File No. 1410/7010-34
Form C28-0405-2

IBM Systems Reference Library

IBM 1410/7010 Operating System (1410-PR-155)

File Organization System for IBM 1301/2302 Disk Storage

MI-977

The File Organization System is a set of programs designed to store and maintain organized data files on either IBM 1301 or IBM 2302 Disk Storage. The programs are capable of loading files, unloading files, reloading files, adding records, and deleting files. The File Organization System also provides the GET and PUT macro-instruction capabilities needed to process the organized data files.

The publication provides information on the functions of the File Organization System programs, the types of data records accepted as input, the control cards needed for program operation, the output produced, and the macro-instructions available for processing the organized data files.

NOTE: The IBM 1302 Disk Storage Unit is now designated the IBM 2302 Disk Storage Unit; there has been no change in the unit itself, in the applications for which the unit may be used, or in the programming parameters used to specify those applications. The IBM 2302 Disk Storage Unit designation has been used in the text of this publication; programming parameters remain unchanged and refer to 1302.

MAJOR REVISION (May 1965)

This publication is a major revision of the publication *IBM 1410/7010 Operating System; File Organization System for IBM 1301/1302 Disk Storage* Form C28-0504-1, and obsoletes that publication and its associated Technical Newsletters (N28-1190 and N28-1195). The presentation of the material on System Generation and System Definition has been changed, and further information on the incorporation of user-written routines has been added. Changes not previously published are indicated by a vertical bar at the left of the affected text, or a bullet (•) at the left of the caption of changed illustrations.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM Corporation, Programming Systems Publications, Dept. 637, Neighborhood Road, Kingston, N. Y. 12401

Contents

This publication describes the use of the File Organization System, a component of the IBM 1410/7010 Operating System. The publication contains information about the capabilities of the File Organization System; the formats of data records; the creation of organized data files; the maintenance of these files through additions, deletions, and reorganization; the processing of organized files with the user's object programs; and the creation, relocation, and execution of File Organization System programs by the user. System Generation considerations relative to the File Organization System are detailed in the publication *IBM 1410/7010 Operating System; System Generation,* Form C28-0352.

Purpose and Advantages of the File Organization System

The File Organization System is a set of IBM 1410/7010 Operating System programs designed to store and maintain data files in IBM 1301/2302 Disk Storage. The File Organization System permits object programs to address records in disk storage through record identifiers, without using disk addresses, or to retrieve data files in record sequence without the use of outside lists or "finder files." Because object programs may reference data in disk storage without using actual addresses, the programs are independent of data location and need not be altered in any way when the size of files exceeds the initial allocation in disk storage or when files are relocated in disk storage.

Some specified portion of each data record, called a record key, identifies each record to the File Organization System. This key, which is usually the part number, customer number, man number within department, etc., also determines the sequence of the data file in disk storage. Thus the key should be chosen to benefit the bulk of the data processing just as the most beneficial sequence for storing data in punched cards or on magnetic tape is chosen. The File Organization System uses an indexing convention to control the location of each record in disk storage. The indexes, controlled and maintained by the File Organization System, locate the data for random, sequential, or skip sequential processing. (Skip sequential processing enables the user to enter an organized file at any point, then process records sequentially starting at that point.) Because all data records are indexed, the absence of data records sought by a user's program can be positively determined.

In addition to the unique advantages of organized files mentioned in the preceding paragraphs, the File Organization System also offers the following advantages:

1. Random processing of individual transactions, as well as sequential processing, is facilitated.
2. Batch processing advantages are retained.
3. Scan processing and report preparation are simplified.
4. Data growth is readily accepted.
5. Disk storage is filled more efficiently.

Functions of the File Organization System

The File Organization System provides the means for performing these functions:

1. Loading data files
2. Unloading all or parts of data files
3. Adding records to existing data files
4. Deleting records or entire data files
5. Reorganizing all or parts of data files
6. Processing data records in organized files

The user directs the File Organization System to perform the first five functions (the organizing functions) by use of control cards. The sixth function is accomplished in the user-written program through the use of the extended IOCS macro-instructions provided by the File Organization System.

Prerequisites

The reader of this publication must have a basic knowledge of programming, and a familiarity with the IBM 1410 or IBM 7010 Data Processing System and the IBM 1410/7010 Operating System.

The reader should also be familiar with the information contained in the following publications:

IBM 1301/2302 Disk Storage Sequential Data Organization, Form A22-6784.

IBM 1410/7010 Operating System; Basic Concepts, Form C28-0318.

Reference will be made in this publication to other IBM 1410/7010 Operating System publications which are listed in *Basic Concepts.*

Minimum Machine Requirements

The minimum machine requirements for the generation of an operating system incorporating the File Organization System, in both tape- and disk-oriented installations, are those listed in the *System Generation*

publication. The File Organization System requires the following minimum machine configuration:

60,000 positions of core storage.
One module of IBM 1301 or IBM 2302 Disk Storage.
One magnetic tape unit (additional units are desirable).
An IBM 1402 Card Read-Punch and an IBM 1403 Printer; or one or two magnetic tape units for the corresponding input/output functions.

General Description of the File Organization System

The File Organization System consists of several relocatable programs designed to accomplish the organizing functions introduced above, extended forms of the GET and PUT macro-instructions used to process an organized file, and a System Definition program.

There are three relocatable programs, or phases, in the File Organization System that perform organizing functions. They are: Load, Add, and Unload. The Load phase contains the programming to load or delete entire data files and to reload all or specified portions of data files. The Add and Unload phases contain programming to add records and unload all or parts of files, respectively. Details on the three phases are found in the sections "Program Description" and "Program Operation."

After a data file has been organized by the File Organization System, its records are readily available for sequential or random processing by object programs that use the extended forms of the GET and PUT macro-instructions provided by the File Organization System. The section "Data File Processing" contains information on the use of the IOCS macro-instructions that may be used in working with organized files.

The System Definition program combines the relocatable programs that perform the organizing functions into a multiphase File Organization System program. A File Organization System program consists of a control phase, and from one to three organizing phases as specified by the user. The Control phase consists of a communication region and routines to analyze control cards and call other phases of the File Organization System program into core storage. Details on the System Definition program are found in the section "System Definition."

An absolute File Organization System program (on the SOF or MJB file) consists of the Control phase and one or more of the following phases: Load, Unload, and Add. The following paragraphs summarize the functions of the four phases.

The Control phase performs a monitoring function within the File Organization System, reading and interpreting all File Organization System control cards. Information specified on the control cards is placed in a reserved area of core storage, the File Organization System Communication Region. The Communication Region is resident in core storage throughout any File Organization System program run. The Control phase, complying with the user-supplied information in the Communication Region, brings the specified phase of the File Organization System program into core storage and gives program control to the operating phase. The operating phase performs its function on the data file and returns program control to the Control phase.

The Load phase provides the functions for loading files, deleting files, and reloading all or specified portions of existing files.

1. The loading process creates the organized data file and the indexes necessary to locate any record in the data file with a minimum of searching. Another function of the loading process is the creation of a File Directory entry. The File Directory contains one entry for each file in the system. Each entry contains all the information needed to describe the data file to the phases in the File Organization System and to the macro-instructions in the user's programs.

2. The deleting process invalidates the index entries for an organized file, effectively (but not physically) deleting the entire organized file by making it inaccessible by the File Organization System.

3. Reloading returns all or specified portions of an unloaded organized file to disk storage.

The Add phase updates existing data files by adding records and making the necessary modifications to the indexes and the File Directory.

The Unload phase moves all or specified portions of organized data files to tape or IBM 1301/2302 Disk Storage for the purpose of reorganization or processing.

Data File Indexes

In the loading process, the File Organization System program establishes a relationship between the record key and the disk storage location of that record. The hierarchy of file indexes, shown in Figure 1, that produces the relationship offers a convenient means for the random processing of organized files, for proper sequential processing, and for absolute data record control.

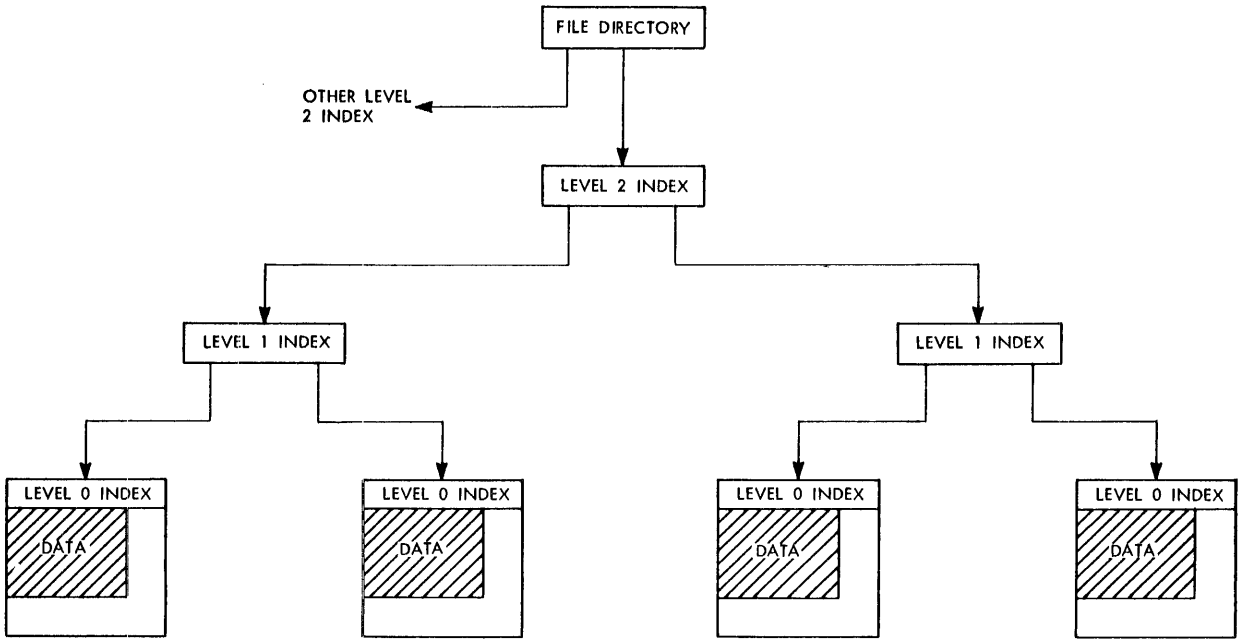


Figure 1. Index Structure

The *Level 1 Index* contains one entry for each cylinder within a data file using two or more cylinders. (No Level 1 Index is created for files that use a single cylinder.) A single entry, that of the highest key in each cylinder, identifies all the records located within a cylinder. The Level 1 Index is formed each time the data file is loaded or reloaded. It does not control the loading of data, but provides a record of where data has been placed.

Within each cylinder a more detailed index is provided containing the highest key of each track of data. This index, identifying specific tracks, is called the *Level 0 Index.* Level 0 Indexes are formed as the data files are loaded and are stored in the same cylinder as their data in order to minimize access time to the data records during processing.

With very large data files, the Level 1 Index may become too large for convenient use. In this case, a higher-level (Level 2) index is formed. It has the same format as each Level 1 Index and contains one entry for each Level 1 Index in the data file.

In addition to relating record keys to track addresses, the indexes permit efficient storage of data; empty space need not be provided for future expansion. Additional records are placed on an overflow track, but an entry is placed in the proper place in the appropriate index.

File Directory

The File Directory contains one entry for each data file in the File Organization System. Each entry describes a data file. The File Directory is on an IBM 1301/2302 Disk Storage unit and contains as many entries as there are data files using the system. The area needed for the directory is a function of the number of data files using the system. If more than one track is required for the File Directory, the tracks are chained together.

Features and Specifications

The File Organization System accepts input data in a wide variety of formats.

The input to the program may be in the form of tape, IBM 1301/2302 Disk Storage, or card records, and may be in Move mode or Load mode. Input from tape or IBM 1301/2302 Disk Storage may be blocked or unblocked fixed-length records. Tape input may be in odd or even parity. The devices containing the input data may be on any channel.

The data files may be loaded into one or more consecutive or nonconsecutive cylinders in any module of IBM 1301/2302 Disk Storage. If the file is loaded into more than one module, the disk units containing the modules must be of the same type, that is, either 1301 or 2302, but not a combination.

The disk area may be loaded with either one record per track or several records per track up to track length. The data records on the organized disk may be blocked or unblocked. The blocking factor of the disk records need not be the same as the blocking factor of the input source.

In addition to the organized data files created by the File Organization System program, two other forms of output are produced, the Exception File and the Result File.

The Exception File, a tape file assigned by the user, contains records that have been selected for removal from the data file for special processing. Records that have duplicate keys or are out of sequence are placed on the Exception File by the File Organization System program. The records written on the file are unblocked. A count of the number of records written in the file is kept, and is later written on the Result File.

The Result File is a listing printed on the SPR after an organizing function is performed. Information is printed as to the number and disposition of records handled, and (where applicable) the status of the file's data areas and index levels.

File Organization System Input/Output flow is shown in Figure 2. The user specifies certain informa-

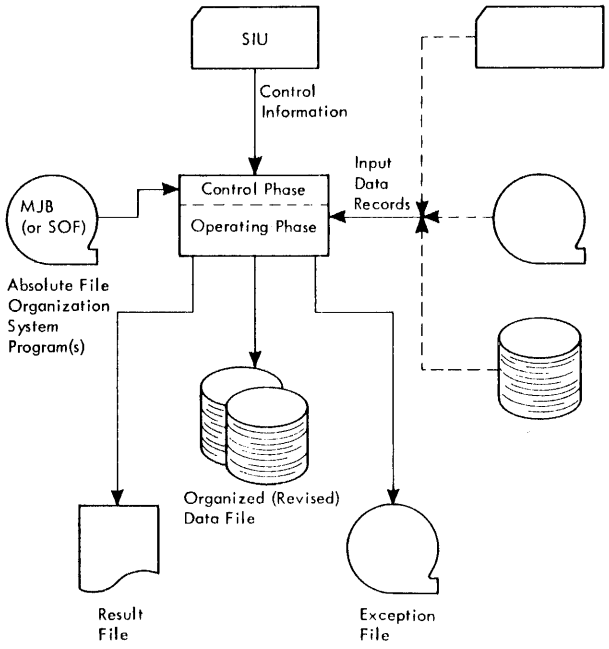


Figure 2. File Organization System Input/Output Flow

tion concerning the files on control cards which are discussed in the Control Card section of "Program Operation."

Data Records

The File Organization System requires data records to be stored in IBM 1301/2302 Disk Storage as fixed-length records. The user must determine if the records are to be unblocked (Form 1 records) or blocked (Form 2 records). The first character of a record may not be a group mark or a segment mark.

Record Key

A *record key* identifies each data record in the file to the File Organization System. The key is specified by the user as a certain portion of each record. The portion of the data record chosen as the key characterizes that record as being different from every other record on the file. The uniqueness of each key is determined by its contents, because the key to every record must be located in the same area of each record.

Each record key can consist of one to 100 characters of the record. The characters need not be consecutive, but can be formed with a maximum of ten segments of varying length.

NOTE: Records with keys consisting entirely of 9s are written on the Exception File. Keys should be selected so that no valid record key will contain only 9s.

The record format in Figure 3 is an example of a record key that a user may specify for a particular data file. Each record can be identified by the character in position 79, the first 5 characters, and the 11 characters starting in position 36. Therefore, the record key is composed of 17 characters found in three segments of the record in the specified sequence.

The user may specify the segments of the record key in any order, independent of the relative positions of the segments in the data record. The order in which the segments are specified indicates the relative importance of each segment, i.e., the first segment in the key is the most significant; the last segment is the least significant.

2nd segment		3rd segment		1st segment
1	5	36	46	79
XXXXX		XXXXXXXXXXX		X

Figure 3. Sample Record Key

Delete Code

The user may specify via a control card one position in each record as a delete code position. The position may *not* be part of the record key. The character in this position may be used to mark the record for removal from the data file. If the user specifies a delete code position, he must also specify in the control card two characters as delete codes. One of these codes indicates to the File Organization System that the record is to be removed from the data file and placed on the Exception File. The other code indicates to the File Organization System that the record is to be removed from the data file and skipped. Any other character in the delete code position indicates that the record is still part of the data file. See the paragraph "Deleting Records" for a discussion of the action taken by the File Organization System at various stages in the organizing and processing of data files if a record contains one of the delete codes.

Figure 4 shows four 81-character records. Assume that the key is positions 1-6, the delete code is position 80, and the delete codes are "$" (skip) and "%" (place on the Exception File). The File Organization System interprets the characters in the delete code position as follows:

1. Record 1 is to be removed from the file.
2. Records 2 and 3 are to remain in the file.
3. Record 4 is to be removed from the file and placed on the Exception File.

Record Address

When the records are placed in IBM 1301/2302 Disk Storage, the File Organization System creates a record address for each block of data records on a track.

For blocked data records (Form 2), a geometric record address is automatically created.

For unblocked data records (Form 1), the user may specify the method of creating the record address. The first method or option available to the user is to choose

1 81

2S0713	SMITH	AR	1	10500 FORTUNA DR	NEWTON	4	$	+
2S0714	SMITH	HL	1	271 HILLSBORO AV	SALEM	4	4700	+
2S0715	SMITH	TA	2	695 SUTTON PL APT 2A	SALEM	4	12145	+
2S0719	SMITH	WE	1	17111 NATIONAL BLVD APT 27	SALEM	4	%	+

Figure 4. Data Records with Delete Codes

any six consecutive characters of the record key to serve as the record address for that record. If a specification is not made, this option is assumed and the low-order six characters of the key constitute the record address. The entire key is used, right-justified and preceded by zeros, if the key is less than six characters.

The user may specify an optional method if he does not want the key used as the record address. The program then generates a six-character record address by dividing the entire record key by 999983. The six-digit remainder is the record address.

IBM 1301/2302 Disk Storage Allocation

Two areas of disk storage must be considered by the user: the area reserved for data records and the area reserved for the File Directory and Indexes.

Data Record Area

The data record area contains the data records and the Level 0 Indexes. The File Organization System assumes that all data record areas contain two-character HA2's and six-character record addresses. The user may specify the mode of operation for a data file as either Move mode or Load mode. He may also format a track of the file to contain several records, or an entire track to contain just one record; however, all tracks on one data file must have the same format. A *Record Load* factor allows the user to specify how much of the available area on each track is to be filled during the initial loading process. The unused record area may be used to accommodate records added to the file. Thus, the added records are in their proper location and access time to these records is no greater than the access time for records placed on the file at initial load.

The File Organization System allows the user to specify any number of cylinders or portions of cylinders to be reserved for the data file. Specified portions may begin with any track of the cylinder and end with any track of the cylinder; a single file may not, however, use separate portions of the same cylinder. Each specified portion on a cylinder must consist of at least two consecutive tracks. The cylinders or portions of cylinders need not be consecutive and may be in one or more physical units of the same type, i.e., IBM 1301 or IBM 2302 Disk Storage.

The entire cylinder area specified need not be filled completely when the original data is placed on the file by the File Organization System. The ratio between the available area within a cylinder and the area actually loaded is called the *Cylinder Load* factor. By leaving some free tracks in a cylinder, data records added to the file at a later time can be reached in approximately the same time as the initially loaded data records.

Figure 5 shows the allocation of a portion of a user's file and the significance of the Cylinder Load factor and the Record Load factor to the allocated portion.

File Directory and Index Area

The File Directory and Index area contains the File Directory, the Level 1 Indexes, and the Level 2 Indexes. This area must be formatted for one record per track with a six-character record address (HA2 of 00) and 2,800 characters for data. The File Directory and Index area must be formatted in Move mode. The number of tracks that must be reserved is a function of the number of data files, the number of records in each data file, and the size of the key in each data file. (Appendix C contains a method which can be used to determine how much area should be allocated to the File Directory and Index area.)

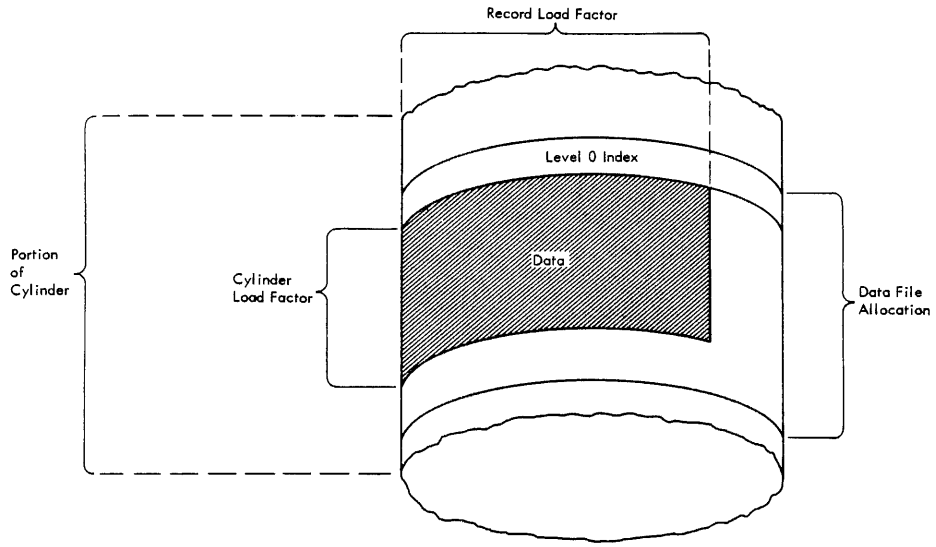


Figure 5. Examples of Disk Storage Allocation Variables

The multiphase File Organization System program created by System Generation and System Definition is executed as any other dependent program. The multiphase program is on the SOF or Job file, the user's control information is in the SIU, and the data records are on an assigned input file.

The types of cards needed to execute a File Organization System program are described in the section "Control Cards." Examples of control card decks to perform specific functions are shown at the end of this section.

The Resident Monitor loads the Control phase into core storage. The phase reads and processes the user's control cards. As the control cards are read, the Control phase analyzes the data and places it in the File Organization System Communication Region, which is resident throughout the execution of the program.

The Control phase calls the operating phase of the File Organization System program that is specified by the File Information control card and passes control to that phase. The operating phase called performs its functions according to the specifications from the Communication Region.

The following discussions describe the functions that can be performed on data files, the control cards needed to specify the desired function, and examples of the execution of the File Organization System programs needed to perform the various functions.

File Organization System Organizing Functions

The functions performed by the three operating phases (Load, Add, and Unload) of the File Organization System are:

Loading files
Unloading files
Reloading files
Adding records
Deleting records
Deleting files

The details of these functions are described in the paragraphs that follow.

Loading Files

Loading all or a part of a data file in key sequence is accomplished by the Load phase of a File Organization System program. The necessary levels of indexes and a File Directory entry are created in the loading process.

Input to the Load phase may be from one or two sources. When two input sources are used, the length of the data records in the two sources must be the same, although the blocking factors may be different. The data records within each source must be in key sequence. The data from the two sources is merged into a single data file in disk storage.

For example, if it is known that a large addition is to be made to a file after it is organized, the recommended procedure is to use two inputs to the Load program rather than perform a Reload followed by an Add. At times it may be advantageous to perform an Unload in order to use this merge-load feature.

Input to the Load phase may be direct output from a tape or disk Sorting program. Use of this type of input is specified on a control card and is discussed in the "Control Card" section.

The File Organization System allows entry to a user-written routine prior to processing each data record. The user-written editing routine is included in the File Organization System at System Definition execution. It can perform any desired processing, such as marking records for deletion, with the following exceptions:

1. It may not alter the position of the record key.
2. It may not alter the contents of the key.

After the data is loaded into the IBM 1301/2302 Disk Storage unit, the Result File is written onto the SPR. The information printed after execution of the Load program includes the number of:

1. Records loaded onto the organized file
2. File Directory and Index area tracks available
3. Index levels developed by the file (i.e., the number of the highest index level)
4. Tracks used for
   a. Data
   b. Track index (Level 0) entries
   c. Cylinder index 1 (Level 1) entries
   d. Cylinder index 2 (Level 2) entries
5. Invalid (including all 9s) or out-of-sequence record keys encountered
6. Partially filled tracks due to
   a. Duplicate record addresses
   b. Insufficient Level 0 index space
7. Records written on the Exception File
8. Records deleted and skipped (not written on the organized file or the Exception File)

Unloading Files

The unloading of all or a specified part of the data file occurs during the Unload phase of the File Organization System program. The user provides the name of the file and the limits between which it is to be unloaded. The specifications are made on the limits control card. If limits are not specified, the entire file is unloaded. The unloaded file is placed on a specified tape or disk area in key sequence. Only the records are written on the output unit. Thus, data for report preparation is readily available on the output file. Records which were previously marked for selection are either written on the Exception File or, if marked for deletion, are ignored.

Before the delete code position of each record is examined, an exit is taken to a user-written routine if the routine was included in the program at System Definition. The user-written routine can perform any desired processing, such as marking records for deletion, with the following exceptions:

1. It may not alter the position of the record key.
2. It may not alter the contents of the key.

NOTE: Unloading a file does not alter the contents of the file on the disk storage unit.

The information written on the SPR following the unloading process includes:

1. Limits of the unload, if partial
2. Number of records unloaded
3. Number of records written in the Exception File
4. Number of records deleted from the file (i.e., not written on the output file or the Exception File)

Reloading Files

All or part of a data file may be reloaded into disk storage using the Load phase of the File Organization System program. Reloading is usually performed following the processing of a data file or the reorganization of a data file. Indexes and the File Directory entry are recreated during reloading. The information written on the SPR is of the same type as in the loading process.

Adding Records

The addition of records to an existing organized file is the function of the Add phase of the File Organization System program. Input to the Add phase does not have to be in key sequence; although processing time will be significantly reduced if input is in key sequence.

NOTE: In the following discussion, the word "block" is used to refer either to a track (Form 1 records) or to a physical record (Form 2 records).

Each new record is inserted into the proper block. If no area is available for the new record, the existing record with the highest key in that block is moved to an Overflow area. The new record may now be inserted, in proper sequence, in the block.

The Level 0 Index entry for the modified block is updated to indicate the new high key in that block, and an additional entry is made in the Level 0 Index for the record moved to the Overflow area. The Overflow area does not have an index. Each record contained in the Overflow area is indexed in the Level 0 Index for its original location.

If space is not available in the Level 0 Index for the new entry, the Level 0 Index is divided in half. The last half of the Index is placed on an overflow cylinder. The new Level 0 Index is linked with the first half and with the next Level 0 Index.

The presence of a new Level 0 Index is noted in the Level 1 Index; the reference to the old half of the Level 0 Index is updated and an entry is made for the new index. If the insertion causes the Level 1 Index to overflow, it is divided and an entry is made in the Level 2 Index.

Figure 6 shows the effect of adding a new record (2107) on the data track and its Level 0 Index. The data records are represented by their record keys.

The record with key 2107 is inserted on track 0243 and is placed in proper sequence between keys 1849 and 2287. Thus, the highest key on the track becomes 2287 instead of 2431. The Level 0 Index entry for the track is changed from 2431 to 2287 and the record with key 2431 is written on track 0618, the currently available overflow track. Reference to the overflow record is inserted in the Level 0 Index. The fields altered by this insertion are underscored in Figure 6.

If the Add program is to be used to make frequent, small changes to an organized file, the file should be set up so that the record load and cylinder load factors (see Figure 5) leave room for growth. When the file is loaded or reloaded, the Environment and Cylinder control cards—described in the "Control Cards" section of this publication—should leave a reasonable portion of each track and a reasonable number of tracks per cylinder unused. This will keep to a minimum the creation of Overflow entries and the manipulation of Index entries.

The Add function also produces a Result File. The information printed on execution of the Add program includes the number of:

1. Records added
2. Tracks used, including both data and index tracks
3. Track index (Level 0) overflows
4. Cylinder index (Level 1 or higher) overflows
5. Records written on Exception File

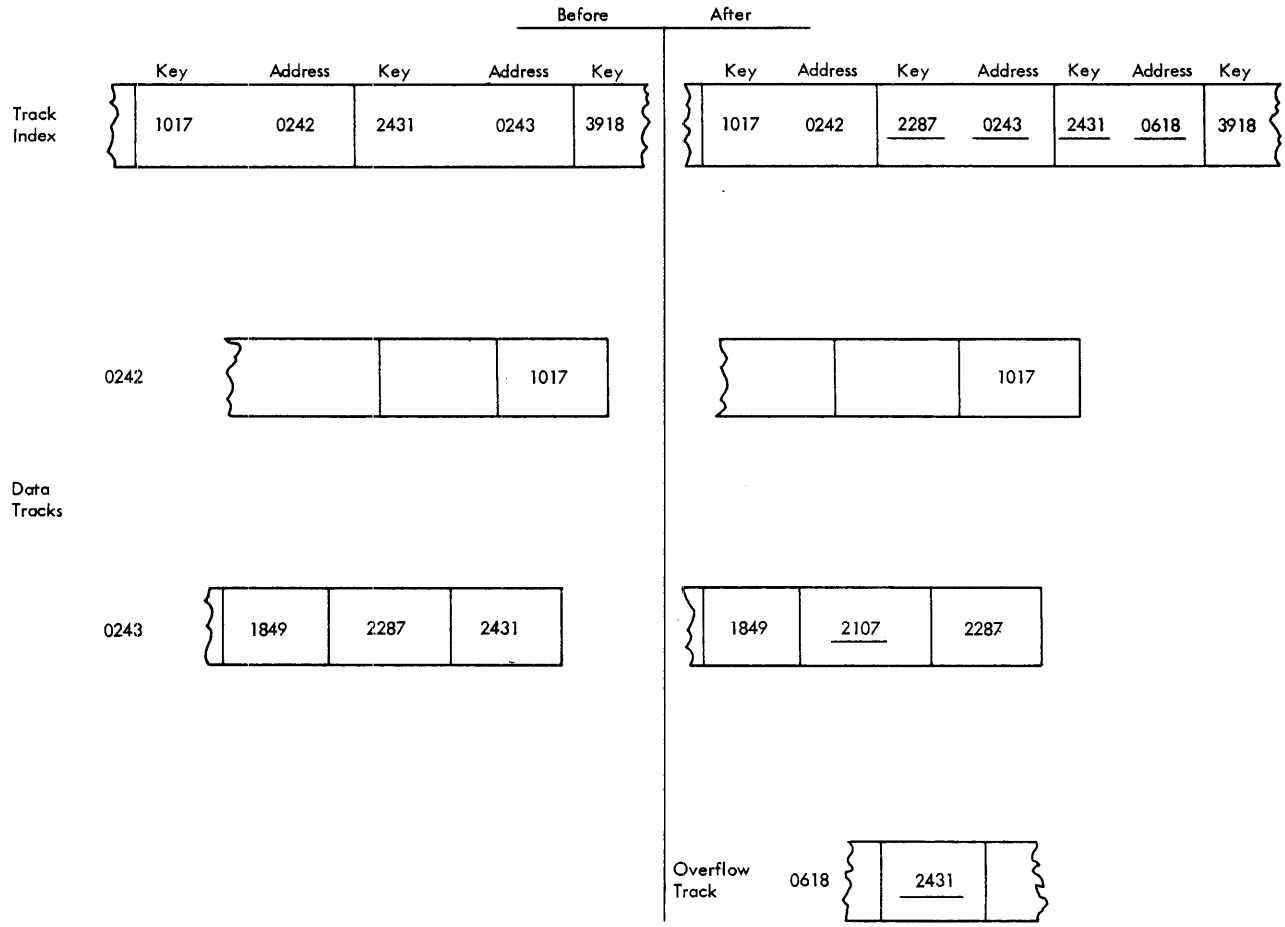


Figure 6. Adding a New Record to the Data File

Deleting Records

The deletion of data records from an organized file is a two-step process. The first step is the setting of the delete code by the user. The second step is the physical deletion of the data record by the Unload or Load phase of the File Organization System.

If the delete code is set before or during the loading or reloading process, the data record is not written on the disk. It is either written on the Exception File or ignored, according to the type of delete code (exception or skip) involved.

If the delete code is set in a user's program by use of GET and PUT macro-instructions, the data record will not be available to subsequent GET macro-instructions. However, the record remains (physically) on the disk. When the data file is unloaded, deleted records are not written on the same unit as the other records. They are either written on the Exception File or ignored.

If the delete code is set by a user-written routine during the unloading process, the record is not written on the same unit as the remainder of the data file. Rather, it is either written on the Exception File or ignored.

Deleting Files

Deleting files is a function of the Load phase. It is specified by DELET entered in the file information card. A file is deleted by removing the entry pertaining to the file from the File Directory.

Shared File Considerations

In a shared-file environment, it is the user's responsibility to ensure that two File Organization System programs are not executed simultaneously if both programs alter the contents of the File Directory or if either alters an organized file used by both.

Control Cards

The File Organization System function to be performed (Load, Unload, Reload, Add, Delete) receives its specifications from user-supplied control cards. Six control cards are available to the user, each designed to sup-

ply certain information to the File Organization System program. The cards are:

File Information card
Input/Output Assignment card
Environment card
Overflow card
Limits card
Cylinder card

The purpose of each card, its use, and the operands it may contain are discussed in the following material. Figure 7 shows the control cards needed to perform each file organization function. A detailed summary of the necessary operands is given at the end of the control card discussion.

FUNCTION	REQUIRED CARDS
Loading files	File Information I/O Assignment Environment Overflow Cylinder
Unloading files	File Information I/O Assignment Overflow Limits (for partial unloading)
Reloading files	File Information I/O Assignment Overflow Limits (for partial reloading) Cylinder
Adding files	File Information I/O Assignment Overflow
Deleting files	File Information

Figure 7. Control Card Requirements

Some of the operands discussed are required in the control card. Other operands are optional and are included only if a particular processing requirement must be met. The program assumes certain information about an omitted optional operand. Certain rules, however, apply to all control cards. They are:

1. The operand field must begin in card column 21.
2. The operands must appear in the order discussed, and must be separated by commas. Operands that are not required for a particular function can be omitted from the control card. However, if an operand is omitted, the comma that would have followed the operand must be included.
3. Operands must be the length specified in the control card descriptions.
4. The operand field cannot contain blanks. A blank signals the end of the operand field.
5. Operand fields that are too long for one control card can be continued on succeeding cards. This case is discussed under the cards to which it applies.
6. The last operand in a control card should not be followed by a comma unless the next card is a continuation card. The use of continuation cards is discussed in the paragraphs "File Information Card" and "Limits Card."
7. For any given function, control cards must always be arranged in the control card deck in the sequence shown in Figure 7. Out-of-sequence control cards can cause execution of one or more functions to be suppressed.

File Information Card

The file information card is used to specify the File Organization System function to be performed. This card is always required and must precede any optional control cards. The format of the file information card is:

6	16	21
filename	*function*	*operands*

Filename is the name of the data file to be operated upon, and function is the mnemonic of the File Organization function to be executed (LOAD, DELET, RELOD, UNLOD, or ADD). The filename may consist of from one to ten characters.

The contents of the operand field are illustrated in Figure 8.

If all operands cannot be specified in columns 21-72 of the file information card, a continuation card should be used. The format of the continuation card is:

16	21	76
function (identical to the preceding card)	*remainder of operands*	2CONT

If the continuation card is required, the first card must end with a complete operand followed by a comma. Columns 76-80 of the file information card must contain the characters "ICONT," indicating that additional operands will follow on the next card. Figure 9 illustrates the correct use of a file information card and its continuation card.

The Reload, Delete, Unload, and Add functions may require only the file name and function.

Input/Output Assignment Card

The input/output assignment (IOA) card contains information about the units on which the input records to the Load, Reload, or Add functions are located, or on which the output records will be written by the Un-

load function. One input/output assignment card is required for each input or output unit. The format of the input/output assignment card is:

16	21
IOA	*operands*

The operands of the input/output assignment card are illustrated in Figure 10. They must appear on the card in the order shown in the figure.

Environment Card

The environment (ENVIR) card supplies information to the Load function about the device onto which the file is to be loaded. The format of the environment card is:

16	21
ENVIR	*operands*

The operands of the environment card are illustrated in Figure 11.

OPERAND	NUMBER OF CHARACTERS	FUNCTION	DESCRIPTION
1	4	Position of delete code	The position of the character within each data record that contains the delete code (e.g., "0079" appearing in this field would indicate that the delete code is contained in the 79th position of each record). If this operand is present, operands 2 and 3 are required. If this operand is omitted, operands 2 and 3 must also be omitted. Their omission must still be indicated by the inclusion of commas. If the operand is missing, no checking for deleted records is done.
2	1	Delete code—exception	The character shows that a record is to be removed from the data file and placed in the Exception File. This operand is required if operand 1 is present.
3	1	Delete code—eliminate	The character, which must be different from the character used for operand 2, shows that a record is to be removed from the data file. This operand is required if operand 1 is present.
4	2	Record address conversion code	*For Form 1 (unblocked) records:* When the record address is to be obtained from the composite key, "06" through "99," or "00" representing 100, indicates the right hand position of the six characters desired (e.g., "06" appearing in this field would indicate that the six most significant characters of the record key are to be used). The least significant position is assumed if the operand is omitted. "DV" indicates that the record address is to be obtained by division of the total key. *For Form 2 (blocked) records:* This operand must be omitted, indicating that the Load phase is to generate geometric record addresses.
5	1	Number of key segments	The number of segments used to create the composite key: "1" through "9" or "0" (for ten). This operand must be present.
6 (8, 10, 12 14,16, 18, 20, 22, 24)	4 (4 each)	Address of key segments	Four-digit addresses of the *right-hand position* of each key segment. (For a key segment designated as positions 35 through 39, "0039.") The number of operands (6, 8, etc.) must be the same as the number of key segments, as specified in operand 5.
7 (9, 11, 13 15, 17, 19, 21, 23, 25)	2 (2 each)	Length of key segments	Two-digit length designations (the number of positions included in the key segment to which the operand applies). The number of odd operands (7, 9, etc.) must be the same as the number of corresponding even operands (6, 8, etc.) and the number of key segments specified in operand 5.

Figure 8. File Information Card Operands

Line (3–5)	Label (6–15)	Operation (16–20)	OPERAND (21–71)	(72–80)
01	FILEXYZ	LOAD	ØØØ1,$,%,,6,ØØ1Ø,Ø9,ØØ75,Ø5,ØØ25,1Ø,ØØ76,Ø1,Ø1ØØ,	1CONT
02		LOAD	1Ø,Ø12Ø,Ø5	2CONT
03				

• Figure 9. Use of File Information Continuation Cards

OPERAND	NUMBER OF CHARACTERS	FUNCTION	DESCRIPTION
1	3	Unit	The Monitor work (MWx), reserve (MRy), or Tele-processing system unit (MTz) of the input file (for the Load, Reload, or Add functions) or output file (for the Unload function). The characters "IPI" indicate that the symbolic unit name is found in location /IPI/ in the Resident Monitor Communication Region. The name of a Monitor work or reserve unit must have been placed in this location by a previous program within the job, such as the Generalized Tape Sorting program.
2	4	Physical record length	The number of characters in each physical record.
3	1	Record form	Blocked data designated by a "2." Unblocked data designated by a "1." Blocked data is assumed if the operand is not specified.
4	1	Mode	"M" indicates Move mode; "L" indicates Load mode. Move mode is assumed if the operand is not present.
5	1	Parity	"B" indicates odd parity. "U" indicates even parity. Odd parity is assumed if the operand is not present.
6	1	Disk input/output form	If the input to the Add, Load, or Reload functions is on disk or if a disk output file is to be produced by the Unload function, the form of the disk records must be denoted by the codes "A," "C," or "G." These correspond to the IOCS disk file forms A, C, and G.
7	2	Disk input/output records per track	Specified when the data file is of form C.
8	3 or 5	Number of disk records	When input data is on disk, the number of records in the file must be given. When the data file is the output of the disk sort or the Unload function of the File Organization System, "IPI" should be specified; this indicates to the File Organization System that the location /IPI/ contains the number of disk records. If "IPI" is not specified, the actual number of records (5 characters) must be given.
9	10	Tape name	Must be provided when input or output is a labeled tape.
10	5	Date of creation	Must be provided when input is a labeled tape.
11	3	Length of label	"080" or "120." Must be provided when input or output is a labeled tape.

Figure 10. Input/Output Assignment Card Operands

OPERAND	NUMBER OF CHARACTERS	FUNCTION	DESCRIPTION
1	4	Device type	The type of storage device: "1301" or "1302" (for 2302).
2	2	First track to use	The numbers can range from "00" through "38." If the operand is missing, "00" is assumed.
3	2	Last track to use	The numbers can range from "01" through "39." If the operand is missing, "39" is assumed.
4	2	Number of records	The number of physical records on a track.
5	4	Record length	The length of each physical record.
6	4	Logical record length	The length of each logical record.
7	1	Mode	"M" indicates Move mode: "L" indicates Load mode. Move mode is assumed if the operand is not present.
8	1	Record form	Unblocked records are indicated by "1." Blocked records are indicated by "2." Blocked records are assumed if the operand is missing.
9	1	Write check option	"W" is used to show that the option is wanted. Write checking is not performed if this operand is omitted. If this operand is specified, all write operations performed on the file will be checked. This includes the loading and reloading of the file, the addition of records to the file and any write operations performed during the execution of PUT macro-instructions.

Figure 11. Environment Card Operands

Overflow Card

The overflow (OVFL) card contains information about the Exception File and about Overflow areas. The format of the overflow card is:

16	21
OVFL	*operands*

Figure 12 illustrates the overflow card operands.

Limits Card

The limits (LIMIT) card must be used to define the limits of a *partial* reload or unload. The format of the limits card is:

16	21
LIMIT	*operands*

The operands of the LIMIT card are illustrated in Figure 13.

If both operands cannot be specified in columns 21-72 of the limits cards, continuation cards must be used. The format of the continuation card is:

16	21	76
LIMIT	*operands*	*n*CONT

where n is 2 for the second limits card (i.e., the first limits continuation card), 3 for the third limits card, or 4 for the fourth limits card. If a continuation card is required, the previous card must end in column 72 or must end with a complete operand followed by a comma. Columns 76-80 of the first card must contain the characters "1CONT," indicating that additional information follows on another card. Figures 14 and 15 illustrate alternate ways of coding limits continuation cards.

OPERAND	NUMBER OF CHARACTERS	FUNCTION	DESCRIPTION
1	3	Exception File	The Monitor work file for the exception information.
2	3	Exception File label length	The number of characters in the Exception File label: "080" or "120." (May be omitted if the Exception File is unlabeled.)
3	8	Beginning of first Overflow area	Beginning address of the first cylinder to be used for overflow in the form camcylha, where c is the channel ("1," "2," "3," or "4"), a is the access mechanism number ("0" or "1"), m is the module number ("0"—"9"), cyl is the cylinder number ("000" to "249"), and ha is the HA2 established for this file. All HA2s for any one file may be identical.
4	3	End of first Overflow area	End address is of the form cyl. (See operand 3.)
5	8	Beginning of second Overflow area	Beginning address is of the form camcylha. (See operand 3.)
6	3	End of second Overflow area	End address is of the form cyl. (See operand 3.)

Figure 12. Overflow Card Operand

OPERAND	NUMBER OF CHARACTERS	CONTENTS	DESCRIPTION
1	1-100	First key	The key designating the beginning of the partial reload or unload.
2	1-100	Ending key	The key designating the end of the partial reload or unload.

Figure 13. Limits Card Operands

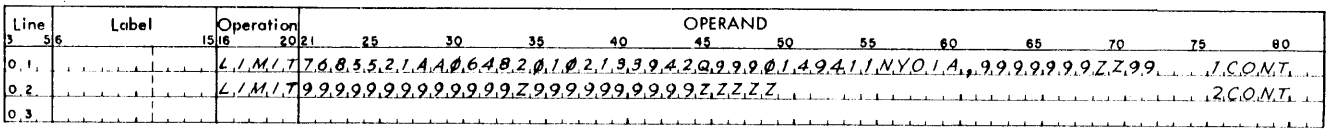


Figure 14. Use of Limits Continuation Card

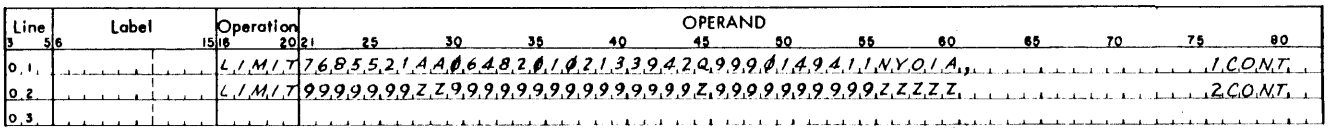


Figure 15. Alternate Use of Limits Continuation Card

OPERAND	NUMBER OF CHARACTERS	FUNCTION	DESCRIPTION
1	6	First cylinder	Address of the first cylinder to load in the form camcyl, where c is the channel ("1," "2," "3," or "4"), a is the access mechanism number ("0" or "1"), m is the module number ("0"-"9"), and cyl is the cylinder number ("000"-"249").
2	3	Last cylinder	The address of the last cylinder to load: "000" to "249."
3	2	HA2	The HA2 used for this file.
4	2	Cylinder load factor	The number of tracks in each cylinder to be loaded. The number may range from "02" to the maximum: the number of tracks in each cylinder made available for the file by the environment card. (If this operand is omitted the maximum load factor is used.)
5	2	Record load factor	The number of logical records to be loaded in each physical record or track. The number may range from "01" to the maximum: the number of records per track (Form 1) or per block (Form 2) specified in the environment card.

Figure 16. Cylinder Card Operands

Cylinder Card

Cylinder (CYL) cards supply cylinder addresses and loading factors required by the Load or Reload functions. At least one cylinder card must be included among the control cards for the Load or Reload functions, and must follow all other file organization cards. The user may provide the function with any number of additional cylinder cards. A cylinder card is read each time the function determines that more cylinders are required for the data file. Should more cylinder cards be supplied than are actually used by the Load or Reload function, extra cards are disregarded. Each is printed on the SPR with the informational message CARD NOT USED. The format of the cylinder card is:

16	21
CYL	*operands*

The operands for the cylinder card are illustrated in Figure 16. The first three operands of the cylinder card are required; the fourth and fifth may be omitted.

Figure 17 is a summary of all File Organization System control cards. The table illustrates which control cards and which operands on these control cards are required for each of the five File Organization System functions. A dash (—) indicates that the card or operand listed at the far left of the table must *not* be included for the particular File Organization System function named at the top of the column.

Messages

This section lists and explains the messages that may be generated during file organization, and provides information on the interpretation of message patterns caused by control card sequence errors. Console and SPR messages are presented separately with unnumbered messages in alphabetic order followed by numbered messages in numeric sequence. The significance of message numbers is explained in Figure 18.

CARD/OPERAND	FUNCTION				
	LOAD	RELOAD	DELETE FILE	UNLOAD	ADD
FILE INFORMATION	Required	Required	Required	Required	Required
Position of delete code	Optional[1]	—	—	—	—
Delete code—exception	Optional[1]	—	—	—	—
Delete code—eliminate	Optional[1]	—	—	—	—
Record address conversion code	Form 1: Key assumed Form 2:—	—	—	—	—
Number of key segments	Required	—	—	—	—
Address of key segments	Required[2]	—	—	—	—
Length of key segments	Required[2]	—	—	—	—
INPUT/OUTPUT ASSIGNMENT	Required	Required	—	Required	Required
Unit	Required	Required	—	Required	Required
Physical record length	Required	Required	—	Required	Required
Record form	"2" assumed	"2" assumed	—	"2" assumed	"2" assumed
Mode	"M" assumed	"M" assumed	—	"M" assumed	"M" assumed
Parity	"B" assumed	"B" assumed	—	"B" assumed	"B" assumed
Disk input/output form	Required for disk	Required for disk	—	Required for disk	Required for disk
Disk input/output records per track	Required for disk	Required for disk	—	Required for disk	Required for disk
Number of disk records	Required for disk input	Required for disk input	—	—	Required for disk input
Tape name	Required for labeled tapes	Required for labeled tapes	—	Required for labeled tapes	Required for labeled tapes
Date of creation	Required for labeled tapes	Required for labeled tapes	—	—	Required for labeled tapes
Length of label	Required for labeled tapes	Required for labeled tapes	—	Required for labeled tapes	Required for labeled tapes
ENVIRONMENT	Required	—	—	—	—
Device type	Required	—	—	—	—
First track to use	"00" assumed	—	—	—	—
Last track to use	"39" assumed	—	—	—	—
Number of records per track	Required	—	—	—	—
Record length	Required	—	—	—	—
Logical record length	Required	—	—	—	—
Mode	"M" assumed	—	—	—	—
Record form	"2" assumed	—	—	—	—
Write check option	Write checking not performed if operand omitted	—	—	—	—
OVERFLOW	Required	Required	—	Required	Required
Exception file	Required	Required	—	Required	Required
Exception file label length	Required for labeled file	Required for labeled file	—	Required for labeled file	Required for labeled file
Beginning of first overflow cylinder	Required	Required	—	—	—
End of first overflow cylinder	Required	Required	—	—	—
Beginning of second overflow cylinder	Required	Required	—	—	—
End of second overflow cylinder	Required	Required	—	—	—
LIMITS	—	Required for partial reload	—	Required for partial unload	—
First key	—	Required[3]	—	Required[3]	—
Ending key	—	Required[3]	—	Required[3]	—
CYLINDER	Required	Required	—	—	—
First cylinder	Required	Required	—	—	—
Last cylinder	Required	Required	—	—	—
HA2	Required	Required	—	—	—
Cylinder load factor	Maximum assumed	Maximum assumed	—	—	—
Record load factor	Maximum assumed	Maximum assumed	—	—	—

[1] If any one of these operands is used, all three are required.
[2] At least one operand of this type is required.
[3] If the control card is included, these requirements must be met.

Figure 17. Control Card Requirements

POSITION	INDICATION	EXPLANATION
Ten-thousands	0	Indicates a "cannot proceed" condition. The program indicates to the Resident Monitor that, unless programs are in the TEST mode, subsequent programs within the same job should not be executed and the System Monitor should skip to the next MON$$ control card. No waiting loop is entered.
	1	Indicates an occurrence such as commencement or conclusion of a particular part of the program. The message is primarily of diagnostic value. No waiting loop is entered.
	2	Indicates that a waiting loop has been entered and that the operator must take a specific corrective action or terminate execution of the program.
	3	Indicates that a waiting loop has been entered and that the operator must take one of two specific corrective actions or terminate execution of the program.
Thousands	0	This digit is always a zero.
Hundreds	5	Identifies the type of program being executed. This digit is a five for all File Organization System messages.
Tens and units	—	Arbitrarily assigned to identify each message uniquely.

Figure 18. Numbered Message Codes

Console Messages

The messages listed below are generated at the console printer. Most messages indicate error conditions that can only be corrected through the action of the programmer, who must analyze the problem, make a change in the program (for example, by altering the contents or sequence of control cards), and re-execute the program. Conditions of this type cause control to be transferred to the System Monitor. No operator action is possible. Execution of the file organization program is discontinued, and any subsequent programs within the job are bypassed. The System Monitor skips to the next job.

ANOTHER CYLINDER CARD NEEDED

Explanation: The data area available to the Load phase was exhausted before the function was completed.

Action: No operator action is possible. The user may add one or more additional CYL cards to the Load phase control-card deck and re-execute the program.

CONTROL CARDS IN ERROR

Explanation: The Control phase has completed its execution, but a control-card error was found. (The console message appears once for each series of control cards found to be in error.) All control cards for this phase have been analyzed; no file organization occurs, no alteration is made to the data file, and no waiting loop is entered. Program control returns to the System Monitor.

Action: No operator action is possible. The System Monitor skips to the next job. The user must refer to the SPR diagnostic message, analyze the problem, correct the control cards in error, and re-execute the program.

CONTROL CARDS MUST BE SUPPLIED

Explanation: The Control phase has been called in by the Resident Monitor. On attempting to read the first file information card, the Control phase received an SIU end-of-file indication.

Action: The control-card deck must be supplied and the program re-executed.

FILE NOT FOUND

Explanation: The Add or Unload phase has searched the File Directory and has not found the file name given in columns 6-15 of the file information card.

Action: No operator action is possible. The user should check columns 6-15 of the file information card to be sure that the name is correctly given. If the file name is correct, the user must alter the control card sequence or provide additional Load function control cards so that the file is loaded before the Add function is executed.

INPUT AREA REQUIREMENTS EXCEED CORE CAPACITY

Explanation: In attempting to assign core-storage input areas for execution of the Load or Reload program the Load phase determined that available core-storage capacity would be exceeded.

Action: No operator action is possible. The user must modify the input presentation as outlined in Appendix A.

LOAD PROGRAM ERROR

Explanation: The Load program has encountered an error during or prior to the execution of a Load function (Load, Reload, or Delete). Execution of the function is suppressed or terminated. No waiting loop is entered. A diagnostic message is printed on the SPR.

Action: No operator action is possible. The System Monitor skips to the next job. The user must refer to the SPR diagnostic message, correct the problem, and re-execute the function.

NO KEY—JOB DISCONTINUED

Explanation: Partial unload of an organized file has been terminated before execution. The beginning key specified on the LIMIT card is not the actual key of a record in the file, or is the key of a record bearing a delete code.

Action: No operator action is possible. The user must change the LIMIT card as required and re-execute the job.

NO RECORDS LOADED

Explanation: The Load phase has reached the end of the input data being processed. No records valid for loading have been found; for example, a deletion code was found in every record. (If the organizing function in process was a Reload, the original file has not been disturbed.)

Action: No operator action is possible. The user must re-examine the input data and take whatever corrective action is required and then re-execute the program.

PROGRAM NOT FOUND

Explanation: A file information card has been read—the card calls for an organizing function not part of the existing File Organization System.

Action: No operator action is possible. To use the organizing function specified, the user must redefine the system to include that function.

RECORD ADDITION DISCONTINUED WITH . . . (key)

Explanation: All overflow areas were exhausted during Add phase execution. Remaining records, beginning with the key shown, have not been added.

Action: No operator action is possible. The user must reorganize the file and re-execute the program to add additional records.

00551 LEVEL *n* WL

Explanation: The File Organization System has encountered a wrong length (too short) index or File Directory entry. The character *n* in the message indicates the level at which the error exists: F for File Directory, 0 for Level 0, 1 for Level 1, etc. An irrecoverable processing loop has been entered.

Action: Operator action must be taken to terminate execution. Either press INQUIRY REQUEST, type $10, and press INQUIRY RELEASE; or press COMPUTER RESET, then START.

30501 TWO ADDITIONAL DATA OVERFLOW TRACKS REQUIRED

Explanation: The Add phase has determined that further addition of records may cause an overflow. A waiting loop has been entered. A minimum of two additional data overflow tracks should be provided to ensure successful completion of the phase. If additional areas are not provided, addition of further records may cause the program to terminate unconditionally.

Action: Press INQUIRY REQUEST, then type one of the following:

$3xcamttttthaceeec—to provide additional overflow areas
where:
x is any character
c is the channel (1 for channel 1, etc.)
a is the access arm
m is the module
tttt is the beginning track address
ha is the home address
ceee is the ending track address.

$3xb—to continue record addition without providing additional overflow area
where:
x is any character
b is blank.

Press INQUIRY RELEASE.

30502 *n* ADDITIONAL INDEX OVERFLOW TRACKS REQUIRED

Explanation: Index adjustment during record addition has created the potential of an index overflow. To protect existing records, execution of the Add program has been suspended and a waiting loop has been entered. An additional index overflow area must be provided if record addition is to be resumed. The *n* in the console message indicates the number of tracks required for the additional area.

Action: Press INQUIRY REQUEST, then type one of the following:

$3xamttttthaamttttha—to provide additional area
where:
x is any character
a is the access arm (beginning track)
m is the module (beginning track)
tttt is the track address (beginning track)
ha is the home address (beginning track)
The second amttttha stands for similar disk control information for the ending track.

NOTE: No channel is specified in the message reply. The channel is the same as that originally specified for the File Directory and Index area on the SYSDF card at System Definition.

$3xb—to terminate execution
where:
x is any character
b is blank

Press INQUIRY RELEASE

30503 ADDRESS VERIFICATION

Explanation: Because of the importance that any disk area assigned at the console be an available area, this message always appears after the reply to message 30501 or 30502 has been entered. The message allows for visual verification of the preceding entry, and provides for final confirmation by the operator.

Action: Press INQUIRY REQUEST, then type:

$3xYES—If the previously entered area assignment is x correct.

$3xNO—If the previously entered area assignment is in error.
where:
x is any character

Press INQUIRY RELEASE.

NOTE: If the reply to this message was NO, the preceding message—30501 or 30502—will again appear at the console printer. This sequence will continue until the reply YES is given to message 30503.

SPR Messages

The messages listed and explained below are those produced on the SPR during file organization. The following terms are used throughout the message explanations that follow:

file information card—a control card containing, in columns 16-20, the mnemonic code of an organizing function. These codes are:

LOAD
RELOD
DELET
ADD
UNLOD

A function code is used whenever a specific function is being discussed. The general term "file information card" is used when the control card being discussed may reflect any of two or more organizing functions.

card set—the series of control cards that provide all of the parameters required for a given organizing function. A correctly arranged card set begins with a file information card and includes all of the "detail" control cards (IOA, ENVIR, etc.) required for the function named on the file information card. In the message explanations that follow, the expression "associated file information card," refers to the file information card of the card set of which the card being discussed is a part.

set terminator—a control card or condition interpreted by the Control phase as indicating that the end of a card set has been reached. A set terminator is considered to exist

whenever the Control phase encounters any one of the following:

1. A CYL card in a set beginning with a LOAD or RELOD file information card
2. A System Monitor (MON$$) control card
3. End of file on the SIU

In addition, any card preceding a file information card is considered the last card of a set.

Control Phase Messages

The Control phase examines the parameters and sequence of all file organization control cards and prints all control cards on the Standard Print Unit. If a parameter or sequence error is found, one of the below-listed diagnostic messages is printed on the SPR and the message CONTROL CARDS IN ERROR is produced at the console printer. (For the effect of a control card error on file organization execution, refer to the explanation of the console message.)

ANOTHER FILE CARD REQUIRED FOR KEY SEGMENTS

Explanation: Sequence error. After reading a file information card containing 1CONT in columns 76-80, the Control phase encountered the card printed immediately above this message. The card is not the required file information continuation card with 2CONT in columns 76-80.

CARD IMPOSSIBLE TO PROCESS

Explanation: The card printed immediately above this message cannot be interpreted by the Control phase. The information in columns 16-20 contains a key-punch error or is otherwise not valid for file organization.

CARD NOT USED

Explanation: This is an informational message to indicate to the programmer that execution of the LOAD or RELOD function performed did not require the use of the additional areas provided through the second or subsequent CYL cards (the cards printed immediately above this message). Those areas remain available for other use.

CONTROL CARDS IN WRONG ORDER

Explanation: Sequence error. A file information card is missing or out of sequence. This message appears in the relative position within the control-card printout where a file information card should have appeared.

CONVERSION CODE EXCEEDS KEY LENGTH

Explanation: There is an inconsistency in the control card printed immediately above this message. The complete key length (i.e., the sum of odd parameters 7 through 25) provides fewer characters than parameter 4 specifies for use in creating record addresses.

CONVERSION CODE INVALID

Explanation: The ENVIR card printed immediately above this message is inconsistent with the associated file information card. The fourth parameter of the file information card was present, indicating Form 1 records. Parameter 8 of the ENVIR card indicates that records are Form 2.

CYLINDER CARD NEEDED

Explanation: Sequence error. A CYL card is missing or out of sequence. This message appears in the relative position within the control card printout where the CYL card should have appeared.

CYLINDER CARDS ALLOWED ONLY WITH LOAD AND RELOAD

Explanation: The CYL card printed immediately preceding this message is out of order or unnecessary. The associated file information card specifies an organizing function other than LOAD or RELOD; no CYL card is required or permitted.

CYLINDER LOAD FACTOR GREATER THAN ENVIRONMENT

Explanation: The CYL card preceding this message is inconsistent with the associated ENVIR card. The fourth parameter of the CYL card specifies that more tracks per cylinder are to be loaded than were defined by the first- and last-track parameters (parameters 2 and 3) of the ENVIR card.

DESIGNATION OF KEY SEGMENTS INVALID

Explanation: Because of a format error or missing continuation card, the file information card does not completely define the key segment lengths and addresses to be used. For every even parameter beginning with the sixth there must be a corresponding odd parameter, and for every odd parameter beginning with the seventh there must be a corresponding even parameter.

DISK RECORD COUNT INVALID

Explanation: The IOA card preceding this message does not meet all of the following conditions:

1. Parameter 6 is mandatory for disk input or output records.
2. If the function is ADD, LOAD, or RELOD and there is disk input, parameter 8 must consist of five digits.
3. If the function is UNLOD and there is disk output, parameter 8 must be the characters IPI (without slashes).

ELIMINATION AND EXCEPTION CHAR MUST DIFFER

Explanation: The same character appears as the second and third parameters of the file information card preceding this message.

ENVIRONMENT CARD ALLOWED ONLY FOR LOAD

Explanation: The ENVIR card is out of order or unnecessary. The associated file information card specifies a function other than LOAD; no ENVIR card is required or permitted.

FIRST THREE PARAMETERS REQUIRED

Explanation: The CYL card preceding this message has been examined by the Control phase and has been rejected because one or more of the first three parameters is missing. (The first three parameters are mandatory.)

GROUP MARK INVALID IN FIRST POSITION

Explanation: The file information card preceding this message specifies a group-mark (≡) as a delete code (parameters 2 or 3). Parameter 1 of the same card indicates that the group-mark will appear as the first character of a record, which is not permitted.

INDICATED VS DESIGNATED KEY SEGMENTS DIFFER

Explanation: There is an inconsistency in the file information card preceding this message. The number of key segments specified by the length and address parameters (parameters 6 through 25) does not match the number of segments specified by parameter 5.

INFORMATION ALLOWED ONLY TO COLUMN 72

Explanation: The parameters contained on the LIMIT card preceding this message extend beyond the maximum permissible operand field; card columns 21-72.

I/O CARD NEEDED

Explanation: Sequence error. No IOA card was found in the card set.

I/O CARD SHOULD NOT BE INCLUDED WHEN DELETING

Explanation: The IOA card is out of order or unnecessary. The associated file information card specifies the function DELET.

I/O OVFL, AND ENVIR CARDS NECESSARY FOR LOAD

Explanation: The card set is incomplete. One or more of these control cards is missing or out of order.

I/O AND OVFL CARDS NECESSARY FOR RELOAD

Explanation: The card set is incomplete. One or both of these control cards is missing or out of order.

KEY LENGTHS DIFFER

Explanation: The two parameters of the LIMIT card are inconsistent; the number of characters in each parameter is not the same. Because each parameter is an actual key, the different lengths indicate that more than one key length exists within a single file, which is not permitted. If the two parameters of the LIMIT card *are* consistent, this message indicates that, for this file, the File Directory and Index area contains keys of a length other than that appearing on the LIMIT card.

LENGTH OF TOTAL KEY MUST NOT BE OVER 100

Explanation: The maximum permissible key length is 100 characters. If this message appears immediately below a LIMIT card, one or both of the card's parameters is more than 100 characters long. If this message appears immediately below a file information card, the sum of the 7th, 9th, and subsequent odd parameters of the card is more than 100.

LIMIT CARDS ARE OUT OF ORDER

Explanation: A LIMIT continuation card has been found out of sequence in the control-card deck. For example, the LIMIT card just read by the Control phase contains 2CONT in columns 76-80, but is preceded by the card shown, which is not a LIMIT card numbered 1CONT.

LIMIT CARD ONLY ALLOWED WITH RELOAD AND UNLOAD

Explanation: The LIMIT card is out of order or unnecessary. The associated file information card specifies an organizing function other than UNLOD or RELOD.

MODE INVALID

Explanation: Parameter 7 of the ENVIR card is in error. Only the characters M or L (for Move or Load mode, respectively) may appear.

MUST INDICATE KEY SEGMENTS

Explanation: Parameter 5 of the file information card is missing or invalid.

NUMBER OF RECORDS PER TRACK TOO GREAT

Explanation: Parameters 4, 5, and 7 of the ENVIR card are inconsistent. The number of records per track (parameter 4) times the length of each physical record (parameter 5) exceeds the track limits based on the mode specified in parameter 7.

ONLY ONE { OVFL / ENVIR } CARD ALLOWED

Explanation: This message follows printout of a misplaced or redundant card. It is the second OVFL or ENVIR card found since the associated file information card was read.

ONLY ONE LIMITS GROUPING ALLOWED

Explanation: This message follows any LIMIT card separated from preceding LIMIT cards by a card of another type.

ONLY TWO I/O CARDS ALLOWED

Explanation: The IOA card is redundant. It is the third IOA card found in the card set.

OVERFLOW CARD NEEDED

Explanation: Sequence error. An OVFL card is missing or out of sequence.

OVERFLOW PARAMETERS ARE NECESSARY

Explanation: The OVFL card is part of the card set for the Load or Reload function. For those functions, all six OVFL parameters are mandatory; one or more is missing.

OVERFLOW CARD NOT ALLOWED FOR DELET

Explanation: The OVFL card preceding this message is out of order or unnecessary. The associated file information card specifies the function DELET.

PARAMETER *n* INVALID

Explanation: Parameter *n*, where *n* is a digit from 1 through 6, is in error on the card printed immediately above this message. Any type of card may appear.
NOTE: This message following a CYL or OVFL card may indicate a conflict between parameters rather than an error in a single parameter. If no specific parameter error is evident, check the entire card for consistency. Also, the message PARAMETER 1 INVALID following an IOA card may indicate that the parameter refers to a symbolic unit for which no assignment has been made.

PARAMETERS REQUIRED FOR LOAD

Explanation: The ENVIR card does not include all of the mandatory parameters: 1, 4, 5, and 6.

POSITION MUST BE GIVEN FOR DELETE CODE

Explanation: The file information card includes parameter 2 and/or parameter 3, but does not include parameter 1.

RECORD FORM INVALID

Explanation: Parameter 8 of the ENVIR card preceding this message is invalid. Only the characters 1 or 2 (for Form 1 or Form 2 records, respectively) may appear.

RECORDS PER BLOCK GREATER THAN ENVIRONMENT

Explanation: The CYL card is inconsistent with the associated ENVIR card. Parameter 5 of the CYL card establishes a blocking factor for the Form 2 records that is greater than the blocking factor implied by parameters 5 and 6 of the associated ENVIR card.

RECORDS PER TRACK GREATER THAN ENVIRONMENT

Explanation: The CYL card is inconsistent with the associated ENVIR card. Parameter 5 of the CYL card specifies a greater number of Form 1 records per track than was specified by parameter 4 of the associated ENVIR card.

RECORDS PER TRACK IS INCLUDED IF DISK FORM IS C

Explanation: Parameters 6 and 7 of the IOA card are inconsistent. Parameter 6 specifies that records are *not* Form C. Parameter 7 is present; parameter 7 is permitted only when records *are* Form C.

RECORDS PER TRACK INVALID

Explanation: Parameters 6 and 7 of the IOA card preceding this message are inconsistent. Parameter 6 specifies Form C records. Parameter 7, mandatory for Form C records, is missing.

TAPE LABEL PARAMETER ERROR
Explanation: The error involves parameters 9, 10, and 11 of the IOA card. If either parameter 9 or 11 is present, both must be present. Parameter 9 may not be longer than ten characters. Parameter 10 may be omitted, but if present may not be longer than five characters. Parameter 11 must be the characters 080 or 120.

WRITE CHECK OPTION INVALID
Explanation: Parameter 9 of the ENVIR card preceding this message is invalid. Only the character W may appear.

Load Program Messages

The SPR messages listed below are generated by the Load program prior to or during execution of a Load function (Load, Reload, or Delete). If an error arises, the message LOAD PROGRAM ERROR is produced at the console printer and one of the following messages is printed on the SPR.

FILE DIRECTORY OVERFLOW
Explanation: The last available File Directory track has been exhausted during execution of the Load phase. Address linkages of the file have been changed; file reorganization is essential to prevent loss of data.
Action: No operator action is possible. The user must take corrective action as outlined in Appendix A.

FUNCTION INVALID FOR THIS FILE
Explanation: An organizing function has been specified that is incompatible with the existing files: e.g., a check of the File Directory indicates that the system has been called upon to load a file already on disk, or to unload or delete a file not on disk.
Action: No operator action is possible. The user must determine and correct the problem, and re-execute the program.

INDEX LIMITS EXCEEDED
Explanation: During execution of the Load phase, the maximum permissible indexing level (Level 2) was exceeded.
Action: No operator action is possible. The user must take corrective action as outlined in Appendix A.

LIMITS PROHIBIT PARTIAL RELOAD
Explanation: Before partial reload execution was begun, it was determined that the indexing level for the file was above the maximum permissible level (Level 2) for that function. Data on the existing file has not been disturbed.
Action: No operator action is possible. The user must take corrective action as outlined in Appendix A.

PARTIAL RELOAD INDEX LIMITS EXCEEDED
Explanation: During execution of the partial reload function, the maximum permissible indexing level for partial reload (Level 2) has been exhausted. The function cannot be completed. Address linkages of the file have been changed; the file must be reorganized before further use.
Action: No operator action is possible. The user must take corrective action as outlined in Appendix A.

Sequence Error Diagnosis

The physical order of File Organization System control cards within the card deck in the SIU must conform to the card sequences specified in this manual. (Required sequences are covered in the sections in which card format is discussed.) Out-of-sequence control cards constitute an error and cause execution to be suppressed.

Most sequence errors may be specifically diagnosed from the message(s) produced; *any* sequence error can be detected by a scan of the control-card printout or of the cards themselves. The relationship between card and error message on the printer page follows these conventions:

1. An error message relating to the parameters of a card is printed on the line below the card printout.
2. Messages that indicate that a card is extraneous to the set in which it appears (such as ENVIRONMENT CARD ALLOWED ONLY FOR LOAD) are printed on the line following the printout of the extra card.
3. Certain control cards must always be followed by a specific type of card. For example, a LIMIT card marked 1CONT must be followed by a LIMIT card marked 2CONT. When these required sequences are interrupted by an out-of-sequence card, a message indicating that the required second card was not found appears after the printout of the card that *was* found.
4. When a card set is incomplete, the error message (e.g., CYLINDER CARD NEEDED) follows the last card of the set (i.e., the last card that was printed before the set terminator was encountered). The message refers to the card set as a whole; it does not necessarily indicate that the immediately preceding card is in error.

Certain types of sequence errors cause a number of apparently contradictory messages to be generated. This is almost certain indication that control cards are out of order. Simply rearranging the deck and re-executing the program should resolve the problem. Detailed analysis of the messages produced is not justified when an out-of-sequence condition exists, because messages will not have their usual significance and will probably conflict. This point is illustrated by the following example.

Figure 19 is the SPR printout resulting from a file organization run in which the sequence of control cards was incorrect. Two separate organizing functions are involved: the loading of file SAMPLE1, and the reloading of file SAMPLE2. (The complete card set for the RELOD function is not shown.) No parameter errors are involved. The point to be noted is that the individual messages produced are of less significance than the *pattern* of the messages. The Control phase interprets the card deck as follows:

1. These control cards (by card type) are read, found to be correct, and printed out:

LOAD
IOA
ENVIR
OVFL

```
SAMPLE1    LOAD ,,,10,1,0010,10

           IOA  MW1,0800

           ENVIR1301,,,02,0800,0080

           OVFL MW2,,10020600,208,10120611,208
     CYLINDER CARD NEEDED



SAMPLE2    RELOD

           CYL  111111,222,HA
     I/O AND OVFL CARDS NECESSARY FOR RELOAD

           IOA  MW1,2000
     CONTROL CARDS IN WRONG ORDER
```

Figure 19. Typical Control Card Sequence Error Printout

2. The next card read is the file information card for the RELOD function. This is a set terminator. The preceding card set is, therefore, reviewed as a whole and is found to include no CYL card. The message CYLINDER CARD NEEDED is printed at the apparent end of the card set. The Control phase skips to a new printer page and begins reviewing the next card set.

3. The file information card for the RELOD card set is read, found correct, and printed. Because it is apparently in correct position as the first card of a set, no message appears.

4. The misplaced CYL card from the first card set is found. It is assumed to belong to the second card set, and is taken as the terminator for that set. Because the second set consisted of only a file information card and a terminating CYL card, the message I/O AND OVFL CARDS NECESSARY FOR RELOAD is printed.

5. The IOA card of the second set is now read. Because the second set was terminated by the out-of-sequence CYL card, the IOA card must be the first of a new card set. It is not the required file information card. The message CONTROL CARDS IN WRONG ORDER is given.

6. Remaining control cards and error messages are not shown. It may be seen that the deck must be reordered before meaningful results can be obtained.

File Organization Examples

Figures 20 and 21 contain lists of sample control card decks. Each Monitor or File Organization System control card in the sample decks contains an identification number in columns 3 and 4. The card descriptions that follow refer to the control cards by their identification number.

Example 1

Figure 20 illustrates the control cards needed to load a data file with two types of tape records. The information on sample cards 03-09 is interpreted as follows:

Card 03: The organized file is to be called WIDGET. The first position of each data record is reserved for a delete code. If a "$" appears in this position, the record is to be removed from the data file and placed on the Exception File. If a "%" appears, the record is to be ignored. The Load function is to generate the record addresses. The composite record key is created from three key segments. The first segment ends at position 10 and is 5 characters long. The second ends at position 30 and is 3 characters long. The third ends at position 20 and is 2 characters long. The complete composite key will consist of the ten characters contained in these locations: 6 through 10, 28 through 30, and 19 and 20 in that order.

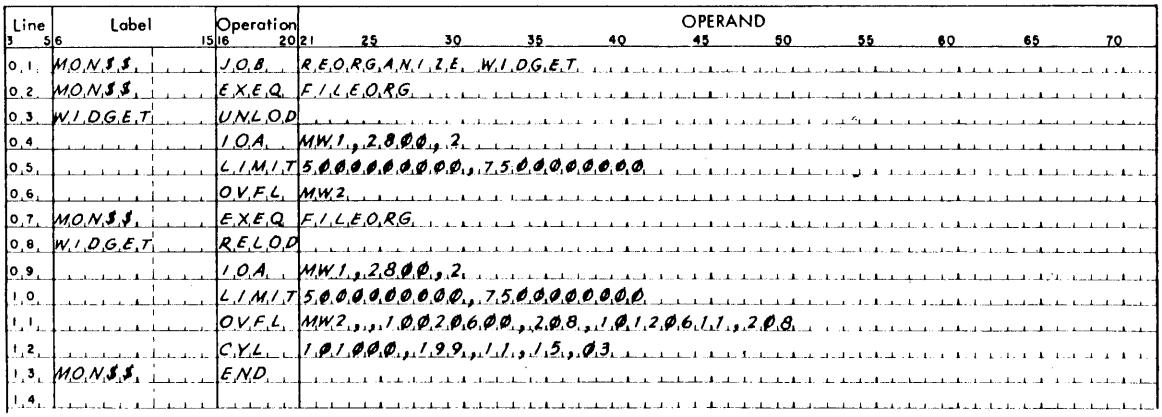
Card 04: Some of the data records are output from the previously executed tape sort program. They are on the unit specified in location /IPI/. The 100-character Form 1 records are to be written in Move mode and even parity.

Card 05: Additional data records are located on MW2. They are blocked into physical records that are 2,000 characters long. These data records are written in Load mode and are in odd parity.

Card 06: The records from both units are to be loaded on tracks 20-39 of an IBM 1301 Disk Storage Unit. Six physical records are to be written per track. The physical record length is 0400 characters; the logical record length is 100 characters. The records, to be written in Move mode, are Form 2. Write Disk Checks are to be executed after each Write Disk operation is complete.

Line	Label	Operation	OPERAND
0,1,	MONS$,	COMT,	LOAD,DATA,FILE,WIDGET,
0,2,	MONS$,	EXEQ,	FILEORG,
0,3,	WIDGET,	LOAD,	ØØØ1,,$,,%,,3,,ØØ1Ø,,Ø5,,ØØ3Ø,,Ø3,,ØØ2Ø,,Ø2,
0,4,		IOA,	IPI,,Ø1ØØ,,1,,M,U,
0,5,		IOA,	MW2,,2ØØØ,,2,,L,,Ø,
0,6,		ENVIR	1301,,2Ø,,39,,Ø6,,Ø4ØØ,,Ø1ØØ,,M,,2,,W
0,7,		OVFL,	MW3,,,1ØØ2ØØØØ,,2Ø5,,1Ø12ØØ1,1,,2Ø5,
0,8,		CYL,	1ØØØØØ,,199,,ØØ,
0,9,		CYL,	1Ø1ØØØ,,199,,1,1,,15,,Ø3,
1,0,	MONS$,	JOB,	NEXT,
1,1,			

Figure 20. Control Cards to Load Data File



Line	Label	Operation	OPERAND
01	MONSS	JOB	REORG,ANIZE, WIDGET
02	MONSS	EXEQ	FILEORG
03	WIDGET	UNLOD	
04		IOA	MW1,,2800,,2
05		LIMIT	5000000000,,7500000000
06		OVFL	MW2
07	MONSS	EXEQ	FILEORG
08	WIDGET	RELOD	
09		IOA	MW1,,2800,,2
10		LIMIT	5000000000,,7500000000
11		OVFL	MW2,,,1000200600,,2808,,1012061,,2808
12		CYL	1010000,,199,,1,,15,,03
13	MONSS	END	
14			

Figure 21. Control Cards for Partial Unload and Partial Reload

Card 07: Exception information is to be placed on MW3. The Exception File is unlabeled. The first area available for overflow records is located on channel 1, access mechanism 0, module 0 beginning at cylinder 200, and ending at cylinder 205. The HA2 in this area is 00. The second Overflow area is located on channel 1, access mechanism 0, module 1. It begins at home address 11, cylinder 200, and ends at cylinder 205.

Card 08: The first area for data records is on channel 1, access mechanism 0, module 0; the first cylinder to be loaded is 000. The last cylinder to be loaded is 199. The home address to be used is 00.

Card 09: If necessary, channel 1, access mechanism 0, module 1, cylinder 000 to 199 will also be used. The home address to be used in this case is 11. Fifteen tracks in each cylinder are to be used and three logical records are to be loaded into each block.

Example 2

Figure 21 illustrates partial unloading and partial reloading of a file. A portion of the data file WIDGET is to be unloaded onto tape and then reloaded from tape back onto disk storage. The Unload function is to block those records whose keys are between 5000000000 and 7500000000 into 2,800-character physical records and write them on tape unit MW1. When the unloading is complete, the records are to be reloaded into the area of disk storage specified on sample card 12 in Figure 21.

Once a data file has been organized by the File Organization System, the programmer may work with records within the data file by using the facilities of IOCS. The general use of IOCS is described in the publication *IBM 1410/7010 Operating System; Basic Input/Output Control System*, Form C28-0322.

Specific information on defining and processing *organized* files within the framework of IOCS is given in the following text.

The DTF Statement

The DTF (Define the File) statement is used to describe the characteristics of a logical data file. The DTF statement consists of a header line and several subsequent entries. Information contained in these entries is used by IOCS to read and write data files.

DTF Header Line

The DTF header line is required. It must appear as the first entry of every DTF statement. The operand of the DTF header is the name of a logical file. If it is not also the name of the organized file, a FILENAME entry that specifies the name of the organized file must be included. The FILENAME entry and its operand are discussed later in this section.

Figure 22 illustrates the coding of the DTF header line for the file FILEABC.

Line	Label	Operation	
01		DTF	FILEABC
02			

Figure 22. DTF Header Line

FILEFORM Entry

The FILEFORM entry is required. It is used to describe the type of data records contained in the defined file. Records of the File Organization System are indicated by the word ORGANIZED in the operand field. An example of how this entry is coded is illustrated in Figure 23.

Line	Label	Operation	
01	FILEFORM		ORGANIZED
02			

Figure 23. FILEFORM Entry

INDEX Entry

The INDEX entry is required. The operand of INDEX (X1-X12) indicates the index register (1-12) the programmer wishes to use in blocking and deblocking data records. An example of how the INDEX entry is coded is illustrated in figure 24. In the example, index register 11 is to be used for blocking and deblocking data records.

Line	Label	Operation	
01	INDEX		X11
02			

Figure 24. INDEX Entry

IOAREAS Entry

The IOAREAS entry is required. It is used to indicate to the IOCS which input/output areas are to be associated with the file.

The IOAREAS entry may have a minimum of one and a maximum of four operands. Each of the first three operands is the label of an input/output area that is to be associated with the file. The label must not be indexed.

The IOAREAS entry causes the IOCS to generate an Input/Output Request Word (IORW) for each input/output area with a label appearing as one of the first three operands of this entry.

The IOAREAS entry may have a fourth operand, a label that consists of an alphabetic character followed by from one to eight alphameric characters. The IOCS creates three new labels by adding an A, B, or C to the end of the given label. The modified labels are assigned, respectively, to the first, second (if any), and third (if any) IORW generated by the other operands of the IOAREAS entry.

An example of how the IOAREAS entry is coded is shown in Figure 25. In the example the input/output areas are IOAREA1, IOAREA2, and IOAREA3. IOCS will assign the labels LABELA, LABELB, and LABELC to the IORW's for the areas.

The input/output areas specified in the IOAREAS entry must be defined using Autocoder DA statements.

Line	Label	Operation	OPERAND
01	IOAREAS		IOAREA1,IOAREA2,IOAREA3,LABEL
02			

Figure 25. IOAREAS Entry

The length of the defined areas (see Figure 26) depends upon:

1. The form of the data records
2. The type of GET macro-instructions that will be used to process the data records

Each input/output area must be preceded by one or two nine-character fields of the form

V V
XXXXXXXX≠

The number of fields required for each combination of record form and type of GET macro-instruction is illustrated in Figure 27. Examples of input/output area definitions are illustrated in Figure 28.

Record Form	Macro-instruction GET FILE	Macro-instruction GET FILE,KEY
Form 1 (unblocked)	Length of physical record X number of physical records per track	Length of physical record
Form 2 (blocked)	Length of physical record	Length of physical record

Figure 26. Length of Input/Output Areas

Record Form	Macro-instruction GET FILE	Macro-instruction GET FILE,KEY
Form 1 (unblocked)	1	2
Form 2 (blocked)	1	1

Figure 27. Number of Fields

Record Description	Input/Output Area Definition Col. 6	16	21
Form 2 records	area	DA	1X8,G
		DA	nXs,G
Form 1 records to be processed by GET FILE,KEY	area	DA	1X8,G
		DA	1X8,G
		DA	1Xm,G
Form 1 records to be processed by GET FILE	area	DA	1X8,G
		DA	rXm,G
Form 1 records to be processed by both GET FILE and GET FILE,KEY	area	DA	1X8,G
		DA	1X8,G
		DA	rXm,G

area: the name of the input/output area as specified on the IOAREAS entry
m: the length of each physical record
n: number of records per block
r: the number of physical records per track
s: length of each logical record

Figure 28. Input/Output Area Definitions

ERRCHECK Entry

The ERRCHECK entry is not required. It may be used to cause IOCS to check for wrong-length records. The user specifies that he wants IOCS to check for wrong-length records by placing WLR in the operand field. This entry should not be used if the user is planning to issue both GET FILE and GET FILE,KEY macro-instructions to process Form 1 records. An example of how this entry is coded is illustrated in Figure 29.

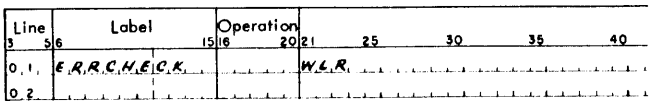


Figure 29. ERRCHECK Entry

ERROPTNS Entry

The ERROPTNS entry is not required. The ERROPTNS entry specifies the action IOCS is to take when a read operation results in an uncorrectable error condition. By using the ERROPTNS entry, the programmer can either accept or skip the erroneous record.

ACCEPT is the operand that causes the IOCS to handle all uncorrectable, erroneous records that occur on a file as if they were error free (i.e., release them to the using program as the IOCS would a record read into core storage without error).

SKIP is the operand that causes the IOCS to read the next logical record from the file into the input area that contains the uncorrectable, erroneous record, thereby destroying that record.

Omission of this entry causes the IOCS to process erroneous records as if they were error free.

Each of the operands discussed excludes the other. Therefore, only one may be specified in the same ERROPTNS entry. Figure 30 illustrates how this entry is coded if the user wishes to skip the erroneous record.

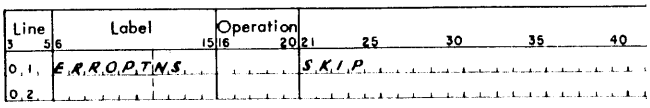


Figure 30. ERROPTNS Entry

EOFADDR Entry

The EOFADDR entry is required for all data files that are to be processed sequentially. The operand of the EOFADDR entry is the label of the end-of-file routine the programmer has provided for the file. This routine is entered whenever the IOCS determines that an end-of-file condition has occurred on the file. An example of how the EOFADDR entry is coded is illustrated in Figure 31.

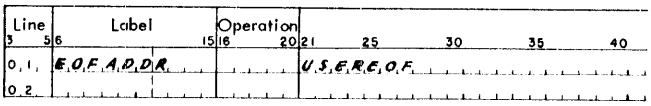


Figure 31. EOFADDR Entry

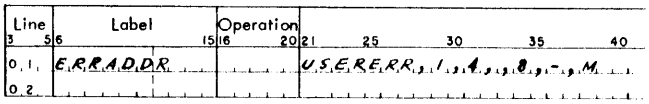


Figure 33. ERRADDR Entry

ERRADDR Entry

This entry is not required. It is used to specify the conditions under which the IOCS is to exit to an error routine the programmer has provided for the file.

The ERRADDR entry may have a minimum of one and a maximum of six operands. The first operand is the label of the error routine the programmer has provided for the file. (This operand may be omitted. If it is, the comma that would normally separate it from the next operand must be included in the entry.) Each of the second through sixth operands consists of a single character. The characters that may appear as these operands are illustrated in Figure 32.

CHARACTER	CHANNEL STATUS INDICATOR(S)
1	Not Ready (i.e., 7631 File Control Off-Line)
4	Data Check
5	Data Check (i.e., Parity Check)
M	Data Check, Wrong-Length Record
V	Data Check, No Transfer (i.e., Invalid Track Number), or Data Check, No Transfer, and Condition (i.e., Mode Check)
8	Condition (i.e., Disk Storage Circuit Check)
Q	Wrong-Length Record, Condition (i.e., Disk Storage Circuit Check plus Wrong-Length Record)
Z	No Transfer, Condition (i.e., No Record Found, Disk Storage Circuit Check, or 7631 File Control Circuit Check)
9	Condition (i.e., Disk Storage Circuit Check, Invalid Operation code, or Write Disk Check Without Mode setting)
b	No Transfer (Write Inhibit Switch on 7631 is on)
—	Wrong-Length Record (i.e., incorrect Format Length, or No Group Mark with Word Mark at End-of-Disk Control Word)

Figure 32. Acceptable Characters for ERRADDR Operands

The IOCS bypasses its normal error procedures and causes a branch to be executed to the user's error routine if the following two conditions are met:

1. The channel status indicator(s) that is turned on by an error condition or conditions that occurred on the file, *exactly matches* the channel status indicator(s) represented by the BCD code of one of the single-character operands.

2. Two commas *do not* appear between the first operand and the single-character operand. (See the second and third operands illustrated in Figure 33.)

If the above two conditions are met, but an error routine has not been specified, the IOCS bypasses its normal error correction procedures and accepts the record.

The IOCS causes a branch to be executed to the user's error routine after normal IOCS error procedures have been executed and were not able to correct the error(s), if the following conditions are met:

1. The channel status indicator(s) that is turned on by an error condition or conditions *exactly matches* the channel status indicator(s) represented by the BCD code of one of the single-character operands.

2. Two commas *do* appear between the first operand and the single-character operand. (See the fourth, fifth, and sixth operands illustrated in Figure 33.)

If a single-character ERRADDR entry whose BCD code includes a B-bit is to have its intended effect, the WLR operand must be specified on the DTF ERRCHECK entry for the file.

A group mark entered as the second or sixth operand of the DTF ERRADDR entry causes the IOCS to exit to the user's error routine if *any* channel status indicator is turned on as the result of an input/output operation performed on the file.

If the group mark is entered as the *second* operand, the normal IOCS error procedures are bypassed, and any subsequent operands are redundant. (See Figure 34.)

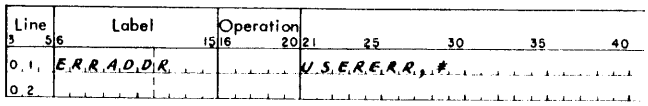


Figure 34. ERRADDR Entry (Second Operand; ≠)

If the group mark is entered as the *sixth* operand, the normal IOCS error procedures are not bypassed. In this case, any operand except the group-mark operand entered after the double-comma entry is superfluous. (See Figure 35.)

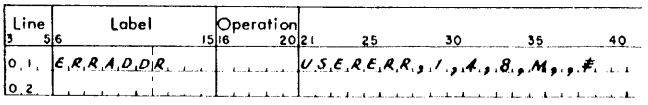


Figure 35. ERRADDR Entry (Sixth Operand; ≠)

If the group mark appears as the sixth operand and the programmer wishes to omit one or more of the other operands, a comma must be substituted for each missing operand. (See Figure 36.)

In Figure 36 the first two commas entered before the group mark operand represent the omitted fourth and fifth operands. The third and fourth commas are the normal double-comma entry.

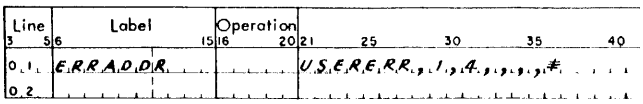


Figure 36. ERRADDR Entry (Operands Omitted)

NOTFOUND Entry

This entry is required. The NOTFOUND entry is used to specify exits to two user-written error routines. The NOTFOUND entry has two operands; the entry, including both operands, is required. The first is the label of a routine designed by the user to handle the error condition resulting from an attempt to open an organized file that is not in the File Organization System. The second operand contains the label of a user-written error routine that is entered whenever an attempt is made to GET a record that is not in the File Organization System. An example of coding a NOTFOUND entry is illustrated in Figure 37. In this example, the user-written error routines are labeled NOFILE and NORECORD.

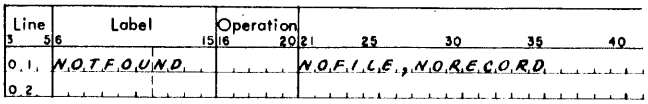


Figure 37. NOTFOUND Entry

FILENAME Entry

The FILENAME entry specifies the name of the organized file to which the previous DTF header refers. This name is used to locate information in the File Directory and Indexes about unit assignment for the organized file. If the FILENAME entry is missing, the operand of the DTF header is considered the name of the organized file. Figure 38 illustrates the coding of a FILENAME entry where FILEXYZ is the name of the organized file.

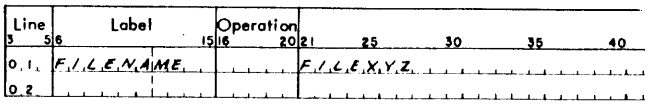


Figure 38. FILENAME Entry

Input/Output Operations

This section deals with the five macro-instructions that operate on organized files. Two of these instructions—GET FILE and GET FILE,KEY—are used to make data available for processing. The third macro-instruction, PUT FILE, is used to return processed data to the organized file. The fourth and fifth instructions are IOCTL macro-instructions used to open and close organized files.

In the following descriptions of the GET and PUT macro-instructions, it is important to note the distinction between logical records and data blocks. A logical record is the individual record made available for processing when a GET is issued. A data block is the unit of data physically transferred between core storage and an I/O device.

GET FILE,KEY Macro-Instruction

The GET FILE,KEY macro-instruction enables the user to enter an organized file at any point, by designating the key of the logical record to be accessed. The format of the macro-instruction is:

6	16	21
anylabel	GET	*file,key*

File is the name which appears as the operand of the DTF header. Key is the address of the units position of a field containing the key of the desired logical record. (A word mark must be set over the high-order position of the key, and no other word mark may appear within the key.)

The GET FILE,KEY macro-instruction causes the data block containing the desired logical record to be brought into core storage. The address of the high-order position of the logical record is placed in the index register specified in the DTF entry INDEX.

NOTE: If the record key specified through a GET FILE,KEY macro-instruction is not the actual key of a record on the organized file, the program takes the NOTFOUND exit specified in the DTF statement. For Form 2 records, the index register named in the INDEX entry of the DTF statement will contain at that time the address of the high-order position of the next record in key sequence after the non-existent key. For Form 1 records, the contents of the index register should be disregarded.

GET FILE Macro-Instruction

The GET FILE macro-instruction provides for sequential processing of logical records. The format of the macro-instruction is:

6	16	21
anylabel	GET	*file*

File is the name which appears as the operand of the DTF header.

If the first entry to a file is through a GET FILE macro-instruction, the first data block of the organized file is brought into core storage and the address of the high-order position of the first logical record is placed in the index register specified in the DTF entry INDEX. Subsequent GET FILE instructions cause the address of each subsequent logical record, in key sequence, to be placed in the index register.

If a file is entered by means of a GET FILE,KEY macro-instruction, subsequent GET FILE instructions operate as described above. The GET FILE,KEY instruction makes the desired starting record available. Subsequent GET FILE instructions make the next and each subsequent logical record available in key sequence.

PUT FILE Macro-Instruction

The only instruction that causes data records to be returned to an organized file is the PUT FILE macro-instruction. The format of the instruction is:

6	16	21
anylabel	PUT	*file*

File is the name which appears as the operand of the DTF header.

The PUT FILE macro-instruction never causes an immediate, physical transfer of data; it does set an indicator to show that a write operation is required. The point at which the write operation is actually performed is a function of the nature of the next I/O macro-instruction issued, the location of the current logical record within the data block, and other variables. For programming purposes, however, it is sufficient to consider that the *effect* of the PUT FILE instruction is to cause the logical record made available by the preceding GET to be written back into its original location within the organized file.

Figure 39 is an example of sequential processing of an organized file. The desired entry point is record 54623: i.e., the logical record whose key is 54623. The key is defined through a DCW statement (line 070). The GET FILE,KEY statement (line 010) causes the data block containing logical record 54623 to be brought into core storage; the core-storage address of the logical record is placed in the index register specified in the DTF header statement for FILEA. The logical record is processed (line 020). The statement PUT FILEA (line 030) sets an indicator to show that a write is required. Then a GET FILE macro-instruction (line 040) is issued. The next logical record—the record with the next higher key—is made available. The record is processed (line 050) and the program branches to the PUT FILE instruction (line 030). Records continue to be made available, in key sequence, until the loop is terminated by an end of file or in one of the processing routines; physical GETs and PUTs occur as required, under the internal control of the applicable macro-instructions.

IOCTL OPEN,ORGANIZED Macro-Instruction

The IOCTL OPEN,ORGANIZED macro-instruction is used to open an organized file. The names of the organized file to be opened appear as operands of this macro-instruction. A maximum of eight files may be specified in each IOCTL OPEN,ORGANIZED macro-instruction. An example of how this macro-instruction is coded is illustrated in Figure 40. In the example, two files, FILE1 and FILE2, are to be opened.

Line	Label	Operation	OPERAND
010	GOFILEA	GET	FILEA,KEYADDR GET FIRST RECORD
020		B	PROCESS1 PROCESS FIRST RECORD
030	LOOP	PUT	FILEA PUT RECORD BACK
040		GET	FILEA GET ANOTHER RECORD
050		B	PROCESSN PROCESS ANOTHER RECORD
060		B	LOOP GO TO PUT RECORD BACK
070	KEYADDR	DCW	@54623@
080			

Figure 39. Sequential File Processing

Line	Label	Operation	OPERAND
010	ANYLABEL	IOCTL	OPEN,ORGANIZED,FILE1,FILE2
020			

Figure 40. IOCTL OPEN,ORGANIZED Macro-Instruction

IOCTL CLOSE,ORGANIZED Macro-Instruction

The IOCTL CLOSE,ORGANIZED macro-instruction is used to close an organized file. The names of the organized files that are to be closed appear as the operands of this macro-instruction. A maximum of eight files may be specified in each IOCTL CLOSE,ORGANIZED macro-instruction. Figure 41 shows an example of how the macro-instruction is coded. In this example three files, named FILE1, FILE2, and FILE3, are to be closed.

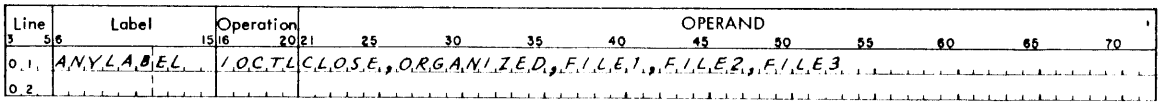


Figure 41. IOCTL CLOSE,ORGANIZED Macro-Instruction

The System Definition program combines the relocatable programs that perform the organizing functions into a multiphase File Organization System program. The user indicates the organizing functions to be included in the File Organization System program on the System Definition (SYSDF) control card.

SYSDF Control Card

The SYSDF control card designates the programs that will be part of the File Organization System and specifies the order in which the programs will normally be executed. The user also assigns a name to his program through this card. The format of the SYSDF card is:

6	16	21
name	SYSDF	*prog1, . . . , progn, number*

name: The user places in this field the name by which the produced File Organization System program will be identified. This name is to be used when the File Organization System program is executed; i.e., it is used as the first parameter on the MON$$ EXEQ card. The name may be one to ten alphameric characters, left-justified in the field. The first character must be alphabetic. Special characters may not be included anywhere in the name.

SYSDF: The letters SYSDF must appear in this field; this is the mnemonic by which the System Definition card is identified.

prog1, . . . ,progn: One to three parameters of this type may appear. Each is the name of one File Organization System program. The possible parameters are LOAD, ADD, ADD2, and UNLOAD. The order in which two or more programs are specified is the order in which they will appear in storage. The inclusion of LOAD provides the load, reload, and data deletion functions. If the Add program is to be included, either ADD or ADD2 (but not both) is specified. ADD is for use with IBM 1301 Disk Storage only; ADD2 is mandatory for 2302, and usable by 1301 but with less efficient use of core storage. (For a description of the modules, IBFOSADD and IBFOSADD2, corresponding to ADD and ADD2, refer to "Appendix B: Core-Storage Requirements.")

number: A two-position field specifying the number of tracks that have been allocated for the File Directory and Index area. (Refer to Appendix C.)

After analyzing the parameters, the System Definition program produces a set of card-image records that specifies the relocatable subroutines to be included in the desired File Organization System program. The records are written on a magnetic tape unit which the user must assign as symbolic unit MW2. The user directs the Linkage Loader to receive input from this unit by using the INPUT control card. The Linkage Loader converts the selected subroutines from relocatable to absolute format.

Relocation of the System Definition Program

The System Definition program, when not available on the SOF in absolute form, must be called in from the Library and relocated prior to execution. To do this, these control cards are required:

6	16	21
MON$$	EXEQ	LINKLOAD
	PHASE	DEFINE
	CALL	IBFOSYSDEF

These control cards cause the System Definition program to be placed on the Job file (MJB) in absolute form.

Execution of the System Definition Program

If the System Definition program is on the SOF in absolute form, the control cards required for execution are:

6	16	21
MON$$	EXEQ	DEFINE
name	SYSDF	*parameters*

These control cards cause input to the Linkage Loader program to be recorded on MW2, so that subsequent execution of the Linkage Loader program (with MW2 input) will produce an absolute File Organization System program on the Job file.

If the System Definition program is *not* on the SOF in absolute form, it must first be relocated as described above under "Relocation of the System Definition Program." Because the output of that step is on the MJB, the control cards required for execution are:

6	16	21
MON$$	EXEQ	DEFINE,MJB
name	SYSDF	*parameters*

These cards produce an absolute File Organization System program as described above.

User-Written Routines

User routines may be incorporated into File Organization System programs at System Definition time. If a routine is included in a program, an exit to that routine is taken during execution of the program.

Programming Considerations

The routine is entered prior to the processing of each record. Each user routine must be a closed subroutine, that is, the first executable instruction must be a Store B-address Register instruction. This instruction, the entry point to the routine, must have a symbolic label defined by the user in terms of a predetermined linkage symbol: OLOD/, OADD/, or OUNL/ for routines appended to the Load, Add, or Unload programs respectively. The last instruction executed by the routine must be a Branch instruction to the location stored by the first instruction. A typical coding sequence is:

```
ADD         TITLE       PROGRMNAME
                .
                .
                .
OADD/       DEFIN       USERLABEL
                .
                .
                .
USERLABEL   SBR         EXIT+5
                .
                .
                .
EXIT        B           0
```

If a user-written routine is incorporated into the Load program, the last position of the routine to be loaded must be defined by the linkage symbol OLDX/. This symbol is the base to which the Load program is relocated; the character defined by OLDX/ is overlaid. A typical coding sequence is:

```
LOAD        TITLE       PROGRMNAME
                .
                .
                .
OLOD/       DEFIN       USERLABEL
                .
                .
                .
USERLABEL   SBR         EXIT+5
                .
                .
                .
EXIT        B           0
                .
                .
                .
            LTORG
OLDX/       DEFIN       LASTCHAR
LASTCHAR    DCW         #1
```

Index register 13 is available to user-written routines as specified for the IBM 1410/7010 Operating System; (see "Information for Dependent Programs" in *IBM 1410/7010 Operating System; System Monitor,* Form C28-0319). The other index registers used in the routine must be saved and restored by the routine. The address of the key of the current record and the address of the current record are available to the user routine. The address of the current key is at symbol OKEY/. The address of the current record may be found in index register 3.

Inclusion of User Routines

User-written routines may be entered into the File Organization System from three sources:

Standard Input Unit (SIU)
Go File
Relocatable library (also contains the relocatable File Organization System programs)

If the routine is entered through the SIU, the user's relocatable routine must follow the SYSDF card in the SIU. The name of the File Organization System program into which the routine is to be inserted must be punched in columns 1-5 of the appropriate TITLE card, as LOAD, ADD, or UNLOD for the respective program.

If the user's relocatable routines are entered from the GO file or library, LINKLOAD CALLN cards for the relocatable routine must follow the SYSDF card in the control deck. The name of the File Organization System program into which the routine is to be inserted must be punched in columns 1-5 of the appropriate CALLN card, as described above for the TITLE card.

System Definition Program Messages

The following messages are produced during execution of the System Definition program. The console message 20540 ERROR ON SYSDF CARD OR USER ROUTINE, which also appears on the SPR, indicates that an error has been encountered. Diagnostic messages appear on the SPR; the action to be taken at the console is indicated by the console message. The action to be taken to correct the program error is suggested by the diagnostic message.

Console Message

20540 ERROR ON SYSDF CARD OR USER ROUTINE
Explanation: The System Definition program completed execution but one or more control-card errors were found. A waiting loop has been entered. The program processed any valid SYSDF cards that were present, but the cards in error were not processed. Any user routine named on an invalid SYSDF card has not been processed. Control cards in error have been printed on the SPR with one of the diagnostic messages listed below.
Action: Press INQUIRY REQUEST, then type:
$31—to continue processing.
$32—to discontinue processing.
Press INQUIRY RELEASE.

SPR Messages

ERROR IN SYSDF IDENTIFICATION
Explanation: The sequence of preceding control cards indicated that the card printed out with this message should be a SYSDF card. The card, however, is either an out-of-sequence card of another type, or a SYSDF card with an error in columns 16-20.

INVALID PARAMETER IN SYSDF CARD
Explanation: One of the parameters specified on the SYSDF card is in error.

NO FILE ORGANIZATION DEFINITION CARD
Explanation: On attempting to read the first SYSDF card, the System Definition program encountered end of file or a System Monitor control card on the SIU.

END OF FILE ORGANIZATION DEFINITION PROGRAM
Explanation: This message is for diagnostic use. It indicates that the program has completed execution, with or without having encountered control-card errors.

Individual organized files, as well as an entire set of organized files, will from time to time require reorganization to maintain full system efficiency. This appendix discusses when and why reorganization may become necessary, and how it may be accomplished.

When to Reorganize

The need for reorganization of a given file can usually be traced to one of three conditions:

1. The file has grown far beyond its original projected size. Data area, including overflow area, is no longer adequate.
2. When the file was created, not enough File Directory and Index area tracks were provided to support the amount of data to be handled, although ample data area was provided.
3. The use of the file is such that very extensive record additions are made. Index levels quickly mount to the point where operating efficiency is reduced. (In this case, a regular schedule for reorganization should be established to maintain efficiency.)

Result Files are printed out on the SPR after each execution of the Load, Add, or Unload functions. The Result Files for the Load and Add functions are the programmer's principal guide in determining when a file is ready for reorganization. Result Files should be analyzed carefully in the light of the file's size and usage. During early usage of a file, it may be of value to maintain a record of the file's rate of growth and the frequency with which overflows and indexing level increases occur.

A typical Load program Result File is shown in Figure 42. The total number of input records is the sum of the four entries *not* followed by asterisks. The number of records loaded onto the file is shown as RECORDS LOADED. The number of records written on the Exception File is the sum of the entries DELETED TO EXCEPTION FILE and INVALID KEY SEQ AND DUP KEYS. Remaining input records (if any) are those DELETED AND SKIPPED. Entries that should be analyzed to determine when reorganization is necessary are followed, for purposes of illustration, by five asterisks. (Asterisks do not appear on the actual SPR printout.)

```
LOAD REPORT FOR        FILENAME

RECORDS LOADED 0000084
FD TRACKS AVAIL      19                    *****
INDEX LEVELS  0                            *****
TRACKS USED                                *****
     DATA         000003                   *****
     TRACK INDEX 00001                     *****
     CYL INDEX 1    000                    *****
     CYL INDEX 2    000                    *****
INVALID KEY SEQ AND DUP KEYS 000005
PARTIALLY FILLED TRACKS DUE TO             *****
     DUP RECORD ADDR              0000     *****
     TRACK INDEX SPACE LACKING 0000        *****
DELETED TO EXCEPTION FILE     000010
DELETED AND SKIPPED           000011
```

Figure 42. Load Program Result File

Figure 43 shows a typical Add program Result File. With the exception of the first and last entries, which are for the programmer's general information, all entries are significant in determining when reorganization is necessary. In addition, the highest indexing level reached may be calculated as follows:

1. Add: NUMBER OF DATA BLOCK OVERFLOWS
   NUMBER OF TRACK INDEX OVERFLOWS
   NUMBER OF CYLINDER INDEX OVERFLOWS
2. Subtract this sum from: NUMBER OF TRACKS USED.
3. The result is the number of *additional* indexing levels introduced by the Add function. To determine the index level now effective for the file, add this number to the INDEX LEVELS digit of the file's current Load program Result File.

Reorganization should always be performed after termination of a program because of an unacceptable overflow, or when it becomes frequently necessary to provide console input to permit program execution to continue. These conditions can be avoided by action based on careful attention to Result Files.

```
ADD REPORT FOR       FILENAME

NUMBER OF RECORDS ADDED 0100

NUMBER OF TRACKS USED 010

NUMBER OF DATA BLOCK OVERFLOWS 010

NUMBER OF TRACK INDEX OVERFLOWS 000

NUMBER OF CYLINDER INDEX OVERFLOWS 000

NUMBER OF RECORDS WRITTEN ON EXCEPTION FILE 0000
```

Figure 43. Add Program Result File

Reorganizing a Single File

The method used to organize a file depends upon the reason for reorganization and the desired end result. Three methods are suggested below.

Method 1: This is the basic reorganization method.

In addition to making more data area available, it permits the user to alter the file's internal characteristics (such as record length, key specification, etc.) exactly as though the file were being loaded for the first time.

a. Unload the entire file.
b. Delete the file.
c. Make any desired control-card changes. To make more data area available, make the required changes in the CYL and/or OVFL cards.
d. Load the file.

Method 2: To make additional data area available only:

a. Unload the entire file.
b. Prepare CYL and/or OVFL cards that provide additional data area.
c. Reload the entire file.

Method 3: To make additional data area available to the portion of an extensive file where the greatest activity can be predicted:

a. Perform a partial Unload using LIMIT cards.
b. Prepare control cards. Use the same LIMIT-card parameters but make the required CYL and/or OVFL changes to provide additional areas. The LIMIT parameters for the Unload and Reload functions *must* be identical.
c. Perform a partial Reload.

Reorganizing All Files

This method must be used in the case where the File Directory and Index area, which is used by all files, is exhausted.

1. Unload *all* files.
2. Delete *all* files.
3. Perform a System Definition run providing additional tracks for the File Directory and Index area.
4. Load all required files. Because this method constitutes, in effect, the re-creation of all files, an opportunity exists for altering any file's internal characteristics such as record length or key designation, by using applicable control card parameters.

Reorganization After Termination of Partial Reload

Two error messages that may be produced on the SPR after termination of a *partial* Reload indicate the need for an immediate reorganization of the file being processed. They are:

FILE DIRECTORY OVERFLOW
PARTIAL RELOAD INDEX LIMITS EXCEEDED

Either of these messages indicates that a partial reload has been terminated during execution. Some, but not all, data has been added to the file. (The first message may also appear in connection with a total Reload. In this case, the input data is available for error recovery.) All data is recoverable, but the file must be reorganized before further use.

For example, assume a file consisting of records 0001 through 5000 (that is, a file of which the first record key is 0001 and the last record key is 5000). On an attempt to reload records 3000 through 4000, execution was terminated with the message PARTIAL RELOAD INDEX LIMITS EXCEEDED. Recovery is as follows:

1. Unload records from the first record through 2999. The ending key for the partial unload *must* be one less than the beginning key of the partial reload that was terminated. For Form 1 records, the exact key of the first record must be used. For Form 2 records, if the key of the first record is not known any hypothetical key, such as 0001, known to be lower than the first existing key may be used. Hypothetical keys must contain the same number of characters as the actual keys of the file.

2. Unload records 4001 (or corresponding actual key) through the last record of the file. The beginning key *must* be at least one greater than the ending key of the partial reload that was terminated, and must be an actual key for Form 1 records. The last record of the file may be specified using a known actual key or a hypothetical key, such as 9999, known to be higher than any actual key in the file.

3. The file now exists in three segments: the portion unloaded in step 1, the portion unloaded in step 2, and the input for the partial reload that was terminated. These three segments should be merged externally to produce a complete input file.

4. The input file created in step 3 may now be reloaded, or the old organized file may be deleted and the input file loaded as an organized file.

The File Organization System, supplied to the user as part of the Master file for both tape- and disk-oriented systems, consists of these elements:

1. The System Definition program, through which the user creates the specific multiphase programs to be employed for file organization. The System Definition (DEFINE) program is included in absolute format in the operating section of the Master file, and as the relocatable module IBFOSYSDEF in the Master file's Relocatable Library.

2. The operating and control modules through which file organization is performed. These modules are:

IBFOSCTL0, IBFOSCTL1, IBFOSCTL2 —the control modules

IBFOSLOAD0, IBFOSLOAD1, IBFOSLOAD2, IBFOSADD, IBFOSADD2, IBFOSUNLOD —the organizing function modules

All of the operating and control modules except IBFOSCTL0 are included in the Relocatable Library. IBFOSCTL0 is created through System Definition; it is present with the defined File Organization System programs on the job file or SOF.

3. The modules of the File Organization System related to IOCS macro-instructions. These modules are also part of the Relocatable Library, while the IOCS macro-instructions are part of the Macro Library:

File Organization System I/O Control Modules	Related Macro-Instructions
IBFOSOOF	IOCTL OPEN,ORGANIZED
IBFOSCOF	IOCTL CLOSE,ORGANIZED
IBFOSSG	GET FILE PUT FILE
IBFOSDG	GET FILE,KEY PUT FILE

The following paragraphs discuss the modules of the File Organization System in core storage during system definition, file organization, and the execution of a user-written program employing organized file input/output macro-instructions. Methods for calculating the number of core-storage positions occupied by each module and its associated input/output areas are also provided. The values given for the core-storage requirements of each module exclusive of I/O areas (i.e., program size) are guidelines; they are subject to change as modifications are made to the system. New figures will be published should a change of more than ten percent occur.

System Definition

The module IBFOSYSDEF occupies 5,000 positions of core storage above Resident Monitor (see Figure 44, part A).

File Organization

The control modules IBFOSCTL0 and IBFOSCTL1 are resident in core storage throughout file organization. They occupy a total of 550 positions of core storage (see Figure 44, part B). The figures provided below for the remaining file organization modules do not include Resident Monitor and the resident control modules. To calculate absolute core-storage requirements, calculate core requirements for the phase in question and add the size of Resident Monitor plus 550 positions.

IBFOSCTL2: This module is brought into core storage only to prepare for the initiation of an organizing function: Load, Add, or Unload. After reading the control cards involved, IBFOSCTL2 calls in the applicable organizing function module and is overlaid by that module. IBFOSCTL2 occupies 14,000 positions of core storage (see Figure 44, part B).

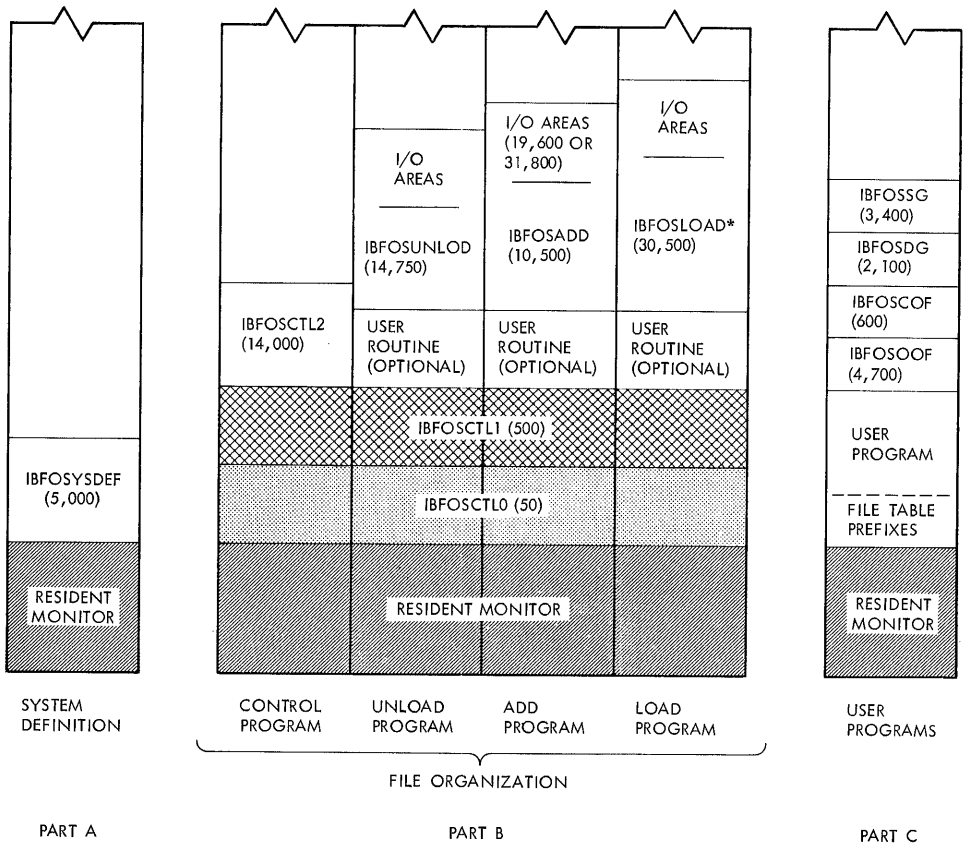
IBFOSUNLOD: The program size for the Unload phase is 14,750 positions (see Figure 44, part B). The number of core positions occupied may be calculated as:

program size + I/O areas + user-written routine (if any)

I/O areas are built up as follows:

- Two I/O areas for input from the organized file (two times physical record length)
- Two I/O areas for output to the Exception file (two times physical record length, where logical record length equals physical record length for unblocked records)
- Two I/O areas for input to the device which receives the unloaded file. (For Form 2 records, two times physical record length. For Form 1 records, two times physical record length; or two times physical record length times number of records per track if the device is disk storage.)

IBFOSADD or IBFOSADD2: When IBM 1301 Disk Storage is used to contain an organized file, the IBFOSADD module is selected. The Add program size is 10,500 positions. Seven I/O and work areas are created; each occupies the equivalent of a full disk track of



*Consists of IBFOSLOAD0, IBFOSLOAD1, and IBFOSLOAD2. Only one of these modules is in core storage at a given time; the maximum size is 30,500 positions.

• Figure 44. File Organization System Core-Storage Requirements

2,800 characters for a total I/O area of 19,600 positions. The number of core-storage positions occupied (Figure 44, part B) is, therefore, always:

30,100 + user-written routine (if any)

When IBM 2302 Disk Storage is used for an organized file, the IBFOSADD2 module is selected. As in the case of IBM 1301 Disk Storage, the Add program size is 10,500 positions. The seven Add program I/O and work areas, however, are structured as three 2,800-character work areas and four 5,850-character I/O areas, to accommodate the IBM 2302 full track length. The number of core-storage positions occupied by the Add program using 2302 Storage is:

$$10{,}500 + (3)(2800) + (4)(5850) = 42{,}300$$

The Master file provided to the user contains both forms of the module: IBFOSADD and IBFOSADD2. By the method outlined in *System Generation*, either one of the two Add program modules can be deleted from the Relocatable Library; both may be included if desired. Note that while IBFOSADD2 is mandatory for IBM 2302 Disk Storage, either form of the module may be used with IBM 1301 Disk Storage.

IBFOSLOAD0, IBFOSLOAD1, and IBFOSLOAD2: Program size for the largest phase of the three-phase Load program is 30,500 positions. The number of core-storage positions occupied (see Figure 44, part B) is:

program size + I/O areas + user routine (if any)

The minimum I/O area size, regardless of record length or number of input files, is 2,330 positions of core storage. The size of the I/O areas for a program requiring more positions may be calculated as 1, 2, 3, or 4 times the physical record length depending on whether one or two input files are used and the number of I/O areas the Load program is able to assign. The program attempts to assign two I/O areas for each file, testing before each assignment to determine whether sufficient core storage is available. Assignment of one area for each file must be possible, or the Load program is terminated. If assignment of a second area is impossible for a single file, or for either or both of two input files,

the program executes, using fewer than the maximum number (2 or 4) of I/O areas.

User-written Programs Employing Extended I/O Macro-instructions

The amount of core storage required for File Organization System modules related to I/O operations varies, depending on the macro-instructions employed in the user-written program. (See Figure 44, part C.) The size of each module is given below.

IBFOSOOF and IBFOSCOF: These modules (Open Organized File and Close Organized File) occupy 4,700 and 600 positions of core storage respectively. Both accompany any program in which an organized file is opened.

IBFOSSG: This module (Sequential Get) occupies 3,400 positions of core storage. It is present in core storage with any program using the macro-instruction GET FILE.

IBFOSDG: This module (Direct Get) occupies 2,100 positions of core storage. It is present in core storage with any program using the macro-instruction GET FILE,KEY.

File Table Prefixes: Each file table relating to an organized file includes a 270-position file table prefix, in addition to the standard file table of 45 to 57 positions.

Appendix C: File Directory and Index Area Calculations

The approximate number of tracks of disk storage used as the File Directory and Index area for the complete set of organized files in a system is:

One track for each 18 files or portions thereof

plus

One track for each file

plus

A factor *for each file* estimated from Figure 45.

Use Figure 45 as follows, for each file in the system.

1. Locate the curve that is equal to or immediately higher than the number of characters per data track for the file. (Characters per data track equals characters per record times records per data track.)
2. Find the key length for the file (horizontal scale). Mark the intersection of the selected curve and the key length. For Form 1 records, use the "Form 1 Cut-Off" curve if reached before the selected curve.
3. The point of intersection provides a specific value on the *vertical* scale. Use this value as follows:

   Form 1 Records: If the total number of physical records in the file is equal to or less than the value found, the desired factor is 1. If not, divide the value into the total number of records and round the result to the next higher integer. The rounded result is the desired factor.

   Form 2 Records: If the total number of *data tracks* used by the file is equal to or less than the value found, the desired factor is 1. (To estimate the number of data tracks, divide the total number of data blocks by the number of data blocks per track.) If the value found is less than the total number of data blocks, divide the value into the number of data blocks, and round the result to the next highest integer. The rounded result is the desired factor.

NOTE: This approximation assumes a maximum value for both record load factor and cylinder load factor, as well as the use of the maximum number of tracks per cylinder. The factor arrived at graphically may require adjustment upward if these assumptions do not apply.

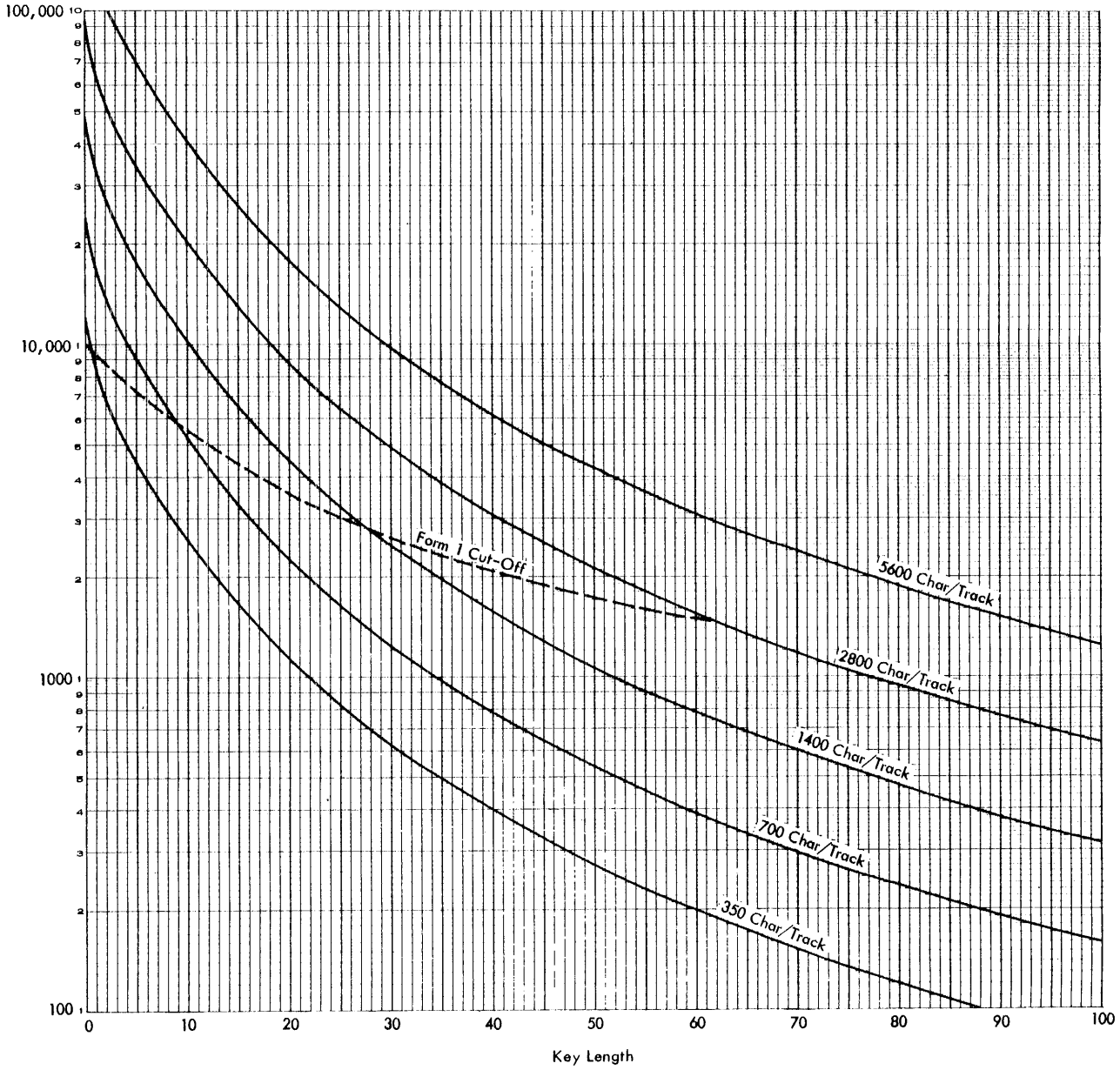


Figure 45. File Directory and Index Area Additive Factor

Index

(Where more than one page reference is given, the major reference appears first.)

IBM **Technical Newsletter**

File Number	1410/7010-34
Re: Form No.	C28-0405-2
This Newsletter No.	N27-1268
Date	December 30, 1966
Previous Newsletter Nos.	None

IBM 1410/7010 FILE ORGANIZATION SYSTEM
FOR IBM 1301/2302 DISK STORAGE

This Technical Newsletter amends the publication IBM 1410/7010 Operating System; File Organization System for IBM 1301/2302 Disk Storage, Form C28-0405-2, to include new information concerning the File Directory and Index Area, and to make other necessary changes and additions.

The attached replacement pages (9-10, 17-18, 21-22, 33-34) should be substituted for the corresponding pages now in the publication. Text changes are indicated by a vertical line to the left of the affected text; figure and table changes are indicated by a bullet (•) to the left of the affected figure/table caption.

Please file this cover letter at the back of the publication. It provides a method of determining if all changes have been received and incorporated into the publication.

IBM Corporation, Programming Publications, Dept. 637, Neighborhood Road, Kingston, N.Y. 12401

PRINTED IN U.S.A.

tion concerning the files on control cards which are discussed in the Control Card section of "Program Operation."

Data Records

The File Organization System requires data records to be stored in IBM 1301/2302 Disk Storage as fixed-length records. The user must determine if the records are to be unblocked (Form 1 records) or blocked (Form 2 records). The first character of a record may not be a group mark or a segment mark.

Record Key

A *record key* identifies each data record in the file to the File Organization System. The key is specified by the user as a certain portion of each record. The portion of the data record chosen as the key characterizes that record as being different from every other record on the file. The uniqueness of each key is determined by its contents, because the key to every record must be located in the same area of each record.

Each record key can consist of one to 100 characters of the record. The characters need not be consecutive, but can be formed with a maximum of ten segments of varying length.

NOTE: Records with keys consisting entirely of 9s are written on the Exception File. Keys should be selected so that no valid record key will contain only 9s.

The record format in Figure 3 is an example of a record key that a user may specify for a particular data file. Each record can be identified by the character in position 79, the first 5 characters, and the 11 characters starting in position 36. Therefore, the record key is composed of 17 characters found in three segments of the record in the specified sequence.

The user may specify the segments of the record key in any order, independent of the relative positions of the segments in the data record. The order in which the segments are specified indicates the relative importance of each segment, i.e., the first segment in the key is the most significant; the last segment is the least significant.

2nd segment		3rd segment		1st segment
1	5	36	46	79
XXXXX		XXXXXXXXXXX		X

Figure 3. Sample Record Key

Delete Code

The user may specify via a control card one position in each record as a delete code position. The position may *not* be part of the record key. The character in this position may be used to mark the record for removal from the data file. If the user specifies a delete code position, he must also specify in the control card two characters as delete codes. One of these codes indicates to the File Organization System that the record is to be removed from the data file and placed on the Exception File. The other code indicates to the File Organization System that the record is to be removed from the data file and skipped. Any other character in the delete code position indicates that the record is still part of the data file. See the paragraph "Deleting Records" for a discussion of the action taken by the File Organization System at various stages in the organizing and processing of data files if a record contains one of the delete codes.

Figure 4 shows four 81-character records. Assume that the key is positions 1-6, the delete code is position 80, and the delete codes are "$" (skip) and "%" (place on the Exception File). The File Organization System interprets the characters in the delete code position as follows:

1. Record 1 is to be removed from the file.
2. Records 2 and 3 are to remain in the file.
3. Record 4 is to be removed from the file and placed on the Exception File.

Record Address

When the records are placed in IBM 1301/2302 Disk Storage, the File Organization System creates a record address for each block of data records on a track.

For blocked data records (Form 2), a geometric record address is automatically created.

For unblocked data records (Form 1), the user may specify the method of creating the record address. The first method or option available to the user is to choose

1								81
2S0713	SMITH	AR	1	10500 FORTUNA DR	NEWTON	4	$	+
2S0714	SMITH	HL	1	271 HILLSBORO AV	SALEM	4	4700	+
2S0715	SMITH	TA	2	695 SUTTON PL APT 2A	SALEM	4	12145	+
2S0719	SMITH	WE	1	17111 NATIONAL BLVD APT 27	SALEM	4	%	+

Figure 4. Data Records with Delete Codes

any six consecutive characters of the record key to serve as the record address for that record. If a specification is not made, this option is assumed and the low-order six characters of the key constitute the record address. The entire key is used, right-justified and preceded by zeros, if the key is less than six characters.

The user may specify an optional method if he does not want the key used as the record address. The program then generates a six-character record address by dividing the entire record key by 999983. The six-digit remainder is the record address.

IBM 1301/2302 Disk Storage Allocation

Two areas of disk storage must be considered by the user: the area reserved for data records and the area reserved for the File Directory and Indexes.

Data Record Area

The data record area contains the data records and the Level 0 Indexes. The File Organization System assumes that all data record areas contain two-character HA2's and six-character record addresses. The user may specify the mode of operation for a data file as either Move mode or Load mode. He may also format a track of the file to contain several records, or an entire track to contain just one record; however, all tracks on one data file must have the same format. A *Record Load* factor allows the user to specify how much of the available area on each track is to be filled during the initial loading process. The unused record area may be used to accommodate records added to the file. Thus, the added records are in their proper location and access time to these records is no greater than the access time for records placed on the file at initial load.

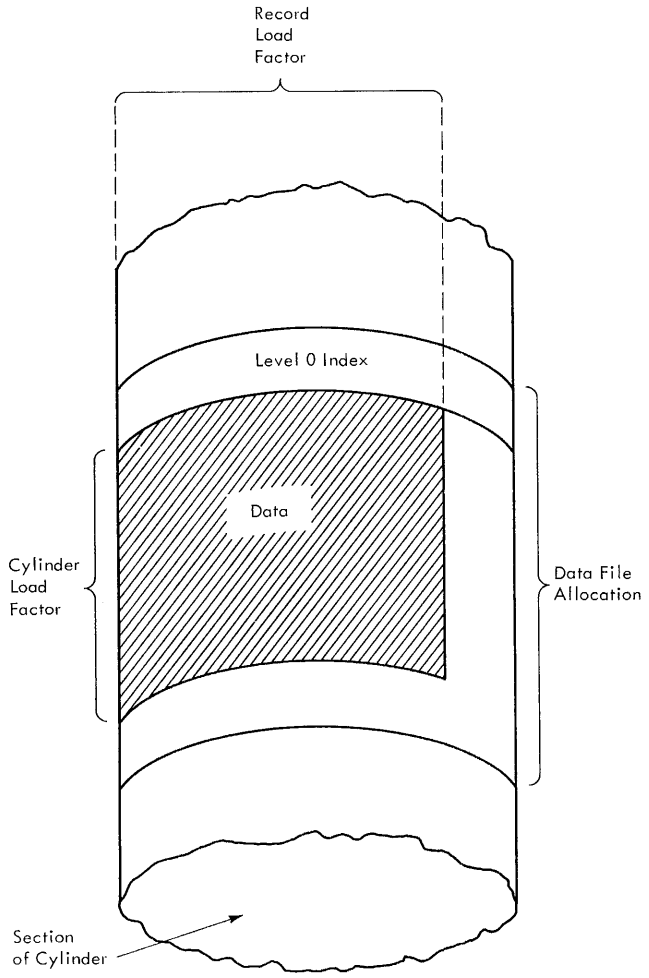


Figure 5. Examples of Disk Storage Allocation Variables

The File Organization System allows the user to specify any number of cylinders or portions of cylinders to be reserved for the data file. Specified portions may begin with any track of the cylinder and end with any track of the cylinder; a single file may not, however, use separate portions of the same cylinder. Each specified portion on a cylinder must consist of at least two consecutive tracks. The cylinders or portions of cylinders need not be consecutive and may be in one or more physical units of the same type, i.e., IBM 1301 or IBM 2302 Disk Storage.

The entire cylinder area specified need not be filled completely when the original data is placed on the file by the File Organization System. The ratio between the available area within a cylinder and the area actually loaded is called the *Cylinder Load* factor. By leaving some free tracks in a cylinder, data records added to the file at a later time can be reached in approximately the same time as the initially loaded data records.

Figure 5 shows the allocation of a portion of a user's file and the significance of the Cylinder Load factor and the Record Load factor to the allocated portion.

File Directory and Index Area

The File Directory and Index area contains the File Directory, the Level 1 Indexes, and the Level 2 Indexes. This area must be formatted for one record per track with a six-character record address (HA2 of 00) and 2,800 characters for data. (A 2302 Disk Storage Unit can be formatted for two (2,800 characters) records per track. However, only the first record is used by the File Organization System; the second record may be used through normal IOCS.) The File Directory and Index area must be formatted in Move mode. The number of tracks that must be reserved is a function of the number of data files, the number of records in each data file, and the size of the key in each data file. (Appendix C contains a method which can be used to determine how much area should be allocated to the File Directory and Index area.)

Overflow Card

The overflow (OVFL) card contains information about the Exception File and about Overflow areas. The format of the overflow card is:

16	21
OVFL	*operands*

Figure 12 illustrates the overflow card operands.

Limits Card

The limits (LIMIT) card must be used to define the limits of a *partial* reload or unload. The format of the limits card is:

16	21
LIMIT	*operands*

The operands of the LIMIT card are illustrated in Figure 13.

If both operands cannot be specified in columns 21-72 of the limits cards, continuation cards must be used. The format of the continuation card is:

16	21	76
LIMIT	*operands*	*n*CONT

where n is 2 for the second limits card (i.e., the first limits continuation card), 3 for the third limits card, or 4 for the fourth limits card. If a continuation card is required, the previous card must end in column 72 or must end with a complete operand followed by a comma. Columns 76-80 of the first card must contain the characters "1CONT," indicating that additional information follows on another card. Figures 14 and 15 illustrate alternate ways of coding limits continuation cards.

OPERAND	NUMBER OF CHARACTERS	FUNCTION	DESCRIPTION
1	3	Exception File	The Monitor work file for the exception information.
2	3	Exception File label length	The number of characters in the Exception File label: "080" or "120." (May be omitted if the Exception File is unlabeled.)
3	8	Beginning of first Overflow area	Beginning address of the first cylinder to be used for overflow in the form camcylha, where c is the channel ("1," "2," "3," or "4"), a is the access mechanism number ("0" or "1"), m is the module number ("0"—"9"), cyl is the cylinder number ("000" to "249"), and ha is the HA2 established for this file. All HA2s for any one file may be identical.
4	3	End of first Overflow area	End address is of the form cyl. (See operand 3.)
5	8	Beginning of second Overflow area	Beginning address is of the form camcylha. (See operand 3.)
6	3	End of second Overflow area	End address is of the form cyl. (See operand 3.)

NOTE: If an N is specified in column 7 of the OVFL card, the exception file will not be rewound between jobs.

- Figure 12. Overflow Card Operand

OPERAND	NUMBER OF CHARACTERS	CONTENTS	DESCRIPTION
1	1-100	First key	The key designating the beginning of the partial reload or unload.
2	1-100	Ending key	The key designating the end of the partial reload or unload.

Figure 13. Limits Card Operands

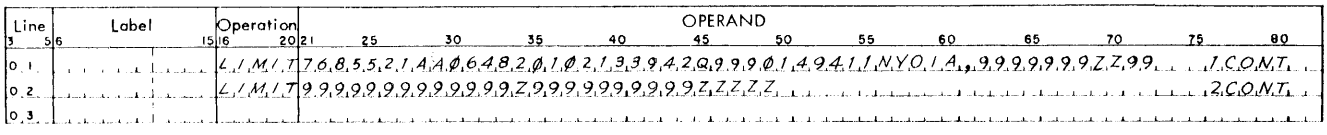


Figure 14. Use of Limits Continuation Card

Line (3–5)	Label (6–15)	Operation (16–20)	OPERAND (21–80)	
01		LIMIT	768552 1AAØ648 2Ø1Ø213394 2Q999Ø14941 1NYO1A,	1CONT
02		LIMIT	9999999ZZ9999999999999Z9999999999ZZZZZ	2CONT
03				

Figure 15. Alternate Use of Limits Continuation Card

OPERAND	NUMBER OF CHARACTERS	FUNCTION	DESCRIPTION
1	6	First cylinder	Address of the first cylinder to load in the form camcyl, where c is the channel ("1," "2," "3," or "4"), a is the access mechanism number ("0" or "1"), m is the module number ("0"-"9"), and cyl is the cylinder number ("000"-"249").
2	3	Last cylinder	The address of the last cylinder to load: "000" to "249."
3	2	HA2	The HA2 used for this file.
4	2	Cylinder load factor	The number of tracks in each cylinder to be loaded. The number may range from "02" to the maximum: the number of tracks in each cylinder made available for the file by the environment card. (If this operand is omitted the maximum load factor is used.)
5	3 2	Form 2 record load factor Form 1 record load factor	The number of logical records to be loaded in each physical record or track. The number may range from 001 (Form 2) or 01 (Form 1) to the maximum: the number of records per track (Form 1) or per block (Form 2) specified in the environment card.

• Figure 16. Cylinder Card Operands

Cylinder Card

Cylinder (CYL) cards supply cylinder addresses and loading factors required by the Load or Reload functions. At least one cylinder card must be included among the control cards for the Load or Reload functions, and must follow all other file organization cards. The user may provide the function with any number of additional cylinder cards. A cylinder card is read each time the function determines that more cylinders are required for the data file. Should more cylinder cards be supplied than are actually used by the Load or Reload function, extra cards are disregarded. Each is printed on the SPR with the informational message CARD NOT USED. The format of the cylinder card is:

16	21
CYL	*operands*

The operands for the cylinder card are illustrated in Figure 16. The first three operands of the cylinder card are required; the fourth and fifth may be omitted.

Figure 17 is a summary of all File Organization System control cards. The table illustrates which control cards and which operands on these control cards are required for each of the five File Organization System functions. A dash (—) indicates that the card or operand listed at the far left of the table must *not* be included for the particular File Organization System function named at the top of the column.

Messages

This section lists and explains the messages that may be generated during file organization, and provides information on the interpretation of message patterns caused by control card sequence errors. Console and SPR messages are presented separately with unnumbered messages in alphabetic order followed by numbered messages in numeric sequence. The significance of message numbers is explained in Figure 18.

NO RECORDS LOADED

Explanation: The Load phase has reached the end of the input data being processed. No records valid for loading have been found; for example, a deletion code was found in every record. (If the organizing function in process was a Reload, the original file has not been disturbed.)

Action: No operator action is possible. The user must re-examine the input data and take whatever corrective action is required and then re-execute the program.

PROGRAM NOT FOUND

Explanation: A file information card has been read—the card calls for an organizing function not part of the existing File Organization System.

Action: No operator action is possible. To use the organizing function specified, the user must redefine the system to include that function.

RECORD ADDITION DISCONTINUED WITH . . . (key)

Explanation: All overflow areas were exhausted during Add phase execution. Remaining records, beginning with the key shown, have not been added.

Action: No operator action is possible. The user must reorganize the file and re-execute the program to add additional records.

00551 LEVEL *n* WL

Explanation: The File Organization System has encountered a wrong length (too short) index or File Directory entry. The character *n* in the message indicates the level at which the error exists: F for File Directory, 0 for Level 0, 1 for Level 1, etc. An irrecoverable processing loop has been entered.

Action: Operator action must be taken to terminate execution. Either press INQUIRY REQUEST, type $10, and press INQUIRY RELEASE; or press COMPUTER RESET, then START.

30501 TWO ADDITIONAL DATA OVERFLOW TRACKS REQUIRED

Explanation: The Add phase has determined that further addition of records may cause an overflow. A waiting loop has been entered. A minimum of two additional data overflow tracks should be provided to ensure successful completion of the phase. If additional areas are not provided, addition of further records may cause the program to terminate unconditionally.

Action: Press INQUIRY REQUEST, then type one of the following:

$3xcamttttheeee—to provide additional overflow areas
where:
x is any character
c is the channel (1 for channel 1, etc.)
a is the access arm
m is the module
tttt is the beginning track address
ha is the home address
eeee is the ending track address.

$3xb—to continue record addition without providing additional overflow area
where:
x is any character
b is blank.

Press INQUIRY RELEASE.

30502 *n* ADDITIONAL INDEX OVERFLOW TRACKS REQUIRED

Explanation: Index adjustment during record addition has created the potential of an index overflow. To protect existing records, execution of the Add program has been suspended and a waiting loop has been entered. An additional index overflow area must be provided if record addition is to be resumed. The *n* in the console message indicates the number of tracks required for the additional area.

Action: Press INQUIRY REQUEST, then type one of the following:

$3xamtttthaamtttttha—to provide additional area
where:
x is any character
a is the access arm (beginning track)
m is the module (beginning track)
tttt is the track address (beginning track)
ha is the home address (beginning track)
The second amttttha stands for similar disk control information for the ending track.

NOTE: No channel is specified in the message reply. The channel is the same as that originally specified for the File Directory and Index Area on the GEN08 card at System Definition.

$3xb—to terminate execution
where:
x is any character
b is blank

Press INQUIRY RELEASE

30503 ADDRESS VERIFICATION

Explanation: Because of the importance that any disk area assigned at the console be an available area, this message always appears after the reply to message 30501 or 30502 has been entered. The message allows for visual verification of the preceding entry, and provides for final confirmation by the operator.

Action: Press INQUIRY REQUEST, then type:

$3xYES—If the previously entered area assignment is x correct.

$3xNO—If the previously entered area assignment is in error.
where:
x is any character

Press INQUIRY RELEASE.

NOTE: If the reply to this message was NO, the preceding message—30501 or 30502—will again appear at the console printer. This sequence will continue until the reply YES is given to message 30503.

SPR Messages

The messages listed and explained below are those produced on the SPR during file organization. The following terms are used throughout the message explanations that follow:

file information card—a control card containing, in columns 16-20, the mnemonic code of an organizing function. These codes are:

LOAD
RELOD
DELET
ADD
UNLOD

A function code is used whenever a specific function is being discussed. The general term "file information card" is used when the control card being discussed may reflect any of two or more organizing functions.

card set—the series of control cards that provide all of the parameters required for a given organizing function. A correctly arranged card set begins with a file information card and includes all of the "detail" control cards (IOA, ENVIR, etc.) required for the function named on the file information card. In the message explanations that follow, the expression "associated file information card," refers to the file information card of the card set of which the card being discussed is a part.

set terminator—a control card or condition interpreted by the Control phase as indicating that the end of a card set has been reached. A set terminator is considered to exist

whenever the Control phase encounters any one of the following:

1. A CYL card in a set beginning with a LOAD or RELOD file information card
2. A System Monitor (MON$$) control card
3. End of file on the SIU

In addition, any card preceding a file information card is considered the last card of a set.

Control Phase Messages

The Control phase examines the parameters and sequence of all file organization control cards and prints all control cards on the Standard Print Unit. If a parameter or sequence error is found, one of the below-listed diagnostic messages is printed on the SPR and the message CONTROL CARDS IN ERROR is produced at the console printer. (For the effect of a control card error on file organization execution, refer to the explanation of the console message.)

ANOTHER FILE CARD REQUIRED FOR KEY SEGMENTS

Explanation: Sequence error. After reading a file information card containing 1CONT in columns 76-80, the Control phase encountered the card printed immediately above this message. The card is not the required file information continuation card with 2CONT in columns 76-80.

CARD IMPOSSIBLE TO PROCESS

Explanation: The card printed immediately above this message cannot be interpreted by the Control phase. The information in columns 16-20 contains a key-punch error or is otherwise not valid for file organization.

CARD NOT USED

Explanation: This is an informational message to indicate to the programmer that execution of the LOAD or RELOD function performed did not require the use of the additional areas provided through the second or subsequent CYL cards (the cards printed immediately above this message). Those areas remain available for other use.

CONTROL CARDS IN WRONG ORDER

Explanation: Sequence error. A file information card is missing or out of sequence. This message appears in the relative position within the control-card printout where a file information card should have appeared.

CONVERSION CODE EXCEEDS KEY LENGTH

Explanation: There is an inconsistency in the control card printed immediately above this message. The complete key length (i.e., the sum of odd parameters 7 through 25) provides fewer characters than parameter 4 specifies for use in creating record addresses.

CONVERSION CODE INVALID

Explanation: The ENVIR card printed immediately above this message is inconsistent with the associated file information card. The fourth parameter of the file information card was present, indicating Form 1 records. Parameter 8 of the ENVIR card indicates that records are Form 2.

CYLINDER CARD NEEDED

Explanation: Sequence error. A CYL card is missing or out of sequence. This message appears in the relative position within the control card printout where the CYL card should have appeared.

CYLINDER CARDS ALLOWED ONLY WITH LOAD AND RELOAD

Explanation: The CYL card printed immediately preceding this message is out of order or unnecessary. The associated file information card specifies an organizing function other than LOAD or RELOD; no CYL card is required or permitted.

CYLINDER LOAD FACTOR GREATER THAN ENVIRONMENT

Explanation: The CYL card preceding this message is inconsistent with the associated ENVIR card. The fourth parameter of the CYL card specifies that more tracks per cylinder are to be loaded than were defined by the first- and last-track parameters (parameters 2 and 3) of the ENVIR card.

DESIGNATION OF KEY SEGMENTS INVALID

Explanation: Because of a format error or missing continuation card, the file information card does not completely define the key segment lengths and addresses to be used. For every even parameter beginning with the sixth there must be a corresponding odd parameter, and for every odd parameter beginning with the seventh there must be a corresponding even parameter.

DISK RECORD COUNT INVALID

Explanation: The IOA card preceding this message does not meet all of the following conditions:

1. Parameter 6 is mandatory for disk input or output records.
2. If the function is ADD, LOAD, or RELOD and there is disk input, parameter 8 must consist of five digits.
3. If the function is UNLOD and there is disk output, parameter 8 must be the characters IPI (without slashes).

ELIMINATION AND EXCEPTION CHAR MUST DIFFER

Explanation: The same character appears as the second and third parameters of the file information card preceding this message.

ENVIRONMENT CARD ALLOWED ONLY FOR LOAD

Explanation: The ENVIR card is out of order or unnecessary. The associated file information card specifies a function other than LOAD; no ENVIR card is required or permitted.

FIRST THREE PARAMETERS REQUIRED

Explanation: The CYL card preceding this message has been examined by the Control phase and has been rejected because one or more of the first three parameters is missing. (The first three parameters are mandatory.)

GROUP MARK INVALID IN FIRST POSITION

Explanation: The file information card preceding this message specifies a group-mark (≡) as a delete code (parameters 2 or 3). Parameter 1 of the same card indicates that the group-mark will appear as the first character of a record, which is not permitted.

INDICATED VS DESIGNATED KEY SEGMENTS DIFFER

Explanation: There is an inconsistency in the file information card preceding this message. The number of key segments specified by the length and address parameters (parameters 6 through 25) does not match the number of segments specified by parameter 5.

INFORMATION ALLOWED ONLY TO COLUMN 72

Explanation: The parameters contained on the LIMIT card preceding this message extend beyond the maximum permissible operand field; card columns 21-72.

The System Definition program combines the relocatable programs that perform the organizing functions into a multiphase File Organization System program. The user indicates the organizing functions to be included in the File Organization System program on the System Definition (SYSDF) control card.

SYSDF Control Card

The SYSDF control card designates the programs that will be part of the File Organization System and specifies the order in which the programs will normally be executed. The user also assigns a name to his program through this card. The format of the SYSDF card is:

6	16	21
name	SYSDF	*prog1, . . . , progn, number*

name: The user places in this field the name by which the produced File Organization System program will be identified. This name is to be used when the File Organization System program is executed; i.e., it is used as the first parameter on the MON$$ EXEQ card. The name may be one to ten alphameric characters, left-justified in the field. The first character must be alphabetic. Special characters may not be included anywhere in the name.

SYSDF: The letters SYSDF must appear in this field; this is the mnemonic by which the System Definition card is identified.

prog1, . . . ,progn: One to three parameters of this type may appear. Each is the name of one File Organization System program. The possible parameters are LOAD, ADD, ADD2, and UNLOAD. The order in which two or more programs are specified is the order in which they will appear in storage. The inclusion of LOAD provides the load, reload, and data deletion functions. If the Add program is to be included, either ADD or ADD2 (but not both) is specified. ADD is for use with IBM 1301 Disk Storage only; ADD2 is mandatory for 2302, and usable by 1301 but with less efficient use of core storage. (For a description of the modules, IBFOSADD and IBFOSADD2, corresponding to ADD and ADD2, refer to "Appendix B: Core-Storage Requirements.")

number: A two-position field specifying the number of tracks that have been allocated for the File Directory and Index area. (Refer to Appendix C.)

After analyzing the parameters, the System Definition program produces a set of card-image records that specifies the relocatable subroutines to be included in the desired File Organization System program. The records are written on a magnetic tape unit which the user must assign as symbolic unit MW2. The user directs the Linkage Loader to receive input from this unit by using the INPUT control card. The Linkage Loader converts the selected subroutines from relocatable to absolute format.

Relocation of the System Definition Program

The System Definition program, when not available on the SOF in absolute form, must be called in from the Library and relocated prior to execution. To do this, these control cards are required:

6	16	21
MON$$	EXEQ	LINKLOAD
	PHASE	DEFINE
	CALL	IBFOSYSDEF

These control cards cause the System Definition program to be placed on the Job file (MJB) in absolute form.

Execution of the System Definition Program

If the System Definition program is on the SOF in absolute form, the control cards required for execution are:

6	16	21
MON$$	EXEQ	DEFINE
name	SYSDF	*parameters*

These control cards cause input to the Linkage Loader program to be recorded on MW2, so that subsequent execution of the Linkage Loader program (with MW2 input) will produce an absolute File Organization System program on the Job file.

If the System Definition program is *not* on the SOF in absolute form, it must first be relocated as described above under "Relocation of the System Definition Program." Because the output of that step is on the MJB, the control cards required for execution are:

6	16	21
MON$$	EXEQ	DEFINE,MJB
name	SYSDF	*parameters*

These cards produce an absolute File Organization System program as described above.

User-Written Routines

User routines may be incorporated into File Organization System programs at System Definition time. If a routine is included in a program, an exit to that routine is taken during execution of the program.

Programming Considerations

The routine is entered prior to the processing of each record. Each user routine must be a closed subroutine, that is, the first executable instruction must be a Store B-address Register instruction. This instruction, the entry point to the routine, must have a symbolic label defined by the user in terms of a predetermined linkage symbol: OLOD/, OADD/, or OUNL/ for routines appended to the Load, Add, or Unload programs respectively. The last instruction executed by the routine must be a Branch instruction to the location stored by the first instruction. A typical coding sequence is:

```
ADD          TITLE        PROGRMNAME
                .
                .
                .
OADD/        DEFIN        USERLABEL
                .
                .
                .
USERLABEL    SBR          EXIT+5
                .
                .
                .
EXIT         B            0
```

If a user-written routine is incorporated into the Load program, the last position of the routine to be loaded must be defined by the linkage symbol OLDX/. This symbol is the base to which the Load program is relocated; the character defined by OLDX/ is overlaid. A typical coding sequence is:

```
LOAD         TITLE        PROGRMNAME
                .
                .
                .
OLOD/        DEFIN        USERLABEL
                .
                .
                .
USERLABEL    SBR          EXIT+5
                .
                .
                .
EXIT         B            0
```

Index register 13 is available to user-written routines as specified for the IBM 1410/7010 Operating System; (see "Information for Dependent Programs" in *IBM 1410/7010 Operating System; System Monitor*, Form C28-0319). The other index registers used in the routine must be saved and restored by the routine. The address of the key of the current record and the address of the current record are available to the user routine. The address of the current key is at symbol OKEY/. The address of the current record may be found in index register 3.

Inclusion of User Routines

User-written routines may be entered into the File Organization System from three sources:

Standard Input Unit (SIU)
Go File
Relocatable library (also contains the relocatable File Organization System programs)

If the routine is entered through the SIU, the user's relocatable routine must follow the SYSDF card in the SIU. The name of the File Organization System program into which the routine is to be inserted must be punched in columns 1-5 of the appropriate TITLE card, as LOAD, ADD, or UNLOD for the respective program.

If the user's relocatable routines are entered from the GO file or library, LINKLOAD CALLN cards for the relocatable routine must follow the SYSDF card in the control deck. The name of the File Organization System program into which the routine is to be inserted must be punched in columns 1-5 of the appropriate CALLN card, as described above for the TITLE card.

System Definition Program Messages

The following messages are produced during execution of the System Definition program. The console message 20540 ERROR ON SYSDF CARD OR USER ROUTINE, which also appears on the SPR, indicates that an error has been encountered. Diagnostic messages appear on the SPR; the action to be taken at the console is indicated by the console message. The action to be taken to correct the program error is suggested by the diagnostic message.

Console Message

20540 ERROR ON SYSDF CARD OR USER ROUTINE
Explanation: The System Definition program completed execution but one or more control-card errors were found. A waiting loop has been entered. The program processed any valid SYSDF cards that were present, but the cards in error were not processed. Any user routine named on an invalid SYSDF card has not been processed. Control cards in error have been printed on the SPR with one of the diagnostic messages listed below.
Action: Press INQUIRY REQUEST, then type:
$31—to continue processing.
$32—to discontinue processing.
Press INQUIRY RELEASE.

SPR Messages

ERROR IN SYSDF IDENTIFICATION
Explanation: The sequence of preceding control cards indicated that the card printed out with this message should be a SYSDF card. The card, however, is either an out-of-sequence card of another type, or a SYSDF card with an error in columns 16-20.

INVALID PARAMETER IN SYSDF CARD
Explanation: One of the parameters specified on the SYSDF card is in error.

NO FILE ORGANIZATION DEFINITION CARD
Explanation: On attempting to read the first SYSDF card, the System Definition program encountered end of file or a System Monitor control card on the SIU.

END OF FILE ORGANIZATION DEFINITION PROGRAM
Explanation: This message is for diagnostic use. It indicates that the program has completed execution, with or without having encountered control-card errors.

Printed in U.S.A. C28-0405-2 665/12 P

IBM

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