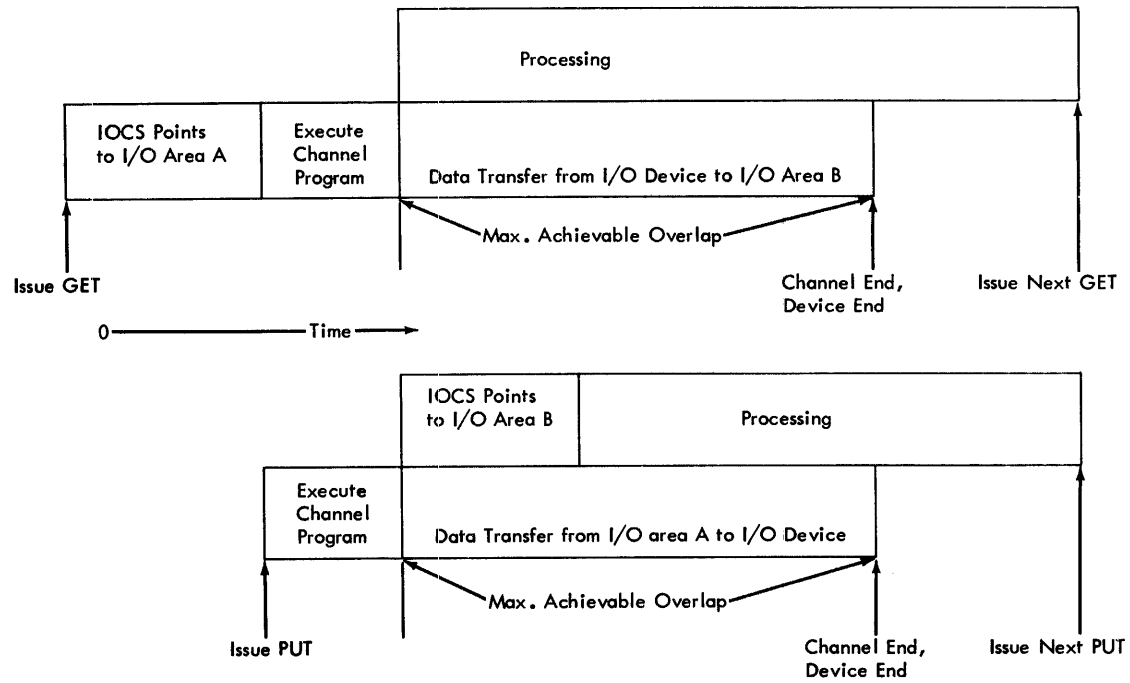


Figure 10. Overlap of Processing and I/O: Two I/O Areas and No Work Area (Unbuffered I/O Device)

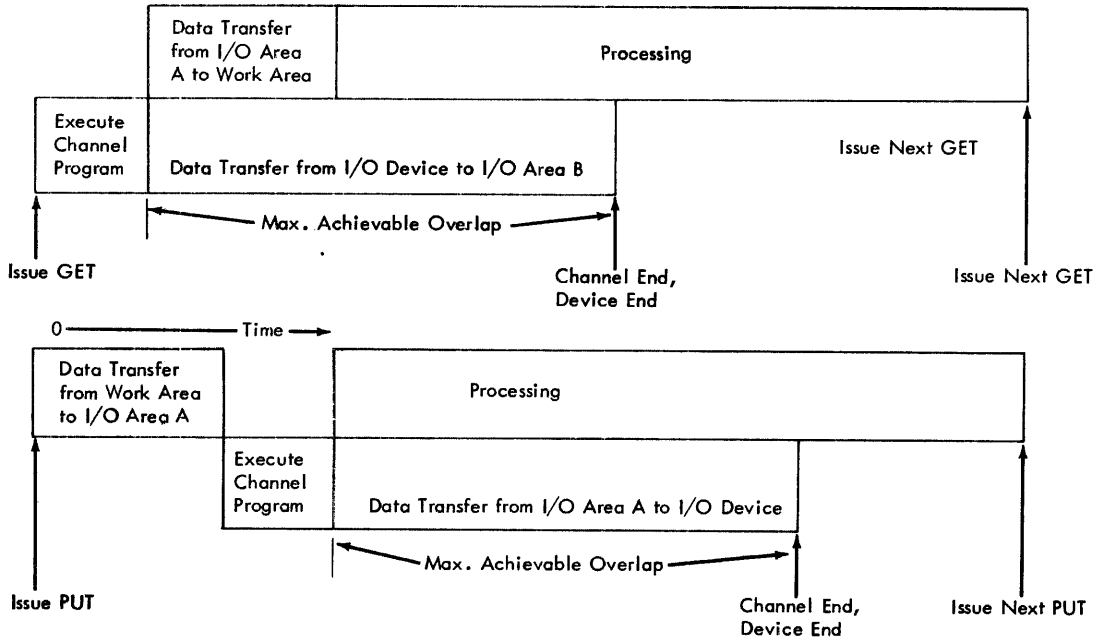


Figure 11. Overlap of Processing and I/O: Two I/O Areas and a Work Area (Unbuffered I/O Device)

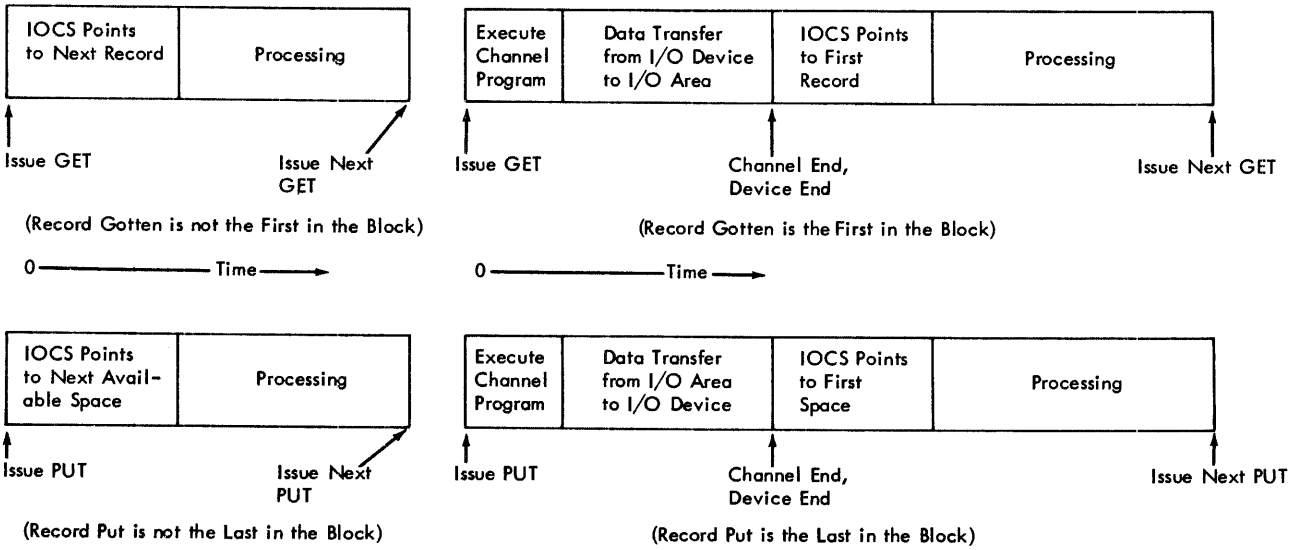
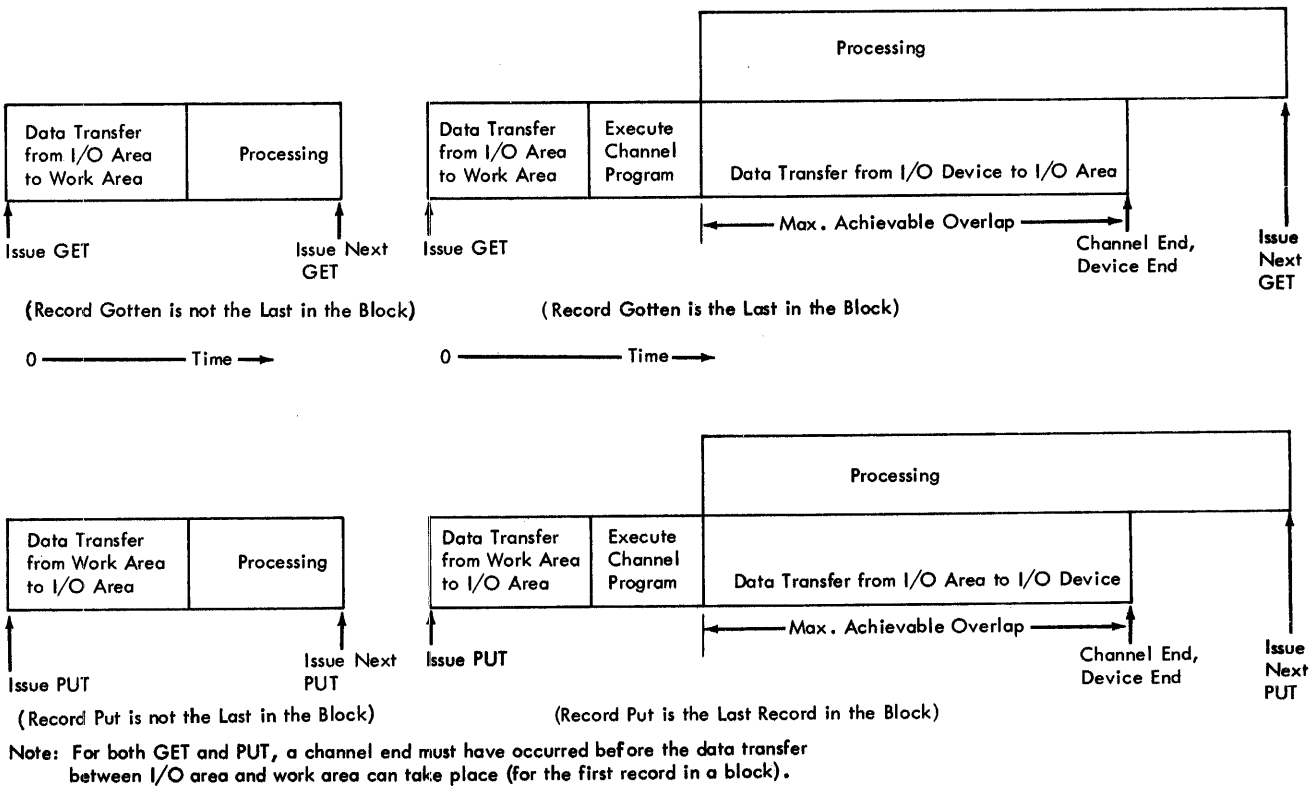


Figure 12. Overlap of Processing and I/O: One I/O Area and No Work Area (Blocked Records)



Note: For both GET and PUT, a channel end must have occurred before the data transfer between I/O area and work area can take place (for the first record in a block).

Figure 13. Overlap of Processing and I/O: One I/O Area and a Work Area (Blocked Records)

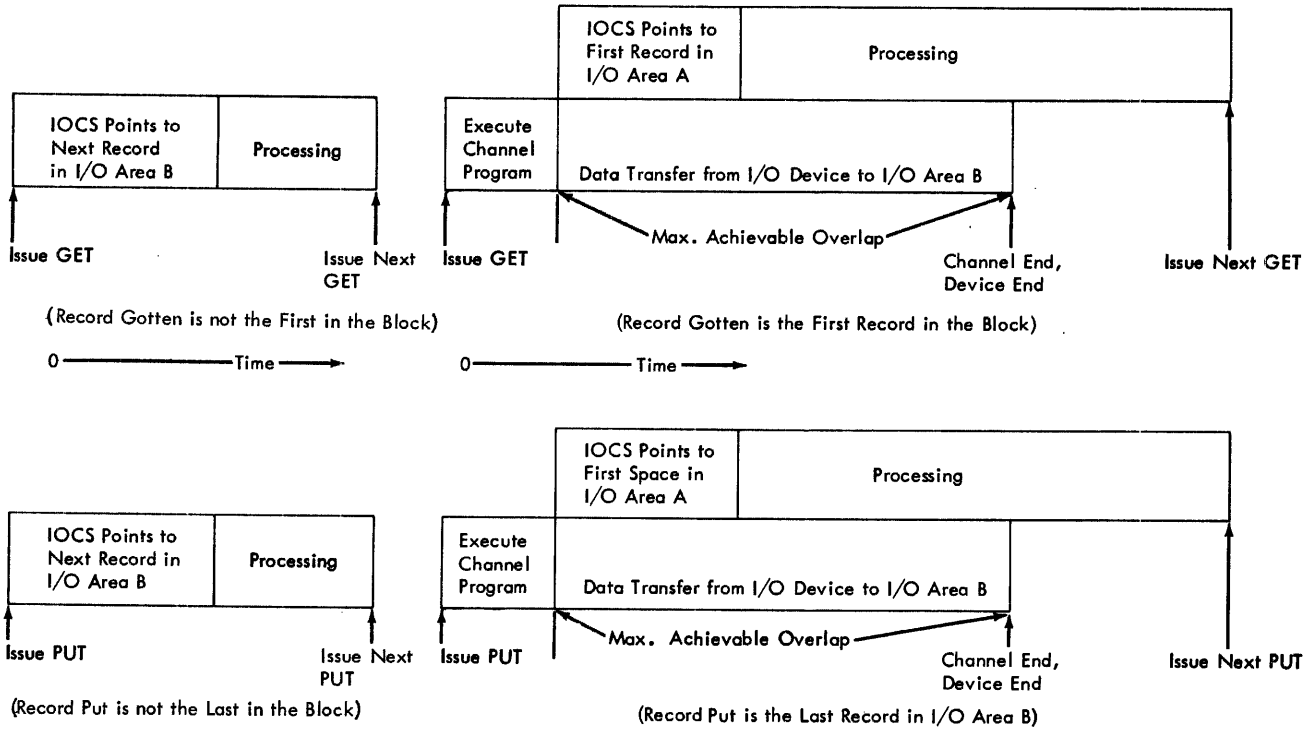


Figure 14. Overlap of Processing and I/O: Two I/O Areas and No Work Areas (Blocked Records)

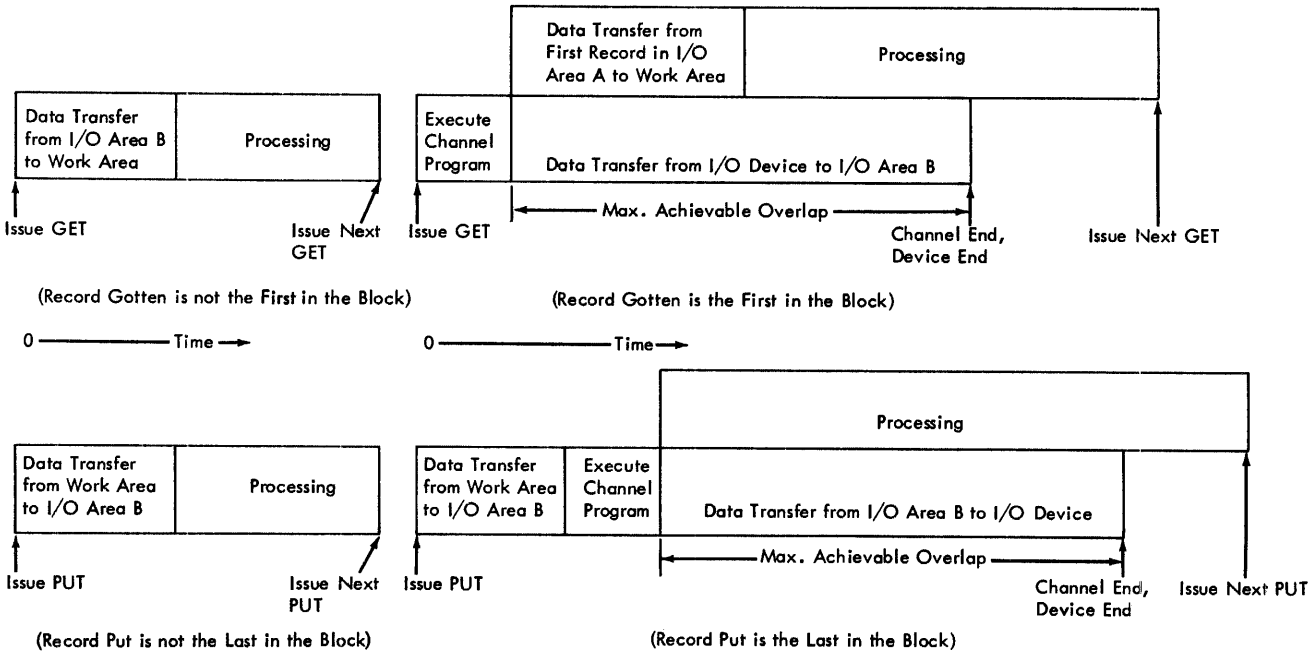


Figure 15. Overlap of Processing and I/O: Two I/O Areas and a Work Area (Blocked Records)

Record Format (Blocked or Unblocked)	Number of I/O Areas	Separate Work Area	Amount of Effective Overlap
Unblocked	1	no	Overlap of the device operation only for buffered devices such as 1403, 1443, 2540. No overlap of magnetic tape, 1052, 1442, 2311, 2671, 1285, 1287.
		yes	Overlap processing of each record. (Record move required.)
	2	no	Overlap processing of each record. (No record move required.)
		yes	Overlap processing of each record. (No advantage to a work area.)
Blocked	1	no	No overlap.
		yes	Overlap processing of last record in each block.
	2	no	Overlap processing of full block.
		yes	Overlap processing of full block. (No advantage to a work area.)

Note: Overlap given is the maximum achievable.

Figure 16. Summary of Achievable Overlap of Processing and Input/Output

Macro Instructions

The following macro-instructions are provided when using the GET-PUT level of the sequential-access method.

GET -- Get a Logical Record: The GET macro-instruction obtains a single logical record from a logical file in either of two operations: move or locate. In the move operation, GET moves the logical record from an input area into a work area specified by the programmer. The record may be processed or extended in the work area.

In the locate operation, GET does not move the logical record from the input buffer, but places into a register the address of the I/O area segment in which the programmer may process that record. The programmer may not extend record size.

GET operates in a strictly sequential manner. As required, the system schedules the filling of input areas, deblocks records, and directs input error recovery procedures. After GET has retrieved all records to be processed and has discovered that no data remains, the system checks labels and passes control to the programmer's end-of-file exit specified in the DTF macro instruction. The system also tests for an end-of-volume condition and initiates automatic volume switching if an input file extends across several volumes. Finally, the system switches from one extent to the next when a file occupies discontinuous areas on a direct-access device.

The following operands must be specified by the programmer in the GET macro-instruction:

1. The name of the file.
2. The address of the programmer's work area for input logical records if the move operation is used, or a register containing the address of the work area.

RDLNE -- Read a Line: The RDLNE macro instruction provides selective on-line correction when processing journal tapes on the IBM 1285 or 1287 Optical Reader. This macro causes the reader to read a line in the on-line correction mode while processing in the off-line correction mode.

RELSE -- Release an Input Block: The RELSE macro-instruction causes the GET macro instruction to ignore the remaining logical records in an input block and to obtain logical records from the next block. When the programmer does not require the remaining contents of an input block that GET is deblocking, he may issue the RELSE macro instruction to release the input area so that the next logical record is retrieved from the next block with the next GET macro.

The file name is the only operand required by the RELSE macro.

DISEN -- Disengage Magnetic Reader: The DISEN macro-instruction causes the magnetic character reader to stop feeding documents.

LITE -- Light Pocket Lights: The LITE macro-instruction permits any combination of pocket lights on the magnetic character reader to be lighted.

PUT -- Put a Logical Record: The PUT macro-instruction places a logical record into an output file in either of two operations: move or locate. In the move operation, PUT moves the logical record from a work area specified by the programmer into an output area. In the locate operation, PUT does not move the logical record into the output area, but places into a register the address of the I/O area segment into which the programmer may build the next record.

Like the GET macro-instruction, PUT operates in a strictly sequential manner. As required, the system blocks records, schedules the emptying of output areas, and handles output error correction procedures, where possible. The system also resolves discontinuities of DASD file extents, tests for an end-of-volume condition, and initiates automatic volume switching and label creation.

The following operands are specified in the PUT macro-instruction.

1. The name of the file.
2. The address of the programmer's work area for output logical records if the move operation is used, or a register containing the address of the work area.
3. An additional operand is provided for the Selective Tape Listing feature. This operand provides forms control for the feature.

TRUNC -- Truncate an Output Buffer: The TRUNC macro-instruction causes the PUT macro-instruction to regard an output area as full, and subsequently to place logical records into the next block. When the programmer does not need the remaining portion of an output area that PUT is blocking, he may issue the TRUNC macro-instruction to truncate the area so that the next logical record is placed into another block. Thus, just as input areas may be released, output areas may be truncated for writing short blocks.

The name of the file is specified in the TRUNC macro-instruction.

The CLOSE macro-instruction effectively truncates the last block of a file.

FEOV -- Force End of Volume: The FEOV macro-instruction causes the system to assume an end-of-volume condition for either an input or output tape file, thereby causing automatic volume switching. When volumes are switched, FEOV creates output labels as required and verifies labels on new input reels. When FEOV is issued to a tape input file, trailer labels are not checked.

The name of the file is specified in the FEOV macro-instruction.

CNTRL -- Control I/O Device: The CNTRL macro-instruction provides the following functions:

For magnetic tape units:

1. Rewind
2. Rewind and unload
3. Erase gap (write blank tape)
4. Write tape mark
5. Backspace to interrecord gap
6. Backspace to tape mark
7. Forward space to interrecord gap
8. Forward space to tape mark

For optical readers:

1. Marking error lines when reading journal tapes.
2. Read keyboard information when reading journal tapes.
3. Ejection of documents.
4. Stacker selection of documents.
5. Incrementing documents.

For direct-access storage units:

1. Seek to specified track
2. Restore strip to data cell (2321 only)

For card readers and punches:

1. Stacker select

For printers:

1. Immediate space specified number of lines
2. Space specified number of lines after print

3. Immediate skip to specified channel

4. Skip to specified channel after print

The following operands must be specified in the CNTRL macro-instruction:

1. The name of the input or output file.

2. A mnemonic code indicating the action to be taken.

3. For card readers and punches - the stacker number.

4. For printers - the number of lines to space or the channel to be skipped to.

Note: If directed to a card reader, CNTRL must follow every GET macro instruction directed to that card reader for the same file, except for a 1442. Furthermore, only one input area may be used. As soon as CNTRL is issued, the card input area may be scheduled for refilling by issuing another GET.

PRTOV -- Test for Printer Overflow: The PRTOV macro-instruction tests overflow indicators for on-line printer channel overflow. If an overflow indicator is on, PRTOV causes either an automatic skip to a new page or a transfer of control to a specified point in the problem program. Before testing overflow indicators, PRTOV waits for completion of all previously requested printing.

The following operands must be specified in the PRTOV macro-instruction:

1. The name of the file.

2. The printer channel to be tested for overflow (either 9 or 12).

3. The address of a user routine may optionally be specified for transfer of control on condition of overflow; if this is not specified, a printer overflow condition causes automatic skipping to channel one.

Updating

A sequential file on a direct-access device, a card input file in a 1442 or 2520, or a card file in the punch feed of a 2540 equipped with the punch-feed-read special feature can be updated. That is, each DASD or card record can be read, processed, and transferred back to the same storage location or card from which it was read. When desired, this function is specified in the DTF macro-instruction that defines the file.

The physical DASD record or card record is transferred to main storage by a GET instruction. After the record is processed, the next PUT instruction causes the updated physical record to be written in the same location or punched in the same card from which it was read. For a card record, PUT transfers the record to the file from the input area of main storage. For a DASD record, the PUT instruction sets up an indication for the next GET instruction, which accomplishes the transfer. (The input area must not be modified between the PUT and GET executions.) If a work area is specified in the GET and PUT instructions, PUT first moves the updated record from the work area back to the input area and then transfers the record to this file.

A GET instruction must always precede a PUT instruction for a DASD or card record, and only one PUT can be issued for each record. A PUT instruction may be omitted, except for the 2540 and 2520 if a particular record does not require updating.

READ-WRITE LEVEL SEQUENTIAL ACCESS

The READ-WRITE level of the sequential-access method provides the programmer with an efficient and flexible means for storing and retrieving the blocks of a sequentially organized disk or tape file. The macro instructions provided with this level of the sequential-access method allow a file to be treated alternately as input and output. It is particularly effective in applications where records are alternately read and written from and to a file used as a temporary extension of main storage.

Storage Areas

A single I/O area equal to one block length is used with these macros. This area is filled or emptied each time a READ-WRITE is issued. The I/O area need not be fixed in location. The programmer supplies the address of the area in the macro instruction itself, each time the macro instruction is issued.

Macro Instructions

The following macro instructions are provided when using the READ-WRITE level of the sequential-access method.

READ -- Read a Block: The READ macro instruction requests that a block be transmitted from a work file to a main-storage area specified by the programmer. READ operates in a strictly sequential manner, starting either at the beginning of the file or at a point previously positioned with one of the POINT macro instructions. READ can also be used to read backwards sequentially from magnetic tape. Each read operation transfers one full block into main storage. To allow overlap of the input operation with processing, READ does not wait for the end of the input operation, but returns control to the problem program. The completion of the operation must be tested by issuing the CHECK macro instruction described below.

After READ has retrieved all records to be processed, and has discovered no data remains, the system passes control to the programmer's end-of-file exit specified in the DTF macro instruction. Work files may not occupy multiple volumes (tape reels or disk packs).

The following operands must be specified by the programmer in the READ macro instruction. For document processing on the IBM 1287 Optical Reader, the READ macro instruction is used to access selected data. The channel command words that effect the reading and the channel command word list to be used in reading the documents from the 1287 file are provided by the user.

1. The name of the file.
2. Type of processing (SQ for sequential or MR for records read from the IBM 1412 or 1419 Magnetic Character Readers).
3. The input area to be used. (Not applicable to 1287, 1412, or 1419 document processing.)
4. For format U records, the record length, or S to specify that the maximum record length must be used. (Not applicable to 1287, 1412, or 1419 document processing.)
5. For 1287 Optical Reader document processing, the address or name of the user provided CCW list used to read a document.

   Note: When processing format F records, the record length in the DTF table can be changed before each read.

The READ macro-instruction, when used for magnetic character readers, permits the use of more than one magnetic reader per program.

WRITE -- Write a Block: The WRITE macro instruction requests that a block be transmitted from a main-storage area specified by the programmer to a work file. Like the READ macro instruction, WRITE operates in a strictly sequential manner. To allow overlap of the output operation with processing, WRITE does not wait for completion of the output operation, but returns control to the problem program. The WRITE operation transfers one full block to the file. The file must be contained within one volume.

The following operands must be specified in the WRITE macro instruction.

1. The name of the file.
2. Type of processing (SQ for sequential).
3. The output area to be used.
4. For Format U records, the record length.

CHECK -- Wait for and Test Completion of Read or Write Operation: The CHECK macro instruction waits for completion of an input/output operation requested by a READ or WRITE macro instruction and tests for errors and exceptional conditions. As required, CHECK passes control to the appropriate exits that are specified by the programmer in the DTF macro instruction for error analysis and end-of-file. The programmer must issue a CHECK macro-instruction to test the input/output operation before issuing any other macro instructions for the same file. Similarly, a problem program must check an input/output operation for completion before altering the input or output area in main storage.

The file name used in the preceding READ or WRITE macro instruction must be specified in the CHECK macro instruction.

NOTE -- Note Positional Data: The NOTE macro-instruction places into a general register the position on a volume of the last block read from or written into a work file. This data identifies the block for subsequent repositioning of that volume.

The identification that NOTE provides is a 3-byte block count for magnetic tape; for direct-access volumes, it is three bytes identifying the cylinder, track, and record number within the track.

The file name must be specified in the NOTE macro instruction.

Notes: The following items should be considered when using the NOTE macro instruction:

1. Before issuing a NOTE macro instruction, the programmer must test the last input/output operation for completion.

2. NOTE is normally used to provide information for a subsequent POINT macro instruction.

3. For output files being written on direct-access volumes, NOTE places into a general register the number of bytes of remaining space on the track containing the noted record. (This data may be used subsequently by a POINTW macro instruction).

POINTR -- Position to Block for Read:

POINTW -- Position to Block for Write:

POINTS -- Position to Start of File: Three POINT macro instructions are provided to allow the programmer to reposition a work file to a specified block for subsequent operations. The POINTS macro instruction, when used for magnetic tape files, causes a rewind to the tape load-point and then positions the tape to the first data block (bypassing any labels to the first tape mark). When the POINTS macro instruction is issued to a DASD file, the file is positioned to the lower limit of the first extent of the file. The POINTR and POINTW macro instructions are used on both DASD and magnetic tape files to position the file to a specified block. The block count (for tape) or the physical cylinder, track, and record address (for DASD) specified in these macro-instructions are obtained from a previous NOTE.

Subsequent READ instructions following a POINTR pick up blocks sequentially, beginning with the one specified. Subsequent WRITE instructions following a POINTW write blocks sequentially immediately following the one specified. Subsequent writes to a tape work file cause any blocks previously written on that portion of the tape to be destroyed and the new blocks to be written in their place. On direct-access device files, however, WRITE instructions following a POINTW place blocks on remaining unused track space immediately following the block specified. The amount of space remaining on the track is determined by the NOTE macro instruction that was previously used to identify the location.

The following operands must be specified by the programmer in the POINT macro instructions.

1. The name of the file.

2. For POINTR and POINTW, the address of the main-storage location containing the tape block count or the DASD address.

CNTRL -- Control Input/Output Devices:

FEOV -- Force End of Volume: The CNTRL and FEOV macro-instructions described for the GET-PUT level of the sequential-access method are also provided with the READ-WRITE level for files defined as work files. Their operation is exactly the same in both cases.

DSPLY -- Display a Field: The DSPLY macro instruction displays a document field on the display screen of the IBM 1287 Optical Reader. This macro is used to key in a complete field on the keyboard when a 1287 read error makes this type of correction necessary.

RESCN -- Reread a Field: The RESCN macro instruction selectively rereads a field on a document when a 1287 read error makes this type of correction necessary.

WAITF -- Wait: The WAITF macro instruction, for use with document processing on the IBM 1287 Optical Reader, is issued to ensure that the transfer of a record has been validly completed.

The WAITF macro-instruction is used with magnetic character readers in a multiprogramming system to determine if any magnetic character reader has documents ready for processing.

INDEXED-SEQUENTIAL FILE MANAGEMENT SYSTEM

INDEXED-SEQUENTIAL ORGANIZATION

An indexed-sequential file is one whose records are organized on the basis of a collating sequence determined by control fields called keys that precede each record or block of data. The key for each block of data is 1-255 bytes in length and contains the identifier of the last logical record in that block. An indexed-sequential file exists in space allocated on direct-access volumes as prime areas, overflow areas, and index areas.

Indexed-sequential organization gives the programmer a great deal of flexibility in the operations he can perform on a file. He has the ability to:

- Read or Write (in a manner similar to that for sequential organization) logical records whose keys are in ascending collating sequence.

- Read or write individual random logical records. If a large portion of the file is being processed, reading records in this manner is somewhat slower per record than reading according to a collating sequence. A search for pointers in indexes is required for the retrieval of each record.
- Add logical records with new keys. The system locates the proper position in the file for the new record and makes all necessary adjustments to the indexes.

The ISFMS has these advantages:

- An IOCS file management system specifically designed for direct access storage devices.
- Sequentially organized files that can be processed in random order or in sequential order.
- Both READ/WRITE and GET/PUT macro instruction routines available to the problem program.
- Routines for processing blocked or unblocked records.
- Record processing directly in the I/O area or in a work area.
- Presorted logical records that are loaded onto disk while a series of indices are established for subsequent processing.
- An efficient chaining method for handling additions requiring an overflow area.

The ISFMS has these restrictions:

- Only one I/O area is permitted when a file is loaded.
- All physical data records must contain key areas, and all key areas must be the same length.
- Data records must be fixed-length only.
- Only Standard Disk Labels are permitted.
- For multipack files, all packs must be online for any function (loading, adding, retrieving randomly or retrieving sequentially) performed for the file.
- The prime data area for a logical file must be contained within one extent on a disk pack or a data cell. It must start on the first track (track 0) of a cylinder, and it must end on the last track (track 9 or track 19) of the same or a different cylinder. Prime data extents cannot start or end in the middle of a cylinder. For a multipack file, the prime data area must continue from the last track of one pack to the first track (track 0) of cylinder 1 on the next pack so that the area is considered continuous by ISFMS. The first cylinder (cylinder 0) is reserved for labels. For a multicell file, the prime data area must continue from the last track of one cell to the first track in the second head position (HH=10) on the next cell so that the area is considered continuous by ISFMS. The first head position (HH=00) is reserved for labels.

Indexes

The ability to read and write records from anywhere in a file with indexed-sequential organization is provided by indexes that are part of the file itself. There are always two types of indexes: a cylinder index for the whole file, and a track index for each cylinder. An entry in a cylinder or track index contains the identification of a specific cylinder or track and the highest key that is associated with that cylinder or track. The system locates a given record by its key after a search of a cylinder index and a track index within that cylinder.

A third index can be created and used if desired. If a file occupies many cylinders, a search of the cylinder index for a key is inefficient. The programmer can request that a master index be created that indexes the cylinder index as shown in Figure 17. Each entry in the master index points to a track in the cylinder index. A master index should be constructed if the cylinder index occupies four or more tracks, and a cylinder index in main storage is not used.

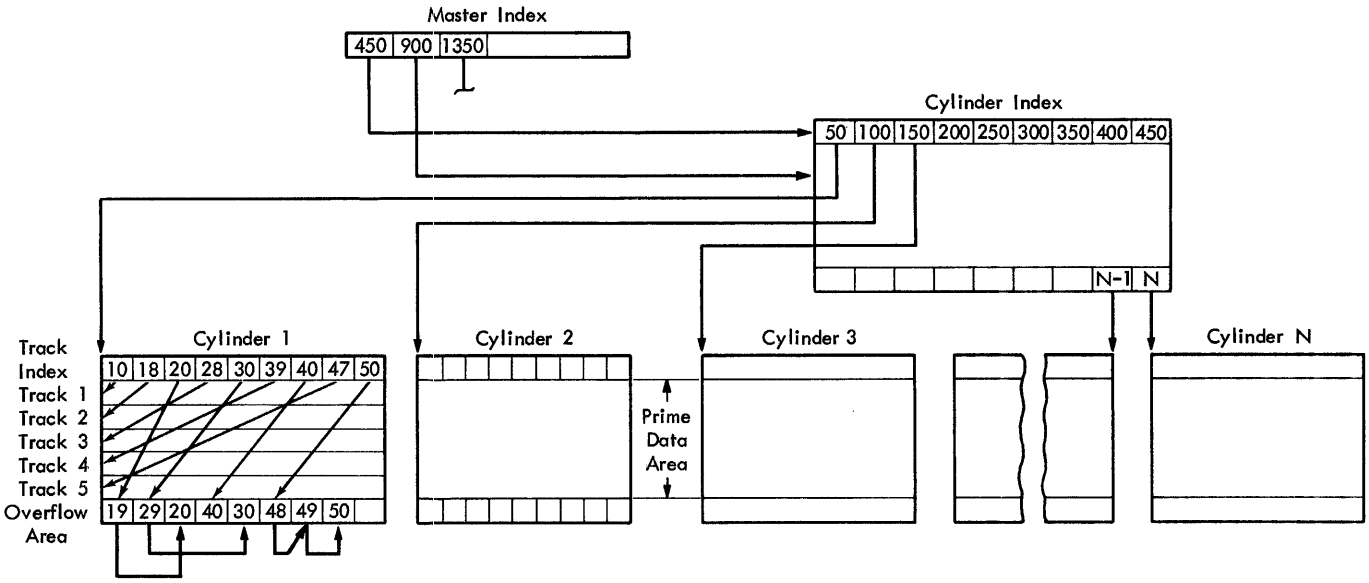


Figure 17. Index Structure for an Indexed-Sequential File

Insertion of Records

A new record added to an indexed-sequential file is placed into a location on a track determined by the value of its key field. If records were inserted in precise physical sequence, insertion would require shifting all records of the file with keys higher than that of the one inserted. However, because an overflow area exists, indexed-sequential file organization allows a record to be inserted into its proper position on a track, with only the records with higher keys on that track being shifted.

Overflow Area

In addition to the prime area, whose tracks initially receive records of an indexed-sequential file, there is an overflow area for records forced off their original tracks by the insertion of new records. When a record is to be inserted, the records already on the track that are to follow the new record are written back on the track after the new record. The last record on the track is written onto an overflow track. Track index entries are adjusted to indicate records on an overflow track.

RESIDENT CYLINDER INDEX

All or part of the cylinder index can reside in main storage. If the area assigned to the cylinder index is large enough for all the index entries to be read into main storage at one time, no presorting of the record keys need be done. If the area assigned to the cylinder index is not large enough, the keys of the records to be processed should be presorted to fully utilize the resident cylinder index. This option speeds up the random retrieve and add functions when the number of records to be processed is significant. Processing time per record is reduced by decreasing the number of times that the cylinder index on secondary storage must be accessed. Searching the master index (if any) is eliminated entirely.

Another option permits the writing and reading of more than one physical record on DASD per I/O operation. This option can be used when adding a new record on a prime data track and shifting over the existing records. It is available to the prime areas only. Fewer I/O operations are required, thereby increasing speed.

Any user not desiring the new options need not modify or reassemble his source decks. The parameters for specifying these options are described in the Supervisor and Input/Output Macros publication listed on the front cover of this publication.

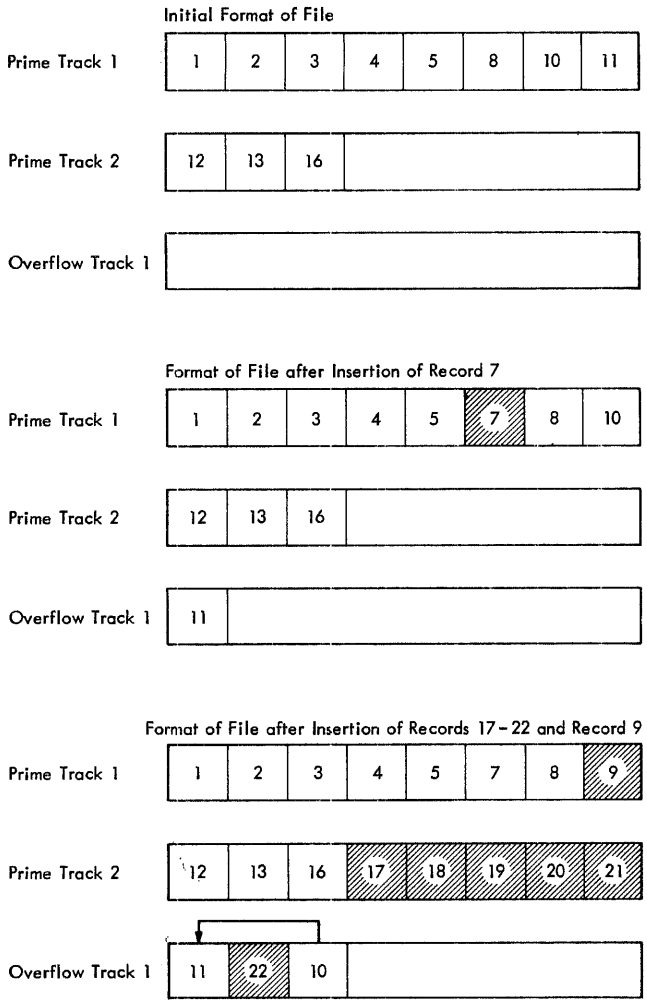


Figure 18. Addition of Records to a 1-Cylinder, 3-Track Indexed-Sequential File

Figure 18 illustrates this adjustment for addition of records to an indexed-sequential file whose keys are in a numerical collating sequence. When this file is created, its records are placed on the prime tracks in the storage area allocated to the file. If a record, e.g., record 7, is to be inserted into the file, the indexes indicate that record 7 belongs on prime track 1. Record 7 is written immediately following record 5, and records 8 and 10 are retained on prime track 1. Since record 11 no longer fits on this track, it is written on an overflow track and the proper track index is adjusted to show that the highest key on prime track 1 is 10 and that overflow records exist. When records 17 through 22 are added to the end of the file, prime track 2 receives records 17 to 21, but record 22 does not fit and is written following record 11 on the overflow track. When record 9 is inserted, record 10 is shifted to the overflow track after record 22. Note that records 10 and 11 on the overflow track are chained together to show their logical sequence and to indicate that they belong on the same prime track.

Two types of overflow areas may be requested by the programmer. He can request a cylinder overflow area that provides a certain number of tracks on each cylinder to hold the overflow of that cylinder. He can also request an independent overflow area that provides a certain number of tracks independent of the rest of the file, perhaps even on a different volume. The independent overflow area is used whenever one of the cylinder overflow areas is filled, or the independent overflow area can be used without a cylinder overflow area.

LOGIC MODULES

The logical IOCS routines provided for indexed-sequential files are much more than an access method. Several routines are available to provide complete file management for direct-access storage files. The complete facility is called the indexed-sequential file management system (ISFMS). The ISFMS routines can be retrieved from the relocatable library by the linkage editor just as with the sequential access method routines. The routines must be assembled by the user before placing them in the relocatable library. This is normally a one-time operation, performed as part of the total system generation process. The routines generated are tailored to provide specific functions but remain generalized as to specific file attributes. Four basic types of routines are available:

1. Load - To load (create) an indexed-sequential file.
2. Add - To add records, in sequence, to an existing indexed-sequential file.
3. Sequential Retrieval - To retrieve records sequentially in key sequence from an indexed-sequential file.
4. Random Retrieval - To retrieve individual records called for by key from any point in the file.

The load routine is always separate. No other operations can be performed on an

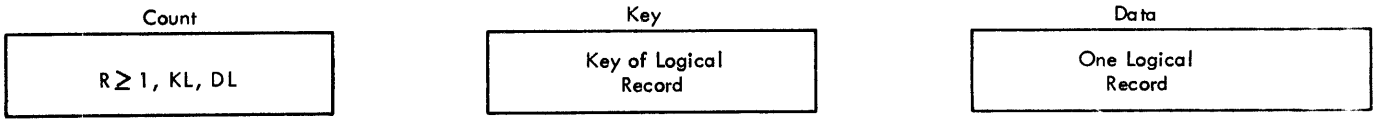


Figure 19. Unblocked Fixed-Length Records

output file as it is being loaded. The add and retrieve routines can be used in any combination. Furthermore, the retrieval routines can be assembled with updating capability, allowing records to be written back into the file after retrieval.

The assembled routines are called logic modules. They are selected from a master source routine in accordance with parameters in a special macro instruction, ISMOD-indexed sequential module. The assembled logic modules are completely file independent and can be used for all indexed-sequential files. If desired, the logic modules can be assembled along with the user's problem program and included in the output object module.

The user's program includes a DTFIS macro instruction for each indexed-sequential file to be processed. Some of the fields within the DTF table generated from this macro instruction are not determined or filled in until the file is opened during execution of the program. Many of the fields in the table are retained with the file in a special format of the standard DASD file label (format 2).

In addition to the parameters that describe the file to be processed, the DTFIS macro instruction includes certain parameters identical to those used in the ISMOD macro instruction.

RECORD FORMATS

The indexed-sequential access method supports fixed unblocked records and fixed blocked records.

The formats of records as they appear on direct-access devices are shown in the following sections. Each block is recorded on a direct-access device with a count field, a key field, and a data field.

The key field is used by the system to locate a requested logical record. The data field contains the logical records.

Fixed Unblocked

Fixed-length unblocked records appear on direct-access devices as shown in Figure 19.

R is the sequence number of the record on the track (R0 is not used).

DL is the logical record length.

KL is the key length. This length must remain constant for the file.

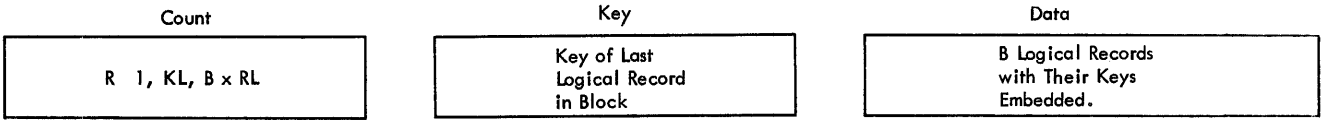


Figure 20. Blocked Fixed-Length Records

Fixed Blocked

Fixed-length blocked records appear on direct-access devices as shown in Figure 20.

R is the sequence number of the record on the track (R=0 is not used).

RL is the logical record length.

B is the number of logical records appearing in a block and is a constant for a file.

KL is the key length. This length must remain constant for the file.

More specifically, the key and the data areas may be pictorially represented as shown in Figure 21.

Keys 1, 2, and 3, respectively, are the keys of logical records 1, 2, and 3 and are physically embedded in the records. The position and length of the key in each logical record of the file must be specified in the DTF macro-instruction. The contents of the key field on the direct-access volume is used by the system for locating the block containing a requested logical record.

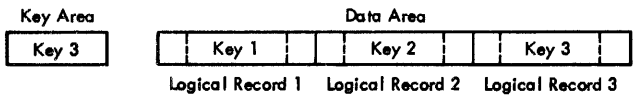


Figure 21. Key and Data Areas

OVERFLOW RECORDS: The preceding figures are for prime (rather than overflow) data records only. Data records in an overflow area are organized somewhat differently, as shown in Figure 22. They are never blocked, even though the prime data records may be blocked. In addition, they contain a link field which links to the next record in the overflow chain. The link field is 10 characters in length.

MACRO INSTRUCTIONS FOR INDEXED-SEQUENTIAL FILES

Both GET-PUT and READ-WRITE input/output macro instructions are provided for indexed-sequential files. The purpose and effect of the instructions vary depending on the logic modules used and conditions preset by other macro instructions. The following macro instructions are provided:

DTFIS Define The File for Index Sequential

SETFL Set File Load Mode

ENDFL End File Load Mode

READ Read a Record (random retrieval)

WRITE Write a Record (load, add, or random update)

WAITF Wait for Completion of READ or WRITE

SETL Set Lower Limit for Sequential Retrieval

ESETL End Sequential Retrieval

GET GET a Record (sequential retrieval)

PUT PUT a Record (sequential update)

This section does not discuss each of these macro instructions separately but, instead, presents them in the context of a particular function.

CREATING AN INDEXED-SEQUENTIAL FILE

In order to create an indexed-sequential file, the programmer writes a DTFIS macro instruction that calls for a load routine. The parameters of the macro instruction completely define the desired output file. Note that this macro instruction provides no information to the system about the

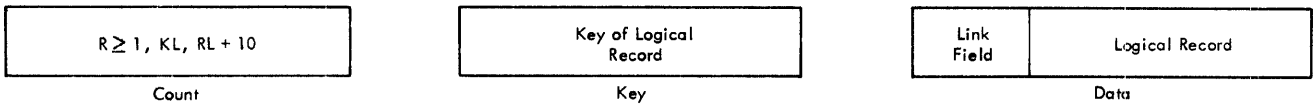


Figure 22. Overflow Records

input file. The flexibility of the system allows the input to be retrieved by any access method. For example, two or more input files can be merged as they are presented to the load routine. These can be defined for sequential, direct, or indexed-sequential retrieval routines from any device. The only requirement of the load routine is that the records be presented in ascending order by key.

A SETFL macro instruction is issued before beginning the file loading process. The SETFL initializes the index areas. Control does not return to the problem program until this operation has been completed. The program can then begin loading the file.

Each logical record is presented to the load routine separately. The programmer issues a WRITE macro instruction to place the record in the file. Note that for blocked files, the WRITE macro instruction in this instance operates like the PUT, blocking records in an output area and then performing the actual I/O operation when the block is full.

When all records have been loaded, the programmer issues an ENDFL macro instruction to terminate the loading process. This instruction writes the last block of records, followed by an end-of-file record. It then completes the indexes with any final index record needed and writes dummy index entries for the rest of the specified prime data extent.

To Extend a File

The same program used to initially set up an indexed-sequential file can be used to extend the file, where all records being added are sequentially higher than the last record previously in the file.

TO ADD RECORDS TO A FILE

The WRITE macro instruction is used to add new records to an existing file. Although the WRITE macro is written just as when loading or extending a file, it is not preceded by the SETFL macro. The operation of the WRITE, when records are being added, is entirely different. When the WRITE is issued, the add routine locates the proper block for the new record (using the indexes to speed the search). It reads this block, inserts the new record in the proper location, shifts all following records on the track over one record position (changing records from one block to the next when necessary) and rewrites the block. The last record previously on the track is then written in the overflow area. (This is always true except in the case of the last track in the file, which may have usable space remaining after the last record.) The WAITF macro instruction is used at the point in the program where processing must be held up until the I/O operation is complete.

SEQUENTIAL RECORD RETRIEVAL AND UPDATE

Sequential retrieval from an indexed-sequential file begins at a location or record specified in a SETL macro instruction. Input blocks are read and each record is presented in sequence in response to the GET macro instruction. When necessary, the sequential retrieval routine reads from the overflow area for records that were displaced from the prime area by added records. The track index is used to indicate when this is necessary. The key field of unblocked records is read along with the data field. With blocked records, however, the key of the block (repeated in the last record of the block) is not read.

The programmer can issue a PUT after processing each record to cause it to be written back into its original location. If the file is blocked, the entire block is written back after all records in the block have been processed and a GET is issued for the first record in the next block or if an ESETL is issued. The PUT macro instruction does not have to be issued for records that have not been changed; a series of GET's can be issued with no intervening PUT. The entire block is written back into the file if, and only if, a PUT is issued for any record in the block.

Once a SETL macro instruction has been issued, GET and PUT are the only I/O operations that can be performed before issuing an ESETL macro instruction. For example, if a WRITE is to be issued to add a record to a file that is being processed sequentially, it must be preceded by an ESETL. After adding the new record, the SETL macro instruction can be reissued, specifying the last record processed as the new starting point.

RANDOM RECORD RETRIEVAL AND UPDATE

Random retrieval from an indexed-sequential file is performed by the READ macro instruction. The programmer first places the key field of the desired record in a special field. In response to the READ instruction, the random retrieval routine searches the indexes to locate the track containing the desired record and then searches the track for the record. The block containing the record is read and the record is made available for processing. For both blocked and unblocked files, only the data portion of the record is read; the key field is not read.

The programmer can issue a WRITE after processing the record to cause it to be written back into its original location. To allow overlap of input and output operations with processing, READ and WRITE do not wait for completion of the operations, but return control to the problem program. The WAITF macro instruction is used at the point in the program where processing must be held up until the I/O operation is complete.

REORGANIZING AN INDEXED-SEQUENTIAL FILE

As new records are added to an indexed-sequential file, existing records are placed in overflow areas. The access time for retrieving records in an overflow area is greater than that required for retrieving other records. Therefore, when many overflow records develop, input/output performance is reduced. For this reason, the programmer should reorganize indexed-sequential files as soon as the need becomes evident. The system maintains a set of statistics to assist the programmer in determining when reorganization is required.

These statistics are maintained in the format 2 file label recorded with the file. During file processing, they occupy fields within the DTFIS table. The DTFIS name (filename) is concatenated with a letter to reference these fields. The programmer can test these 4-byte fields as he processes the file. The following statistics are kept:

- Prime Record Count (filenameP) - A count of the number of records in the prime record area. (filenameP is used for DTFIS ADD, while filenameP+4 is used for DTFIS LOAD.)

- Overflow Record Count (filenameO) - A count of the number of records in the overflow area(s).

- Available Independent Overflow Tracks (filenameI) - A count of the number of tracks remaining in the independent overflow area.

- Cylinder Overflow Areas Full (filenameA)- A count of the number of cylinder overflow areas that are full, necessitating use of the independent overflow area.

- Non-First Overflow Reference (filenameR)- A count of the number of times a random reference (READ) is made to records that are the second or higher links in an overflow chain.

In addition to these statistics maintained by the system, there is another field (filenameT) that can be used by the programmer to keep a count of tagged (for deletion) records. This field is kept in the format-2 label and is available in the DTFIS table during file processing.

No facility is provided in the Disk Operating System for tagging or deleting records. The programmer can perform these functions in any way that he wishes. The data portion of the records can be tagged or deleted as desired. The key fields should not be changed in any way that modifies their sequence in the file. Tagged or deleted records can be eliminated when reorganizing the file.

Reorganization is accomplished by creating a new version of the file, using the existing version as input. The program that performs this operation would include two DTFIS macro instructions: one describing the input file (sequential retrieval), and the other describing the output file (loading). Note that these are completely unrelated files as far as the system is concerned. They must even be called by different names in the macro instructions. (These names are not stored with the files; a file can be created using one name and processed using another.)

INDEXED SEQUENTIAL DISK STORAGE SPACE FORMULAS

Three formulas compute IBM 2311 disk storage requirements for an Indexed Sequential file. The known quantities for the computations given are:

D_L = Data Length
K_L = Key Length
B_L = Block Length (Data Length x Number of Records)
X = Number of prime data tracks per cylinder
L = Number of bytes (10) for overflow link information.

I. TO CALCULATE THE NUMBER OF PRIME DATA RECORDS PER CYLINDER (Npr)

Let: A = Number of prime data records on a shared track
B = Number of records on a non-shared track.

(Note: These values must be whole numbers.)

Then: a. Determine the size of the track index (T_1),

$$T_1=[2X+1][91.49+1.049(K_L)]$$

b. Determine the number of bytes remaining on a track for prime records (T_2),

$$T_2=3625-T_1$$

c. Determine the size of the last prime record on a track (T_3),

$$T_3=20+K_L+B_L$$

d. Determine the number of prime data records on a shared track (A),

$$T_4=T_2-T_3$$

if the result (T_4) is negative, set A=0,

if the result (T_4) is zero, set A=1,

if the result (T_4) is positive, set

$$A=1+\frac{T_4}{81+1.049(K_L+B_L)}$$

e. Determine the number of records on a non-shared track (B),

$$B=1+\frac{3605-(K_L+B_L)}{81+1.049(K_L+B_L)}$$

Compute the number of prime records per cylinder (Npr) by substituting for A, B and X in

$$Npr=A+B(X-1)$$

II. TO DETERMINE THE NUMBER OF OVERFLOW RECORDS PER TRACK (Nor)

Compute:

$$Nor=1+\frac{3605-(K_L+D_L+L)}{81+1.049(K_L+D_L+L)}$$

III. TO DETERMINE THE NUMBER OF CYLINDER OR MASTER INDEX RECORDS PER TRACK (Nir)

Compute:

$$Nir=1+\frac{3595-K_L}{91.49+1.049(K_L)}$$

(Note: Allow for a dummy record.)

DIRECT-ACCESS METHOD

The direct-access method (DAM) is a flexible access method provided specifically for use with direct-access storage devices. The flexibility of the system is derived from the advanced design of the direct-access devices available with the System/360 rather than being the result of intricate and lengthy programming routines. All of the direct-access devices supported by the Disk Operating System use the same recording techniques. Some of the outstanding features of these devices are:

- Flexible record format, at the record level.
- Flexible record reference; either on record ID (physical track and record address) or on record key (control field of the logical record).
- Ability to search sequentially through an area for a record, using a minimum of central processing unit process time.

The macro instructions of the direct-access method allow the programmer to take advantage of the flexibility of the System/360 direct-access devices with a minimum of effort. The function of the READ-WRITE macro instructions is basically the same as the physical IOCS EXCP macro instruction, except that the system provides the appropriate CCW chains. There are no elaborate routines for handling file maintenance functions such as adding records to existing files, handling overflows, locating synonyms (multiple records with duplicate addresses), deleting records, etc. Many of the problems associated with these functions that had to be solved with involved programming procedures on prior systems are virtually eliminated because of the advanced recording and addressing technique of the direct-access devices used with the System/360.

The ease with which random files can be created and maintained illustrates this flexibility. The user is still faced with the problem of developing a randomizing formula to convert record keys (control fields) to valid device addresses. However, it is no longer of such paramount importance that the addresses produced be nearly perfectly distributed or that the number of duplicate addresses be kept to a bare minimum. The formula can randomize to track instead of to record. In some cases, it is sufficient to randomize to a cylinder address. Records can be written in any order on a track. If there is no room on the track, the record is simply written on the first track that does have an available location. There is no need to chain records with duplicate addresses. To retrieve the record, the system can start at any specified track (the randomized address) and search for the record on the basis of its actual key. It will search exactly as far to locate and read the record as it did to locate the available location when writing it.

Another example of the flexibility of these devices is the ease with which selected records can be processed in a sequential file. A presequenced file can be loaded into consecutive record locations and then processed:

1. In straight sequential order, reading every record, for jobs that require the entire file.
2. In selective sequential order, reading specific records (determined by a sequenced transaction file), for jobs that require only selected records out of a file.

The direct-access method provides two programmed features to enhance the flexibility of the System/360 direct-access device recording techniques:

1. Maintenance of capacity record
2. Return of ID for next read, write, or start of search.

LOGIC MODULE ASSEMBLY

The logic modules available with the direct-access method must be assembled by the user from a master source routine supplied by IBM. Once assembled, they can be stored in the relocatable library and linked automatically with any program that requires them. If preferable, they can be assembled along with the user's problem program and included in the same output object module. The modules are assembled in accordance with parameters in a special macro instruction DAMOD (direct-access module). These parameters specify the functions that the particular module is to provide.

A file-definition macro instruction, DTFDA (define the file for direct access), is written to define each file to be processed.

RECORD FORMATS

The direct-access method supports fixed unblocked records and undefined records. Both format F and format U records can be written on direct-access storage with or without keys. The data portion of undefined records can be variable in length. If key fields are written, however, they must be present on all records and they must all be the same length. Record blocking and deblocking must be handled by the programmer.

CAPACITY RECORD

When creating variable-length sequential files and in most random-file applications, records are written by searching on the ID of the last record currently on the track and then writing count, key, and data of the new record. The direct-access routines maintain a special record on each track to facilitate this type of operation. The record is called the capacity record. It is written in the data portion of the first record (R0) of each track. The capacity record contains the ID of the last record currently on the track and the count of the number of bytes that can be written after the last record. When a record is to be written, the system can:

1. Read the capacity record,
2. Determine whether there is room on the track for the record.
3. If the record fits, write the record and the updated capacity record.
4. If no room, notify the problem program.

The capacity record is not always used. The description of the WRITE macro instruction explains when it is used.

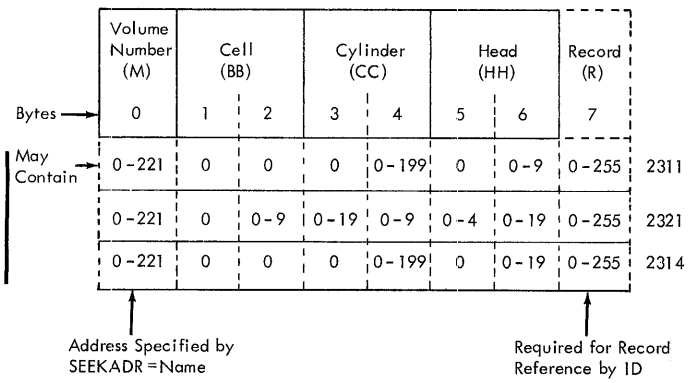


Figure 23. DASD Address Formats

ID LOCATION

The direct-access method requires that the programmer provide actual DASD addresses for all READ-WRITE operations. These are 8-byte binary addresses (Figure 23) of the form MBBCCHHR, where:

M identifies the file volume. A single logical file can occupy several volumes (i.e., disk packs or data cells). When this is the case, the physical units must be assigned (in XTENT control cards) to a sequential set of symbolic unit numbers. For example, a single logical file on three disk drives could be assigned to SYS002, SYS003, SYS004. The value M is always 0 for the first volume, 1 for the second, 2 for the third, etc. In the previous example, M=0 refers to SYS002; M=1 refers to SYS003; and M=2 refers to SYS004. Note that M is never read or written on the storage device by the Disk Operating System. It is used to reference the proper logical unit block for the symbolic name (see Symbolic Device Addressing under Physical IOCS).

BB is a 2-byte field containing cell numbers (0-9) for IBM 2321 Data Cell Drives. This is always a 2-byte binary zero for IBM 2311 or IBM 2314 Disk Storage.

CC is a 2-byte field, containing either cylinder number (0-199) for 2311 or 2314 disk storage, or subcell (0-19) and strip (0-9) for 2321 data cell.

Note: The last four strips of subcell 19 are reserved for alternate tracks, and consequently are invalid addresses.

HH is a 2-byte field, containing either head number (0-9) for 2311 disk storage, head number (0-19) for 2314 disk storage, or cylinder (0-4) and head (0-19) for 2321 data cell.

R is the record number. This byte can have a binary number of 0 to 255 to identify the physical location of the record on the track. The value R does not always have to be specified. The descriptions of the READ and WRITE macro instructions explain when R is needed.

These 8-byte addresses are used either as the starting point for a search on record key or as the actual address for a READ or WRITE ID. When searching for a record key, the programmer has the option of specifying that the search be only with-

in the specified track or from track to track, starting at the address given and continuing either until the record is found or until the end of the cylinder is reached.

For certain types of operations, the system can be requested to return the ID (CCHHR) of the record read or written or of the next record following the one read or written. The programmer can place these in the 8-byte address field to either READ or WRITE a new record or to update the one read. For example, to delete a record from a random file with keys, the programmer can randomize the record key to a starting location, search on key to read the record, and then use the ID returned to write a blank or zeroed record (key and data) back into the same location. The descriptions of the READ and WRITE macro instructions explain when the ID can be returned and whether the ID returned is that of the same or of the next record.

When the ID returned is that of the next record, the system obtains the ID by chaining to a read-count command. This command will skip to the next track if the record read or written was the last currently on the track. The system does not attempt to read the next ID if the end of a cylinder is reached. In this case, it adds one to the CC portion of the previous ID, forces the HH portion to 0, and forces R to 1 for a 2311 or 2314 file. For a 2321 file, it adds one to the high order H, forces the low order H to zero, and forces R to one. An overflow from the high order H will increase the low order C by one, force both H's to zero, and force R to one. Subsequent overflows of address locations will cause increases in the next higher positions of the addresses. (It is the user's responsibility to check the validity of the address returned in IDLOC.)

WRITE -- Write a Block

The WRITE macro instruction can be used to execute the following operations:

Instruction Format		Operation
WRITE filename,KEY	1.	Search on key
	2.	Write data
WRITE filename,ID	1.	Search on ID
	2.	Write key (≥0) and data
WRITE filename,RZERO	1.	Search on specified ID.
	2.	Restore the maximum capacity of the track in R0.
	3.	Erase the remainder of the track following R0.
WRITE filename,AFTER	1.	Read R0 (capacity record) of specified track
	2.	Calculate whether room exists to write record after last record.
	3.	Search on previous ID (ID of last record taken from R0).
	4.	Write count, key (≥0), and data.
	5.	Update R0 with new last ID and count
WRITE filename,AFTER,EOF	1.	Write EOF on DASD

WRITE filename,KEY: The programmer supplies the key of the record to be updated and the track address (MBBCCHH) at which the search is to begin. Only the data portion of the referenced record is written. If the DTFDA macro instruction specifies that the system is to return record

ID, the ID returned depends on whether or not the search multiple option is specified.

- With search multiple - ID of record written
- Without search multiple - ID of the record following the one written.

WRITE filename,ID: The programmer supplies the track and record address (MBBCCHHR) of the record to be written. The system searches on this ID and then executes a write key and data (or write data only) operation. The key field length can be zero. If the system is requested to return ID, the ID returned is that of the next record in the file.

WRITE filename,RZERO: The programmer supplies the track address after which the track is to be erased. The system searches for this track, restores the maximum capacity of the track in R0, and erases the remainder of the track after R0.

WRITE filename,AFTER: The programmer supplies the track address (MBBCCHH) of the track at the end of which the record is to be written. The system reads the capacity record and checks to see if there is enough room for the record to be written. If not, the problem program is notified. If there is enough room, the system searches on the ID of the last record and then executes a write count, key, and data operation for the new record. The capacity record is then rewritten with the updated count of remaining bytes and the new last-record ID.

This format of the WRITE cannot return any ID field.

If requested in the DTF macro instruction, each of the preceding write operations is verified by a read command chained to the write.

In all cases, the WRITE macro instruction returns control to the problem program after requesting execution of the CCW chain. The programmer can perform any processing desired and then issue a WAITF macro instruction to check for completion of the operation.

WRITE filename,AFTER,EOF: This operation will cause an EOF record to be written after the last record on the track.

READ -- Read a Block

The READ macro instruction can be used to read from a random file in either of two ways:

Instruction Format	Operation
READ filename,KEY	1. Search on key 2. Read data
READ filename,ID	1. Search on ID 2. Read key (≥ 0) and data

In the first format (reference by key), the programmer supplies the key field of the record to be read and the track address (MBBCCHH) at which the search is to begin. Only the data portion of the referenced record is read. The system searches either the track specified or to the end of the cylinder, depending on whether the search multiple option is selected. If requested to return record ID, the system returns:

- With search multiple - ID of record read
- Without search multiple - ID of record following the record read.

In the second format (reference by ID) the programmer supplies the track and record address (MBBCCHHR) of the record to be read. The system searches on this ID and then executes a read key and data (or read data only) operation. The key-field length can be zero. If requested to return ID, the system will supply the ID of the next record.

The READ macro instruction returns control to the problem program after requesting execution of the CCW chain. The programmer can perform any processing desired and then issue a WAITF macro instruction to check for completion of the operation.

CNTRL -- Control Seek and Control Restore

The CNTRL macro instruction provided with the direct-access routines can be used in two forms:

CNTRL - Seek	To initiate a seek operation in anticipation of a subsequent READ or WRITE
CNTRL - Restore	To restore a data cell

strip from the read/write head to the cell.

The READ and WRITE macro instructions do not have to be preceded by a CNTRL-Seek. They automatically seek to the correct cylinder if necessary. However, it is often possible to issue a seek to position the access mechanism to the correct cylinder before the actual ID or key required for a READ or WRITE is available.

Use of the CNTRL-Restore macro instruction is not essential. The 2321 data cell automatically restores a strip from the read/write drum:

1. When a seek is given for another strip
2. After 800 milliseconds pass without an operation being performed on the strip.

If the programmer knows that he is finished with a strip, he should issue a CNTRL-Restore to avoid unnecessary delay when the next READ is issued.

The CNTRL macro instruction returns control to the problem program as soon as the operation is initiated.

The programmer must provide the following operands for the CNTRL macro instruction:

1. The name of the file
2. The operation to be performed; either SEEK or RESTR
3. For a seek, the symbolic name of the field containing the seek address.

WAITF-Wait for and Test Completion of Read or Write Operation

The WAITF macro instruction waits for completion of an input/output operation requested by a READ or WRITE macro instruction and tests for errors and exceptional conditions. Any exceptional conditions discovered are passed to the problem program in a special 2-byte location defined by the programmer. The programmer must issue a WAITF macro instruction after a READ or WRITE before issuing any other macro instruction for the same file.

The only operand required for this macro instruction is the name of the file.

CREATING DIRECT-ACCESS FILES

There are no rigid rules governing the organization of files processed by the direct-access method. The programmer is free to do anything he wishes within the power of the macro instructions provided. There are, however, certain standard, typical file organizations and processing techniques for which the obvious direct-access method possibilities can be outlined. The following sections are intended to suggest to the reader possible ways in which he can use direct access routines for his more typical files. The techniques shown are not the only ones that can be used. In cases where more involved procedures are required, such as linking together multiple files, master and trailer records, etc., the user may have to use variations on these techniques to achieve the desired results.

Sequential Files

FIXED-LENGTH RECORDS, WITH KEYS: To load the file - Either of the following techniques can be used to produce identical files.

1. a. Preformat the area

   b. WRITE filename,ID

   Note: DTFDA macro instruction should request the system to return the next ID for each successive WRITE.

2. WRITE filename,AFTER

To process the file - Use the first of the following techniques to process the entire file. Use the second technique to process selected records (against sequenced input).

1. READ filename,ID
   WRITE filename,ID (if updating)

   Note: DTFDA macro instruction should request the system to return the next ID for each successive READ.

2. READ filename,KEY
   WRITE filename,KEY (if updating)

   Note: DTFDA macro instruction should request the system to return the next ID to be used as the starting point for each successive READ operation.

FIXED-LENGTH RECORDS, WITHOUT KEYS: To load the file - Fixed-length records with no keys can be created and processed with the GET-PUT level of the sequential-access method. If any of the direct-access techniques shown for Fixed-Length Records, With Keys, is used, the output will be identical to that produced by the sequential-access routine, providing the user has included the WRITE EOF options during creation of the files.

To process the file - Use the sequential-access GET or the direct-access READ Filename,ID. Files should be processed by the access method used to create them.

UNDEFINED RECORDS, WITH KEYS: To Load the File -

1. a. Use clear or initialize utility program or the macro WRITE filename,RZERO to initialize the capacity record.

   b. WRITE filename,AFTER

   Note: Use track address of first track, update to next track or track and cylinder when system indicates no more room on a track.

To process the file - Use the same techniques as listed for Fixed-Length Records, with Keys.

UNDEFINED RECORDS WITHOUT KEYS: To load the file - Undefined records with no keys can be created and processed with the GET-PUT level of the sequential-access method. The direct-access method can be used to create a file identical to that produced by the sequential-access method. Use the same techniques shown for Undefined Records with Keys.

To process the file - Use the sequential-access GET or the direct-access READ filename,ID. Files should be processed by the access method used to create them.

Random Files

FIXED-LENGTH RECORDS, WITH KEYS: To load the file - Either of the following techniques can be used.

1. a. Preformat the area.

   b. Randomize to track and record, leaving one or more tracks on each cylinder for overflow.

   c. READ filename,ID to determine whether record position is occupied.

   d. If record position available, WRITE filename,ID.

   e. If record position not available, place record in overflow area, using WRITE filename,AFTER.

2. a. Use clear or initialize utility program to set up the capacity records.

   b. Randomize to track (or, possibly, only to cylinder).

   c. WRITE filename,AFTER.

   d. If record not written (no room on track) re-issue WRITE for next track or overflow area.

To process the file - Use the same randomizing formula used to create the file.

1. If randomizing to track and record:
   READ filename,ID
   WRITE filename,ID (update)

2. If randomizing to track (or cylinder):
   READ filename,KEY
   WRITE filename,KEY (update)

FIXED-LENGTH RECORDS, WITHOUT KEYS:

To load the file - Use first technique listed for Fixed-Length Records, with Keys.

To process the file - Use first technique listed for Fixed-Length Records, with Keys.

UNDEFINED RECORDS, WITH KEYS: To load the file:

1. a. Use clear or initialize utility program to set up the capacity records.

   b. Randomize to track (or cylinder)

   c. WRITE filename,AFTER

   d. If record not written (no room on track) re-issue WRITE for next track or overflow area.

To process the file:

1. a. Randomize to track (or cylinder)

   b. READ filename,KEY

UNDEFINED RECORDS, WITHOUT KEYS: This is normally feasible. File would have to be

written using the ID of the previous record for reference (AFTER or AFTERID). It would have to be read for processing with the READ filename,ID. The ID field, however, would not normally be available to the processing program.

DIRECT-ACCESS STORAGE DEVICES

IBM 2311 DISK STORAGE DRIVE

The IBM 2311 Disk-Storage Drive features removable, interchangeable disk packs, offering virtually unlimited data-storage capacity. A disk pack can be easily removed and replaced with another pack in less than a minute. These units have flexibility comparable to a tape system, plus the advantage of direct-access processing.

Cylinder Concept

The corresponding recording tracks on each disk surface are physically located one above the other, and may be pictured as forming 203 concentric cylinders of 10 data tracks each. Figure 24 is a schematic representation of the cylinder concept.

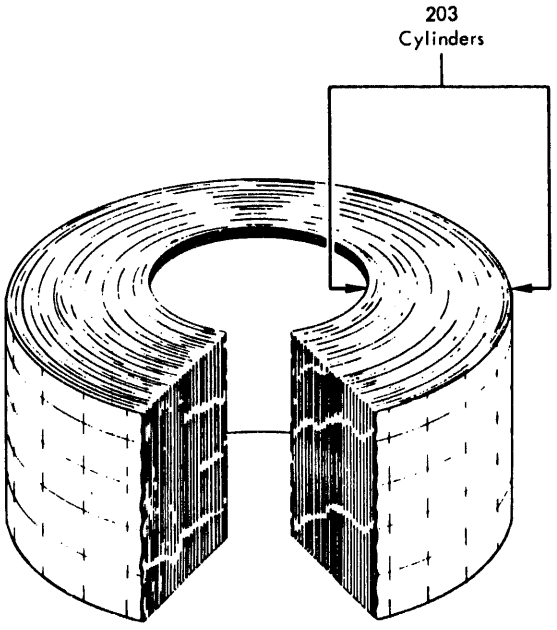


Figure 24. Cylinder Concept

Access Mechanism

Ten read/write heads are mounted on a vertical assembly. The heads are aligned vertically and are all moved together horizontally to any of 203 positions. Therefore, each time the read/write heads are moved into position, one entire cylinder of ten data tracks is accessible for reading and writing. Only electronic switching of the heads is necessary to select a particular track within the cylinder. Figure 25 illustrates the access mechanism and the disk pack.

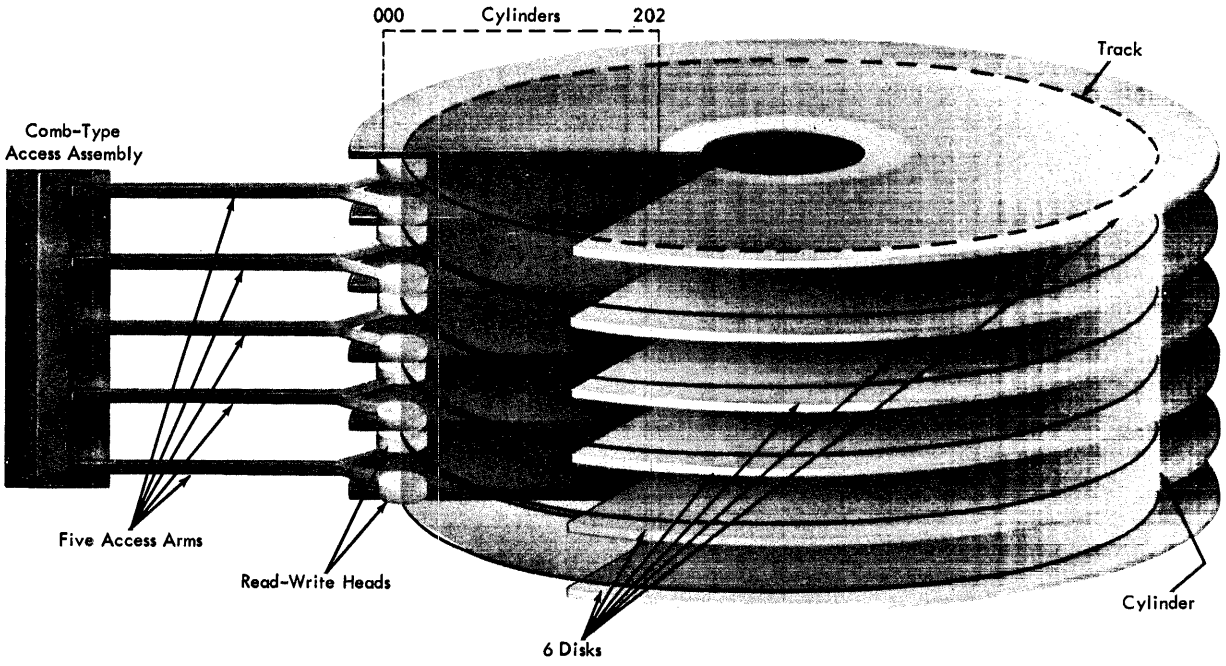


Figure 25. IBM 2311 Access Assembly and Disk Pack

	Per Track	Per Cylinder	Per Disk Storage Drive	Per Storage Control Unit
Disk Storage Drives				8
Cylinders			200	1,600
Tracks		10	2,000	16,000
Bytes (Alphameric Characters)	3,625	36,250	7,250,000	58,000,000
Packed Decimal Digits (Numeric Only)	7,250	72,500	14,500,000	116,000,000

NOTE: All figures are based on one record per track.

Figure 26. 2311 Disk Storage Drive Capacity

Storage Capacity

The 7.25 million byte capacity of each disk pack is based on 200 tracks per disk surface. With the high-density recording of the 2311, minute contamination particles can affect data reading and writing. Therefore, 203 tracks per disk surface are provided to ensure that the stated capacity is maintained for the life of the disk pack.

Because each record has certain non-data characters, such as disk addresses, the net data-storage capacity of tracks may vary.

Figure 26 indicates that on the basis of one 3625-byte record per track each disk pack can store 7,250,000 bytes. Figure 27 shows the number of bytes per record according to the number of equal-length records per track. The figure is used only for illustration; in actual practice records on the same track can vary in length in both the key areas and the data areas. Formulas for determining record capacity per track are available in the publication, IBM 2841 Storage Control Unit, Form A24-3254.

	Number of Equal-Length Records per Track																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Maximum Number of Bytes per Record without Key Field	3625	1740	1131	830	651	532	447	384	334	295	263	236	213	193	177	162	149	138	127	118
Maximum Number of Bytes per Record with Key Field	3605	1720	1111	811	632	512	428	364	315	275	244	217	194	174	158	143	130	119	108	99

Figure 27. Bytes per Record vs Records per Track (2311)

	Per Track	Per Cylinder	Per Disk Storage Module	Per IBM 2314
Disk Storage Drives				8
Cylinders			200	1,600
Tracks		20	4,000	32,000
Bytes (Alphameric Characters)	7,294	145,880	29,176,000	233,409,000
Packed Decimal Digits (Numeric Only)	14,588	291,760	58,352,000	466,816,000

NOTE: All figures are based on one record per track.

● Figure 28. IBM 2314 Storage Capacity

IBM 2314 DISK ACCESS STORAGE FACILITY

The IBM 2314 Direct Access Storage Facility consists of eight on-line disk storage modules, one spare (off-line) module, and an integral control unit. The spare module is available for immediate use if servicing or routine maintenance is necessary on any of the other modules. Of the total of nine modules, any eight can be on line at a time. Each disk storage module has an individually addressable access mechanism, and uses a removable and interchangeable IBM 2316 Disk Pack to provide programming flexibility and virtually unlimited offline storage capacity.

Access Mechanism and Disk Organization

Information is written on and read from disk surfaces of the 2316 disk packs by read/write heads in the 2314. The 20 read/write heads at each disk-module location are positioned by a movable comblike access mechanism. Each disk module has its own access mechanism. Two read/write heads are attached to each arm of an access mechanism. Because all 20 read/write heads are always positioned in the same vertical plane, 20 tracks are available for reading or writing purposes without movement of the access mechanism. Figure 24 shows how the entire disk pack consists of 203 concentric cylinders of information. The numbering is from 000 (outermost cylinder) to 202 (innermost cylinder). Tracks in cylinders 200 through 202 are alternate tracks that can be used if any of the tracks in cylinders 000 through 199 should become defective.

Storage and Record Capacity

Because each record has certain nondata areas (such as count fields and gaps), the net data storage capacity of tracks varies with the number of records. With one record per track, each track has a data capacity of 7294 bytes.

Because of the high-density recording method used by the 2314, minute contamination particles can affect data reading and writing and may cause loss of bits. Therefore, 203 tracks per disk surface are pro-

Storage Device	Track Capacity in Bytes When R_0 is Used as Specified By IBM Programming Systems	Bytes Per Data Record			
		Data Records (except for last record)		Last Record	
		Without Key	With Key	Without Key	With Key
2314	7294	$101 + \frac{2137}{2048}(D_L)$	$146 + \frac{2137}{2048}(K_L + D_L)$	D_L	$45 + K_L + D_L$

Record R_0 used as specified by IBM Programming Systems. No application data; $K_L = 0$; $D_L = 8$	Number of Equal Length Records That Can Be Stored on a 2314 Track																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Bytes per record without key	7294	3520	2298	1693	1332	1092	921	793	694	615	550	496	450	411	377	347	321	298	276	258
Bytes per record with key	7249	3476	2254	1649	1288	1049	877	750	650	571	506	452	407	368	333	304	277	254	233	215

R_0 = Record 0 (track descriptor record)

D_L = Data Length

K_L = Key Length

● Figure 29. Record Capacities

vided to ensure that a capacity of 29.17 million bytes (based on 200 tracks) is maintained for the life of the disk pack (Figure 28).

Record capacities for the 2314 are listed for records with keys and without keys in Figure 29. Further information on the 2314 is available in the publication, IBM System/360 Component Descriptions - 2314 Direct Access Storage Facility and 2844 Auxiliary Storage Control, Form A26-3599.

IBM 2321 DATA CELL DRIVE

The IBM 2321 Data Cell Drive is a device for storing data on magnetically coated strips. Two hundred strips are contained in a single removable and interchangeable cell assembly (see Figure 30). Ten cell assemblies, each containing twenty subcells, can be mounted on a data cell drive at one time. A rotary positioning system positions a selected subcell of ten strips beneath an access station. At this station, a selected strip is withdrawn from the subcell and rotated past a read/write head element for data transfer. The strip is then returned to its original location in the subcell.

Read/Write Head Block

The read/write head block contains 20 magnetic elements. It can be positioned to one of five positions, creating five cylinders of 20 data tracks each and providing read/write access for the 100 recording tracks per strip. The head block is positioned during access time.

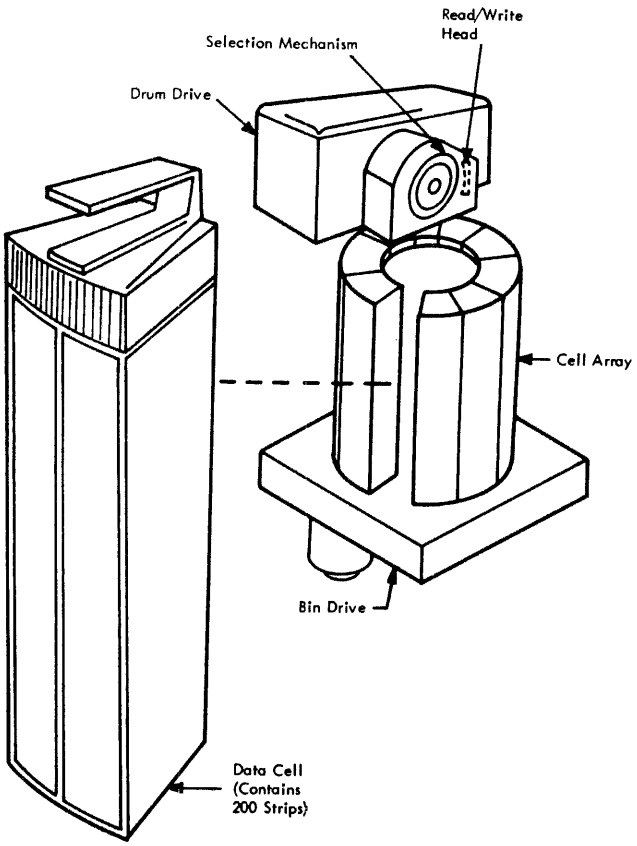


Figure 30. Data Cell and Strip Selection (2321)

Storage Capacity

Figure 31 illustrates the capacity of a 2321 data cell, based upon 2,000 bytes per track. Formulas for determining record capacity per track are available in IBM 2841 Storage Control Unit, Form A24-3254.

	MODEL 1 Capacity	
	8-Bit Alpha-numeric Mode	Packed Decimal Digit Mode
Track	2,000	4,000
Cylinder	40,000	80,000
Strip	200,000	400,000
Subcell	2,000,000	4,000,000
Data Cell	40,000,000	80,000,000
Full Array	400,000,000	800,000,000

Figure 31. 2321 Data Cell Capacity

DASD TRACK FORMAT

The 2311 Disk Storage Drive and the 2321 Data Cell Drive use a track format consisting of an index marker, a home address, and one or more data records. An address marker precedes each data record (except the first one, as explained later) to indicate the beginning of a new record. Each track area is separated by a gap. Figure 32 is a schematic representation of a DASD track. The various areas of the track format are described as follows:

Index Marker: The index marker indicates the physical beginning of each track. There is one index marker per track.

Home Address: There is one home address per track. This address is 7 bytes in length and is recorded in binary code. The home address defines the location of the track in terms of the physical parameters of the files. Home addresses are written on the track by an initializing utility program, which will be explained later.

Figure 33 is a schematic representation of the home-address area.

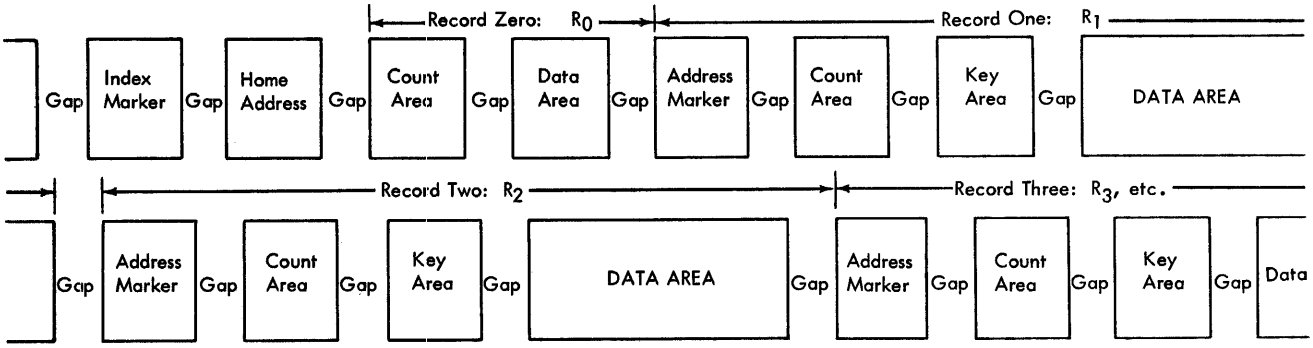


Figure 32. Schematic Representation of DASD Track (with Key Area)

The Flag Byte: The flag byte is recorded on the track during a write-home-address operation. This flag information byte indicates the condition of the track and is automatically propagated to all records as they are recorded on the track.

The Address: The four bytes containing the cylinder number and head number given the physical location of the track.

The Check Bytes: Two check bytes contain the 16 check bits used to verify the validity of reading and writing. These check bytes are a function of the record-verification circuits of the systems. They are automatically appended to each separate area written on a DASD track. The two bytes are not included in the count field's key length and data length when the length of the corresponding areas are defined.

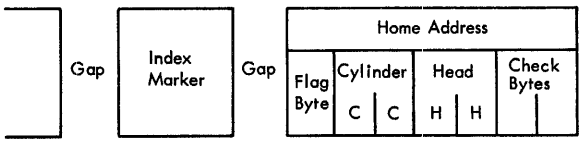


Figure 33. Schematic Representation of the Home Address

Record Zero (R0)

The first record on every track (record number zero: R0) is used primarily to facilitate the use of an alternate track, when the original track is found to be defective. Referred to as the track-descriptor record or record zero (R0), this record is unique in that it is not preceded by an address marker and does not contain a key area (key length is always zero). Figure 34 is a schematic representation of the Record Zero.

The count area is similar to other records as described in the following paragraph. The data area can be used to maintain updated information about the data records on the track. A discussion of the capacity-record portion of this data area is contained in the section Direct-Access Method.

DASD RECORD FORMAT

There are three basic parts to each DASD record: the count area, the key area (optional), and the data area. Figures 35 and 36 are schematic representations of a record with a key area and without a key area.

The Count Area: The count area consists of the flag byte recorded from the home address flag byte, the identifier field, the key length, the data length, and check bytes. This area is recorded in binary notation.

The Identifier Field: The identifier field (record ID) of five bytes contains the cylinder number, the head number and the record number to define the physical location of the record. The record number is the sequential position of the record on the track. Record number zero (R0) is the first record on the track and each succeeding record is numbered in ascending order.

The Key Length: The key length is one byte and denotes the number of bytes in the key portion of the record,

but does not include the two check bytes. If the record does not contain a key area, key length is recorded as zero.

The Data Length: The data length is two bytes long and specifies the number of bytes in the data portion of the record, but does not include the two check bytes.

The Key Area: The key area consists of the key field and two check bytes. The key is an external number, such as part number or employee number, that identifies the information stored in the data portion of the record. This field will usually be the major control field of the logical data record to which it is appended. The key length can vary from zero to a maximum of 255 bytes.

The Data Area: The data area contains the information stored in the file, plus two check bytes. The data area can be one logical record or many logical records blocked together. Records on the same track can vary in length in the key area and in the data areas. The length of the data area is defined by the data-length field.

DASD INITIALIZATION AND MAINTENANCE

INITIALIZE DISK/DATA CELL PROGRAMS

Initialize-disk and initialize-data-cell utility programs are provided as part of the IBM System/360 Disk Operating System package. Initialization programs are provided to prepare disk packs and data cells to be used. As the packs and cells are received by an installation, the programs are used to write standard home addresses and track description records (record zero) and to make a surface analysis to identify defective recording surfaces (if any). The program will write the volume label and

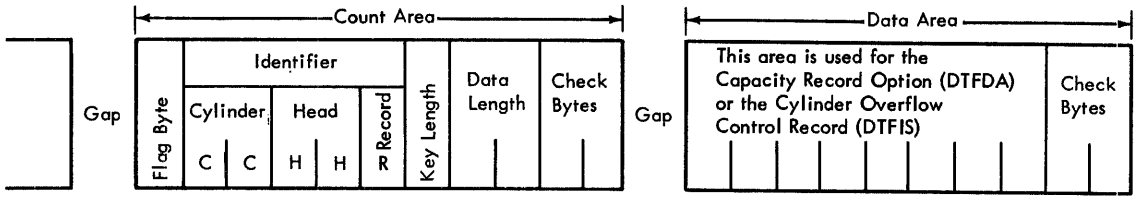


Figure 34. Schematic Representation of Record Zero

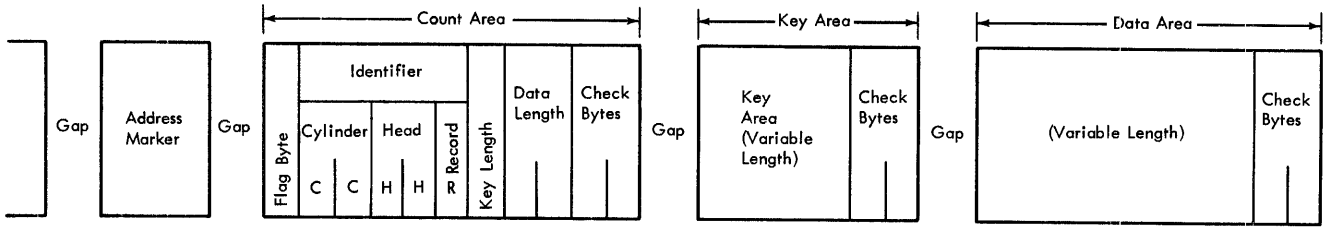


Figure 35. Schematic Representation of a DASD Record with a Key Area

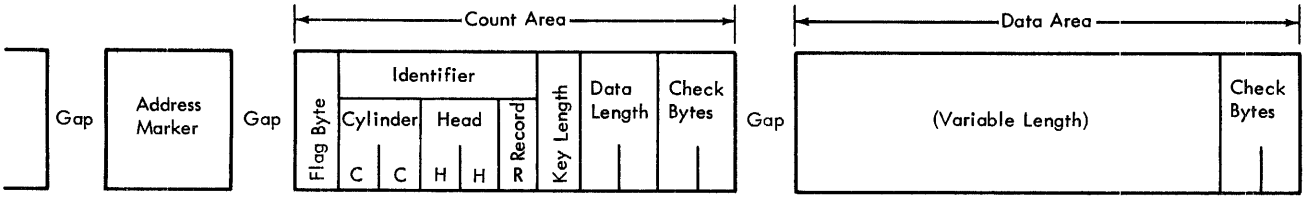


Figure 36. Schematic Representation of a DASD Record Without a Key Area

establish the volume table of contents (VTOC) area to be used for cataloging file labels. For further details on the VTOC see the section Labels.

The initialization programs are also used to reinitialize disk packs or cells when job requirements change. To guard against accidental reinitialization of devices containing valid data fields, the VTOC is checked for labels reflecting unexpired data files.

Home Address and Record Zero Generation

When the initialization programs write the home address and record zero fields on each track, a verification of the accuracy of recording is made. If a home address and record zero (R0) cannot be successfully written on a track, a message will be printed indicating the error. The pack/cell will be deleted from the job if a home address and/or record zero cannot be written on one or more tracks. The remainder of the storage area will be analyzed before the device is deleted. The program will continue processing the next device, if present.

PHYSICAL IOCS AND DEFECTIVE DASD TRACKS

Discovery of a Defective Track

Whenever a record cannot be successfully read from a DASD track with standard error recovery procedures, a message indicating the error condition and the address of the track which failed is issued to the operator. If the error condition is not acceptable to the program (processing cannot continue), the program is terminated. DASD IOCS provides the facility, in the Verify Option, to read each record as it is written to verify that it was written correctly. The user must request the Verify Option by specifying VERIFY=YES in his DTF entry for the file. The CCW Skip bit provides this verification capability without requiring an I/O area for the record. If the Verify procedure is not used, write data checks will not be discovered until the record is later read for processing.

Correction of a Defective Track

The operator should note the address of the defective track and later process the IBM System/360 DOS Alternate Track Assignment program which flags the defective tracks and assigns alternate tracks to replace them. A portion of each DASD volume is reserved for alternate tracks, i.e., the last three cylinders of each disk and the last four strips of each cell.

Previously Established Defective Tracks

Whenever a DASD device detects that a flagged defective track is being accessed (every count field on the track contains this flag), it indicates the Track Condition Check error condition. When this condition occurs, physical IOCS error recovery procedures retrieve the address of the assigned alternate track from record zero (R0) of the defective track and restart the channel program on the assigned alternate track.

Whenever a channel program operating in multiple track mode reaches the end of an alternate track, the DASD device detects this and indicates the Track Condition Check error condition. Physical IOCS retrieves the address of the defective track from record zero (R0) of the alternate track and restarts the channel program on the track following the defective track. The problem program is therefore never concerned with flagged defective tracks or alternate tracks.

DASD LABELS

The IBM System/360 Disk Operating System provides positive identification and protection of all DASD files by recording labels on each volume (pack or cell). These labels ensure that the correct volume is used for input and that no current information is destroyed on output.

Certain standard labels are required for all DASD files, although it is possible to process files with the physical I/O macro instructions (EXCP) without processing any labels. When using any of the logical IOCS routines, labels must be processed for each data file. A special IOCS facility (DTFPH) is also provided to allow label processing when using physical I/O macro instructions. In addition to the required standard labels, logical IOCS provides facilities for optional user labels.

The standard labels include one volume label for each volume and one or more file labels for each logical file on the volume. There may be user volume labels and, in some cases, user header labels and user trailer labels.

Note: User file labels are processed by IBM System/360 Operating System.

STANDARD VOLUME LABEL

The standard volume label identifies the entire volume (cell or pack). Every volume used in the Disk Operating System environment must have a standard volume label. It is always the third record on cylinder 0, track 0. The first two records on this track of every resident pack are IPL records, otherwise the records contain binary zeros. The volume-label record has a 4-byte key field and an 80-byte data field. Both the key field and the first four bytes of the data field contain the label identifier VOL1. The format of the data field is shown in Appendix A.

The volume label contains a volume serial number. This number is assigned when the volume is prepared for use in the system, and the number is never changed. It is repeated in the labels for all files on the volume.

The only other field in this label that is used by the Disk Operating System programs is the address of the area containing the file labels.

Additional Volume Labels

The standard volume label can be followed by one to seven additional volume labels (starting with the fourth record on cylinder 0, track 0). These labels must contain the label identifier VOL2, VOL3, etc. in the 4-byte key fields and in the first four bytes of the data fields. The other 76 bytes can contain whatever information the user requires. These labels are not read or processed by IOCS. If required, they must be read by using physical I/O macro instructions.

Creation of Volume Labels

All volume labels (the standard label and any additional labels) are written by the initialize-disk or initialize-data-cell utility programs at the time the pack or cell is prepared for use. The information in the standard volume label is checked, but never altered, by IOCS during file processing.

STANDARD FILE LABELS

A standard file label or set of file labels identifies a particular logical file, gives its location(s) on the pack or cell, and contains information to prevent premature destruction of current files. The indexed-sequential file management system also supplies and maintains information in the file labels to further define an indexed-sequential file. Other fields within the file labels are set aside for use by the Operating System/360 management features. The number and format of labels required for any one file depends on the file organization structure and the number of separate areas (extents) used by the file (see Standard File Label Formats).

Volume Table of Contents (VTOC)

All standard file labels are grouped together and stored in a specific area on the pack or cell. Because each file label contains file limits, the group of labels is essentially a directory of all data records on the volume. Therefore, it is called the volume table of contents (VTOC). The VTOC itself is a file of records (one or more standard label records per logical file) and is defined as such with its own file label. The label of the VTOC is the first record on the VTOC. This label identifies the file as the VTOC file and gives the file limits of the VTOC.

Preformatting the VTOC: The VTOC is preformatted by the initialize-disk or initialize-data-cell program. The user specifies its location and length when initially preparing the volume for use. It can be placed anywhere within the volume with the following restrictions:

1. For the 2311, it must be within cylinders 0-199 (cylinders 200-202 are used as the alternate track area). For the 2321, it must be within subcell 0, strip 0, cylinder 0 and subcell 19, strip 5, cylinder 4 (strips 6-9 are used as the alternate track area).
2. If it is on a system residence disk pack, it must be outside of the residence area.
3. It must be one or more full tracks, with the single exception noted in

number 5.

4. It must be contained within one cylinder. It cannot overflow onto another cylinder.
5. If it is on a disk pack that is not used for system residence, it can begin on cylinder 0, track 0, immediately following the last volume label.

The utility program used for preformatting writes the foundation records that are to be filled in later. Each record location is written with a 44-byte key field and a 96-byte data field. Both these fields in each record are filled with binary zeros. The initializing program then writes the label for the VTOC itself in the first record location. This label is described as Format 4 under Standard File Label Formats.

The second record in the VTOC is also reserved at this time by inserting a hexadecimal 05 in each of the first four bytes of the key field and a EBCDIC value 5 in the first byte of the data field. This label is used by the Direct-Access Device Space Management (DADSM) facility of Operating System/360. It is not used or maintained by the Disk Operating System programs. The label is described as Format 5 in the next section.

STANDARD FILE LABEL FORMATS

All standard file labels are written in the preformatted 140-byte records in the VTOC (44-byte key and 96-byte data). The field types contained within the labels written for data files follow three standard formats. In addition to these three formats, there are the two special labels: the one used for the VTOC itself, and the DADSM label (see Preformatting the VTOC). The format of a label is identified by the value in the first byte of the data field. This is an EBCDIC value from 1 to 5 indicating label format 1 to 5.

Format 1: This format is used for all logical files. It is always the first of the series of labels when a file requires more than one label (as discussed in formats 2 and 3).

The format-1 label identifies the logical file (by a file name assigned by the user and included in the 44-byte key area), and it contains file and data-record specifications. It also provides the addresses for three separate areas (extents) for the file. If the file is scattered over more than three separate areas, a format-3 label is also required. In this case, the format-1 label points to the second label set up for the file on this volume.

If a logical file is recorded on more than one volume, a format-1 label is always in the VTOC for each volume. The format-1 label is illustrated in Appendix B.

Format 2: This format is required for any file that is organized by the indexed-sequential file management system. The 44-byte key area is not used by the Disk Operating System programs. Operating System/360 uses this area to describe the second-level and third-level master indexes. The 96-byte data area contains additional specifications unique to this file organization (such as the track reserved for indexes).

If an indexed-sequential file is recorded on two or more volumes, the format-2 label is used on the first volume only. It is not repeated on the additional volumes (as the format-1 label is). The format-2 label is illustrated in Appendix C.

Format 3: If a logical file uses more than three extents of any volume, this format is used to specify the addresses of the additional extents. It is used only for extent information, and the entire 140 bytes provide for as many as 13 extents)

The format-3 label is pointed to by the format-1 label for the logical file or by a preceding format-3 label. It is included as required on the first volume or any additional volumes if the logical file is recorded on two or more volumes. The format-3 label is illustrated in Appendix D.

Format 4: The format-4 label is used to define the VTOC itself. This is always the first label in the VTOC. This label is also used to provide the location and number of available tracks in the alternate track area. The format-4 label is illustrated in Appendix E.

Format 5: This label is used by the System/360 Operating System for Direct-Access Device Space Management (DADSM). The Disk Operating System programs do not use or maintain this label, although it is reserved by the initialize-disk and initialize-data-cell utility programs. The format-5 label is illustrated in Appendix F.

DASD USER HEADER AND TRAILER LABELS

The user can include additional labels to further define his file, if he desires, provided the file is processed by the routines of either the sequential or the direct-access methods or with the physical

I/O macro instructions (DTFPH). The DTFSD and DTFSR routines allow as many as eight user header labels and as many as eight user trailer labels. The DTFDA and DTFPH routines allow as many as eight user header labels, but no user-trailer labels. The DTFIS routine (for indexed-sequential files) makes no provision for any user labels.

Note: All eight additional user labels may be used with a 2311 file. Five additional user labels are the maximum possible with a 2321 file.

User header and trailer labels are not stored in the VTOC. Instead, they are written on the first track of the first extent allocated by the user for the logical file. The user label track is defined by IOCS as a separate extent in the format-1 label for the file. If a file is written on two or more volumes, the user label track is reserved in the first extent of each.

The user header labels are read after the standard file labels are processed. As each user header label is read, the DTF routine branches to a routine supplied by the user to process the label. User trailer labels are read when DTFSD or DTFSR reaches the end of the last extent on each volume. They are furnished to the user's routine in the same way as the header labels.

All user labels must be 80 bytes long and they must contain standard information in the first four bytes. The remaining 76 bytes may contain information desired by the user.

The standard information in the first four bytes of a user header label is used as a key when reading or writing the label. The header labels are identified by UHL1, UHL2...UHL8. The trailer labels, when applicable, are identified in the key field by UTL0, UTL1...UTL7, although the first four bytes of the 80 byte labels contain UTL1, UTL2...UTL8. Each user label set (header or trailer) is terminated by an end-of-file record (a record with data length 0). For example, if a file has five header labels and four trailer labels, the user label track contains:

	Key Field
R0	Standard Information
R1	UHL1--user's 1st header label
R2	UHL2--user's 2nd header label
R3	UHL3--user's 3rd header label
R4	UHL4--user's 4th header label
R5	UHL5--user's 5th header label
R6	UHL6--end-of-file record
R7	UTL0--user's 1st trailer label
R8	UTL1--user's 2nd trailer label
R9	UTL2--user's 3rd trailer label
R10	UTL3--user's 4th trailer label
R11	UTL4--end-of-file record

When files are processed by the direct-access method, or by physical IOCS defined with the DTFPH statement MOUNTED=ALL, only user header labels can be used. In this case the user label track contains:

	Key Field
R0	Standard information
R1	UHL1--user's 1st header label
R2	UHL2--user's 2nd header label
R(n)	UHL(n)--user's nth header label, where n ≤ 8
R(n+1)	UHL(n+1)--end-of-file record
R(n+2)	UTL0--end-of-file record

DASD LABEL PROCESSING

All DASD label processing is performed by the transient label-processing routines of the supervisor. These routines use the information stored in the label information area of the resident pack. This information is supplied by the DLBL and EXTENT job control cards. A DLBL card must be supplied for each logical file, and an EXTENT card must be supplied for each extent in which the file is located.

Note: Job Control information previously supplied on VOL, DLAB and XTENT statements should now be supplied on

the simplified DLBL and EXTENT statements. However, DOS will continue to accept the information in the previous form.

The DTFSD and DTFSR routines process the labels of a sequential file (input or output) one volume at a time. For DTFSR, as each extent is checked, IOCS can pass control to a user's extent-exit routine. When the end of the last extent on a volume is reached, an automatic OPEN is issued for the next volume. The DTFDA (direct-access method) and DTFIS (indexed sequential) routines require that all volumes be on-line for the initial OPEN. DTFPH can be used to process Sequential or Direct Access files.

The actual label processing consists of the following checks:

DASD Input Files

- The file name entry is the only one required for the DLBL control card. All other entries are optional. If any optional entry is specified, it is checked as described in the following. Any entry not specified will not be checked.
- The volume serial numbers in the volume labels are compared to the file serial numbers in the EXTENT cards.
- Fields 1-3 in the format-1 label are compared to the corresponding fields in the DLBL card. Fields 4-6 are then checked.
- Each of the extent definitions in the format-1 and format-3 labels is checked against the limit fields supplied in the EXTENT cards.
- If user header labels are indicated (when using DTFSD, DTFSR, DTFPH, or DTFDA), they are read as each volume is opened. After reading each label, the OPEN routine branches to the user's label routine to perform any processing necessary.
- If user trailer labels are indicated (when using DTFSD or DTFSR), they are read after reaching the end of the last extent on each volume or an end-of-file read by logical IOCS. As with the user header labels, the trailer labels are processed by the user's routine.

DASD Output Files

- The file name entry is the only one required for the DLBL control card. All other entries are optional. If any optional entry is specified, it is checked as described in the following. For any entry not specified, IOCS fills in the required information with default options.
- The volume serial numbers in the volume labels are compared to the volume serial numbers in the EXTENT cards.
- The extent definitions in all labels in the VTOC are checked to determine if any overlap into areas defined in the EXTENT cards. If any do overlap, the expiration date is checked against the "today's date" in the communication region. If the expiration date has passed, the old labels are deleted. If not, the operator is notified of the condition.
- The new format-1 label is written with information supplied in the DLBL card. If an indexed-sequential file is being processed, the DTFIS routine supplies information for the format-2 label.
- The information in the EXTENT cards is placed in the format-1 labels and, if necessary, additional format-3 labels.
- If user header labels are indicated (when using DTFSD or DTFSR, DTFPH, or DTFDA), the user's label routine is entered to furnish the labels as each volume is opened. This can be done for as many as eight user header labels per volume. As each label is presented, IOCS writes it out on the first track of the first extent of the volume.
- If user trailer labels are indicated (when using DTFSD or DTFSR), the user's label routine is entered to furnish the labels when the end of the last extent on each volume is reached. This can be done for as many as eight user trailer labels. As each label is presented, IOCS writes it out on the first track of the first extent of the volume. The CLOSE macro instruction must be issued to create trailer labels for the last volume of a file.

TAPE LABELS

A tape file processed by the logical IOCS routines must conform to certain standards. These standards concern labels, placement of tape marks, and the grouping (or blocking) of tape records. Considerations of blocked records were discussed previously under Types of Records. This section (TAPE Labels) discusses the topics of tape labeling and the placement of tape marks (on both labeled and unlabeled files).

Tape files can be processed with or without labels. If labels are present, they are classified as either standard or nonstandard. The standard label set includes the following types of labels:

1. Standard volume label - fixed in length and format, processed by IOCS.
2. Additional volume labels - fixed in length and identifier, but not fixed format; bypassed by DOS/360 IOCS.
3. Standard file label - fixed in length and format, processed by IOCS.
4. Additional file labels - fixed in length, identifier, and format; bypassed by DOS/360 IOCS.
5. User labels - fixed in length and identifier, but not fixed format; read and written by IOCS, processed by user routine.

The additional volume labels, additional file labels and user labels are classified as part of the standard label set even though they are not fixed-format. They are, however, standard in length (80 bytes) and have standard label identifier fields. Nonstandard labels, on the other hand, are unrestricted in size, format, or identification. All these label types and the rules governing their positioning are described in the following sections.

STANDARD TAPE LABEL SET

When standard tape labels are specified in the DTFMT or DTFSR entries, the minimum set of labels (Figure 37) allowed for DOS/360 consists of:

1. One standard volume label per reel.
2. Two standard file labels for each logical file on the reel (one header label preceding the file and one trailer label following the file).

The user has the option of adding additional header and trailer labels (see example in Figure 38):

1. Up to seven additional volume labels.
2. Up to eight user header labels and up to eight user trailer labels.

Note: On 7-track tape, standard labels are written in the same density as the data on the tape. (All information on a tape reel must be written in a single density.) These standard labels are written with even parity in the translation mode.

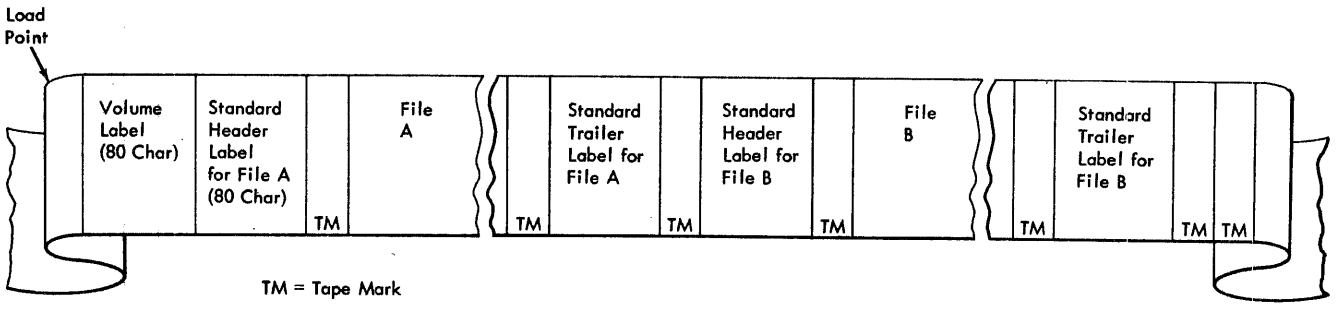


Figure 37. Tape File with Standard Labels

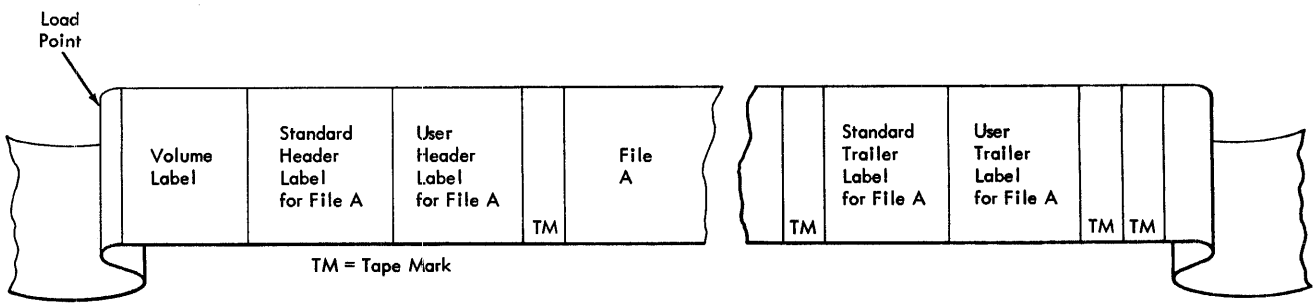


Figure 38. Tape File with Standard and User Labels

Standard Volume Label

The standard volume label identifies the entire volume (or reel). If standard labels are specified for a file, every reel used for the file must have the standard volume label. It is always the first record on the reel. It is 80 bytes long and follows a fixed format. The first four bytes contain the label identifier VOL1. The standard tape volume label is identical to the standard DASD volume label except that the DASD label contains the address of the file label area on the volume. The format of the standard volume label is shown in Appendix A.

The standard volume label contains a volume serial number. This number is assigned to the reel when it is prepared for use in the system. This number is never changed. It is repeated in the file labels for all files on the reel.

Additional Volume Labels

The standard volume label can be followed by up to seven additional volume labels. These labels are 80 bytes long and must contain the label identifier VOL2, VOL3, etc. in the first four bytes. The other 76 bytes can contain whatever information the user requires. These labels are not processed by IOCS. IOCS bypasses all additional volume labels on input files.

Creation of Volume Labels

All volume labels (the standard label and any additional labels) are written by an IBM-supplied utility program at the time a reel is prepared for use. The information in the standard volume label is checked, but never altered, during file processing. IOCS bypasses all additional volume labels when building output files. If an output tape specified to have standard labels is found to have no volume label by the OPEN routine, a diagnostic message is issued. This gives the machine operator an opportunity to supply a volume serial number so that a volume label can be written on the output tape.

STANDARD TAPE FILE LABELS

Standard file labels are written before and after every logical file on a reel. These labels are referred to as file header labels or file trailer labels, depending on their position and use. They are always 80 bytes long and always have the same format and content, with the following exceptions:

1. The label identifier field (bytes 1-3) contains:

   a. HDR to indicate a header label (precedes the data file).

   b. EOV to indicate an end-of-volume (end of reel) trailer label (written at the end of a reel, indicating that the file is continued on another reel).

   c. EOF to indicate an end-of-file trailer label (written at the end of the logical file).

2. The block count field is used only in the EOF and EOV trailer labels. This field is blank in the HDR label.

The standard tape file label is illustrated in Appendix G.

ADDITIONAL FILE LABELS

Each standard file label (one header and one trailer) can be followed by up to seven additional file labels. These labels are 80 bytes long and must contain the label identifier HDR, EOV, or EOF in the first three bytes. The fourth byte should contain a character 2, 3,...8, indicating the second, third... and up to the eighth file label. These labels are not processed by IOCS. If required, these labels must be written in the user's LABADDR routine by using physical I/O macro instructions. IOCS bypasses additional header labels on input files.

USER HEADER AND TRAILER LABELS ON TAPE

The user can include additional header and trailer labels to further define his file, if he desires. As many as eight additional header labels can be written after the standard file header label, and as many as eight additional trailer labels can be written after the standard file trailer label (EOF and EOV). Each additional label in the set is 80 characters long. The first four characters of each additional label must contain standard identifying information. The remaining 76 characters can contain any information and arrangement desired by the user. The user header labels are identified by UHL1, UHL2...UHL8 in bytes 1-4. The user trailer labels are identified by UTL1, UTL2...UTL8 in bytes 1-4.

TAPE MARKS WITH STANDARD TAPE LABELS

Figures 37 and 38 illustrate the use of tape marks with files that use the standard label sets. The sequence of items on the tape is:

1. No tape mark preceding header label set.

2. Header label set:
   - Standard volume label (required)
   - Additional volume labels (none to seven, optional)
   - Standard file header label (required)
   - Additional file labels (none to seven, optional)
   - User header labels (none to eight, optional)

3. Tape mark between header label set and first data record.

4. Physical records for file.

5. Tape mark between last data record and trailer label set.

6. Trailer label set:
   - Standard file trailer label (required at end of file and end of volume)
   - Additional file labels (none to seven, optional)
   - User trailer labels (none to eight, optional)

7. Tape mark after trailer label set.

8. If multi-file reel, (EOF label) next standard file header label follows here. If single-file reel (EOF label) or if last file of a multi-file reel, another tape mark follows here. If multi-reel file (EOV label), one tape mark follows the EOV label.

STANDARD TAPE LABEL PROCESSING

Standard tape label processing is performed by the IOCS transient routines. These routines use the information stored in the label information area of resident pack. This information is supplied by the TLBL job control cards. Note that only one TLBL card need be supplied for each logical file, regardless of the number of reels required to make up the file.

Note: Job Control information previously supplied on VOL and TPLAB statements should now be supplied on the simplified TLBL statement. However, DOS continues to accept the information in the previous form.

The actual label processing consists of the following checks:

Tape Input File

- The file name entry is the only one required for the TLBL control card. All other entries are optional. If any optional entry is specified, it is checked as described in the following. Any entry not specified will not be checked.

- The volume serial number in the standard volume label on the first or only reel is compared to the file serial number in the TLBL card. All other volume labels on all reels of the file are bypassed.

- The contents of the TLBL card are compared to the corresponding fields in the standard file header label on the first reel. For successive reels of a multi-reel file, the volume sequence number is increased by one for each reel.

- If user labels are indicated, they are read into main storage by the OPEN routine for processing by the user's label routines. The user labels are read one at a time, until all have been processed.

- When a standard file trailer label is read, the block count is compared to a count accumulated by IOCS.

- If user trailer labels are indicated, they are read into main storage by the CLOSE routine for processing by the user's label routine. The user trailer labels are read one at a time until all have been processed.

Tape Output File

- The file name entry is the only one required for the TLBL control card. All other entries are optional. If any optional entry is specified, it is checked as described in the following. For any entry not specified, IOCS fills in the required information with default options.

- The volume serial number in the standard volume label on the first or only reel is compared to the file serial number in the TLBL card. All other volume labels on all reels are bypassed.

- The expiration date in the standard file header label is checked against the "today's date" in the communication region. If the expiration date has passed, the reel is backspaced to write the new standard file label. If not the operator is notified of the condition. This check is performed on each reel of a multi-reel output file. If no file label is present, the tape is considered expired.

- The new standard file label is written with the information supplied in the TLBL card. For multi-reel files, the volume sequence number is increased by 1 for each successive reel.

- If user header labels are indicated, the user's label routine is entered to furnish the labels as each reel is opened. This can be done for as many as eight user header labels per file.

- If end of reel is sensed before completing the file, an EOV trailer label is written with all fields presented in the TLBL card plus a block count.

- When end of file is reached, an EOF trailer label is written identical to the EOV label previously mentioned.

- If user trailer labels are indicated, the user's label routine is entered to furnish the labels after each trailer (EOV or EOF) label is written. This can be done for as many as eight user trailer labels)

NONSTANDARD TAPE LABELS

Any tape labels that do not conform to the standard label specifications are considered nonstandard. If they are to be read, checked, or written, it must be done by the user. On input files, the nonstandard labels may or may not be followed by a tape mark. Therefore, four conditions are possible:

1. Nonstandard label(s), followed by a tape mark, to be checked.

2. Nonstandard label(s), not followed by a tape mark, to be checked.

3. Nonstandard label(s), followed by a tape mark, which are not to be checked.

4. Nonstandard label(s), not followed by a tape mark, which are not to be checked.

For conditions 1 and 2, the DTFMT or DTFSR entries must specify nonstandard labels and the address of a user-written routine to do the reading or writing.

For condition 3, nonstandard labels must be specified, but the address of a user routine is omitted. IOCS skips all labels, passes the tape mark, and positions the tape at the first data record to be read.

For condition 4, nonstandard labels and a user address are specified. IOCS can not distinguish labels from data records because there is no tape mark to indicate the end of the labels. Therefore, to position the tape at the first data record, the user must read all labels.

With nonstandard labels when an end-of-file or an end-of-volume condition exists,

the user indicates to IOCS which condition it is. On end of file, IOCS branches to the user's end-of-file address. On end of volume, IOCS initiates the end-of-volume procedures to close the completed volume and open the next volume for processing.

On output files, nonstandard labels are written by the user's routine by using physical IOCS. The OPEN routine writes a tape mark between the user's nonstandard header labels and his first data record unless the DTF macro instruction has the entry: TPMARK=NO. The CLOSE routine writes a tape mark after the user's last data record before he writes his nonstandard trailer labels, and after the trailer labels.

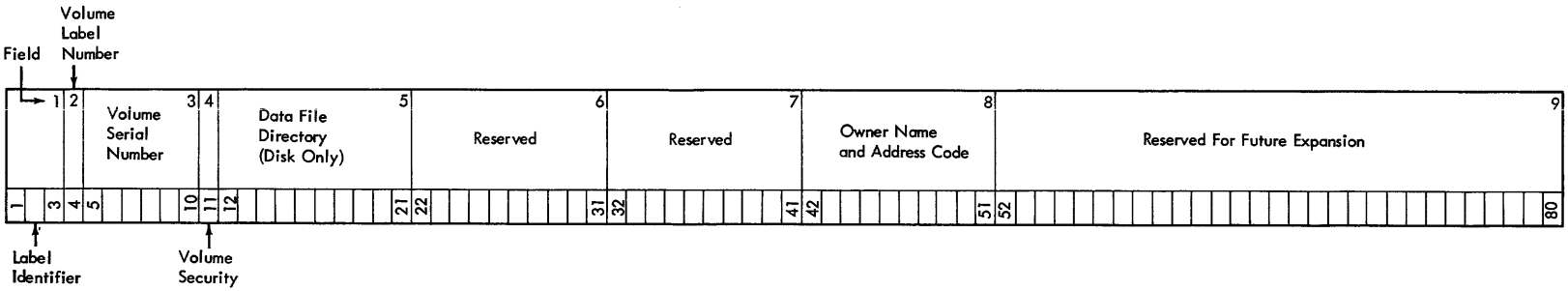
UNLABELED TAPE FILES

The DTF macro instruction specifies whether the first record of an unlabeled file is a tape mark.

Unlabeled input tape files may or may not have a tape mark as the first record. (If the first record is not a tape mark, IOCS assumes it is a data record.) Any tape that is to be read backward must have a tape mark as the first record on tape. Unlabeled output tape files (which are written by IOCS) may be written with a tape mark as the first record. This allows for the use of the read-backward feature.

Note: Seven-track tapes may be read backward only if they were written by a System/360, and they must not have been written in the conversion mode.

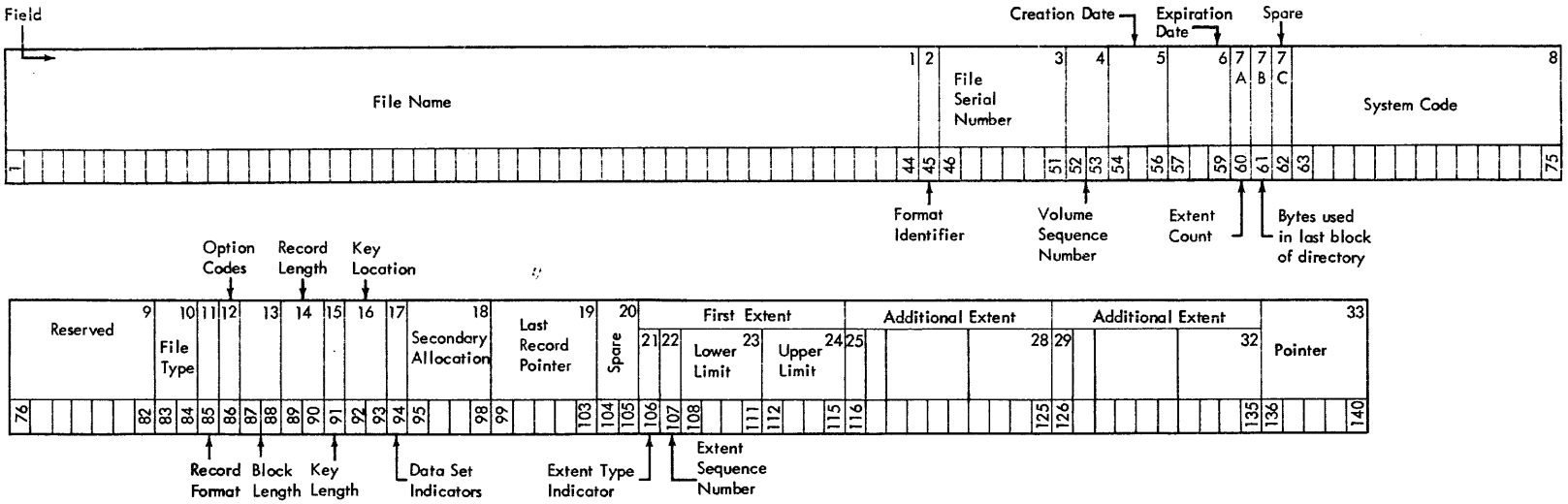
When an unlabeled output file is specified, the OPEN routine assumes the mounted scratch tape is also unlabeled. Therefore, any existing labels, including the volume label, are destroyed.



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 (OS/360 only)	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. (OS/360 only)
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 Disk Operating System programs as well as the Operating System/360 programs.



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

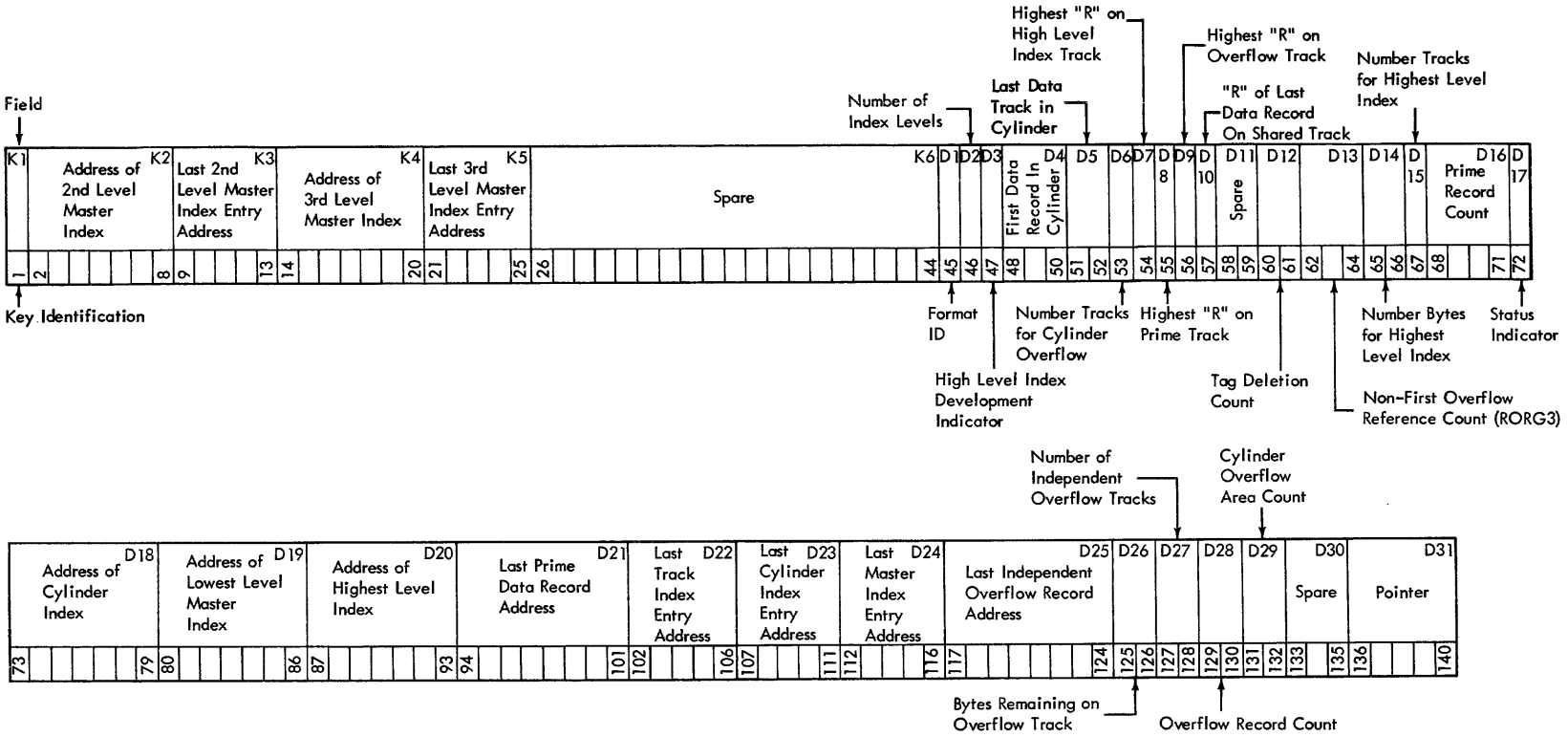
FIELD	NAME AND LENGTH	DESCRIPTION
1.	FILE NAME 44 bytes, alphameric EBCDIC	This field serves as the key portion of the file label. Each file must have a unique file name. Duplication of file names will cause retrieval errors. The file name can consist of three sections: 1. File ID is an alphameric name 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.
		Note: The Disk Operating System compares the entire 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 relative 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	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 Disk Operating System 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 the Disk Operating System.
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. (OS/360 only)
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 = 0 1 = Reserved 2 = Master index present 3 = Independent overflow present 4 = Cylinder overflow present 5 = Reserved 6 = Delete record (OS/360 only) 7 = Reorganize (OS/360 only)
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 position 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 Disk Operating System.

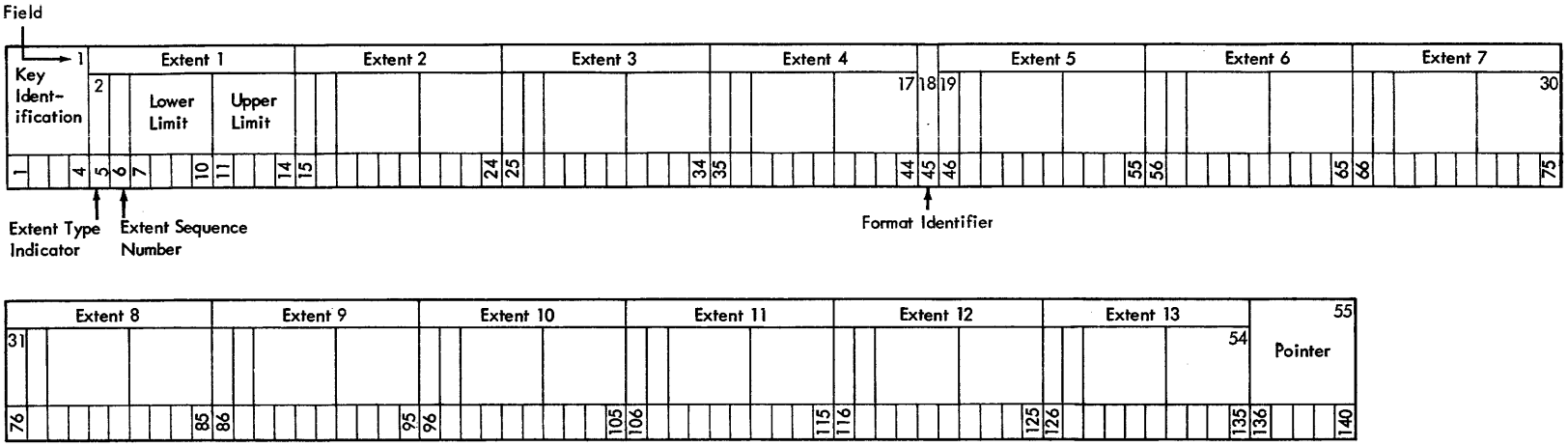
FIELD	NAME AND LENGTH	DESCRIPTION
		BIT
		1 If on, indicates that the data set described by this file must remain in the same absolute location on the direct access device. (OS/360 only)
		2 If on, indicates that Block Length must always be a multiple of 8 bytes. (OS/360 only)
		3 If on, indicates that this data file is security protected; a password must be provided in order to access it. (OS/360 only)
		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 the Disk Operating System routines. The first byte of this field is an indication of the type of allocation request. Hex code C2 (EBCDIC B) 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. (OS/360 only)
20.	SPARE 2 bytes	Reserved.
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.



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

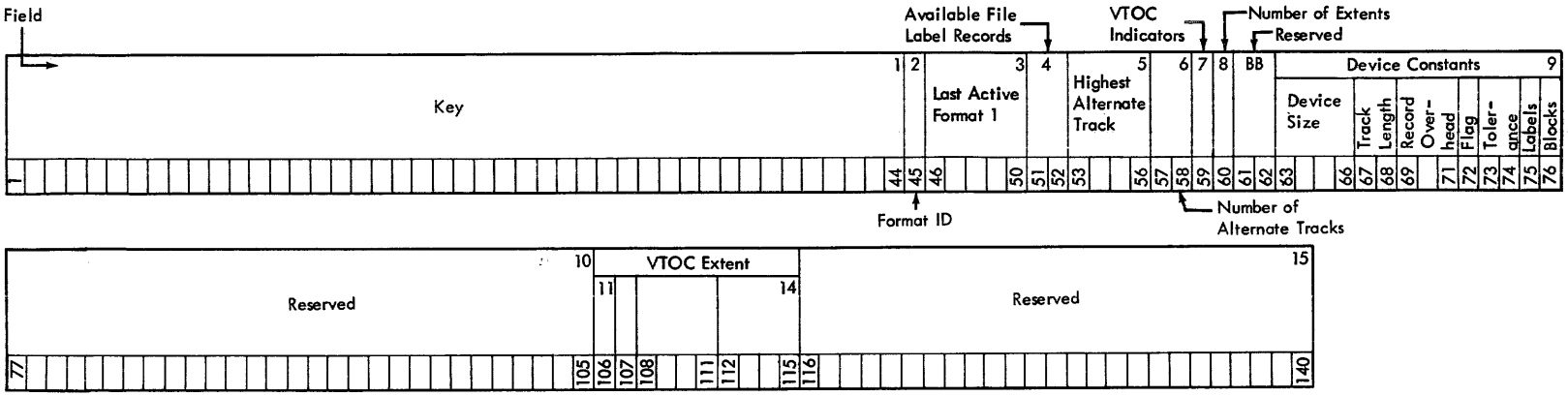
FIELD	NAME AND LENGTH	DESCRIPTION
K1	KEY IDENTIFICATION 1 byte	This byte contains the Hex Code 02 in order to avoid conflict with a file name.
K2	ADDRESS OF 2ND LEVEL MASTER INDEX 7 bytes, discontinuous binary	This field contains the address of the first track of the second level of the master index, in the form MBBCCHH. (OS/360 only)
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. (OS/360 only)
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. (OS/360 only)
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. (OS/360 only)
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. (OS/360 only)
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. (OS/360 only)
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 non-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
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. (OS/360 only)
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.
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. (OS/360 only)



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 byte	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. (OS/360 only)
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
7	2321. 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).

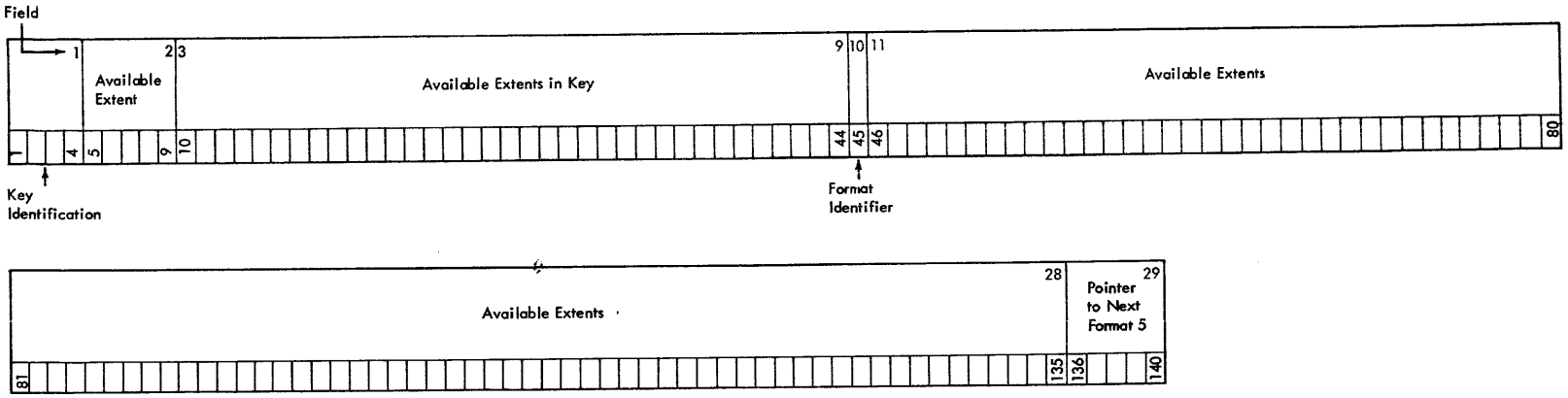
Directory Blocks/Track (1 byte) - A count of the number of directory blocks that can be written on each track for an Operating System/360 partitioned data set. (Number of full records of 8-byte key and 256-byte data lengths that can be contained on one track of this device.)

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
2314	203	20	7294	146	45	45	1	2137	25	17
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.

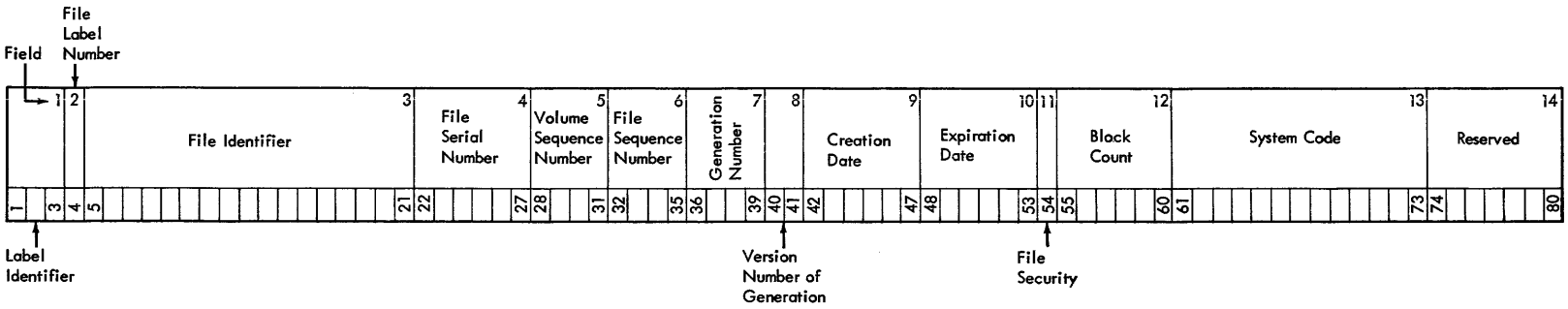
10.	RESERVED 29 bytes	Reserved.
11-14.	VTOC EXTENT	These fields describe the extent of the VTOC, and are identical in format to fields 21-24 of the Format 1 file label. Extent type 01 (prime data area).
15.	RESERVED 25 bytes	Reserved.



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.
3-9	AVAILABLE EXTENTS IN KEY 35 bytes	These fields are identical to field 2. They are in relative track address sequence.
10.	FORMAT IDENTIFIER 1 byte EBCDIC numeric	5 = Format 5
11-28	AVAILABLE EXTENTS 90 bytes	These fields are the same as Field 2. There are 26 available extent fields in the Format 5 label.
29.	POINTER TO NEXT FORMAT 5	Contains the address (in the form CCHHR) of the next Format 5 file label if one exists.

Note: Format 5 label used by OS/360 only.



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. This number must be numeric (0000-9999). Multiple volumes of an output file will be numbered in consecutive sequence.
6.	FILE SEQUENCE NUMBER 4 bytes	assigns numeric sequence to a file within a multi-file set.
7.	GENERATION NUMBER 4 bytes	numerically identifies the various editions of the file.
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.

APPENDIX H: PLANNING INFORMATION FOR FUTURE SYSTEM RELEASE

This appendix describes a future release of the Disk Operating System and is included for planning purposes only. When coding for the future release is available, this appendix will be deleted, and the information will be incorporated in the appropriate sections of the publication. A future release will provide:

- A new system of DASD addressing, called Relative Track Addressing.
- Trailer Label Processing for Direct Access Management.

RELATIVE TRACK ADDRESSING

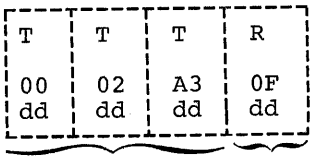
Relative Track Addressing is more convenient to use than the physical address (MBBCCHHR). In the new system, the programmer has two main advantages:

1. The data in the file appears to be one logically continuous area, although it may be physically non-contiguous.
2. The user needs to know only the relative position of the data within the file; its actual address is not required.

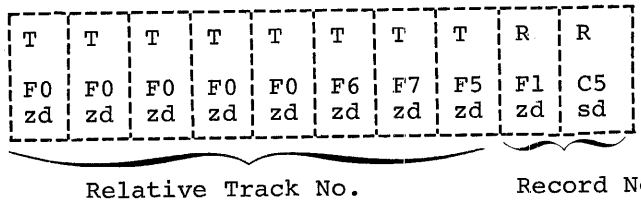
The relative address may be specified by the user in either of two formats: Hexadecimal (TTTR), or Zoned Decimal (TTTTTTTTRR). In the hexadecimal format, TTT represents the track number relative to the start of the data file, and R represents the record number on that track. In the zoned decimal format, TTTTTTTT represents the track number relative to the start of the data file, and RR represents the record number on the track.

The hexadecimal format requires 4 bytes, while the zoned decimal format requires 10 bytes, as shown in the following illustration. In this illustration the relative address of the 15th record of the 675th track of the data file is used as an example:

Hexadecimal -- 4 bytes, (X'0002A30F') in the form TTTR



Zoned Decimal -- 10 bytes, (C'0000067515') in the form TTTTTTTTRR



Specific information on the implementation of Relative Track Addressing can be found in the Supervisor and I/O Macros publication referenced on the front cover of this manual.

DASD TRAILER LABEL PROCESSING

This new DTFDA option allows user standard labels to be read or written at CLOSE time. If used, this option requires that header labels must also be read or written. Specific information on the implementation of Trailer Label Processing can be found in the Supervisor and I/O Macros publication referenced on the front cover of this manual.

INDEX

Whenever one reference has more significance than the others for an item, that page number is listed first.

IBM S/360 Printed in U.S.A. C24-3427-3

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM **Technical Newsletter**

File No. S360-30 (DOS)

Re: Form No. C24-3427-3

This Newsletter No. N24-5349

Date: May 3, 1968

Previous Newsletter Nos. None

IBM System/360
Disk Operating System
Data Management Concepts

This Technical Newsletter adds Appendix H to your publication. This appendix contains information, to be used for planning purposes only, concerning a future release of the Disk Operating System. When coding for the future release is available, this appendix will be deleted and the information will be incorporated in the proper location in the publication.

The pages attached to this Newsletter replace the following pages of your publication.

3 and 4
67 and 67.1
Blank and 68

Changes on modified pages are indicated by a vertical line to the left of the affected text and to the left of affected parts of figures. A dot (●) next to a figure title or page number indicates that the entire figure or page should be reviewed. Please insert this page to indicate that your publication now includes the following modified pages:

Modified Pages

3, 67.1, 68

IBM Corp., Programming Publications Dept., Endicott, N.Y. 13760

PRINTED IN U.S.A.

Whenever one reference has more significance than the others for an item, that page number is listed first.

Reader's Comments Form

IBM System/360
Disk Operating System
Data Management Concepts C24-3427-3

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. Please give specific page and line references with your comments when appropriate. All comments will be handled on a non-confidential basis. Copies of this and other IBM publications can be obtained through IBM branch offices.

	Yes	*No*
● Does the 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?	☐

Your comments:

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

Staple

C24-3427-3

Your comments, please . . .

This publication is one of a series that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, help us produce better publications for your use. Each reply is carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in using your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 170
ENDICOTT, N. Y.

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

POSTAGE WILL BE PAID BY . . .

IBM Corporation
P. O. Box 6
Endicott, N. Y. 13760

Attention: Programming Publications, Dept. 157

Fold Fold

Cut Along Line

IBM S/360 Printed in U. S. A. C24-3427-3

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Additional Comments: