File Number S360(Mod.20)-30
Form C24-9007-5

DPS

IBM Systems Reference Library

IBM System/360 Model 20
Disk Programming System
Input/Output Control System

This publication provides information required for using the Disk Programming System (DPS) Input/Output Control System (IOCS) for the IBM System/360 Model 20. The publication contains the following information:

1. General description of the various input and output functions provided by the IOCS.
2. Definition of the record formats processed by the IOCS.
3. Description of the relationship between overlapping operations and the specification of different combinations of I/O areas and work areas.
4. Introduction to the concepts of file organization and file processing.
5. Detailed descriptions of the IOCS imperative macro instructions and the file definition statements.

The reader of this publication should be familiar with basic programming concepts and with the operating principles of his system as described in the applicable SRL publications. For a list of pertinent publications, refer to the IBM System/360 Model 20, Bibliography, Form A26-3565.

Sixth Edition (March, 1969)

This is a major revision of, and obsoletes C24-9007-4.

Most of the text has been reorganized and rewritten to make the publication easier to understand. These improvements are not marked.

The technical changes incorporated in the publication relate to the availability of the IBM System/360 Model 20, Submodel 5. The sections headed "Monitor Macro Instructions" and "The ATENT Routine" have been added. These technical changes and additions are marked in the following way: Changes to the text, and small changes to illustrations are indicated by a vertical line to the left of the change; changed or added illustrations are denoted by the symbol • to the left of the caption; added pages are flagged by the symbol • to the left of the page number.

This edition applies to the following components of IBM System/360 Model 20 Disk Programming System and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Input/Output and Monitor Macro Definitions version 3 modification 0
Printer-Keyboard Macro Definitions version 2 modification 0

Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/360 Model 20 SRL Newsletter, Form N20-0361, for the editions that are applicable and current.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

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

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Laboratories, Programming Publications, 703 Boeblingen/Germany, P.O. Box 210.

Contents

Introduction

The Disk Programming System (DPS) Input/Output Control System (IOCS) described in this publication consists of macro instructions which in turn select and generate routines that perform all input/output operations for card devices, printer, printer-keyboard, magnetic tape, and disk. The IOCS also supports the Magnetic Character Readers and the Binary Synchronous Communications Adapter. For details about the functions and features of the 1419/1259 DPS IOCS and the BSCA IOCS, refer to the SRL publications IBM System/360 Model 20, Disk and Tape Programming Systems, Input/Output Control System for the 1419 and 1259 Magnetic Character Readers, Form C33-6001; and IBM System/360 Model 20, Input/Output Control System for the Binary Synchronous Communications Adapter, Form C33-4001.

You can use the IOCS only in programs written in Assembler language. For details of the Assembler language refer to the SRL publication IBM System/360 Model 20, Disk and Tape Programming Systems, Assembler Language, Form C24-9002. Writing Assembler language programs with IOCS macro instructions enables you to achieve optimum time performance. Moreover, you can make use of extended overlay techniques, which results in a decrease of main-storage requirements and facilitates exit handling.

Routines for reading input data, writing output data, and controlling the input/output (I/O) devices form a large part of most programs written in the Assembler language. By using the routines supplied by IBM, you can avoid writing I/O routines for each of your programs. The time normally required for writing and testing I/O routines, you can thus use for actually solving the problem. The IOCS routines perform all required input and output operations. They ensure that machine interrupt conditions are handled properly, and that optimum overlapping of processing and input/output operations occurs.

The IOCS routines are stored, in the form of macro definitions, in the macro library of the disk-resident DPS. They can be included in a problem program through the use of macro instructions. You are required to:

1. describe the file by means of declarative macro instructions (referred to as file definition statements in this publication),
2. refer to the file in imperative macro instructions that cause the desired I/O operations, and
3. write your own exit routines that are entered automatically by the IOCS when an exit condition (e.g., end-of-file) occurs.

The Assembler uses the file definition statements and the imperative macro instructions to select macro definitions and generate routines that perform all I/O functions required by the problem program.

Data processing operations that can be performed by the IOCS include record storage, record retrieval, and record updating. A detailed description of each of these operations is given where the appropriate macro instructions are described. A brief summary is given below.

Record Storage: The IOCS provides for the storing of information by generating the routines required to punch records into cards, to list them on a printer or on the printer-keyboard, and to write them onto magnetic tape or disk.

Record Retrieval: The IOCS allows you to retrieve records from card, magnetic tape, and/or disk files, and records entered on the printer-keyboard in sequential order. For files in disk storage, the IOCS provides for the retrieval of records either in sequential or random order.

Record Updating: The IOCS allows you to retrieve a record from disk storage, update it, and then return it to the same location from which it was retrieved. (The updating of records in a card or tape file requires the entire file to be read as input to produce a new updated file as output. In the case of a card file, the updated information may be punched into the input cards.)

In addition to the functions described above, the IOCS is capable of:

- blocking and deblocking magnetic tape and disk records;
- switching between two I/O areas (if two areas are specified);
- handling end-of-file conditions;
- handling end-of-volume conditions;

- handling I/O error conditions; and
- performing I/O control functions such as card stacking, tape rewinding, seeking data on disk, etc.

All of these functions are provided by the IOCS for the processing of files organized according to any of the three available methods of file organization:

1. Sequential file organization, which provides for sequential processing of card, printer, printer-keyboard, magnetic tape, and disk records.
2. Direct-access file organization, which provides for random and sequential processing of disk records.
3. Indexed-sequential file organization, which provides for both sequential and random processing of disk records.

Machine Requirements

MINIMUM SYSTEM CONFIGURATION

Submodel 2

- An IBM 2020 Central Processing Unit, Model BC2 (12,288 bytes of main storage);
- an IBM 2311 Disk Storage Drive, Model 11 or 12;
- one of the following card reading devices:

  IBM 2501 Card Reader, Model A1 or A2,
  IBM 2520 Card Read-Punch, Model A1,
  IBM 2560 Multi-Function Card Machine (MFCM), Model A1;
- one of the following printers:

  IBM 1403 Printer, Model N1, 2, or 7,
  IBM 2203 Printer, Model A1.

Submodel 4

- An IBM 2020 Central Processing Unit, Model BC4 (12,288 bytes of main storage);
- an IBM 2311 Disk Storage Drive, Model 12;
- an IBM 2560 MFCM, Model A2;
- an IBM 2203 Printer, Model A2.

Submodel 5

- An IBM 2020 Central Processing Unit, Model BC5 (12,288 bytes of main storage);
- an IBM 2311 Disk Storage Drive, Model 11 or 12;
- one of the following card-reading devices:

  IBM 2501 Card Reader, Model A1 or A2,
  IBM 2520 Card Read-Punch, Model A1,
  IBM 2560 Multi-Function Card Machine (MFCM), Model A1;
- one of the following printers:

  IBM 1403 Printer, Model N1, 2, or 7,
  IBM 2203 Printer, Model A1.

MAXIMUM SYSTEM CONFIGURATION

Submodel 2

- An IBM 2020 Central Processing Unit, Model D2 (16,384 bytes of main storage); with or without a Binary Synchronous Communications Adapter (Feature No. 2074);
- two IBM 2311 Disk Storage Drives, Model 11 or 12 (both must be the same model);
- an IBM 2415 Magnetic Tape Unit, Model 1 through 6;
- an IBM 2501 Card Reader, Model A1 or A2;
- an IBM 1442 Card Punch, Model 5;
- one of the following card units:

  IBM 2520 Card Read-Punch, Model A1,
  IBM 2520 Card Punch, Model A2 or A3,
  IBM 2560 MFCM, Model A1;
- one of the following printers:

  IBM 1403 Printer, Model N1, 2, or 7,
  IBM 2203 Printer, Model A1;
- an IBM 2152 Printer-Keyboard;
- a 1419 or 1259 Magnetic Character Reader.

Submodel 4

- An IBM 2020 Central Processing Unit, Model D4 (16,384 bytes of main storage); with or without a Binary Synchronous Communications Adapter (Feature No. 2074);
- two IBM 2311 Disk Storage Drives, Model 12;

- an IBM 2560 MFCM, Model A2;
- an IBM 2203 Printer, Model A2;
- an IBM 2152 Printer-Keyboard.

Submodel 5

- An IBM 2020 Central Processing Unit, Model E5 (32,768 bytes of main storage); with or without a Binary Synchronous Communications Adapter (Feature No. 2074;
- four IBM 2311 Disk Storage Drives, Model 11 or 12;
- an IBM 2415 Magnetic Tape Unit, Model 1 through 6;
- an IBM 2501 Card Reader, Model A1 or A2;
- an IBM 1442 Card Punch, Model 5;
- one of the following card units:

  IBM 2520 Card Read-Punch, Model A1,
  IBM 2520 Card Punch, Model A2 or A3,
  IBM 2560 MFCM, Model A1;

- one of the following printers:

  IBM 1403 Printer, Model N1, 2, or 7,
  IBM 2203 Printer, Model A1;

- an IBM 2152 Printer-Keyboard;
- a 1419 or 1259 Magnetic Character Reader.

Data Files

Many types of data files are used in data processing applications. Theoretically there is no restriction on the logical content of information that can be processed, on the relationship of various units of information in the file, on the organization, or on the format.

To simplify the description of the use of the IOCS for card input or output, card files are considered to be either combined files or simple files. A combined file, which must be fed from one hopper of the I/O device, is a set of cards for which the IOCS performs both input and output operations, i.e., cards are to be read and punched during one pass through the I/O device. (Output data may be punched either into cards containing input data or into interspersed blank cards.) All other card files (input only or output only) are considered to be simple files.

LOGICAL RECORDS

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

RECORD BLOCKING

Blocking of records is the process of grouping a number of logical records before writing them on a storage device. A group of logical records is referred to as a block. Blocking improves processing efficiency by reducing the number of I/O operations required to process a file, and also saves storage space on the external medium on which the file resides because there are no gaps between the individual logical records in a block.

RECORD FORMATS

Logical records may be in one of three formats: fixed length (format-F), variable length (format-V) or undefined (format-U). The record format and whether or not the file is blocked are specified in the file definition statement for the file.

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

Format F

Format-F records are fixed-length records. Figure 1 shows one example of format-F records on magnetic tape (part A) and of format-F records on disk (part B). The number of logical records within a block (blocking factor) is normally constant for every block in the file unless the block is truncated (short block) by a TRUNC macro instruction.

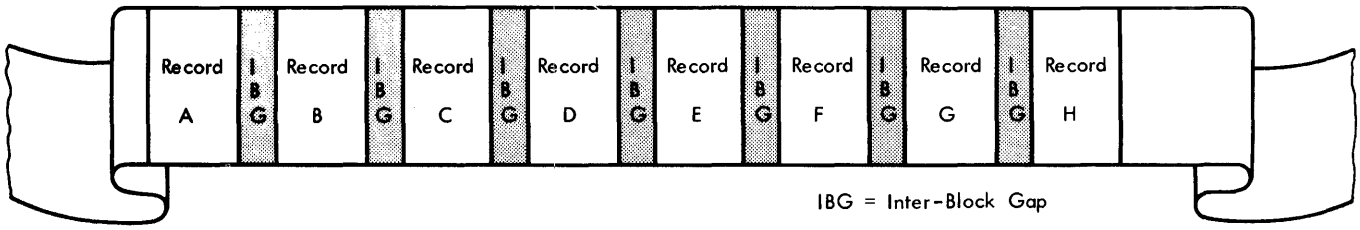
The TRUNC macro instruction serves to write truncated blocks on magnetic tape. Truncated blocks that may be contained in a tape input file are handled automatically by the IOCS.

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

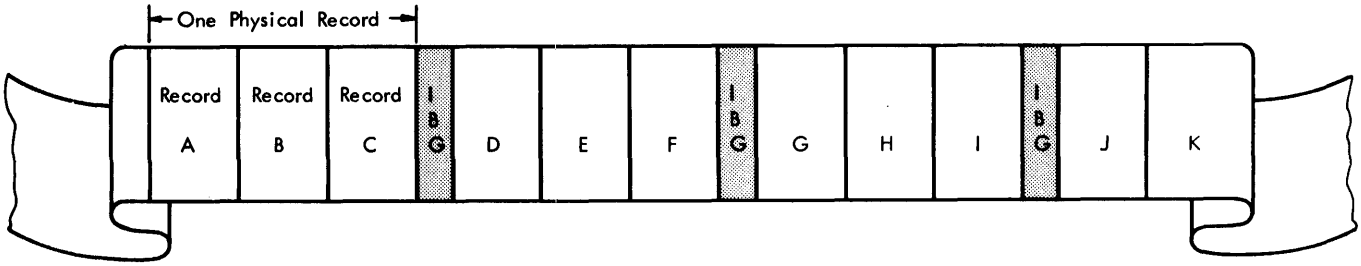
The IOCS performs physical-length checking on blocked format-F records and automatically handles truncated blocks. Because the channel and interruption system can be used for length checking and because blocking and deblocking are based on a constant record length, the IOCS processes format-F records faster than format-V records.

Format V

Format-V records are variable-length records, each of which describes its own length. Format-V records can be blocked. Each block of variable-length records includes a block length. The IOCS performs length checking of the block and makes use of the record length information in deblocking and blocking. Figure 2 is an example of format-V records on magnetic tape. The first four characters of each logical record contain control information. Specify the length of the logical record in the first-two characters when you create the record; the next two characters are reserved and must be binary zeros. The four bytes required for the length indication are included in the byte count for the record.

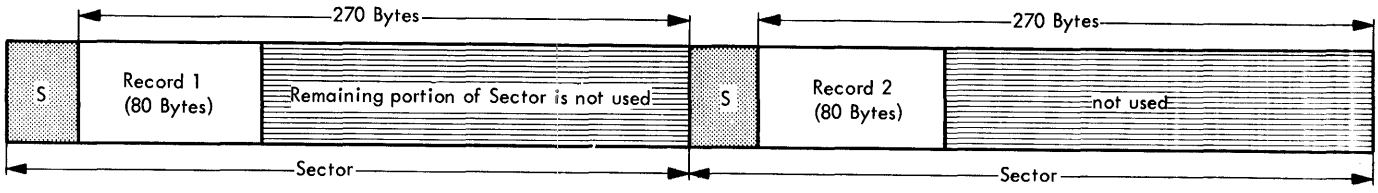


a. Unblocked Record Format

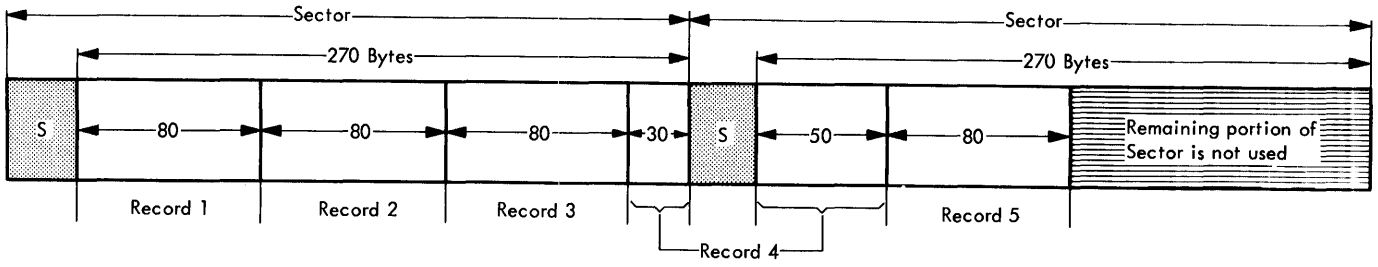


b. Blocked Record Format

Figure 1A. Example of Format-F Records on Magnetic Tape

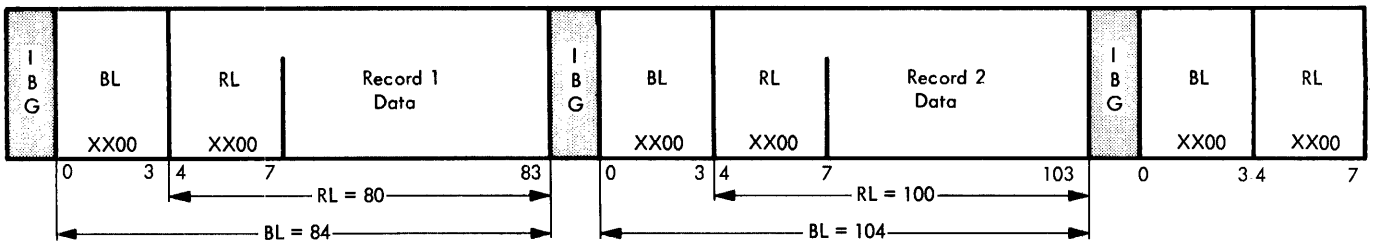


S = Sector Address
Unblocked Record Format

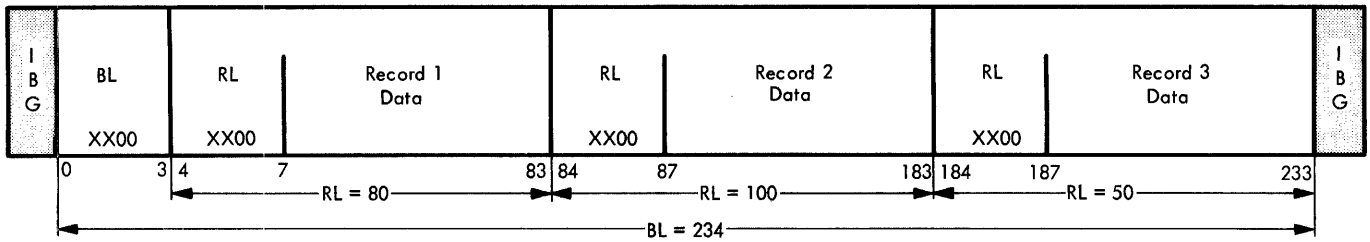


S = Sector Address
Blocked Record Format (assuming five Records per Block)

Figure 1B. Example of Format-F Records on Disk



a. Variable Length - Unblocked Record Format



b. Variable Length - Blocked Record Format

BL = Block Length
RL = Record Length
IBG= Inter-Block Gap

●Figure 2. Example of Format-V Records on Magnetic Tape

The first four characters of each block of format-V records contain block control information. The first two characters, which are provided by the IOCS at the time the records are blocked, specify the length of the block; the next two characters are reserved and must be binary zeros. Although these four characters do not appear in the record furnished to the problem program, the input and output areas must be large enough to accommodate them.

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

Format U

If the record format of a file is referred to as undefined, the record characteristics are unknown to the IOCS. Because each block is treated as an unblocked logical record, any blocking or deblocking must be performed in the problem program.

Allowable Formats for Files

The format of a file depends upon the type of I/O device used. See Figure 3 for the record format(s) permitted with each type of I/O device.

RECORD FORMATS	Format-F (fixed length)	Format-V (variable length)	Format-U (length undefined)
Blocked	Tape** Disk	Tape* **	
Unblocked	Card Printer Printer-Keyboard Disk Tape**	Tape* **	Tape**

* If a tape file is to be read backwards, format-V records are not allowed.

**The Data Conversion feature is required if a 7-track tape is used and one of the following conditions exists:

- Format-V records are to be read or written.
- Format-F or format-U records to be written contain EBCDIC characters other than those included in the BCD character set.
- Format-F or format-U records to be read have been written using the Data Conversion feature.

●Figure 3. Formats Valid in Accordance with I/O Devices

The IOCS is designed to overlap I/O operations with each other and/or with the processing of data.

Submodels 2 and 4. In the case of Model 20 Submodels 2 and 4, input/output overlap with processing is provided for the printer, the printer-keyboard, and for card reading and punching. Due to hardware characteristics of the Submodels 2 and 4, magnetic tape and disk input and output operations cannot be overlapped with processing or with card and printer I/O operations, except as follows:

1. The execution of tape or disk control operations (e.g., tape rewind, seek operation on disk, etc.) can be overlapped with card and printer I/O operations and/or processing.

2. Magnetic tape and disk I/O operations can be overlapped with printing on the IBM 1403 if the print operation has been started before the I/O instruction for tape or disk is issued.

Submodel 5. With the Submodel 5, full overlapping between I/O operations and processing is possible if the read/compute, write/compute (RWC) feature is utilized. If the RWC feature is not utilized, the I/O-processing capability is the same as with Submodels 2 and 4. The overlapping capabilities available with the Submodel 5 utilizing the read/compute, write/compute feature are shown in Figure 4.

The extent of overlapping is governed by the assignment of I/O areas and work areas in the source program. The choice of these areas can affect the amount of time that the CPU is available for processing. These areas and the effects of various combinations of them are described below.

I/O AREAS

An I/O area is an area into which input data is read or from which output data is written. The data read into or written from an I/O area consists of one block (i.e., one physical record).

For each card file, provide either two I/O areas and one work area or one I/O area and one work area in the source program.

For printer files, an output area must only be provided if the dual-feed carriage feature is used. Otherwise, only a work area should be provided.

Record Format	Number of I/O Areas	Separate Work Area	Amount of Effective Overlap
Unblocked	1	no	No overlap.
		yes	Overlap processing of all records. (Record move required).
	2	no	Overlap processing of all records. (No record move required).
		yes	Overlap processing of all records.
Blocked	1	no	No overlap.
		yes	Overlap processing of last record in each block.
	2	no	Overlap processing of full block.
		yes	Overlap processing of full block.
Note: Overlap given is the maximum achievable.			

Figure 4. Summary of Overlapping Capabilities with Read/Compute, Write/Compute Overlap Feature of Submodel 5

For each magnetic tape and sequential disk file you must define at least one I/O area in the source program; you may also provide a work area.

For sequential processing of disk and magnetic tape files with a Submodel 5 utilizing the RWC feature, you can specify two I/O areas to decrease the throughput time.

For direct-access disk files you must define one I/O area but no work area. For indexed-sequential files you can specify one to three I/O areas (IOAREAL, IOAREAR, IOAREAS) depending on the type of processing involved. You may specify the same area for IOAREAS, IOAREAR, and IOAREAL if you make sure that you finished processing the contents of one I/O area before you start using the area as another I/O area.

Specify the symbolic address of the I/O area(s) in the file definition statement for the appropriate file to be processed. The size of an I/O area must be equal to the length of the longest block to be processed and, for disk files, must be a multiple of 270. When unblocked records in indexed-sequential files are processed, add six bytes to the I/O areas to accommodate the sequence-link field. With the exceptions mentioned in the section One I/O Area and a Work Area, under the heading Non-Overlap Mode, do not use these I/O areas for any other purpose in the problem program.

WORK AREAS

A work area is an area that is used for processing one logical record. The IOCS moves one logical record from an input area to a work area or from a work area to an output area. If a work area is to be used (a work area must be used for card and printer files, for printer-keyboard output files, and for some indexed-sequential files), you must define it in the source program and indicate in the file definition statement for the appropriate file that a work area is to be used. In addition, any GET or PUT macro instruction that refers to the file must specify the symbolic address of the work area used. The use of work areas is not limited to one per file. For example, you may use a different work area for every alternate GET or PUT macro instruction. However, you must specify only one work area in any GET or PUT macro instruction. The advantages of using a work area are explained below under I/O-Work Area Combinations.

If a work area is not specified, the IOCS makes all records of that file available in the I/O area(s).

The length of a work area must be equal to the length of the longest logical record. Note that the record length of format-V records is contained within the first four bytes of the record (see Figure 2). The problem program must include provisions for handling the record length (e.g., it must insert the record length into output records). The use of a work area permits the overlapping of I/O operations and internal processing, thus reducing processing time.

I/O-WORK AREA COMBINATIONS

For a particular file, you can specify one of the I/O-work area combinations as shown in Figure 5.

I/O operations may require the use of up to two registers. The record format in conjunction with the I/O-work area combination used determines whether none, one or two registers must be specified. To determine when it is necessary to specify a register, refer to Figure 26 in the section Programming Considerations.

No I/O Area and a Work Area

Data to be printed on the standard carriage of an IBM 2203 or on the IBM 1403 is printed from the first 144 main storage positions that are used as a print buffer. This is a hardware characteristic. A PUT macro instruction for a printer file (1) causes the output data to be transferred from the specified work area to the print buffer area and (2) initiates the print operation.

One I/O Area

The specification of just one I/O area is permitted for magnetic tape and disk files and is mandatory for printer-keyboard input files.

When a GET or a PUT macro instruction is issued while another I/O operation is being executed, the program enters a waiting loop and remains there until the current I/O operation is completed.

One I/O Area and a Work Area

You must indicate the use of a work area in the file definition statement for the file. Also, define the work areas to be used in your program and assign a name to each of them. That name is then specified as the second operand of each GET or PUT macro instruction you issue.

CARD AND PRINTER FILES: For card and printer files, the use of a work area is mandatory. It permits the IOCS to overlap an I/O operation with processing and/or with another I/O operation.

The following considerations apply to the use of I/O areas as work areas for files being processed in the overlap and the non-overlap mode:

Overlap Mode. I/O areas for files processed in the overlap mode must not be used as work areas. During processing, a given record is processed in the specified work area while other records are simultaneously read into an input area or punched or printed from an output area.

Non-overlap Mode. For combined files, only the punch area may also be used as a work area. For simple files, the input or output area may be used as a work area. Card-print areas must not be used as work areas.

PRINTER-KEYBOARD OUTPUT FILES: Printer-keyboard output files require a work area that you must provide in the problem program. In addition, an output area is required. If no output area is provided in the problem program, the output area allocated at the time the Monitor is generated is used.

MAGNETIC TAPE AND DISK FILES: The use of a work area with Submodels 2 and 4 and with a Submodel 5 not utilizing the RWC feature may provide the advantage of additional processing time becoming available by allowing to optimize overlap between I/O operations with processing.

When a work area is used, processing can be done in a fixed area and no I/O register is required. These are further advantages offered by the use of a work area.

Two I/O Areas

For disk and magnetic tape files, the use of two I/O areas with a Submodel 5 utilizing the RWC feature is recommended if the processing time for the last (or only) record of a block is shorter than the time required to read the next record (block). Two I/O areas are used for sequential disk or magnetic tape files.

Two I/O Areas and a Work Area

Two I/O areas and a work area must be specified for combined files and may be specified for simple card files read on a 2501 Card Reader that is working in overlap mode. This allows the IOCS to maintain maximum card reading speed.

For magnetic tape and sequential disk files you should consider the use of two I/O areas and a work area only if these files are processed by a Submodel 5 utilizing the RWC feature. No additional throughput advantage can be gained from specifying a work area in addition to the two I/O areas. However, it might be of advantage that no I/O register is required when a work area is specified.

I/O-Work Area Combinations for Indexed-Sequential Files

For indexed-sequential files you can specify up to three I/O areas and up to four work areas depending on the type of processing involved (see Figure 5).

•Figure 5. I/O-Work Area Combinations

DEVICE & PROCESSING MODE		FIRST IOAREA		SEC. IOAREA		THIRD IOAREA		FIRST WORKAREA		SEC. WORKAREA		THIRD WORKAREA		FOURTH WORKAREA	
2560 MFCM1/MFCM2	simple	IOAREA1	M	-		CRDPRA	O	WORKA	M	-		-		-	
	combined	INAREA	M	OUAREA	M	CRDPRA	O	WORKA	M	-		-		-	
2560 Card Read Punch	simple	IOAREA1	M	-		-		WORKA	M	-		-		-	
	combined	INAREA	M	OUAREA	M	-		WORKA	M	-		-		-	
2520 Card Punch		IOAREA1	M	-		-		WORKA	M	-		-		-	
1442 Card Punch		IOAREA1	M	-		-		WORKA	M	-		-		-	
2501 Card Reader		IOAREA1	M	IOAREA2	O	-		WORKA	M	-		-		-	
1403 Printer		-		-		-		WORKA	M	-		-		-	
2203 Printer	standard	-		-		-		WORKA	M	-		-		-	
	dual feed	IOAREA1	M	-		-		WORKA	M	-		-		-	
2152 Printer-Keyboard	input	IOAREA	M	-		-		-		-		-		-	
	output	IOAREA	O	-		-		WORKA	M	-		-		-	
2415 Magnetic Tape Unit		IOAREA1	M	IOAREA2*	O	-		WORKA	O	-		-		-	
2311 Disk (sequential)		IOAREA1	M	IOAREA2*	O	-		WORKA	O	-		-		-	
2311 Disk (direct-access)		IOAREA1	M	-		-		-		-		-		-	
2311 Disk (indexed-sequential)															
LOAD		IOAREAL	M	-		-		WORKL	M	-		-		-	
ADD		IOAREAL	M	-		-		WORKL	M	WORKA	M**	-		-	
ADDRTR															
sequential		IOAREAL	M	IOAREAS	M	-		WORKL	M	WORKA	M**	WORKS	O	-	
random		IOAREAL	M	IOAREAR	M	-		WORKL	M	WORKA	M**	WORKR	O	-	
random/sequential		IOAREAL	M	IOAREAR	M	IOAREAS	M	WORKL	M	WORKA	M**	WORKR	O	WORKS	O
RETRVE															
sequential		IOAREAS	M	-		-		WORKS	O	-		-		-	
random		IOAREAR	M	-		-		WORKR	O	-		-		-	
random/sequential		IOAREAR	M	IOAREAS	M	-		WORKR	O	-		WORKS	O	-	

* Only useful with a Submodel 5 utilizing the read/comput, write/compute feature
** Mandatory only for a file containing blocked records
M Mandatory Specification
O Optional Specification

When planning your input and output files, consider the following:

1. processing requirements for storing, updating, or displaying data, and

2. the I/O devices available.

Card, magnetic tape, printer, and printer-keyboard files are organized in a sequential order, because they can be processed only in the existing sequence. Disk files can also be organized and processed consecutively. However, for disk files you are not restricted to sequential file organization, but have the option of working with three methods of file organization and corresponding methods of file processing.

For disk files, it is important to distinguish between two terms:

1. File Organization refers to the method of arranging data records on a direct-access storage device; it is the technique used to "load" the file.

2. File Processing is the method of retrieving records from, adding records to, or updating records in a file.

Note: Files on disk may consist of more than one disk area (extent); the extents of a disk file need not be adjacent and may be contained in more than one volume. The lower and upper limit of a single extent must be contained within one volume. The XTENT control statement, which is used to specify the extents of a disk file, is described in the SRL publication IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006.

FILE ORGANIZATION

Card, magnetic tape, printer and printer-keyboard files are organized as sequential files. For disk files, the data records can be organized as a sequential, direct-access or indexed-sequential file. With disk, more than one method of processing may be used for a single method of file organization. The method of file organization best suited to a particular file depends on the processing requirements for the file.

Sequential File Organization

Sequential file organization means that the records are written consecutively on the storage medium. The physical order of the records prior to organization of the file determines both the physical order of the organized file and the sequence in which the records will be subsequently processed. A sequentially organized file is normally established by sequentially "loading" records that have been pre-sorted on a significant control field within each record. In this case, the last logical record is located in the last physical position of the file on the storage medium.

A sequentially organized card or magnetic tape file can only be processed in the order in which the records physically occur, i.e., sequentially. (This is also the most efficient method for processing a sequential disk file.) Thus, sequentially organized files are subject to certain processing limitations such as:

- The only way to retrieve or update a record in a sequential file is to read every record in the file beginning with the first. Therefore, sequential file organization is the most efficient method if a large number of records in the file are updated or examined every time the file is processed. An extremely low level of activity, on the other hand, justifies the use of another method of file organization that permits random processing of the file.

- Additions and deletions can only be accomplished by copying the entire file. During the copying, the records to be added are merged in and the records to be deleted are excluded.

Sequential file organization is used for all card and magnetic tape files, printer, and printer-keyboard files.

The sequentially organized disk file is similar in concept to a sequentially organized card or magnetic tape file. Sequential disk files differ from card and magnetic tape files in two ways:

1. If processing involves only the updating of records already in the file, an updated record may be rewritten into the same physical location from which it was retrieved and records that are to remain unchanged need not be rewritten at all. (With a card or

magnetic tape file, the updated records, together with any unchanged records, must be stored in a newly created file.)

2. A sequentially organized disk file may be processed randomly by specifying the IOCS instructions used for direct-access files. You must know and specify the actual physical disk address of the record to be retrieved. In your routines, you must also consider the blocking factors of the sequential file, i.e., you must deblock the file.

Direct-Access File Organization

Both the sequential and indexed-sequential methods of file organization involve records that are stored in some logical sequence and are usually processed in that sequence. With the direct-access method of file organization, records are retrieved from or written onto a physically addressed location on disk. The physical disk address of a record to be loaded or retrieved must be calculated in the problem program. Determine a randomizing "formula" to convert certain data within the record to a physical address on disk; the record is stored at the physical address developed by the randomizing formula. Thus, normally, in a file that is being loaded, the records are not placed in contiguous locations on the disk but are "scattered" throughout the area of the pack that is to contain the file.

In selecting the best method for loading a direct-access file, it is necessary to keep two things to a minimum:

1. the number of different records for which the same disk address is derived, and

2. the amount of storage space required, i.e., minimize the amount of wasted storage space.

A file written on disk by the direct-access method may be processed randomly or sequentially. Random retrieval from a direct-access file is generally faster than random retrieval from an indexed-sequential file. The direct-access method, however, is not best suited to retrieval of records in a logical sequence.

Indexed-Sequential File Organization

An indexed-sequential file is organized from records that have been sorted according to specific control information, i.e., keys, contained in each record. The structure of an indexed-sequential file is basically sequential, but this type of file organization has the following features which distinguish it from sequential file organization:

- The process of locating records by referencing record keys permits the option of processing an indexed-sequential file in either sequential or random order.

- When an indexed-sequential file is loaded the IOCS constructs indexes to be used to locate records in subsequent processing. Sequential retrieval through use of these indexes is almost as efficient as sequential processing with a sequential file. In addition, these indexes make it possible to retrieve individual records in random order.

- In a sequential file, the original sequence can be maintained only by copying the entire file and inserting the additions in the appropriate location. In an indexed-sequential file, overflow areas can be reserved to accommodate additions.

As the number of additions increases, the efficiency of processing an indexed-sequential file decreases. This is due to the additional access-arm movement required to read records that have been forced onto the reserved overflow tracks. Therefore, there is a point at which it becomes advisable to reorganize an indexed-sequential file. (That is, to create a new file from the old one, and, in the process, to exclude all records tagged for deletion. In the same operation, the IOCS merges all records in the overflow area into the main file.)

When the number of additions and deletions (or even updates) to be made regularly in a file is high, sequential file organization saves processing time.

Two other factors should be considered when indexed-sequential file organization is used:

1. An indexed-sequential file may be stored on more than one volume, but all of these volumes must be on line during any type of processing, whereas a sequential file may be stored on any number of volumes, which can be mounted and processed consecutively.

2. An indexed-sequential file cannot be direct input to the Model 20 DPS Sort/Merge program and to the file-to-file Utility programs.

FILE PROCESSING

The IOCS provides for the processing of records in sequential order for sequentially organized files, in random or sequential order for direct-access files, and in random or sequential order for indexed-sequential files. Both the direct-access and the indexed-sequential file organization methods apply only to disk files.

Processing Sequential Files

Sequential processing is used to read, write, and process consecutive records in a file. Cards are processed in the order in which the cards are read. Tape records are processed beginning with the first record continuing through the records to the last one. Disk records are processed beginning with a starting disk address and continuing through the records on successive tracks and cylinders to the ending disk address.

The macro instructions GET and PUT are used to cause the transfer of data from and to sequential files. In the case of a Model 20, Submodel 5, the transfer of data to and from the I/O devices overlaps fully with processing. In all other cases, the transfer of data in printer, printer-keyboard, and card files overlaps with processing, unless processing in non-overlap mode has been specified. The extent of overlapping depends on the assignment of I/O areas and work areas. Regardless of the extent of overlapping, when a GET macro instruction has been executed, the desired record is available for processing. Similarly, when a PUT macro instruction has been executed, you can begin building the next output record for the same I/O device.

Processing Direct-Access Files

The IOCS provides routines to read, write, and process disk records that are organized according to the direct-access method. The IOCS locates a disk record for processing by referring to the physical disk address which must be supplied in the problem program.

The macro instructions READ and WRITE cause the transfer of data from and to files when the direct-access method is used. These macro instructions permit records to be retrieved from or placed into a file. They also permit the updating and replacing of records in a file. When the record is required for processing, the problem program must use a WAITF macro instruction to ensure that the transfer of data has been completed before processing continues.

Direct-access files can be processed sequentially. However, these files are not best suited to retrieval of records in a logical sequence.

Processing Indexed-Sequential Files

For indexed-sequential files, the IOCS provides routines to perform the following functions:

1. Loading the file.
2. Extending the file with records that are all higher in sequence than those already loaded.
3. Adding records in sequence without copying the entire file.
4. Retrieving records (with or without updating) either sequentially or randomly.

Any record stored at any location in the logical file can be retrieved randomly. The problem program supplies the control information (key) of the desired record; the IOCS initiates a search for the record and makes it available for processing.

If an indexed-sequential file is processed sequentially, the key of the first record to be processed is specified in the problem program. The records are made available, one after the other. When a macro instruction requires another record, the IOCS retrieves the succeeding record from the logical file in the order determined by the key, until the problem program terminates the operation.

The macro instructions WRITE and READ cause the transfer of data to and from an indexed-sequential file when the records are loaded or when they are processed in random order. The macro instructions GET and PUT are used when the records of an indexed-sequential file are processed sequentially.

A READ or WRITE macro instruction causes the I/O operation to be initiated. When the record is required for processing, the problem program must use a WAITF macro instruction to ensure that the transfer of data has been completed before processing continues. With GET and PUT macro instructions, no subsequent WAITF macro instruction is necessary.

IOCS Macro Instructions

IBM supplies two types of macro instructions for the input/output control of records from various I/O units:

- declarative macro instructions (hereafter referred to as file definition statements), and
- imperative macro instructions.

These instructions are discussed in detail on the following pages. The description of the instructions is divided into several sections according to the device used and the type of file being processed: card, printer, printer-keyboard, magnetic tape, sequential disk, direct-access disk, and indexed-sequential disk.

Some of the IOCS macro instructions pertain to all files irrespective of the device used or the type of file organization involved. These instructions are described in the sections Begin and End Definition Statements (DTFBG, DTFEN) and Instructions for Opening and Closing Files (OPEN, CLOSE).

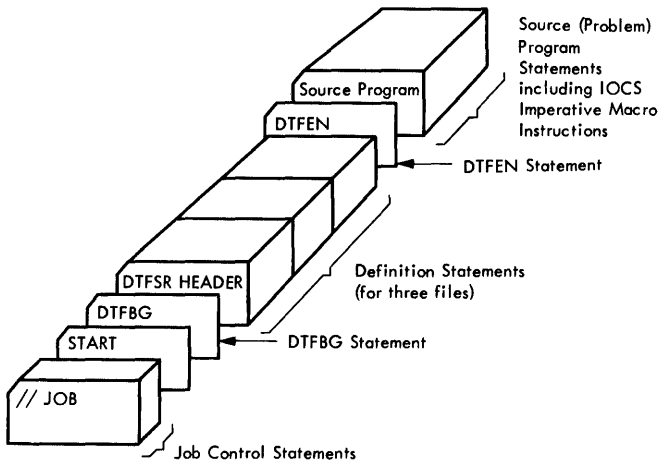
The following conventions apply to the description of the IOCS macro instructions in this publication:

1. Upper-case letters and punctuation marks (except as described in items 3 and 4 below) represent information that must be coded exactly as shown.
2. Lower-case letters and terms represent information that you must supply.
3. Information that is contained within brackets [ ] represents an option that can be included or omitted depending on the requirements of the program.
4. An ellipsis (a series of three periods enclosed by commas) indicates that a variable number of items may be included.

Assembly Procedure

The file definition statements are used by the Assembler to generate those routines that are required to perform the desired I/O operations when the program is executed. The imperative macro instructions cause the generation of linkages to these generated routines. The IOCS routines and the problem program written in the Assembler language are assembled in one run.

Figure 6 shows the arrangement of the assembly input deck for a source program using IOCS macro instructions.



•Figure 6. Arrangement of Source Program Cards Using the IOCS

Diagnostic Messages

Diagnostic messages are provided to indicate error conditions at assembly time such as missing operands of macro instructions, inconsistent combinations of operands, etc. This checking is in addition to that normally performed by the Assembler program.

File Definition Statements

The file definition statements describe the logical file, indicate the type of processing to be used for the file, and specify main-storage areas for the file. It depends on the device used and on the type of processing involved which of the different file definition statements applies to your file. Only two definition statements (DTFBG, DTFEN) are used irrespective of the type of file. The file definition statements are:

DTFBG Define The File BeGin. This definition statement, if present, must precede all other file definition statements. The statement is mandatory if the generated program is to be used as an inquiry program or if it is to be used as a mainline program that permits interrupts by inquiry programs.

DTFSR Define The File for a Serial Record device. This definition statement is used in conjunction with card and printer files.

DTFPK Define The File for a Printer-Keyboard. This definition statement is used to define a printer-keyboard file. When form skipping and overflow printing are desired for a printer-keyboard output file, you must also provide a DTFLC statement.

DTFLC Define The Line-Counter table. The DTFLC statement is used in conjunction with a printer-keyboard output file if form skipping and overflow printing are desired. The DTFLC statement describes the line-counter table, which simulates a carriage-control tape for the printer-keyboard.

DTFMT Define The File for a Magnetic Tape. This definition statement is used to define a file associated with a magnetic tape device.

DTFSD Define The File for a Sequential file organization on Disk. This definition statement is used whenever a sequentially organized disk file is to be processed.

DTFDA Define The File for a Direct-Access file organization. This definition statement is used whenever a disk file of direct-access organization is to be processed.

DTFIS Define The File for an Indexed-Sequential file organization. This definition statement is used whenever a file of indexed-sequential organization is to be processed.

DTFEN Define The File ENd. A DTFEN statement must follow the last definition statement for each program.

At the time of assembly, the file definition statements for the file to be processed must follow the START statement. The file definition statements may appear in any order with the following exceptions:

- The DTFBG statement, if specified, must be the first file definition statement in the program.
- All DTFSR statements must be written contiguously, i.e., DTFSR statements must not be separated by another type of definition statement.
- The DTFEN statement must be the last file definition statement in the program.

A summary of the file definition statements is given in Appendix A.

FORMAT OF FILE DEFINITION STATEMENTS

A file definition statement consist of (1) a header entry that assigns a name to the specified file and (2) detail entries that are required tc define parameters such as the device to be used, the mode of processing, etc.

Note that all file definition cards, except the last one for each file, must have a continuation punch in column 72. This continuation punch may be any character. Punching in continuation cards must begin in column 16.

Figure 7 is an example of a DTFMT file definition statement followed by a DTFEN statement.

Header Entries

A header entry consists of a file name in the name field (starting in column 1) and the mnemonic of the file definition statement in the operation field, which follows the name entry with at least one intervening blank. The name entered in the name field may consist of up to seven characters; the first of these characters must be an alphabetic character other than "I".

The file name assigned in a header entry for a file must be used in all imperative macro instructions that refer to this file.

Detail Entries

A detail entry (except in the DTFEN and DTFLC statements) is composed of a keyword immediately followed by an equal sign (=) which is, in turn, followed by a specification.

Each specification must correspond to the rules and restrictions of programming in the Assembler language. Expressions are permitted for all detail entries that require the specification of a symbolic address. The length of a specification is limited to eight characters, and blanks are not permitted. (Note that a blank within a detail-entry specification causes the Assembler to treat the remaining detail entries as comments). A comma must immediately follow the specification of each detail entry, except the last (see Figure 7).

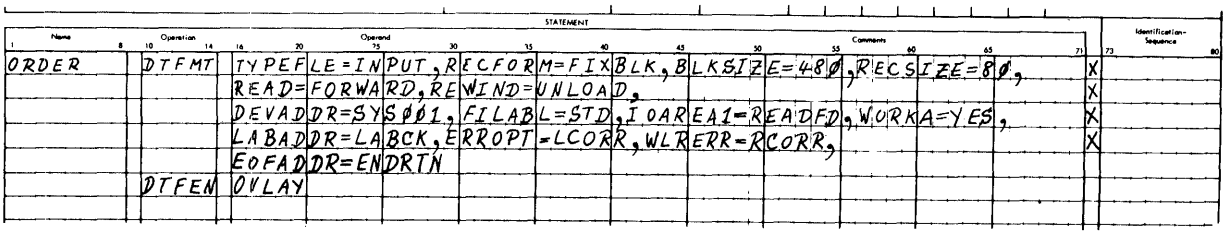


Figure 7. DTFMT Statement Followed by a DTFEN Statement

The detail entries describe the file and specify symbolic addresses of routines and areas used during the processing of a file. The detail entries may appear in any order. You should include only those entries that are applicable to a particular file or program.

Note: The DTFEN and DTFLC statements are written according to the coding rules for positional macro instructions as described in the SRL publication IBM System/360 Model 20, Disk and Tape Programming Systems, Assembler Language, form C24-9002. The detail entries of positional macro instructions must be written in a given sequence. A comma must immediately follow the specification of each detail entry, except the last; a blank indicates the end of the sequence of entries.

Imperative Macro Instructions

Imperative macro instructions are included in the problem program. They perform such functions as opening a file, making records available for processing, writing records that have been processed, etc. The macro instructions IBM provides for input/output control are described in separate sections as follows:

- Instructions for opening and closing files: OPEN, CLOSE.
- Instructions for processing card files: PUT, GET, CRDPR, CNTRL, EOM, LOM, WAITC.
- Instructions for processing printer files: PUT, CNTRL, PRTOV.
- Instructions for processing printer-keyboard files: PUT, READ, WAITF, CNTRL, PRTOV.
- Instructions for processing magnetic tape files: PUT, GET, CNTRL, TRUNC, RELSE, LBRET, FEOV.
- Instructions for processing sequential disk files: PUT, GET, CNTRL.
- Instructions for processing direct-access disk files: WRITE, READ, WAITF, CNTRL, CNVRT.
- Instructions for processing indexed-sequential disk files: WRITE, READ, WAITF, PUT, GET, SETFL, ENDFL, SETL, ESETL.

All macro instructions listed above are written according to the coding rules for positional macro instructions as described in the SRL publication IBM System/360 Model 20, Disk and Tape Programming Systems, Assembler Language, Form C24-9002.

The possible variations of the imperative macro instructions are summarized in Appendix B.

DTFBG Statement

This statement, if used, precedes all other file definition statements in the source program. The DTFBG statement is mandatory if a program is to be executed as an inquiry program and/or as a mainline program permitting inquiry interrupts. The statement is optional for all programs that are not executed as inquiry programs and do not allow inquiry interrupts (see No Operands Specified below).

The DTFBG statement has the following format.

Name	Operation	Operands
	DTFBG	[detail entry]

The name field must be blank and the operation field must contain DTFBG. The operand field may be blank, or may contain one or two detail entries. The detail entries that may follow a DTFBG header entry are shown in Figure 8.

Note: To be compatible with the Tape Programming System, the specification RWC=YES in the operand field of the DTFBG statement does not lead to an error but is ignored.

MAINPRG=YES

Specify this entry if the program is to function as a mainline program that permits interrupts by inquiry requests. You cannot use the program as an inquiry program. The Open routines for disk files provide for file protection (see File Protection in the section The Inquiry Program). If you specify ATENT=YES together with MAINPRG=YES, the MAINPRG entry is ignored because a program using the ATENT entry cannot be executed as a mainline program that allows inquiry interrupts.

INQPRG=YES

This specification in the DTFBG statement allows you to use the program as an inquiry program. When an inquiry request calls a program assembled with the entry INQPRG=YES, the Open routine for any disk files provides for file protection as described in the section The Inquiry Program.

A program you specified as an inquiry program may be executed as a mainline program. When the program is loaded as a mainline program, a warning halt occurs. The operator may continue the job if the Monitor input area is not used in the program.

Detail Entries	Execution of Program		Inquiry Interrupt Permitted
	Mainline Program	Inquiry Program	
MAINPRG=YES,INQPRG=YES	Yes	Yes	Yes (Mainline only)
MAINPRG=YES	Yes	No	Yes
INQPRG=YES	Yes*	Yes	No
ATENT=YES	Yes	No	No
No Operand or no DTFBG Statement	Yes	No	No

* A program specified as an inquiry program may also be executed as a mainline program. When the program is loaded as a mainline program, a warning halt occurs. The operator may continue the job if the Monitor input area is not used in the program.

●Figure 8. Detail Entries of the DTFBG Statement

Enter both operands if the program is to be executed as both mainline and inquiry program. If you specify these two detail entries, do not use the IQIPT macro instruction to process data in the inquiry input area in the Monitor. A record is read into this area only if the program is used as an inquiry program, but not if it is executed as a mainline program.

Note: The main-storage requirements of the problem program increase if you specify MAINPRG=YES,INQPRG=YES.

File protection is provided in the Open and Close routines for programs assembled with the specification MAINPRG=YES,INQPRG=YES. For details refer to the section The Inquiry Program.

ATENT=YES

Specify the ATENT entry if you provide your own ATENT subroutine in the problem program. (For details on the ATENT subroutine refer to the section The ATENT Routine). You can enter this subroutine by pressing the Request key on the printer-keyboard.

Since a program using the ATENT=YES entry does not allow inquiry interrupts, do not specify MAINPRG=YES and/or INQPRG=YES together with ATENT=YES. An error occurs if you specify ATENT=YES and INQPRG=YES. Specifying MAINPRG=YES and ATENT=YES leads to ignoring the MAINPRG entry.

No Operands Specified

Specifying the DTFBG statement without an operand has the same effect as not specifying a DTFBG statement at all, i.e., the object program can be used only as a mainline program that does not allow inquiry interrupts. All inquiry requests are rejected, and no protection is incorporated in the Open routines for disk files.

This option is recommended to minimize the main-storage requirements of the object program if the installation does not use the inquiry function.

DTFEN Statement

To indicate that all files have been defined you must issue a DTFEN statement as the last file definition statement in the problem program. The DTFEN statement has the following format.

Name	Operation	Operand
	DTFEN	[OVLAY]

The name field of a DTFEN statement must be blank. The operation field contains DTFEN. The operand field may be blank, or it may contain OVLAY (overlay). The overlay technique as described in the section Programming Considerations allows you to reduce the number of main storage positions required by the program when magnetic tape or disk files are involved.

This section discusses the macro instructions required to activate and deactivate a file, and the actions the IOCS performs when a file is opened or closed.

Before the first record can be read from any input file or transferred to any output file by IOCS macro instructions, you must ready that file by issuing an OPEN macro instruction. Likewise, you must deactivate the file and terminate all pending requests by issuing a CLOSE macro instruction after all records of that file have been processed. The OPEN and CLOSE macro instructions are described below.

OPEN MACRO INSTRUCTION

The format of the OPEN macro instruction is as follows:

Name	Operation	Operands
[name]	OPEN	filename1,...,filename16

Each specification in the operand field is the name of a file (assigned to it by an entry in the name field of the appropriate file definition statement) to be opened with this macro instruction. Any number of files from one to sixteen on various devices may be opened with one OPEN macro instruction. The operations performed depend on the type of device involved and the labeling technique (if applicable).

For information on label processing and label formats refer to the SRL publication IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006.

The actions the OPEN macro instruction performs for the different devices are described in this section under Initializing Files.

CLOSE MACRO INSTRUCTION

Use this macro instruction to deactivate any file that was previously made available by an OPEN macro instruction. You can close a file at any time by issuing a CLOSE macro instruction. You must issue the CLOSE macro instruction after all records in an input file or output file have been processed. When writing your own Exit routines, make sure that you close your files properly in these routines before you abort the job.

Name	Operation	Operands
[name]	CLOSE	filename1,...,filename16

Each operand is the name of a file to be closed by this macro instruction; the name of a file is the symbol appearing in the name field of the header entry for the file definition statement that describes the file. You may close up to 16 files for various devices with one CLOSE macro instruction. The operations performed depend on the type of device involved and the labeling technique (if applicable). For information on label processing and label formats refer to the SRL publication IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006.

The operations the CLOSE macro instruction performs for the different devices are discussed in this section under Terminating Files.

Reopening Closed Files

CARD AND PRINTER FILES. If you issue a CLOSE macro instruction for a card or printer file, that file cannot be reopened by a subsequent OPEN macro instruction.

PRINTER-KEYBOARD FILE. A printer-keyboard file that has been closed can be reopened.

MAGNETIC TAPE FILE. If further processing of a magnetic tape file is desired, the file can be reopened. If you do this, keep in mind that the previous CLOSE for the file has caused the tape to be positioned in accordance with the rewind option specified in the DTFMT statement for the file. Therefore, you should specify REWIND=NORWD in the DTFMT statement to resume processing of tape records at the point where the CLOSE macro instruction occurred. The first record read from the reopened file must be a file label if standard labels are specified for that file. If the tape file to be reopened is unlabeled or contains non-standard labels, you must determine whether the first record read is a data record or a file label.

If you reopen a multi-volume tape file for which you included the detail entry ALTTAPE in the DTFMT statement, the IOCS continues to read from (or write in) the same volume that was used as input (or

output) tape at the time the file was closed, i.e., only the volume which was processed last is reopened.

DISK FILE. You can reopen a closed disk file. If this is done, the IOCS performs all checks as though the file had never been opened before.

Initializing Files

This section describes the processing the IOCS performs for the various types of files when an OPEN macro instruction is executed.

OPENING CARD FILES

When an input file is processed in the overlap mode, the OPEN macro instruction causes the first card to be read. The data read from this card can then be moved from the input area to the work area when the first GET for the file is encountered. When an input file is processed in the non-overlap mode, the function of the OPEN macro instruction depends on the type of file as follows.

1. In the case of a simple file, the OPEN macro instruction makes the file available for processing.

2. In the case of a combined file, the OPEN macro instruction causes the first card to be read while it is being moved to the pre-punch station.

OPENING PRINTER AND PRINTER-KEYBOARD FILES

For a printer or printer-keyboard file, the OPEN macro instruction makes the file available for processing. The OPEN routine also tests whether the Monitor includes the PIOCS routines for the printer-keyboard.

OPENING MAGNETIC TAPE FILES

When a magnetic tape file with standard labels is opened, the IOCS expects the first record read to be a label. An OPEN macro instruction causes the tape to be rewound prior to processing, unless you prevent rewinding by including REWIND=NORWD in the DTFMT statement for the file. If you specified REWIND=NORWD, or if you open a file that begins at some location within a volume (reel of tape), you can position this volume at the beginning of the required file by means of a FILES control statement submitted to the Job Control program. When a volume has been positioned in this manner, the first record read is a label. If the first record is <u>not</u> a label, the IOCS regards this as an error condition. However, an unlabeled file can be opened in the middle without causing an error condition.

When two or more files of a multi-file tape volume are to be processed by one problem program, processing of each file specified must be completed before the next file in succession is opened.

<u>Example</u>: If the first, second, and third files of a multi-file tape volume are to be processed by one problem program, you must write the OPEN instructions for these files in the following sequence:

```
OPEN  first file
 .
 .
 .
CLOSE first file
 .
 .
OPEN  second file
 .
 .
 .
CLOSE second file
 .
 .
OPEN  third file
 .
 .
 .
CLOSE third file
```

The concurrent processing of two or more files of a multi-file tape volume is not possible.

All files in a multi-file volume must either contain the same type of labels (either standard or non-standard) or no labels whatsoever.

<u>Opening Tape Input Files</u>

The processing done by the IOCS when an OPEN macro instruction is executed depends on whether the file has standard labels, non-standard labels, or no labels. An OPEN macro instruction causes the following:

1. If standard labels are specified, the IOCS will:

   a. read and check the volume label if the tape is at load point;

   b. bypass any user volume labels;

   c. read and check the standard file header label (HDR1);

   d. bypass any additional standard header labels (HDR2-HDR8);

e. test the user labels UHL1-UHL8 (if you specified your own label routine) and make them available to your label routine as they are read. If you did not specify your own label routine, the user labels (if present) are skipped; and

f. properly position the tape to read the first data record.

If the file is to be read backward, the IOCS performs steps e, d, and c, in this sequence; steps a and b are omitted. (The IOCS processes trailer labels instead of header labels).

2. If you specify non-standard labels, the file is spaced forward to the first record following the first tapemark. Therefore the non-standard labels must be followed by a tapemark.

3. If no labels are specified, the first record on tape may be a data record or a tapemark followed by one or more tapemarks. The IOCS reads the record and determines whether it is a tapemark. If it is, control is returned to the problem program. If the record is not a tapemark, it is assumed to be a data record, and the tape is backspaced by one record.

Read-Backward Considerations: 9-track tape files written on System/360 tape units can be read backward if they do not contain variable-length blocked records; 7-track tapes can be read backward if they were written on System/360 tape units without using the Data Conversion feature. Note that 7-track tapes containing format-V records have been written using the Data Conversion feature and therefore cannot be read backward. A file to be read backward is limited to one reel. Any tapemark sensed while reading data records is considered to indicate an end-of-file condition.

When opening a tape file that is to be read backward, the job is terminated if the first record read is not a tapemark.

It is your responsibility to properly position files that are to be read backward prior to issuing an OPEN macro instruction. The proper positions are as follows:

Files with standard labels should be positioned so that the first record read will be the tapemark following the trailer label set. Because the file trailer label is the first label to be checked on a read backward operation, this trailer label must be complete and contain both the trailer and the header information, except HDR, to properly identify the file. If the file labels were originally written by the IOCS, the trailer labels will be complete.

Files with non-standard labels should also be positioned so that the tapemark following the trailer-label set is the first record to be read. However, no label checking is performed.

Unlabeled files should be positioned so that the first record read will be the tapemark after the last record of the file. Unlabeled tape files to be read backward must have a tapemark as the first record of the file (preceding the first data record). If this tapemark is not present, no end-of-file (EOF) condition is detected and an attempt is made to read past the load point.

Specify the NORWD (no rewind) option in the file definition statement for the file to be read backward.

Opening Tape Output Files

The processing done by the IOCS when an OPEN macro instruction is executed depends on whether or not the file is labeled. An OPEN macro instruction causes the following:

1. If standard labels are specified, the IOCS will:

   a. check for a volume label if the tape is at the load point;

   b. read the file header label (if present) and check the expiration date to make sure that the data on the tape is no longer active and may be destroyed; in a multi-file volume only the first file is checked for the expiration date in the header label;

   c. backspace the tape and write the new file header label with the information supplied by the TPLAB statement (refer to the section Control Statements); and

   d. enter your label routine (if you specified one) to allow user header labels (UHL1-UHL8) to be created and written.

2. If no labels are specified, the IOCS will perform the rewind operation and write a tapemark as the first record on the tape. The volume label and the expiration date are not checked, and any existing label set is destroyed.

   Note: The writing of a tapemark may be suppressed by a TPMARK=NO entry in the DTFMT statement.

3. If non-standard labels are specified for a file, a diagnostic message is printed during assembly because the specification of non-standard labels for an output file is not permitted.

OPENING DISK FILES

When disk files are opened, the processing done by the IOCS depends on whether the file is a sequential file, a direct-access file, or an indexed-sequential file. Depending on the type of file, the label processing may occur at different times during the execution of the problem program.

For sequential files, each disk pack of the file is opened when it is required. For direct-access files and indexed-sequential files, all packs of the file are opened at one time; if this is not done, the job cannot continue. Details of the processing performed when each type of file is opened are described below for disk input and disk output files.

You must specify the disk storage areas used by the file by means of XTENT statements which you submit to the Job Control program. For detailed information about these statements, refer to the SRL publication IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006.

Note: If a program is to be executed as an inquiry program, you must follow special rules for opening disk files. These rules are described in the section The Inquiry Program.

Opening Disk Input Files

An OPEN macro instruction causes the following:

1. If the file is a sequential file, the IOCS will:

   a. locate and check the volume label to verify that the proper disk pack is mounted;

   b. locate and check the file label against data furnished in the DLAB control statement at Job Control time; and

   c. check all extent limits in the format-1 file label and, if applicable, in the format-3 file label against the limits specified in the XTENT control statement. The checked extent limits are stored within the processing routines for the file that is being opened. During processing, these extent limits are used to check for end-of-extent conditions and to automatically switch to the next extent when an end-of-extent condition occurs. If an end-of-volume condition is detected (no more extents available within the volume), the file processing routines issue an internal OPEN for the next volume. The functions for this internal OPEN are the same as described above.

2. If the file is a direct-access file, the IOCS will:

   a. locate and check the volume labels of all volumes used for the file to verify that the proper disk packs are mounted;

   b. locate and check the file labels in all volumes of the file against data furnished in the DLAB control statement; and

   c. make all disk areas (defined by XTENT statements) available for processing; if one or more areas are not available, the job will be terminated.

3. If the file is an indexed-sequential file, the IOCS will:

   a. locate and check the volume labels of all volumes used for the file to verify that the proper disk packs are mounted;

   b. locate and check the format-1 file labels in all volumes of the file against data furnished in the DLAB control statement;

   c. locate and process the format-2 file label; and

   d. make all disk areas (defined by the XTENT statements) available for processing; if one or more areas are not available, the job will be terminated.

Opening Disk Output Files

An OPEN macro instruction causes the following:

1. If the file is a sequential file, the IOCS will:

   a. locate and check the volume label to verify that the proper disk pack is mounted;

b. check to make sure that the volume table of contents (VTOC) does not contain a label with the same file identifier as the file to be opened; should the file identifier be identical, the Open routine checks the expiration date to determine whether the file has expired or not. If it has not expired, a halt occurs and the operator has the option to erase the file or to terminate the job.

c. check all extents described in the VTOC for the particular volume against those extents of the file that refer to the same volume. If an extent overlay is detected, the OPEN routines check the expiration date of the file to which the overlaid extent belongs. If the expiration date has been reached, the file label(s) of the expired file are erased. All labels in the VTOC with an address that is higher than the addresses of the erased labels are shifted to use up the space that was previously occupied by the erased labels. The program halts if an extent overlay is detected for an active file. This allows you to either erase the file or terminate the job.

d. cause the format-1 label and, if more than three extents are specified, the format-3 label(s) to be created and written behind the last label in the VTOC. If no space is available within the VTOC, a programmed halt occurs.

e. store the checked extent limits within the processing routines for the file that is being opened. During processing, these extent limits are used to check for end-of-extent conditions and to automatically switch to the next extent when an end-of-extent condition occurs. If an end-of-volume condition is detected (no more extents available within the volume), the file processing routines issue an internal OPEN for the next volume. The functions for this internal OPEN are the same as described above.

2. If the file is a direct-access or an indexed-sequential file, the IOCS will:

a. locate and check the volume labels of all volumes to be used for the file to verify that the proper disk packs are mounted;

b. check to make sure that the VTOC does not contain a label with the same file identifier as the file to be opened; should the file identifier be identical, the IOCS checks the expiration date to determine whether the file has expired or not. In the case of an indexed-sequential LOAD file, a halt occurs if the file is not expired. The operator then has the option to extend the file or to perform an original load.

c. check all extents described in the VTOCs of all volumes used by the file against all the extents specified for the file; if an extent overlay is detected, the OPEN routines check the expiration date of the file to which the overlaid extent belongs. If the expiration date has been reached, the file labels of the expired file are erased. All labels in the VTOC with an address that is higher than the addresses of the erased labels are shifted to use up the space that was previously occupied by the erased labels. The program halts if an extent overlay is detected for an active file. This allows you to either erase the file or terminate the job.

d. cause the format-1 label and, if more than three extents are specified, the format-3 label(s) to be created and written after the last label in the VTOC. In the case of an indexed-sequential file, the IOCS will also cause the format-2 label to be created and written. The program halts if no space is available within the VTOC.

e. store the checked extent limits within the processing routines for the file that is being opened. (During processing, these extent limits are used to check for end-of-extent conditions and to automatically switch to the next extent when an end-of-extent condition occurs.)

Terminating Files

After all records of a file have been processed, that file must be closed.

The need to close, or deactivate, a file is indicated by an end-of-file (EOF) condition. The EOF condition is determined in various ways for different types of files and I/O devices as follows:

1. Card input files. Four cards with the characters /* in columns 1-2 are required by the IOCS to properly perform end-of-file operations. For the number of /* cards required during stacked job processing, refer to Figure 11.

2. Printer-Keyboard input files. The characters /* typed in on the printer-keyboard indicate the end-of-file condition.

3. Tape input files without labels or with non-standard labels. A tapemark indicates an EOF condition. In your end-of-file routine you must determine whether an end-of-file or an end-of-volume condition exists. The IOCS cannot determine this.

4. Tape input files with standard labels. The characters EOF appear as the first three characters of a trailer label.

5. Tape files with standard labels that are read backwards. The characters HDR appear as the first three characters of a header label.

6. Sequential disk input files or indexed-sequential input files processed sequentially. The characters /*b appear in an end-of-file record, or the end of the last extent has been reached.

7. All output files, and direct-access input files. The problem program determines the end of a file.

End-of-File Processing

When EOF occurs in a card input file, the IOCS branches to your end-of-file routine. The address of this routine must be provided in the EOFADDR=name entry of the definition statement for the file.

When EOF occurs in a tape input file with standard labels, the IOCS branches to the label checking routine to check the EOF1 label. In this routine, the IOCS compares the block count recorded in the label with the block count that has been accumulated during processing. An 'unequal' condition is indicated to the operator who has the option to either terminate or continue the job. If user labels (UTL1-UTL8) are to be checked, the IOCS branches to the LABADDR routine when the checking of the EOF1 label has been completed. (Refer to the description of the DTFMT detail entry LABADDR and of the LBRET macro instruction in the section Instructions for Processing Magnetic Tape Files). After all trailer labels have been checked, the IOCS branches to the EOFADDR routine.

If the tape input file has been read backward, the functions performed by the IOCS are essentially the same. On reaching the tapemark preceding the first record of the file, the IOCS branches to check the user header labels (UHL1-UHL8), if present, and then checks the HDR1 label. After these checks are completed, the IOCS branches to the EOFADDR routine.

When EOF occurs in a tape input file without labels or with non-standard labels, the IOCS branches to the EOFADDR routine when the tapemark following the last data record is read.

When EOF occurs in a sequential disk input file or in an indexed-sequential file processed sequentially, the IOCS branches to the EOFADDR routine when the EOF record containing /*b is read, or the end of the last extent has been reached.

End-of-Volume Processing

Some of the actions performed by the CLOSE routine are also required when an end-of-volume condition occurs while processing a magnetic tape or sequential-disk file. Except for tape input files without labels or with non-standard labels, the IOCS detects EOV conditions and takes the required actions without the need for additional routines in the problem program.

During the processing of a magnetic tape file or of a sequential disk file, an EOV condition can occur. This indicates that the next volume of the same file is needed either for reading more input records or for writing more output records. The method of detecting an EOV condition and the action taken are described below.

TAPE EOV CONDITION: The end of a volume of a standard-labeled tape file is indicated in the trailer label of an input reel or by the reflective marker on an output reel. The IOCS processes an EOV condition as follows:

1. For input files, the IOCS (1) checks the block count, (2) branches to your LABADDR routine, if specified, and (3) rewinds the tape if required. The IOCS then processes the header label(s) of the next volume and makes the first record of the volume available to the problem program.

2. For output files, the IOCS causes the EOV trailer label (including the accumulated block count) to be written. If a LABADDR routine is specified, the IOCS branches to this routine to write additional user trailer labels (UTL1-UTL8) and to perform the functions as specified. The IOCS then

processes the header label(s) of the next volume as described above under Opening Tape Output Files.

If no labels or non-standard labels have been specified for an input file, you must determine an EOV condition and issue an FEOV macro instruction (refer to the section Instructions for Processing Magnetic Tape Files) to have the IOCS perform the desired end-of-volume functions. To determine an EOV condition, you must provide in your EOFADDR routine a subroutine to which the IOCS branches as soon as it detects a tapemark. For multi-volume files, refer to the description of the detail entry ALTTAPE.

DISK EOV CONDITIONS: The end-of-volume for sequential disk files is indicated after the contents of all disk areas in a volume (as specified in XTENT control statements) have been processed. The IOCS processes EOV conditions as follows:

1. For input files, the IOCS checks the file label(s) of the next volume and makes the first record in the next volume available for processing.

2. For output files, the IOCS checks whether or not any extents are overlaid, writes a volume label and one or more file labels on the next volume, and makes the first location in that volume available for an output record.

CLOSING CARD AND PRINTER FILES

For card and printer files, the CLOSE macro instruction makes the file unavailable for further processing. Specifically, the CLOSE macro instruction ensures:

1. that records remaining in the output area upon completion of processing are printed and/or punched,

2. that all processed data cards remaining in the card feed path (not end-of-file cards) are selected and sorted into the appropriate stackers,

3. that all pending interrupts for the closed file(s) have been handled.

CLOSING PRINTER-KEYBOARD FILES

You should close a printer-keyboard input or output file after all records in the file have been processed. The characters /* in the first two positions of the record designate the end-of-file for an input file. For an output file, end-of-file is determined by the problem program.

CLOSING MAGNETIC TAPE FILES

The operations performed when a magnetic tape file is closed depend upon whether it is an input or an output file.

Closing Tape Input Files

The CLOSE macro instruction causes the input tape to be rewound if this has been specified in the REWIND entry of the DTFMT statement for the file. The IOCS then deactivates the file; no labels are read or checked.

Closing Tape Output Files

The CLOSE macro instruction causes the writing of any record or block of records that has not yet been placed into the file. If a record block is only partially filled, it will be written on tape as a short block. A tapemark is written following the last record.

If labels have not been specified, a second tapemark is written and the tape is rewound if this has been specified in the DTFMT statement for the file.

If standard labels have been specified for the file, the IOCS writes the trailer label after the tapemark. The trailer label includes the block count accumulated by the IOCS during the run and the header label information (except that HDR is replaced by EOF).

When additional labels are to follow the standard trailer label, the IOCS branches to your label routine specified by the LABADDR=name entry in the DTFMT statement for the file. This occurs after the standard label has been written. After building each label, return to the IOCS using the LBRET macro instruction. After all trailer labels have been written, the IOCS writes two tapemarks, executes the rewind function, if specified, and deactivates the file.

Two tapemarks are written at the end of a tape output file to indicate that no further data follows. If you specified REWIND=NORWD for the file, the IOCS causes the tape to be backspaced by one record. As a result, the second tapemark is overwritten if another output file is written onto the same tape.

CLOSING DISK FILES

The operations performed when you close a disk file depend upon whether it is an input or an output file.

Closing Disk Input Files

The CLOSE macro instruction merely makes the file unavailable for further processing.

Closing Disk Output Files

The CLOSE macro instruction causes the writing of any record or block of records that has not yet been placed onto disk. The IOCS writes an end-of-file record following the last data record in the file. The CLOSE macro instruction also updates format-2 labels when you load or extend an indexed-sequential file or when you add records to it.

Note: In a mainline program that permits inquiry interrupts, you must provide for the closing of all disk input and output files. Details on closing files in a mainline program are given under File Protection in the section The Inquiry Program.

The IOCS provides routines for sequentially processing card files. All records in a card file must be unblocked format-F records with a maximum length of 80 card columns.

You must define a card file in the problem program using the DTFSR file definition statement, which is described in detail below. The discussion of the DTFSR statement is followed by a description of the imperative macro instructions supported for card input and output files.

DTFSR Statement

This file definition statement applies to card and printer files. The name field of the header entry must contain the name for the file, and the operation field must contain DTFSR.

The header entry is followed by detail entries. For ease of reference, the detail entries for card files are described below in alphabetical order.

BINARY=code

Specify this entry if the cards are to be read in the column binary mode. You may provide the entry for both simple and combined files.

Code	Type of File
YES	Simple input file
INPUT	Combined file

The twelve punch positions of a card column read in column binary mode are stored in the six low-order bits of two adjacent bytes of the input area. Therefore, the input area must be large enough to accommodate a number of bytes that is twice the number of card columns to be read.

When you use the BINARY entry for a particular file, you are not allowed to specify the entries SEQNCE and RFORMTn for the same file.

BLKSIZE=n

This entry specifies the length of the I/O area(s) to be used by a simple file. The value of n must be equal to or less than the number of bytes of the area reserved by the DS or DC statement in the source program.

If two I/O areas are used for a file (IOAREA1 and IOAREA2), specify the area size only once because it applies to both areas. The maximum area length acceptable to the IOCS is 80 bytes (160 bytes if the input is in column binary mode).

The minimum area length specifications are:

For input files : two bytes (four bytes for column binary mode)
For output files: one byte.

CONTROL=YES

Specify this detail entry if you intend to issue a CNTRL macro instruction for your card file. The CNTRL macro instruction causes the I/O device to perform stacker selection.

CRDPRA=name

Use this entry in conjunction with CRDPRLn entries when printing on cards is desired. The maximum number of lines that can be printed at one time is six, i.e., the number of print heads available.

The CRDPRA entry specifies the name of the area in main storage that contains the data to be printed by the lowest numbered MFCM print head. Define contiguous 64-byte areas from which the remaining print heads are to print (refer to Figure 9). The CRDPRLn entries serve to define the print heads to be used in ascending order according to the print head numbers. Figure 10 shows the detail entries required to allow printing from the areas shown in Figure 9.

It suffices to specify the CRDPRA and CRDPRLn entries in only one DTFSR statement of a program because they do not refer to a particular file. You must issue a CRDPR macro instruction to cause printing of data from the areas specified in the CRDPRA and CRDPRLn entries. Refer to the description of the CRDPR macro instruction in this section.

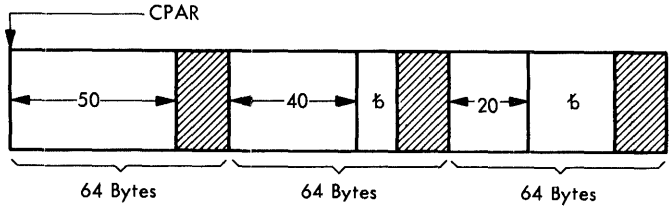
CRDPRLn=m

Entries of this type are used in conjunction with the CRDPRA=name entry to specify the print heads used. You must specify one CRDPRLn entry for each print head you use, i.e., you can specify up to six CRDPRLn entries.

The keyword is CRDPRLn, where n is the number of the print head (1 to 6). The specification m indicates to the IOCS the number of bytes to be printed by this print head. Specification of the number of bytes to be printed by each individual print head is required because, when filling a print area with data to be printed, the IOCS moves into the print area only the number of bytes specified for the particular print head.

Refer to the example in Figures 9 and 10. In this example, print head 1 is to print the first 50 bytes of its 64-byte print area, print head 2 is to print the first 40 bytes of its 64-byte print area, and print head 5 is to print the first 20 bytes of its 64-byte print area. However, all three print heads actually print the first 50 bytes of their 64-byte print areas, i.e., the largest number of bytes specified for a particular print head. At the time of printing, the unused byte positions of a print area contain blanks, since the IOCS clears all print areas up to and including byte 50 (i.e., the largest number of bytes specified) to blanks after every card-print operation.

You may utilize those portions of the print areas that are not cleared by the IOCS. In the example, bytes 51 through 64 of all three 64-byte print areas could be used for other processing (shaded areas in Figure 9).



b = blank

Figure 9. MFCM Card-Print Areas

Figure 10. CRDPRA Detail Entry with CRDPRLn Entries

DEVICE=code

This entry, which is required for all card files, specifies the I/O device to be used to process the particular file. Enter one of the following specifications immediately after the equal sign (=) in this entry.

Code	Explanation
CRP20	A file is to be read and/or punched by the IBM 2520 Card Read-Punch.
MFCM1	A file is to be read and/or punched from the primary feed of the IBM 2560 Multi-Function Card Machine.
MFCM2	A file is to be read and/or punched from the secondary feed of the IBM 2560 Multi-Function Card Machine.
PUNCH20	A file is to be punched by an IBM 2520 Card Punch.
PUNCH42	A file is to be punched by an IBM 1442 Card Punch, Model 5.
READ01	A file is to be read by an IBM 2501 Card Reader.

EOFADDR=name

This entry specifies the symbolic name of the routine in the problem program to which the IOCS should branch on an end-of-file condition. In that routine, you can perform any operation required for the end of the job. Normally, a CLOSE macro instructions is issued.

Note: If, in the end-of-file routine, you want to terminate the execution of a mainline program, you must first close all disk files. The EOFADDR entry is mandatory for card input and combined files.

The IOCS recognizes an end-of-file condition for card input and combined files when the required number of end-of-file cards (/* in columns 1 and 2) have been read. During single-job processing, use four end-of-file cards for all card I/O devices to ensure that the device remains in a ready status after end-of-file is detected. The number of end-of-file cards required during stacked-job processing is device dependent as shown in Figure 11.

MFCM1	MFCM2	2501 one I/O area	2501 two I/O areas	2520
2	2	1	2	1

Figure 11. Number of End-of-File Cards Required During Stacked-Job Processing

INAREA=name

This entry specifies the name of the input area to be used by a combined file. The specified name must be the symbol used in defining the area in the source program.

INBLKSZ=n

This entry specifies the number of bytes in the input area required by a combined file. The specified length applies to the area reserved by the DS or DC statement in the source program and referred to in the INAREA entry. The maximum area length permitted is 80 bytes (160 bytes for column binary mode). The minimum length of the input area is two bytes (four bytes for column binary mode).

IOAREA1=name

This entry specifies the name of the I/O area to be used by a simple file. The specified name must be the symbol used in defining the area in the source program.

A work area must be specified in addition to an I/O area. Refer to the description of the WORKA=YES entry.

IOAREA2=name

This entry can be used for simple input files to specify a second input area when the IBM 2501 Card Reader, Model A2, is used in overlap mode. The name in the specification part of this entry must be the same as the symbol used in defining the area in the source program. That area must be of the same length as the area referred to in the IOAREA1 entry.

The IOAREA2 entry permits a card to be read into the area specified in the DTFSR entry IOAREA1 while the data in the area specified in the DTFSR entry IOAREA2 (from the preceding card) are waiting to be moved to the work area. This may be of significance if only a number of selected cards of the file that is read on the IBM 2501 require extensive processing while all other cards require very little. If only one input area is specified, the data from a card that requires extensive processing may have to be held available for too long a period of time to permit continuous card feeding. In the majority of cases, specifying a second input area permits the IOCS to maintain the maximum card reading speed of the IBM 2501.

Do not use the IOAREA2 entry for a file being read or punched by any other card input or output device or when the IBM 2501 is used in non-overlap mode.

OUAREA=name

This entry is used in conjunction with the INAREA entry and specifies the name of the output area used by a combined file. The name specified must be the symbol used in defining the area in the source program.

OUBLKSZ=n

This entry is used in conjunction with the OUAREA entry to specify the length of the output area required by a combined file. The specification n is the number of bytes in the area. The maximum area length permitted is 80 bytes. The minimum length of the output area is one byte.

OVERLAP=NO

This entry specifies that the file is to be processed in non-overlap mode. If this entry is omitted, the file is processed in overlap mode.

When the OVERLAP=NO entry is used, the IOCS routines that are inserted in the problem program at the time of assembly require less storage space than they require when this entry is omitted.

PFORMTn=xxyy

This entry applies to combined files only. It allows you to check a specified field (or fields), in which data is to be punched, for all blanks. These fields are checked in those cards of a combined file that are not read but only punched.

The keyword of this entry is PFORMTn, where n is any number from 0 to 9. The n allows you to write up to ten different PFORMTn entries per file and thus have a maximum of ten fields checked. The xx specifies the first and the yy the last card column of the field to be checked. For columns 1 through 9, the leading zero is required.

If the field does not contain all blanks, the PUT macro instruction is not executed. Instead, the IOCS either transfers control to the routine specified in a PFXIT entry or it causes a programmed halt.

The specified input area must be large enough to accommodate the information contained in the columns specified in this entry.

You may use up to ten different PFORMTn entries, but only one PFXIT entry for each file.

PFXIT=name

This entry is used in conjunction with the PFORMTn entry. It specifies the name of your routine to which the IOCS transfers control if the test of the field specified by the PFORMTn entry indicates an error condition. To return to the main program, branch to the address contained in register 14.

If a PFORMTn check occurs, the main program branches immediately to the PFXIT routine. In this case, the contents of the work area are not moved to the punch area.

If a PUT macro instruction is issued that refers to a combined file and the program branches to the PFXIT routine, a subsequent GET will place the contents of the card causing the PFORMTn error into the work area. If this GET is in non-overlap mode, it is possible to punch this card by means of an additional PUT macro instruction.

If the PFXIT entry is omitted and the test shows an error condition, the machine halts before punching is initiated. This enables the operator to remove the card that caused the error condition from the pre-punch station.

RFORMTn=xxyyz

This entry enables you to check whether a specified input card field (or fields) contain(s) numeric characters only or all blanks.

If the input data are to be read in the column binary mode, an RFORMT entry must not be included for the file.

The keyword of this entry is RFORMTn, where n is any number from 0 to 9. The n allows you to write up to ten different RFORMTn entries per file and thus have a maximum of ten fields checked. The xx specifies the first and the yy the last card column of the field to be checked. For columns 1 through 9, the leading zero is required. If the field is to be checked for blanks, z must be a zero (0). If the field is to be checked for numeric characters, z must be a one (1). When checking for numeric characters, the maximum field length is 16 columns.

When a field is tested for all blanks, control is transferred to your RFXIT routine if the field is not blank.

When a field is tested for numeric characters, the test fails if the field contents are not of the following format (where at least the last character is numeric with or without an 11 or 12 zone punch):

bbb.....n

where b = blank
n = numeric character.

If the input cards are read in overlap mode from either an IBM 2520 or an IBM 2560, an RFORMT error with a subsequent branch to your RFXIT routine causes the IOCS to change the processing mode (from overlap to non-overlap) for the GET that detected the error.

Before branching to the RFXIT routine, the IOCS places the record containing the field that led to the error condition into the work area. If the error card was read by the IBM 2560 MFCM or the IBM 2520 Card Read-Punch, that card is positioned at the pre-punch station. The next record will be read by the next GET or EOM macro instruction.

In the RFXIT routine, save the contents of register 14 before issuing any macro instruction. If this is not done the re-entry address is lost during the execution of the macro instruction in the RFXIT routine.

You may use up to ten different RFORMTn entries, but only one RFXIT entry for each file.

If a SEQNCE error and an RFORMTn error are both detected in the same card, only the action specified for the SEQNCE error will be performed. Refer to the description of the SEQNCE=xxyy entry.

RFXIT=name

This entry is used in conjunction with the RFORMTn entry. Specify the name of the routine to which control is to be transferred if the test of the field specified in the RFORMTn entry is negative (i.e., the tested field contains characters other than blanks or numerics, respectively). To return to the main program, branch to the address contained in register 14.

If this entry is omitted and the test is negative, the IOCS causes a programmed halt. This enables the operator to replace the card that led to the error condition.

SEQNCE=xxyy

This entry enables you to check whether the contents of a specified field in successive input records are equal or in ascending order.

The xx is the first and the yy the last card column of the card field to be checked. For card columns 1 through 9, the leading zero is required. The maximum length of the card field to be checked is 16 columns.

Only one SEQNCE entry is permitted for each file. If the input data is to be read in column binary mode, a SEQNCE entry must not be included for the file.

If the input cards are read in overlap mode from either an IBM 2520 or an IBM 2560, a sequence error with a subsequent branch to your SEQXIT routine causes the IOCS to change the processing mode (from overlap to non-overlap) for the GET that detected the error.

Before branching to the SEQXIT routine, the IOCS places the record containing the field that led to the error condition into the work area. If the error card was read by the IBM 2560 MFCM or the IBM 2520 Card Read-Punch, it is positioned at the pre-punch station. The next GET or EOM macro instruction will cause the next record to be read. This record will then be compared with the record preceding the error record.

If the SEQXIT routine contains a macro instruction, the contents of register 14 should be saved before this macro instruction is executed. If this is not done, the re-entry address is lost during the execution of the macro instruction.

If a SEQNCE error and an RFORMTn error are both detected in the same card, only the action specified for the SEQNCE error will be performed.

SEQXIT=name

This entry must be used in conjunction with the SEQNCE entry. It specifies the name of your routine to which control is to be transferred if a sequence error occurs. To return to the main program, branch to the address contained in register 14.

Note: If the sequence-error routine discontinues the processing of a program that is a mainline program, it must contain CLOSE macro instructions for all disk files.

TYPEFLE=code

This entry, which is required for all files, is used to specify the type of file (i.e., input, output, or combined).

Code	Type of File
INPUT	A simple input file.
OUTPUT	A simple output file.
CMBND	A combined file.

WORKA=YES

The WORKA=YES detail entry is mandatory for all card files. Enter the name of the work area as the second operand in all GET, PUT, or CRDPR macro instructions for the particular file. The length of a work area must always be the same as that of the I/O area. For additional information regarding the use of a work area, see Work Areas under the section Overlapping and Storage Areas.

Imperative Macro Instructions

The imperative macro instructions cause the desired I/O operation. They are described below in the following order: PUT, GET, CRDPR, CNTRL, EOM, LOM, WAITC. For the description of OPEN and CLOSE refer to the section Instructions for Opening and Closing Files.

PUT MACRO INSTRUCTION

This instruction punches logical records that have been built in a specified work area.

Name	Operation	Operands
[name]	PUT	filename,workname

The first operand specifies the name of the file, the second operand specifies the name of the work area in which the records are built.

The PUT macro instruction moves the record built in the work area to an output area. When the output area is full, the IOCS will take the data in that output area and punch them on the output device specified in the DTFSR statement for the file.

Individual records for a logical file may be built in the same work area or in different work areas, Each PUT macro instruction specifies the work area in which the completed record was built. However, only one work area can be specified in any one PUT macro instruction.

When a card file is processed in the non-overlap mode, a PUT macro instruction for the file (1) moves a record from the work area to the output area, (2) initiates the punch operation (and the next read operation in the case of a combined file), and (3) transfers control to the main program when the punch operation has been completed.

When a card file is processed in the overlap mode, a PUT macro instruction for the file (1) moves a record from the work

area to the output area, (2) initiates the punch (print) operation, and (3) immediately transfers control to the main program.

If PFXIT has been specified for a combined file and a card to be punched does not contain all blanks in the field to be punched, the PUT macro instruction causes control to be transferred to the specified PFXIT routine.

The work area is not cleared by the IOCS after a PUT macro instruction. To avoid having interspersed characters from preceding records in the output records, ensure that the records use every position of the work area, or clear the work area before the next record is built.

Programming Considerations for Combined Files

Assume that a combined file is being processed by means of the following sequence of instructions:

```
--------------------
GET  F1,W1
-------------------- no GET, EOM, or PUT
-------------------- macro instruction
-------------------- referring to file F1
PUT  F1,W2
--------------------
```

In this case, the following rules apply:

Non-overlap Mode. The statement PUT F1,W2 causes punching into the card that has been made available by the statement GET F1,W1.

Overlap Mode. The statement PUT F1,W2 causes punching into the card following the card that has been made available by the statement GET F1,W1. The card that has been made available by the statement GET F1,W1 has already passed the punch station when the statement PUT F1,W2 is encountered.

GET MACRO INSTRUCTION

This macro instruction makes the next sequential logical record from an input file available for processing in a specified work area.

When a branch condition is detected (end-of-file, sequence check, or read-format check), the IOCS transferred control to the appropriate routine specified in the EOFADDR, SEQXIT, or RFXIT entry of the file definition statement.

Name	Operation	Operands
[name]	GET	filename,workname

The first operand specifies the name of the file. The second operand specifies the symbolic name of the work area to be used.

The GET macro instruction moves the record to be processed from an I/O area to the work area specified by the second operand. There can be more than one work area for a file.

All records from a card file may be processed in the same work area, or different records from the same file may be processed in different work areas. In the first case, each GET macro instruction for the file specifies the same work area. In the second case, different GET macro instructions specify different work areas. It might be advantageous to plan two work areas, and to specify each area in alternate GET macro instructions. This permits the comparison of each record with the preceding one to determine a possible change of the control level. However, only one work area can be specified in any one GET macro instruction.

When a card file is processed in the non-overlap mode, a GET macro instruction for the file (1) initiates the reading of the next record, (2) moves the data from the input area to the work area when the read operation is complete, and (3) transfers control to the main program. When a card file is processed in the overlap mode, the GET macro instruction for the file (1) moves a record, as soon as it is available, from the input area into the work area, (2) initiates the next read operation, and (3) immediately transfers control to the main program.

When a combined file is processed and data are to be punched into the input cards, use one of the programming methods described in this section in the discussion of the LOM macro instruction. Also refer to Programming Considerations for Combined Files in the preceding description of the PUT macro instruction.

CRDPR MACRO INSTRUCTION (IBM 2560 MFCM)

This macro instruction (CaRD PRint) applies only to an IBM 2560 MFCM equipped with the card-print feature. The format is as follows:

Name	Operation	Operands
[name]	CRDPR	,workname,printarea

Because this instruction does not refer to a specific file, it does not have a file name as operand 1; the absence of this operand is indicated by a comma. The second operand is the name of the work area, and the third operand is the name of the card-print area.

A CRDPR macro instruction moves one line of information from the specified work area to the card-print area. However, printing does not take place until the card is being moved into and through the print station by the execution of a subsequent GET, PUT, or EOM macro instruction. It is therefore very important to write the CRDPR macro instruction in proper relationship to PUT, GET, or EOM macro instructions pertaining to the same card. The same rules that apply to the stacker-select CNTRL macro instruction for the IBM 2560 MFCM are also applicable to the CRDPR macro instruction (see No File Name Specified under Stacker Selection (SS) for the IBM 2560 MFCM).

You must write one CRDPR macro instruction for each line to be printed. If two CRDPR macro instructions are issued for the same line, only the second one will be executed. At the time of printing, all print lines, i.e., up to six, are printed simultaneously. It is not possible to print only with print head 1 during one print operation and then with print head 2 and/or another print head during another print operation. If no data is to be printed on a line, simply do not enter any data into the associated print area or, if processing was performed in the area, clear the area before printing takes place.

CNTRL MACRO INSTRUCTION

The CNTRL macro instruction causes stacker selection to be performed on the device associated with the card file. You must include a CONTROL=YES entry in the DTFSR file definition statement if you intend to issue a CNTRL macro instruction for the card file.

Name	Operation	Operands
[name]	CNTRL	filename,SS,n

The first operand specifies the name of the card file for which the device operation is described. The mnemonic SS indicates that stacker selection is to be performed. Specify the stacker into which the cards are to be selected as third operand (n).

Stacker Selection (SS) for the IBM 2520, Model A1 and A2

Either of two stackers can be selected. Cards fed into the IBM 2520 normally fall into stacker 1. The stacker selection mnemonic (SS) is used to select a card into the stacker, specified by the third operand n in this macro instruction. Specify 1 for stacker 1 and 2 for stacker 2.

If two stacker select CNTRL macro instructions are issued for the same file before the next GET or PUT macro instruction for that file, the second CNTRL macro instruction overrides the first. When using stacker selection ensure that the instruction is in proper relationship to the GET, PUT, or EOM macro instruction referring to the card to be selected:

1. Processing in overlap mode. The stacker select CNTRL must be the last macro instruction preceding the GET or PUT that refers to the card to be selected. The example below selects the card, the contents of which are transferred to or from the work area by the GET (or PUT) macro instruction.

```
--------------
CNTRL AAA,SS,n
-------------- no GET or PUT
-------------- referring to file AAA
GET (or PUT) AAA
--------------
```

2. Processing in non-overlap mode. The stacker select CNTRL must be issued after the GET macro instruction or before the PUT macro instruction that moves the card to be selected. The example below selects the card read by the GET macro instruction.

```
--------------
GET   AAA
-------------- no PUT, GET or EOM
-------------- referring to
-------------- file AAA
CNTRL AAA,SS,n
--------------
```

The example below selects the card moved by the PUT macro instruction.

```
--------------
CNTRL AAA,SS,n
-------------- no PUT, GET, or EOM
-------------- referring to
-------------- file AAA
PUT   AAA
--------------
```

Stacker Selection (SS) for the IBM 2560 MFCM

Any of five stackers can be selected. Since the MFCM connected to a Submodel 4 has no stacker 5, all cards selected for stacker 5 go into stacker 4.

Operands			Operation
filename	code	n	
[filename]	SS	1	Select stacker 1
[filename]	SS	2	Select stacker 2
[filename]	SS	3	Select stacker 3
[filename]	SS	4	Select stacker 4
[filename]	SS	5	Select stacker 5

The CNTRL macro instruction for the IBM 2560 MFCM has two forms. One form does not specify the name of a file, the other does. The manner in which the forms of the CNTRL macro instruction are used in the problem program is different.

No Filename Specified: When the first operand is omitted, the CNTRL macro instruction does not specify a file containing a card to be selected; it merely specifies the desired stacker. If the name of the file is omitted, its absence must be indicated by a comma. Hence, the format of the macro instruction is:

```
CNTRL ,SS,n
```

When two CNTRL macro instructions with this format are issued before the stacker select operation is performed, the second CNTRL macro instruction overrides the first.

Execution of this stacker select CNTRL macro instruction for the MFCM requires that the card to be selected is in the pre-print station when the subsequent PUT, GET, or EOM macro instruction referring to an MFCM file is executed.

To ensure that the instruction is in proper relationship to the GET, PUT, or EOM macro instruction referring to the card to be selected, observe the following rules:

1. Processing in overlap mode. If the card to be selected is punched by a PUT macro instruction or if the contents of the card are moved to a work area by a GET macro instruction, issue the CNTRL macro instruction prior to any subsequent PUT, GET, or EOM macro instruction referring to an MFCM file. The example below illustrates this requirement.

```
--------------
PUT (or GET) AAA
-------------- no PUT, GET, or
-------------- EOM referring to
-------------- MFCM files
CNTRL ,SS,n
--------------
```

2. Processing in non-overlap mode. If the card to be selected is punched by a PUT macro instruction, issue the CNTRL macro instruction prior to any subsequent GET, PUT, or EOM macro instruction referring to an MFCM file. The example below illustrates this requirement.

```
--------------
PUT AAA
-------------- no PUT, GET or
-------------- EOM referring to
-------------- MFCM files
CNTRL ,SS,n
--------------
```

There is one exception to the above usage. Between the PUT macro instruction for a card to be selected and the CNTRL macro instruction for this card, a GET macro instruction for the same file may be inserted. The example below illustrates this exception.

```
--------------
PUT AAA
-------------- no PUT, GET, or
-------------- EOM referring to
-------------- MFCM files
GET AAA
-------------- no PUT, GET, or
-------------- EOM referring to
-------------- MFCM files
CNTRL ,SS,n
--------------
```

If the card to be selected is read by a GET macro instruction, another GET,EOM, or PUT macro instruction referring to the file must be issued prior to the CNTRL macro instruction for this card. The example below illustrates this requirement.

```
-------------
GET AAA
------------- any combination of
------------- macro instructions
------------- referring to
------------- another file
GET AAA
(or PUT AAA
or EOM AAA)
------------- no PUT, GET, or
------------- EOM referring to
------------- MFCM files
CNTRL ,SS,n
-------------
```

File Name Specified: When the first operand is present, the CNTRL macro instruction specifies the file containing a card to be selected. Therefore, the format of the macro instruction is:

CNTRL filename,SS,n

If this format is used, the functions are the same as described for the stacker select CNTRL macro instruction for card files that are to be processed on the IBM 2520, Models A1 and A2.

EOM MACRO INSTRUCTION (COMBINED FILES)

This macro instruction (Enter Overlap Mode) applies only to combined files for which a previous LOM (Leave Overlap Mode) macro instruction was issued.

Name	Operation	Operand
[name]	EOM	filename

Enter EOM in the operation field and the name of the file to which the macro instruction applies as the operand.

This macro instruction causes (1) the next card to be read into the input area, and (2) subsequent GET macro instructions referring to the same file to be executed in overlap mode. Processing of the file in overlap mode begins immediately after the EOM macro instruction has been executed. For further details regarding the use of EOM macro instructions, refer to Programming with LOM and EOM Macro Instructions below.

LOM MACRO INSTRUCTION (COMBINED FILES)

This macro instruction (Leave Overlap Mode) applies to combined files for which overlap mode was specified.

Name	Operation	Operand
[name]	LOM	filename

Enter LOM in the operation field and the name of the file to which the macro instruction applies as the operand.

When a LOM macro instruction is issued, processing of the file in non-overlap mode begins when the next GET macro instruction for the specified file is executed. This permits reading a card and punching data into the same card of a combined file that is being processed in overlap mode. If a LOM macro instruction is issued for a particular file, all subsequent GET macro instructions for that file are performed in non-overlap mode until an EOM macro instruction is issued.

Programming with LOM and EOM Macro Instructions

If a card has to be read and then punched, it must be read by a GET macro instruction in non-overlap mode. There are three possible ways to cause the GET macro instruction to operate in non-overlap mode during this reading and punching of the same card:

1. Provide an OVERLAP=NO detail entry in the file definition statement for the file. In this case, the IOCS generates GET and PUT routines for this file that operate in non-overlap mode.

2. Do not provide an OVERLAP=NO detail entry in the file definition statement for the file and, in the source program, issue an LOM macro instruction between the OPEN and first GET macro instructions for the file. In this case, GET and PUT routines that operate in the overlap mode are generated for the file. However, all GET macro instructions for the file operate in non-overlap mode.

3. Do not provide an OVERLAP=NO detail entry in the file definition statement for the file and, in the source program, precede each GET macro instruction with a LOM macro instruction and follow each GET with a test to determine if a punching operation is to be performed on this card. If not, operation of this file can be changed back to the overlap mode by an EOM macro instruction.

The first method keeps storage requirements at a minimum, but results in a decrease of program speed.

The second method is the most satisfactory solution when almost all cards of a file must be both read and punched. The program speed does not decrease as much as with the first method because the PUT routines will operate in the overlap mode.

The third method is usually the most satisfactory solution when only a few specified cards in a combined file must be both read and punched. When this method is used, each card is read in the non-overlap mode and can therefore be subsequently punched.

However, when punching is not required, the program immediately begins operation in the overlap mode. This method requires some additional main storage positions for the extra LOM and EOM macro instructions, but it results in a program that runs at nearly the same speed as a program operating entirely in the overlap mode.

The coding below is an example of the use of the LOM and EOM macro instructions. This coding example assumes that (1) a combined file (AAA) is to be processed and (2) data is to be punched into each card of the file that contains a 7-punch in column 1. It is further assumed that an area named WORKAAA has been defined.

```
COMPR1      LOM   AAA
COMPR2      GET   AAA,WORKAAA
            CLI   WORKAAA,C'7'
            BE    PUNCHR
            EOM   AAA
            ------------------
            ------------------
            ------------------
            B     COMPR1
PUNCHR      ------------------
            ------------------
            PUT   AAA,WORKAAA
            B     COMPR2
```

The macro instruction "LOM AAA" causes the subsequent GET for the file AAA to be executed in non-overlap mode. This permits the punching of data into the same card that has been read by means of the GET macro instruction. If punching is required (a 7-punch in column 1), control is transferred to the punch routine (PUNCHR). The PUT macro instruction for the file may be followed immediately by a branch to the GET macro instruction for the file because the system is still operating in non-overlap mode.

If punching is not required (no 7-punch in column 1), the EOM macro instruction is executed, which causes the operating mode for the file to be changed back to overlap.

WAITC MACRO INSTRUCTION

The format of this macro instruction (WAIT Card) is:

Name	Operation	Operand
[name]	WAITC	

Since the WAITC macro instruction neither refers to a particular file nor requests a particular function, no operand is required.

The WAITC macro instruction causes the problem program to wait for the completion of all pending card and printer I/O operations before the next sequential instruction is executed. This macro instruction allows you to establish uniform operating conditions for all card and printer I/O devices that are used in the program.

In a program using the IOCS, a WAITC macro instruction must precede the appropriate programmed halt statement if one of the following three conditions exists:

1. Card-input is read in overlap mode. (In the case of a read error, the WAITC macro permits a programmed halt to occur, thus allowing the replacement of the card in error.)

2. Card-input is read on an MFCM in overlap mode from one hopper and in non-overlap mode from the other hopper. (Function of the WAITC macro instruction as above.)

3. The FETCH macro is used to load another phase of a multi-phase program into main storage.

Except for condition 2 above, a WAITC macro instruction need not be issued for the replacement of an error card if the cards of the file are to be read in non-overlap mode.

Programming with the WAITC Macro Instruction

A GET macro instruction that refers to a card file may or may not immediately initiate a read operation. This depends on the operating condition of the I/O device involved. If the initiation of the I/O operation is delayed, the IOCS places the device request into a waiting queue. The IOCS handles the device requests in this waiting queue and executes the appropriate I/O operation as the requested I/O devices become available.

I/O Device	Mode of Operation	WAITC required	Number of Dummy GETS	Number of Cards to be Returned	
				Error Feed	Non-error Feed
2501	Non-overlap	No	0	2	
	Overlap with one I/O area	Yes	1	3	
	Overlap with two I/O areas	Yes	2	4	
2560 Feed 1	Non-overlap	No*	0	3	3
	Overlap	Yes**	1	4	3
2560 Feed 2	Non-overlap	No*	0	2	2
	Overlap	Yes**	1	3	2
2520	Non-overlap	No	0	2	
	Overlap	Yes	1	3	

*WAITC macro instruction is required if a file in the other feed is processed in overlap mode.
**A dummy GET is required for both files.

Figure 12. Programming with the WAITC Macro Instruction -- Halt and Restart Information

When a GET macro instruction is issued, the IOCS makes the desired card record available to the problem program in the specified work area. If the problem program determines that this record contains an error, you can provide a halt (HPR instruction) to enable the operator to (1) remove and correct the error card, (2) return it to the hopper, and (3) resume normal system operation.

Since you have no means to determine the status of the waiting queue at the time the error is detected or the exact position of the error card in the I/O device, the standard restart procedures cannot be applied.

Before writing the HPR instruction, you must issue a WAITC macro instruction to (1) establish uniform operating conditions for all card and printer I/O devices and (2) determine the exact position of the error card.

After the execution of the WAITC macro instruction, the waiting list contains no pending I/O device requests, except those for card printing. The error card (to be fed as the first card on restart) is determined by the number of cards that have to be returned to the input deck after the non-process runout.

The number of cards to be returned to the input deck depends on the I/O device used and, in the case of an MFCM file, on the mode of operation. For details refer to Figure 12 which is a summary of the halt and restart information.

Dummy GET Macro Instructions. To ensure proper program functions on restart, i.e., resume processing with the corrected card record, issue either one or two dummy GET macro instructions as shown in Figure 12. For the explanations below, processing in the overlap mode is assumed, unless it is stated that the information applies to files that are processed in the non-overlap mode.

After the execution of a WAITC macro instruction, the contents of the card following the error card are already in the I/O area. Therefore, the first GET macro instruction that is encountered after restart causes the record from the card following the error card to be moved into the work area. To make sure that the contents of the corrected error card have been moved into the work area before normal processing is resumed, the first GET macro instruction encountered after restart must be a dummy GET, i.e., no processing must be performed on the record moved into the work area by means of this GET macro instruction. If an IBM 2501 is used to read the

cards for a file and two I/O areas have been defined for this file, two dummy GET macro instructions are required.

If an IBM 2560 MFCM is used to process two input and/or combined files in one program, an error card in one file requires one dummy GET macro instruction on restart for each of the files with one exception: Only one dummy GET macro instruction is required for the file that contains the error card if (1) the other (non-error) file is an input file whose cards are read in non-overlap mode and (2) no GET has yet been issued for the non-error file. You must provide a switch to determine whether or not a GET has already been executed for the non-error file. This is illustrated in the coding example shown in Figure 13.

A GET macro instruction for a file that is to be processed in overlap mode may be preceded by a CNTRL macro instruction referring to the same file. If this GET macro instruction detects an error card, do one of the following in your restart routine:

1. Repeat the CNTRL macro instruction after the dummy GET macro instruction for the file in your restart routine.
2. Branch to the CNTRL macro instruction preceding the GET macro instruction that detected the error card.

Similar rules apply if two files are processed on the IBM 2560 MFCM in one program. Any file-dependent CNTRL macro instruction that precedes the last GET macro instruction in either file must be repeated after the dummy GET macro instruction for the file and before resuming normal processing. A preceding file-independent CNTRL macro instruction (no file name specified) need be repeated only once.

Figure 12 is provided to facilitate programming of restart routines and to furnish you with the required card-handling information. You must inform the operator about the number of cards to be returned to and placed in front of the remaining cards of the input deck. Any run-out cards that are not to be returned to the input deck must be placed into the proper stacker manually.

A halt caused by the IOCS (due to a machine check) may occur during or immediately after your restart routine, and the number of cards in the I/O device may be less than stated in the appropriate standard procedure. In this case, only those cards must be stacked manually which were in the card feed of the I/O device at the time the halt occurred and they do not have to be returned to the respective hopper.

The coding example in Figure 13 illustrates programming with the WAITC macro instruction. The example includes a simplified restart routine. For the purpose of this coding example, it is assumed that:

1. two files (AAA and BBB) have been defined to be read in the two feeds of the IBM 2560 MFCM,
2. file AAA is to be processed in the overlap mode and the cards of this file are to be fed from hopper 1 of the 2560 MFCM. This file may be an input or a combined file,
3. file BBB is an input file whose cards are to be read in non-overlap mode, and
4. any card of file AAA that does not have a 1-punch in column 1 is an error card and must be replaced.

Only those instructions that illustrate programming with the WAITC macro instruction are shown in Figure 13. These instructions are identified by sequence numbers in parentheses in the rightmost column of Figure 13. These sequence numbers are used as references in the explanations below.

```
┌──────┬─────────┬──────────────────┬──────┐
│Name  │Operation│Operand           │Instr │
│      │         │                  │Sqnce │
├──────┼─────────┼──────────────────┼──────┤
│      │.        │                  │      │
│      │.        │                  │      │
│      │.        │                  │      │
│      │GET      │BBB,WORK2         │ (1)  │
│      │MVI      │SW+1,X'00'        │ (2)  │
│      │.        │                  │      │
│      │.        │                  │      │
│      │.        │                  │      │
│      │CNTRL    │BBB,SS,4          │ (3)  │
│      │.        │                  │      │
│      │.        │                  │      │
│      │.        │                  │      │
│RETPT │CNTRL    │AAA,SS,2          │ (4)  │
│      │GET      │AAA,WORK1         │ (5)  │
│      │CLI      │WORK1,C'1'        │ (6)  │
│      │BE       │NOERR             │ (7)  │
│      │WAITC    │                  │ (8)  │
│      │HPR      │X'FFF',0          │ (9)  │
│      │GET      │AAA,WORK1         │(10)  │
│SW    │B        │BYPASS            │(11)  │
│      │GET      │BBB,WORK2         │(12)  │
│      │CNTRL    │BBB,SS,4          │(13)  │
│BYPASS│B        │RETPT             │(14)  │
│NOERR │.        │                  │      │
│      │.        │                  │      │
└──────┴─────────┴──────────────────┴──────┘
```

Figure 13. Coding Example -- Programming with the WAITC Macro Instruction

If a card of file AAA does not contain a 1-punch in column 1, the branch to NOERR (7) is not performed and the program executes the WAITC macro instruction (8) that precedes an HPR instruction (9). On restart, the program executes either one or two dummy GET macro instructions. Only one dummy GET macro instruction for file AAA (10) is executed if no GET macro instruction has yet been executed for the file BBB. In this case, the branch instruction named SW (11) is executed and the second dummy GET macro instruction for file BBB (12) and the stacker select CNTRL macro instruction (13) for this file are bypassed. Control is returned to the problem program by a branch to RETPT to repeat the CNTRL macro instruction preceding the GET macro instruction that caused the error card to be detected.

If a GET macro instruction has already been executed for the file BBB at the time the error card is detected, the branch instruction named SW (11) is not executed. This instruction has been changed to a no-operation (BC 0) instruction by means of the MVI instruction (2) following the GET macro instruction (1) for the file BBB.

The CNTRL macro instruction for file BBB (3) is only effective when no error card is detected.

If an error card is detected, four cards have to be returned for file AAA and two cards for file BBB.

If the cards of the file BBB are to be read in overlap mode, instructions (2) and (11) have to be omitted.

If the cards of a combined file are also to be card-printed and this file is to be processed in non-overlap mode, consider the following:

Unless successive cards are to be read which are not to be punched, a GET macro instruction for a card does not initiate card movement. Card movement is initiated by the PUT macro instruction for the preceding card. Therefore, you must issue a dummy GET macro instruction prior to the WAITC macro instruction to ensure that the desired card-print operation for the card preceding the error card is properly executed.

This is further explained in the coding example shown in Figure 14.

The coding example in Figure 14 is based on the following assumptions:

(1) the first card of the file CMBF has already been read;

(2) data is to be punched into all input cards; and

(3) all cards that do not contain a 1-punch in column 1 are error cards and must be replaced by the operator.

The sequence numbers shown in the rightmost column of Figure 14 are used as references in the explanations below.

If the card that is made available by the normal GET (1) is not an error card, the next PUT for the same file (8) causes data to be printed on the preceding card. If the card made available by the normal GET is an error card, the dummy GET (4) causes the error card to be moved past the punch station and the card preceding the error card is properly card-printed. On restart, the corrected error card is read by means of the normal GET (1), punched by means of the subsequent PUT (8), and card-printed at the time this PUT macro instruction is executed for the following card.

```
┌──────┬─────────┬──────────────────┬──────┐
│Name  │Operation│Operand           │Instr │
│      │         │                  │Sqnce │
├──────┼─────────┼──────────────────┼──────┤
│      │.        │                  │      │
│      │.        │                  │      │
│      │.        │                  │      │
│REPT  │GET      │CMBF,WRKC         │ (1)  │
│      │CLI      │WRKC,C'1'         │ (2)  │
│      │BE       │8,NERR            │ (3)  │
│      │GET      │CMBF,WRKC         │ (4)  │
│      │WAITC    │                  │ (5)  │
│      │HPR      │X'FFF',0          │ (6)  │
│      │B        │REPT              │ (7)  │
│NERR  │.        │                  │      │
│      │.        │                  │      │
│      │.        │                  │      │
│      │PUT      │CMBF,WRKC         │ (8)  │
│      │CRDPR    │                  │ (9)  │
│      │B        │REPT              │      │
│      │.        │                  │      │
│      │.        │                  │      │
└──────┴─────────┴──────────────────┴──────┘
```

Figure 14. Coding Example -- Programming with the WAITC Macro Instruction Involving Card Printing

The programming considerations that apply to card printing are also applicable to stacker-select CNTRL macro instructions without a file name as the first operand.

Instructions for Processing Printer Files

The IOCS routines to process printer files can be included in the problem program through the use of the macro instructions described below.

Printer files are always organized sequentially. The records in a printer file have to be unblocked format-F records. The length of each record must not exceed the length of a print line.

DTFSR Statement

This file definition statement describes the characteristics of the printer file to be processed. The name field of the header entry must contain the name of the file, and the operation field must contain DTFSR. For ease of reference, the detail entries to be used in the operand field of the DTFSR statement for printer files are described below in alphabetical order.

BLKSIZE=n

This entry specifies the minimum length of the area(s) to be used by a printer file. The value of n must be equal to or less than the number of bytes of the reserved area(s). You must specify a BLKSIZE entry for all printer files even though the IOAREA1 entry is not provided for a printer using the standard carriage.

Maximum area lengths acceptable to the IOCS are 120, 132, or 144 bytes, depending on the number of print positions available. One byte is the minimum length you must specify for a printer file.

CONTROL=YES

Provide this detail entry if a CNTRL macro instruction is to be issued for the file. The CNTRL macro instruction causes the printer to perform form spacing and/or form skipping.

DEVICE=code

This entry is mandatory for all files. It specifies the I/O device to be used to process the particular file. One of the following specifications must be entered immediately following the equal sign(=) in this entry.

Code	Explanation
PRINTER	A file is to be printed by an IBM 2203 Printer with a standard carriage or by an IBM 1403 Printer.
PRINTLF	A file is to be printed on the lower feed of an IBM 2203 Printer with the dual-feed carriage.
PRINTUF	A file is to be printed on the upper feed of an IBM 2203 Printer with the dual-feed carriage.

Note: If both feeds of an IBM 2203 Printer with dual-feed carriage are used, write separate DTFSR statements for the file printed on the lower feed and for the file printed on the upper feed. If the application requires only one feed of the dual-feed carriage, the lower feed must be used. In this case, the DEVICE=PRINTER entry and not the DEVICE=PRINTLF entry must be provided in the printer-file DTFSR statement.

IOAREA1=name

This entry applies only if DEVICE=PRINTLF or DEVICE=PRINTUF has been specified. It designates the name of the I/O area to be used by the printer file. The specified name must be the symbol used in defining the area in the problem program.

For a printer file for which DEVICE=PRINTER has been specified (standard carriage or single file on lower feed of dual-feed carriage), do not provide the IOAREA1 entry. The printer automatically uses the first 144 main-storage positions as a print buffer (printer output area). You cannot use these 144 main-storage positions in the problem program.

Two files printed on the dual-feed carriage require two IOAREA1 entries, i.e., one for each file. The print areas for the lower and upper feed of the dual-feed carriage must be defined as contiguous areas in main storage. The print area for the lower feed precedes the print area for the upper feed (see Figure 15).

Note that you must specify a work area in addition to an I/O area. Refer to the description of the WORKA=YES entry.

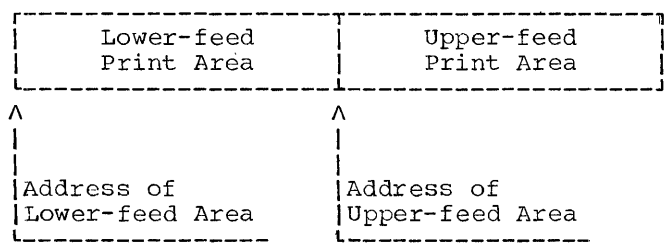


Figure 15. Print-Area Format for Dual-Feed Carriage

PRINTOV=YES

Include this entry for a printer file if a PRTOV macro instruction referring to this file is used in the source program.

TYPEFLE=OUTPUT

This entry, which is required for all printer files, is used to specify the type of file.

WORKA=YES

The WORKA=YES detail entry is mandatory for all printer files. Enter the name of the work area, which must be defined in your problem program, as the second operand in your PUT macro instructions for the particular file. The length of a work area must always be the same as that of the I/O area (if specified). For additional information regarding the use of a work area, see Work Areas under the section Overlapping and Storage Areas.

Imperative Macro Instructions

The imperative macro instructions for processing sequential printer files are described in the following order: PUT, CNTRL, PRTOV. For a description of OPEN and CLOSE refer to the section Instructions for Opening and Closing Files.

PUT MACRO INSTRUCTION

This instruction prints logical records that have been built in a specified work area.

Name	Operation	Operands
[name]	PUT	filename,workname

The first operand specifies the name of the file; the second operand specifies the name of the work area in which the records are built. The PUT macro instruction moves the record from this work area to an output area. When the output area is full, the IOCS takes the data in that area and prints them on the output device specified in the file definition statement for that file.

Individual records for a logical file may be built in the same work area or in different work areas. Each PUT macro instruction specifies the work area in which the completed record was built. However, only one work area can be specified in any one PUT macro instruction.

A PUT macro instruction for the printer file (1) moves a record from the work area to the output area, (2) initiates the print operation, and (3) immediately transfers control to the main program.

The IOCS does not clear the work area when the PUT macro instruction is executed. To prevent having interspersed characters from preceding records in the output record, ensure that

a) the records use every position of the work area, or

b) the work area is cleared before the next record is built.

CNTRL MACRO INSTRUCTION

The CNTRL macro instruction for printer files causes form spacing or form skipping. A CONTROL=YES entry must be included in the file definition statement for a printer file if one or more CNTRL macro instructions are issued for the file.

Name	Operation	Operands
[name]	CNTRL	filename,code,n,m

The first operand specifies the name of the file for which the device operation is to be performed. As the second operand, enter the mnemonic SP for form spacing or SK for form skipping. The mnemonics SP and SK as well as the operands m and n are described in detail below.

Form Spacing (SP) for Printers

The form spacing mnemonic SP is used to control line spacing. The operands n and m specify the number of lines to be spaced; n specifies immediate spacing (i.e., spacing when the CNTRL macro instruction is executed), and m specifies delayed spacing (i.e., spacing after the next line has been

(i.e., spacing after the next line has been printed by means of a PUT macro instruction). The values of both m and n range from zero to three (0 = no spacing, i.e., printing on the same line; 3 = two blank lines, i.e., printing on the third line).

Operands			Operation
code	n	m	
SP	n		Space n (n = 0, 1, 2 or 3) lines immediately
SP	n	m	Space n (n = 0, 1, 2, or 3) lines immediately and m (m = 0, 1, 2, or 3) lines after printing
SP		m	Space m (m = 0, 1, 2, or 3) lines after printing

You may omit either operand n or m. If operand n is omitted, indicate the absence of the operand by a comma.
Example: CNTRL filename,SP,,2.

Both delayed spacing and immediate spacing may be specified in a single CNTRL macro instruction preceding a PUT macro instruction for the same file.

The form will then be spaced n lines before, and m lines after the PUT macro instruction is executed. If two separate CNTRL macro instructions are issued (one for the delayed spacing and one for the immediate spacing) it is immaterial which of the two instructions is issued first. Normally, however, only one CNTRL is issued, e.g., CNTRL filename,SP,1,2 (space one line immediately and two lines after printing).

If a delayed-spacing CNTRL macro instruction is not used before the next PUT macro instruction for the file, the form is automatically spaced one line after printing. If two delayed-spacing CNTRL macro instructions are issued before the next PUT macro instruction for the file, only the second CNTRL macro instruction is effective. If both delayed spacing and skipping are specified before a PUT macro instruction for the file, only the last operation specified will be performed.

To increase the rate of output, use delayed instead of immediate spacing whenever possible.

Form Skipping (SK) for Printers

You can control the skipping of lines of a printed form using the form skipping mnemonic SK. Use operands n and/or m to specify the channel of the carriage control tape to which the form is to be skipped immediately and/or after printing of a line.

Operands			Operation
code	n	m	
SK	n		Skip to carriage-tape channel n (n=1,2,...,12) immediately
SK	n	m	Skip to carriage-tape channel n (n=1,2,...,12) immediately and to carriage-tape channel m (m=1,2,...,12) after printing
SK		m	Skip to carriage-tape channel m (m=1,2,...,12) after printing

You may omit either operand n or m. If operand n is omitted, indicate the absence of the operand by a comma.
Example: CNTRL filename,SK,,12.

When you issue two delayed skipping CNTRL macro instructions before the next PUT macro instruction, only the skipping specified in the second CNTRL macro instruction is effective. When both delayed and immediate skipping are specified either in one or in two successive CNTRL macro instructions, skipping is performed as indicated in both specifications together (i.e., skip to channel n before printing, and to channel m after printing). To increase the rate of output, use delayed rather than immediate skipping whenever possible.

PRTOV MACRO INSTRUCTION

Name	Operation	Operands
[name]	PRTOV	filename,n[,address]

Use this macro instruction (PRinT-OVerflow) for printer files to enable the program to recognize the end of a page. In the operand field, you must specify the name of the file to which the instruction pertains and the carriage-tape channel indicator (n equal to 9 or 12) to be tested. If you provide your own routine to which the program should branch on an overflow condition, specify the name of the routine as the third operand.

The PRTOV macro instruction allows you to check for printer-overflow conditions by testing whether the channel 9 or channel 12 indicator has been set on:

- before the execution of the last (preceding) PUT macro instruction referring to a printer with the standard carriage,
- before the execution of the last PUT macro instruction referring to a printer with the dual-feed carriage when only the lower feed is used, and
- before the execution of the next to last PUT macro instruction referring to a printer with the dual-feed carriage when both feeds are used.

However, if a skip has been performed or more than one line has been spaced after the last PUT macro instruction (or after the next to last PUT if both feeds of a dual-feed carriage printer are used), a punch in channel 9 or 12 that may then be sensed is lost and cannot be determined by a PRTOV macro instruction.

The program branches to your end-of-page routine if the tested indicator is on and the name of your routine has been specified as the third operand. In the end-of-page routine, any IOCS macro instruction (except PRTOV) may be issued, e.g., to print page totals and, upon a skip to channel 1, heading lines on the new page. At the end of the routine, control must be returned to the IOCS by branching to the address contained in register 14.

If IOCS macro instructions are used in the end-of-page routine, the contents of register 14 must be saved before these instructions are executed.

If a third operand has not been specified in the PRTOV macro instruction, an automatic skip to channel 1 is performed when the tested indicator is on.

The DTFSR file definition statement must have a PRINTOV=YES entry when a PRTOV macro instruction is issued for the file.

Instructions for Processing Printer-Keyboard Files

Input and output records for printer-keyboard files must be unblocked format-F records. The input record length may range from 2 to 511 bytes, the output record length, from 1 to 511 bytes. However, if the simulated carriage-control tape feature is used, the output length is limited to 125 bytes.

The IOCS routines to process input or output files on the IBM 2152 Printer-Keyboard can be included in the problem program through the use of the macro instructions described in this section. Use the DTFPK file definition statement to describe your file. Printer-keyboard output files with carriage control also require a DTFLC statement to define a line-counter table. This line-counter table simulates a carriage-control tape.

DTFPK Statement

This file definition statement describes the characteristics of the file to be processed. Since both printer-keyboard input and output files can be processed in a single program, you must write a separate DTFPK statement for each file. However, only one printer-keyboard input and one printer-keyboard output file can be defined in a single program. If both input and output operations are performed on the printer-keyboard in a single program, and if skipping or use of the print-overflow routine is desired, take into consideration that line advances resulting from READ macro instructions are not registered by the line-counter.

The name field of the header entry must contain the file name, and the operation field must contain DTFPK.

The detail entries to be made in the operand field describe the file and specify symbolic addresses of routines and areas used during the processing of the file. They may appear in any order. For ease of reference, they are discussed below in alphabetical order.

BLKSIZE=n

This entry specifies the length of the printer-keyboard input or output records. The specification n must be equal to, or greater than, the number of bytes contained in the longest record. The maximum record length is 511 bytes. If a line-counter table (LCTABLE=YES or PRINTOV=YES) is specified for an output file, the record length and, therefore, the BLKSIZE must not exceed 125 bytes to avoid line overflow and the resulting unaccountable line advances. The minimum record length is two bytes for input files (to allow for EOF indicator /*), and one byte for output files.

Note that the actual length of an input record is determined not by the BLKSIZE entry, but by pressing the EOT (End-of-Transmission) key to indicate the end of a data record.

A BLKSIZE entry is mandatory for input and output files.

CONTROL=YES

This entry is required if a CNTRL macro instruction is issued for an output file. A CNTRL macro instruction causes spacing or skipping of the form on the printer-keyboard. If skipping is desired, enter the detail entry LCTABLE=YES in the DTFPK statement and define a line-counter table by the DTFLC statement.

Note that a CONTROL=YES entry is required if LCTABLE=YES is specified.

EOFADDR=name

This entry is mandatory for input files. It specifies the name of the routine in the problem program to which the IOCS branches if the WAITF macro instruction in the problem program detects an end-of-file condition. In the end-of-file routine you can perform any operation required for the end of the file. Usually, a CLOSE macro instruction is issued.

To indicate the end-of-file condition on the printer-keyboard enter /* as the first two characters of a record.

Note: If in a mainline program the end-of-file routine calls EOJ, you must also close all disk files in this end-of-file routine.

IOAREA=name

This entry specifies the name of the input or output area to be used for the file. The name must be the symbol used to define the area in the source program. The area must be large enough to accommodate the largest record as defined in the BLKSIZE entry. The I/O area is not cleared by the IOCS.

The IOAREA entry is mandatory for an input file. Although the length of the input area must be equal to the maximum record size, the records entered need not fill the entire area, since the operator indicates the end of the data record by pressing the End-of-Transmission (EOT) key on the printer-keyboard. If you omit the IOAREA entry for an output file, then the output area (INQOPT) allocated at the time the Monitor is generated is used for the file. This output area is provided if the generated Monitor contains the inquiry facilities. (For a description of the use of the printer-keyboard output area in the Monitor refer to the section The Inquiry Program).

LCTABLE=YES

This entry is optional for an output file. It indicates the presence of a table that simulates a carriage-control tape for the printer-keyboard. The table associates lines with channel numbers. These channel numbers may then be specified in a CNTRL macro instruction in the program to provide form skipping, and in a PRTOV macro instruction to provide overflow-printing.

You must describe the line-counter table by the DTFLC statement.

PRINTOV=YES

Include this entry if a PRTOV macro instruction for an output file is used in the program to test an overflow condition. If PRINTOV=YES is specified, control and line-counter routines are generated automatically and the entries CONTROL=YES and LCTABLE=YES are not required in the DTFPK file definition statement. However, you must define a line-counter table by a DTFLC statement.

RECSIZE=(register)

This entry specifies a register that contains the length of the output record at the time a PUT macro instruction is executed in the program. You may specify in parentheses any one of the registers 8 through 13, or a symbolic name that has been equated to one of the registers 8 through 13.

In your problem program, load the length of the record into the specified register before issuing the corresponding PUT macro instruction. The maximum record length is 511 bytes. If a line-counter table is specified, the record length must not exceed 125 bytes. The register must contain the length of the record in binary format. If the record length loaded into the register exceeds the length specified in the BLKSIZE entry, the record is truncated on the right by the number of characters in excess of the specified BLKSIZE.

The RECSIZE entry is optional. However, you are strongly recommended to use this entry to save processing time. If you omit the entry, the number of bytes specified in the BLKSIZE entry will be printed; e.g., if your record is 5 bytes long, BIKSIZE=125, and RECSIZE is not specified, 125 characters (5 as desired and 120 blanks) are printed.

TYPEFLE=code

This mandatory entry is used to specify the type of file.

Code	Type of File
INPUT	An input file
OUTPUT	An output file

If you use both a printer-keyboard input and a printer-keyboard output file, you must issue a DTFPK statement for each file.

WORKA=YES

The WORKA=YES entry is mandatory for printer-keyboard output files. Enter the name of the work area as the second operand in your PUT macro instructions for the particular file. The IOCS assumes the length of a work area to be the same as that of the I/O area, or to be equal to the value in the RECSIZE register if RECSIZE was specified. When WORKA=YES is omitted, a warning message is given, but the assembly continues.

DTFLC Statement

The DTFLC statement describes the characteristics of the simulated carriage-control tape. It is used for an output file on the printer-keyboard to define the line-counter table. If DTFLC is specified, either the LCTABLE and CONTROL entries or the PRINTOV entry must be included in the DTFPK statement for the file.

The format of the DTFLC statement is as follows:

Name	Operation	Operands
	DTFLC	formsize,E1,E2,...,E48

The first operand (formsize) refers to a decimal value (1 to 254) specifying the

total number of lines on the form. The remaining operands, E1,E2,...,E48, refer to five-digit decimal values. The first three digits indicate a line number (001 to 254), to which a channel number (01 to 12) is assigned in the fourth and fifth digits. Only one channel must be assigned to a line, but more than one line can be assigned to a channel.

Channel 01 must be specified at least once. There are no limitations as to the sequence of entries. The maximum number of line and channel entries is 48. The name field of the DTFLC statement must be blank.

An example of the entries in the DTFLC is given below. Note that high-order zeros must be present in each entry except the formsize entry.

Example:

Name	Operation	Operand
	DTFLC	72,00601,01203,06612,06712
		Number of lines (Formsize) / Line Number / Channel Number

Note: The line-counter registers only those carriage advances that are initiated by the execution of PUT macro instructions (i.e., the automatic one-line space which occurs during the execution of a PUT) and by CNTRL macro instructions. The line counter does not register carriage advances caused by:

1. any external manipulation of the form, i.e., advancing the carriage by hand or switching the On-line, Off-line key;
2. execution of READ macro instructions for the printer-keyboard;
3. line overflows due to repeated printing with suppressed spacing (CNTRL with delayed space 0).

It is your responsibility to consider the effects of the above conditions during the execution of the program.

Imperative Macro Instructions

The imperative macro instructions for processing printer-keyboard files are described in the following order: PUT, READ, WAITF, CNTRL, PRTOV. For a description of OPEN and CLOSE refer to the section Instructions for Opening and Closing Files.

PUT MACRO INSTRUCTION

Name	Operation	Operands
[name]	PUT	filename,workname

The first operand specifies the name of the file; the second operand specifies the name of the work area in which the records are built. You can have more than one work area if you issue separate PUT macro instructions for writing records from each area.

When a printer-keyboard output file is processed, a PUT macro instruction (1) moves a record from the work area to the printer-keyboard output area, (2) initiates the print operation, and (3) returns control to the main program. The length of the record moved to the output area is derived from the BLKSIZE entry in the DTFPK statement, or, if RECSIZE is specified, from the contents of the register specified under the RECSIZE entry.

Note that the IOCS does not clear the work area or the output area after a PUT macro instruction. You should ensure that no characters from a preceding record remain in the work area.

READ MACRO INSTRUCTION

Name	Operation	Operand
[name]	READ	filename

This macro instruction is used for printer-keyboard input files. The operand specifies the name of the file from which the record is to be read.

The READ macro instruction transfers the characters typed on the printer-keyboard to the I/O area specified in the IOAREA detail entry of the DTFPK file definition statement. An incorrectly typed-in record can be cancelled by pressing the Cancel key. The input area is then cleared and the input record can be re-entered.

The READ macro instruction is executed in the overlap mode, i.e., after the READ has been issued, program execution continues while the operator enters the input record. The actual length of the record is determined by pressing EOT. The maximum length of the record is defined by the BLKSIZE entry. If the maximum length is reached before EOT is pressed, the keyboard locks. Before the input record can actual-

ly be processed, you must issue a WAITF macro instruction to ensure that the record has been completely transferred to main storage.

Note that the input area is not cleared by the IOCS when you issue a READ macro instruction. If your input records do not have the same length, clear the input area before issuing a READ macro instruction.

WAITF MACRO INSTRUCTION

Name	Operation	Operand
[name]	WAITF	filename

The operand specifies the name of the file for which you issue the WAITF macro instruction. The WAITF macro instruction ensures that the execution of the preceding READ macro instruction has been successfully completed. You must issue a WAITF macro instruction before processing the record read by the associated READ macro instruction.

CNTRL MACRO INSTRUCTION

Use the CNTRL macro instruction to specify line spacing or form skipping for a printer-keyboard output file. You must include a CONTROL=YES entry in the DTFPK file definition statement if you issue a CNTRL macro instruction.

For skipping, you must also specify the DTFPK detail entry LCTABLE=YES and a DTFLC statement to define a line-counter table simulating the carriage-control tape.

Name	Operation	Operands
[name]	CNTRL	filename,code,n,m

The CNTRL macro instruction contains CNTRL in the operation field, and the name of the file as the first operand. As the second operand (code), enter the mnemonics SP (for spacing) or SK (for skipping). The mnemonics SP and SK, and the operands n and m are described in detail below.

Form Spacing (SP) for the Printer-Keyboard

Enter SP as the second operand (code) in a CNTRL macro instruction to specify line spacing. The operands n and m specify the number of lines to be spaced; n specifies immediate spacing (i.e., spacing when the CNTRL macro instruction is executed), and m specifies delayed spacing (i.e., spacing after the next line has been printed by means of a PUT macro instruction). The values of both m and n range from zero to three (0 = no spacing, i.e., printing on the same line; 3 = two blank lines, i.e., printing on the third line).

Operands			Operation
code	n	m	
SP	n		Space n(n = 0, 1, 2 or 3) lines immediately
SP	n	m	Space n(n = 0, 1, 2, or 3) lines immediately and m(m = 0, 1, 2, or 3) lines after printing
SP		m	Space m(m = 0, 1, 2, or 3) lines after printing

You may omit either operand n or m. If you omit operand n, indicate the absence of the operand by a comma.
Example: CNTRL filename,SP,,2.

Both delayed spacing and immediate spacing may be specified in a CNTRL macro instruction preceding a PUT macro instruction for the same file. If you use two separate CNTRL macro instruction for immediate and delayed spacing, specify immediate spacing before delayed spacing in order for both specifications to be effective. If a delayed-spacing specification precedes an immediate-spacing specification, only the immediate spacing is executed.

If a delayed-spacing CNTRL is not issued before the next PUT macro instruction for the same file, the form is automatically spaced one line after printing. When two delayed-spacing CNTRL macro instructions are issued before the next PUT macro instruction for the file, only the second delayed-spacing CNTRL is effective. If both delayed spacing and skipping are specified before a PUT macro instruction for the file, only the last operation specified will be performed.

Note: If you issue a CNTRL macro instruction with delayed space 0, the automatic space performed during the execution of the PUT macro instruction is suppressed and the type head remains in the print position following the last character printed. Hence, if the end of the print line is reached during execution of a succeeding PUT macro instruction, an automatic advance to the next line occurs. This advance is not registered in the line counter table.

To increase processing time, use delayed rather than immediate spacing whenever possible.

Form Skipping (SK) for the Printer-Keyboard

Enter SK as the second operand (code) in a CNTRL macro instruction to specify form skipping. Operands n and/or m specify the channel to which immediate and/or delayed skipping is to be performed.

Operands			Operation
code	n	m	
SK	n		Skip to carriage-tape channel n (n=1,2...,12) immediately
SK	n	m	Skip to carriage-tape channel n (n=1,2,...,12) immediately and to carriage-tape channel m (m=1,2,...,12) after printing
SK		m	Skip to carriage-tape channel m (m=1,2,...,12) after printing

You may omit either operand n or m. If you omit operand n, indicate the absence of the operand by a comma.
Example: CNTRL filename,SK,,12.

When two delayed-skipping CNTRL macro instructions are issued before the next PUT macro instruction, only the skipping specified in the second CNTRL macro instruction is effective.

When you specify delayed and immediate skipping in two separate CNTRL macro instructions, issue the specification for immediate skipping first in order for both specifications to be effective. If you specify delayed skipping first, only the immediate-skipping specification is executed.

If you use form skipping for a printer-keyboard output file, you must define a line counter table in the program. This requires a LCTABLE=YES entry (unless PRINTOV=YES has been specified) in the DTFPK statement and a DTFLC line-counter definition statement.

To increase processing time use delayed rather than immediate skipping whenever possible.

PRTOV MACRO INSTRUCTION

Name	Operation	Operands
[name]	PRTOV	filename,n[,address]

This macro instruction (PRinT OVerflow) applies to printer-keyboard output files. In the operand field, specify the name of the file to which the instruction pertains and the channel indicator (n equal to 9 or 12) to be tested. If you provide a routine to which the program should branch on an overflow condition, specify the name of this routine as the third operand.

The PRTOV macro instruction allows you to check for printer overflow conditions by testing the channel 9 or the channel 12 indicator. The channel 9/12 indicator is set on whenever a PUT or CNTRL macro instruction increases the line counter to or beyond the line number associated with channel 9 (12). The indicator is set off after it has been tested by a PRTOV statement or when channel 1 is reached.

If you specify the address of a routine as third operand, the program branches to this routine if the tested indicator is on. In the specified routine, you may issue any IOCS macro instruction except PRTOV. This allows you, for instance, to print overflow page headings. At the end of the routine, return control to the IOCS by branching to the address contained in register 14.

If you use IOCS macro instructions in your routine, you must first save the contents of register 14 to prevent losing the return address. If you do not provide your own overflow routine, an automatic skip to channel 1 is performed when the tested indicator is on.

Include a PRINTOV=YES in the DTFPK file definition statement when you use the PRTOV macro instruction for a printer-keyboard output file, and provide a line-counter table simulating the carriage-control tape by means of a DTFLC statement.

The IOCS provides routines for processing magnetic tape files. These files are processed sequentially. Records in magnetic tape files can be blocked or unblocked format-F, format-V, or format-U records. You must describe your magnetic tape file in the source program using the DTFMT file definition statement. To cause the required I/O operation, you have to issue imperative macro instructions. The DTFMT file definition statement and the imperative macro instructions referring to the magnetic tape files are described in this section.

DTFMT Statement

The DTFMT statement applies to magnetic tape files only. The name field of the header entry must contain the name for the file and the operation field must contain DTFMT. For ease of reference, the detail entries to be made in the operand field are described below in alphabetical order.

ALTTAPE=code

This entry specifies the symbolic address of a magnetic tape drive that will be used as an alternate when a magnetic tape file has two or more reels (volumes) of data.

Code	Magnetic Tape Drive
SYSIPT	System input tape drive.
SYSOPT	System output tape drive.
SYSnnn	Any other magnetic tape drives attached to the system. The specification nnn may be any number from 000 to 019.

You can assign the physical tape-drive address to the symbolic address either when building the system or by means of an assign (ASSGN) statement read by the Job Control program before the problem program is executed. If you use the ASSGN statement, you can mount the second (fourth, sixth, etc.) reel of tape on any one of the magnetic tape drives attached to the system by merely assigning that drive to the specified symbolic address. You could then mount the first (third, fifth, etc.) reel of tape on the magnetic tape drive specified in the DEVADDR entry of the DTFMT statement for the file.

The method described above allows the operator sufficient time to mount the third reel on the magnetic tape drive specified in the DEVADDR entry while the records on the second reel are processed. He can mount the fourth reel on the magnetic tape drive specified in the ALTTAPE entry while the records on the third reel are processed; and so on.

You can specify the detail entry ALTTAPE for both input and output files. If you specify it for an output file, the IOCS switches the magnetic tape drives in accordance with the ALTTAPE specification on detecting an end-of-volume condition, i.e., when the reflective marker at the end of the magnetic tape is sensed.

If you issue the entry for an input file, the functions of the IOCS vary depending on the type of labels (if any) specified in the file.

1. Standard Labels. The IOCS switches the magnetic tape drives in accordance with the ALTTAPE specification.

2. Non-standard Labels or No Labels. The IOCS has no means of determining the end of a volume. When a tapemark is sensed, the IOCS transfers control to the EOFADDR routine, which you may code to determine whether an end-of-file or an end-of-volume condition exists. In the case of an end-of-volume condition, issue an FEOV macro instruction. This causes the IOCS to switch the magnetic tape drives in accordance with the ALTTAPE specification, and then to return control to the instruction following the FEOV macro instruction.

Note: ALTTAPE must not be specified when READ=BACK or DTFEN OVLAY is specified.

I/O device assignment is described in detail in the SRL publication IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006.

BLKSIZE=n

This entry specifies the length of the I/O area to be used by the file. The specification n must be equal to the length, in bytes, of the area reserved in the problem program. If the record format is variable-length, n must be equal to the number of bytes contained in the longest block of records.

The maximum length acceptable to the IOCS is 4095 bytes, which is equal to the maximum block length for IBM 2415 Magnetic

Tape Units connected to an IBM System/360 Model 20. The minimum block length is 18 bytes, except for tape input files containing checkpoint records. For these tape files, the minimum block length is 20 bytes.

If unblocked records or records of undefined format are to be processed in a work area, consider the following: A GET macro instruction causes the IOCS to move the number of bytes specified in the detail entry BLKSIZE from the input area to the work area. A PUT macro instruction causes the IOCS to move this number of bytes from the work area to the output area. Therefore you must ensure that the address of the work area you use for an output file is equal to or lower than the upper main-storage limit minus the BLKSIZE specification.

Note: For variable-length unblocked records, the first four bytes of the work area contain control information (two bytes record length and two bytes binary zeros).

CKPTREC=YES

This entry is required if a tape input file contains checkpoint records interspersed among the data records. When this entry is provided, the IOCS recognizes the checkpoint records and bypasses them.

Magnetic tape files created by means of the Model 20 IOCS do not contain any checkpoint records. Therefore, this entry is required only if a magnetic tape is to be read that was created by means of another program and contains interspersed checkpoint records.

If the detail entry CKPTREC is specified for a tape input file, you must specify a block length (BLKSIZE entry) of at least 20 bytes.

A group of checkpoint records is identified by a header and a trailer identifier, each of which contains the characters ///bCHKPTb// (where b = blank). You must ensure that none of the input blocks contain this character combination in the first twelve positions.

The IBM System/360 Model 20 DPS IOCS does not provide for the creation of checkpoint records on magnetic tape.

CONTROL=YES

This entry is required if a CNTRL macro instruction is to be issued for the file.

The CNTRL macro instruction causes the associated magnetic tape drive to perform operations such as tape rewind, rewind and unload, backspace, and so on.

DEVADDR=code

This entry specifies the symbolic address of a magnetic tape drive to be associated with the particular file. The symbolic address represents a physical tape drive address.

Code	Magnetic Tape Drive
SYSIPT	System input tape drive.
SYSOPT	System output tape drive.
SYSnnn	Any other tape drives attached to the system. The specification nnn may be any number from 000 to 019.

You may assign a physical tape drive address to the symbolic address either when building the system or by means of an assign (ASSGN) statement read by the Job Control program before the problem program is executed. If the ASSGN statement is used, a reel of tape may be mounted on any magnetic tape drive that is available at the time the job is ready to be run; this drive need then merely be assigned to the specified symbolic address. Refer to the SRL publication IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006.

EOFADDR=name

This entry is mandatory for input files. It specifies the name of the routine in the problem program to which the IOCS branches on an end-of-file condition. In that routine, you can perform any operation required for the end of the file, e.g. issue a CLOSE macro instruction. However, you must not issue a GET macro instruction for this file in your EOFADDR routine.

Note: If, in the end-of-file routine, you wish to discontinue the processing of a program that is a mainline program, you must first close all disk files.

An end-of-file condition is detected by reading a tapemark and the characters EOF in the trailer label when standard labels are specified. If standard labels are not specified, the IOCS assumes an end-of-file condition when it reads a tapemark. Since the IOCS does not distinguish between an end-of-file condition and an end-of-volume condition if no labels or non-standard labels are specified, include in your EOFADDR routine a test to determine whether EOF or EOV has occurred.

ERRIO=name

This entry specifies the symbolic name of a two-byte area, in which the IOCS places the address of:

1. The I/O area containing the block that caused an irrecoverable read error (if the name of the error routine is specified in the ERROPT entry),

2. The I/O area containing the wrong-length record (if the name of the wrong-length record routine is specified in the WLRERR entry).

This entry may only be issued if ERROPT=name, and/or WLRERR=name, and two I/O areas are specified.

ERROPT=code

This entry is required for a tape input file if you do not want to terminate the job when a tape read error cannot be corrected by the error routine.

Code	Explanation
IGNORE	The block containing the error is to be handled as if there were no errors (it is included in the block count).
SKIP	The block containing the error is to be skipped, i.e., it should not be made available for processing. The next block is read from tape and processing continues with the first record of that block. However, the block containing the error is included in the block count.
name	This is the name of a routine in the problem program to which the IOCS should branch on a tape read error. When the IOCS branches to this routine, the block containing the error is in the I/O area. In this routine, you must not issue any GET macro instructions for records in the block containing the error because a subsequent GET macro instruction referring to the file will read the next block of records. If you use any other IOCS macro instructions in this routine, save the contents of register 14. At the end of this routine, you must return control to the IOCS by means of a branch to the address contained in register 14. When this return branch has been performed, the IOCS executes the instruction following the GET macro instruction that made the error record available in the input area. The next GET macro instruction referring to the file that contains the block causing the error will make the first record of the next block available for processing. Note: If, in the error routine, you wish to discontinue the processing of a program that is a mainline program, you must first close all disk files.

This entry also applies to wrong-length records if the WLRERR detail entry is not included. Absence of the ERROPT entry causes the job to be discontinued automatically when a permanent read error occurs.

FILABL=code

This entry specifies the type of label processing to be performed.

Code	Type of Processing
STD	For a tape input file, standard labels are to be checked. For a tape output file, standard labels are to be written.
NSTD	Non-standard labels exist. The code NSTD is possible for input files only. Because the non-standard labels are skipped, the non-standard label set must be terminated by a tapemark.
NO	No labels exist. Note that any existing volume label on an output file will be overwritten if you specify FILABL=NO for an output file.

For a magnetic tape without labels, you can omit the FILABL entry. The IOCS then assumes that there are no labels.

IOAREA1=name

This entry specifies the name of the I/O area to be used by the file. This name must be the symbol used in the DS or DC statement that defines the area in the source program.

If the file contains format-V records, reserve four bytes of the I/O area for the blocksize field. The I/O area must be on a halfword boundary. An IOREG=(register) entry is required unless unblocked records are processed or a work area is used.

IOAREA2=name

In a Model 20, Submodel 5, you can specify two input or output areas for a file to permit overlapping of data transfer with processing operations. The name must be identical with the symbol used in the DS or DC statement defining the area in the source program. When you use an IOAREA2 entry you must also specify either IOREG or WORKA. The I/O area must be on a halfword boundary. If the file contains format-V records, reserve 4 bytes of the I/O area for the blocksize field.

IOREG=(register)

This entry specifies a register that contains either the address of a logical input record that is available for processing or the address of the area that is available for building the next output record. Immediately after the equal sign you must specify a register number enclosed in parentheses. Any of the registers 8 through 13 can be used. You may also specify (in parentheses) a symbolic name that has been equated to one of the registers 8 through 13. You can use one and the same register for several files.

You must include an IOREG entry in the DTFMT file definition statement when:

- blocked input or output records are processed in the I/O area; or
- variable-length unblocked records are read backward and are processed in the input area; or
- IOAREA2, but no WORKA is specified.

The register specified in the IOREG entry is loaded by the IOCS during the execution of the Open routine for the file concerned.

A GET (PUT) macro instruction you issue for the file causes the IOCS to increment the contents of the register specified in the IOREG entry by the number of bytes contained in the input (output) record.

Omit this entry if you include a WORKA=YES entry in the DTFMT statement for the file.

LABADDR=name

You may use up to eight user labels in addition to the IBM standard file header or trailer labels. If so, write a routine to check or build the user label(s). Specify the name of the routine in the LABADDR entry. The IOCS branches to this routine after it has processed the IBM standard label or a preceding user label. (Refer to the description of the LBRET Macro Instruction). If this entry is omitted for an input file containing user labels, these additional labels are not checked.

For input files, you can determine the type of label from the first three bytes of the label contained in the IOCS label area. For output files, the IOCS indicates the type of label that is to be written by putting one of the following codes into the low-order byte of register 8.

Code	Type of Label
C 'O'	Header label (when a file is opened)
C 'F'	End-of-file label (on an end-of-file condition)
C 'V'	End-of-volume label (on an end-of-volume condition)

The high-order byte is blank.

The IOCS places the address of the IOCS label area into register 9 at the time a LABADDR routine is being entered. At the end of a LABADDR routine, you must issue a LBRET macro instruction to return to the IOCS. (Refer to the description of the LBRET Macro Instruction).

Note: Do not destroy the contents of registers 14 and 15. Refer to the discussion on register usage under Programming Considerations.

READ=code

This entry specifies the direction in which an input tape is to be read. If you omit the entry, the IOCS assumes forward reading.

Code	Explanation
FORWARD	A magnetic tape is to be read in forward direction.
BACK	A magnetic tape is to be read backward. However, you cannot specify READ=BACK when the tape input file contains variable-length block records or when the entry ALTTAPE is specified for the file.

RECFORM=code

This entry specifies the record format of the file. The IOCS can handle all of the different types of records in the same program. However, all records in one file must be of the same format. If you omit the RECFORM entry, the IOCS assumes unblocked format-F records.

Code	Record Format
FIXUNB	Unblocked format-F records.
FIXBLK	Blocked format-F records.
VARUNB	Unblocked format-V records.
VARBLK	Blocked format-V records.
UNDEF	Format-U records.

For a description of the record formats refer to Data Files.

When variable-length records are specified for a tape output file, the I/O area must include four additional bytes in which the block-length indication is built. If these records are unblocked, the four additional bytes are used to build the length indication for each record since each record is at the same time a block. If the records are blocked, the four additional bytes are used to build the length indication for the entire block. The minimum and maximum record lengths permitted are as shown below. Lengths are given in number of bytes.

	Minimum		Maximum	
Record Type	Input	Output	Input	Output
FIXUNB	1	18	4095	4095
FIXBLK	1	18	4095	4095
VARUNB*	14	14	4091	4091
VARBLK*	14	14	4091	4091
UNDEF	1	18	4095	4095

* Excluding the four bytes required for record length indication.

RECSIZE=code

This entry applies to magnetic tape files containing either blocked format-F or format-U records.

Code	Explanation
n	The operand n specifies the number of bytes in an individual record for a tape file containing blocked format-F records.
(r)	For a tape file containing records of undefined format, the operand (r) specifies a register number enclosed in parentheses, or a symbolic name (in parentheses) that has been equated to a register number. You may use any of the registers 8 through 13. The IOCS uses the register specified to indicate the record length of an input file or to derive from its contents the record length of an output file. You must place the number of bytes contained in an output record into the specified register before you can issue a PUT macro instruction for the file.

REWIND=code

This entry is used to specify the desired rewind and unload operation when an OPEN or CLOSE macro instruction is issued or when an end-of-volume or end-of-file condition is sensed.

Code	Operation
UNLOAD	Rewind the magnetic tape when an OPEN macro instruction is issued, and rewind and unload the magnetic tape when a CLOSE macro instruction is issued or an end-of-volume or an end-of-file condition occurs.
NORWD	No rewind is desired. This entry is mandatory for files to be read backward.

If the REWIND entry is not included, an OPEN or CLOSE macro instruction, an end-of-volume, or an end-of-file condition causes the magnetic tape file to be rewound but not unloaded.

TPMARK=NO

This entry applies only to unlabeled tape output files (FILABL=NO). If this entry is included, it will prevent the writing of a tapemark as the first record on a tape. If you omit this entry, a tapemark will be written as the first record.

TYPEFLE=code

This entry is used to specify the type of file (i.e., input or output).

Code	Type of File
INPUT	An input file
OUTPUT	An output file

VARBLD=(register)

You must specify this entry if you process an output file with blocked format-V records and if no work area is specified. The operand (register) must contain a register number enclosed in parentheses, or a symbolic name (in parentheses) that has been equated to a register number. Any of

the registers 8 through 13 can be specified. The contents of the register, which are loaded by the IOCS, indicate how many bytes are available in the output area for building the next record.

After a PUT macro instruction is issued for a format-V record, the IOCS calculates the number of bytes still available and loads this number into the register specified in the VARBLD entry. You must then compare the length of the next record with the available space. If the record will not fit, you must issue a TRUNC macro instruction to cause the completed block of records to be written on the magnetic tape file. When the block has been written the current record is placed into the beginning of the output area and becomes the first record in the next block. For information regarding the PUT and TRUNC macro instructions, refer to the descriptions under PUT Macro Instruction and TRUNC Macro Instruction.

WLRERR=name

This entry does not apply to files containing undefined records. It specifies the name of a routine to which the IOCS will branch if a wrong-length physical record (block) is read.

When the IOCS branches to the specified routine, the I/O area contains the wrong-length record. In the routine, you may perform any desired operation for wrong-length blocks except issuing a GET macro instruction that refers to any logical record in the wrong-length block, because the GET macro instruction following the GET that detected the length error makes the first record of the next block available for processing. A wrong-length block is included in the block count.

Note: Do not destroy the contents of registers 14 and 15. Refer to the discussion on register usage under Programming Considerations.

At the end of your routine, return to the IOCS by providing a branch to the address contained in register 14. When this return branch has been performed, the IOCS executes the instruction following the GET that made the wrong-length record available for processing.

Whenever blocked format-F records or format-V records are specified (RECFORM=FIXBLK, =VARUNB, or =VARBLK), the machine check for wrong-length records is ignored, and the IOCS generates a programmed check of record length. For blocked format-F records, the record length is considered incorrect if the block that is read is not an integer multiple of the record length (specified in the RECSIZE entry) up to the maximum length of the block (specified in the BLKSIZE entry). This permits short blocks of records to be read without a wrong-length record indication.

For format-V records, the record length is considered incorrect if the length of the block is not the same as the block length specified in the first two bytes of the block.

If unblocked format-F records are specified (RECFORM=FIXUNB), the IOCS utilizes the machine check to determine whether or not a record is of correct length. Specifying RECFORM=FIXUNB causes the number of bytes specified in the detail entry BLKSIZE to be inserted in the generated XIO instructions. Any record whose length is not equal to the specified number of bytes causes a wrong-length-record indication.

Note that the IOCS does not provide the number of bytes contained in the wrong-length record.

If you omit the WLRERR entry and a wrong-length record is detected by the IOCS, one of the following results:

1. If you included the ERROPT entry for this file, the IOCS treats the wrong-length record as an error block and handles it according to your specifications for an error (IGNORE, SKIP, or branch to the error routine).

2. If you omitted the ERROPT entry, the job will be terminated.

WORKA=YES

This entry specifies that records are to be processed in a work area rather than in an I/O area. The name in the DS or DC statement that reserves the work area must be used as the second operand of each GET or PUT macro instruction for the file. You must specify WORKA=YES if you include an IOAREA2 entry but omit the IOREG entry.

Note: Define the work area on a halfword boundary if it is used to process blocked format-V records.

Imperative Macro Instructions

The imperative macro instructions for magnetic tape files are described in the following order: PUT, GET, CNTRL, TRUNC, RELSE, LBRET, FEOV. For a description of OPEN and CLOSE refer to the section Instructions for Opening and Closing Files.

PUT MACRO INSTRUCTION

This instruction writes logical records that have been built directly in an output area or in a specified work area.

Name	Operation	Operands
[name]	PUT	filename[,workname]

You can write the PUT macro instruction with one or two operands, depending on the area in which the records are built.

1. One Operand Specified. Use this format if output records are to be built directly in an I/O area. The operand specifies the name of the magnetic tape file in which you want to place the record. The file name must be the same as the one used in the header entry of the DTFMT file definition statement.

When blocked records are processed directly in an output area, the use of an I/O register is required (refer to the description of the IOREG=(register) detail entry under DTFMT Statement). After an OPEN macro instruction for the file the I/O register contains the first available position of the output area. A PUT macro instruction causes the IOCS to (1) include the current record in a record block and (2) change the address in the I/O register to identify the next available position of the output area in which you can build the next record.

For a file with unblocked records, a PUT macro instruction causes the IOCS to write the affected record onto magnetic tape. You do not need an I/O register when processing unblocked records, unless IOAREA2 is specified for the file.

If blocked format-V records are built in the output area, specify an additional register into which the IOCS places the number of bytes that are still available in the output area for the next record (refer to the VARBLD=(register) entry under DTFMT Statement and to the description of the TRUNC Macro Instruction).

Note: The IOCS does not clear the output area after a PUT macro instruction. To avoid having interspersed characters from preceding records in your output record, use every position of the output area or clear the output area after each PUT macro instruction that writes a block out on magnetic tape.

2. Two Operands Specified. Use this format if records are processed in a work area. The first operand specifies the name of the file. The second operand specifies the name of the work area from which records are moved to the output area. When an output area is full, the IOCS takes the data in that output area and writes them onto magnetic tape. You can use more than one work area if you issue separate PUT macro instructions for writing records from each work area.

If blocked format-V records are built in a work area, the PUT routine checks the length of the output record to determine whether or not the record will fit into the remaining portion of the output area. If it does, the IOCS moves the record into the output area. If the record does not fit, the PUT routine causes the completed block to be written and then moves the record into the output area.

Do not specify IOREG=(register) if you use a work area (refer to the description of WORKA=YES under DTFMT Statement).

Note that the IOCS does not clear the work area after a PUT macro instruction. You should ensure that no characters from preceding records remain in the area either by clearing the work area after each PUT macro instruction or by using every position of the work area.

GET MACRO INSTRUCTION

This macro instruction makes the next sequential logical record from a magnetic tape input file available for processing in either an input area or a specified work area.

When an end-of-file condition is detected, the IOCS transfers control to the end-of-file routine specified by the EOFADDR entry in the DTFMT statement.

The GET macro instruction may cause a read-forward or a read-backward operation. Specify the type of read operation by the READ=code entry in the DTFMT statement.

Name	Operation	Operands
[name]	GET	filename[,workname]

You can write the GET macro instruction with either one or two operands, depending on the area in which the records are processed.

1. One Operand Specified. Use this format if records are to be processed directly in an input area. The operand specifies the name of the file from which the record is to be read. This name must be the same as the one you used in the DTFMT statement for the file.

When blocked records are processed in an input area, you must specify an I/O register that the IOCS needs to identify the next record to be processed. (Refer to the description of the IOREG=(register) entry under DTFMT Statement). This I/O register always contains the address of the currently available record. The GET routine places the proper address into the I/O register.

No I/O register is needed for a file with unblocked records, unless IOAREA2 is specified for the file.

2. Two Operands Specified. Use this format if records are to be processed in a work area. The first operand specifies the name of the file. The second operand specifies the symbolic name of the work area to be used (refer to the description of the WORKA=YES entry under DTFMT Statement). You can specify more than one work area for a file if you issue separate GET macro instructions to move the records into the different work areas (only one work area can be specified in any one GET macro instruction). It might be advantageous to plan two work areas, and to specify each area in alternate GET macro instructions. This allows you to determine a possible change in the control level by comparing each record with the preceding one.

An I/O register must not be used when the records are processed in a work area.

CNTRL MACRO INSTRUCTION

The CNTRL macro instruction is used to control magnetic tape functions that are not concerned with reading data from, or writing data on, the magnetic tape. Include a CONTROL=YES entry in the DTFMT file definition statement if you issue a CNTRL macro instruction for the file.

Name	Operation	Operands
[name]	CNTRL	filename,code

The control macro instruction contains CNTRL in the operation field, and the name of the tape file for which the operation is described as the first operand. As a second operand, enter one of the mnemonics listed below to specify the operation.

Operand (Code)	Operation
BSF	Backspace file, i.e., backspace tape to preceding tapemark
BSR	Backspace record, i.e., backspace tape to preceding interblock gap
ERG	Erase gap, i.e., erase tape to produce a gap
FSF	Forward space file, i.e., forward space tape to next tapemark
FSR	Forward space record, i.e., forward space tape to next interblock gap
REW	Rewind tape
RUN	Rewind and unload tape
WTM	Write a tapemark

BSF (Backspace File). Use this mnemonic if you want to backspace the tape file. When a CNTRL macro instruction with BSF as the second operand is executed, the IOCS causes the tape to be stopped at the tapemark preceding the first data record of the file. In the case of an input file, the tapemark is read during the next read-forward operation and the program then branches to your end-of-file routine. In the case of an output file, the next PUT macro instruction that refers to the file causes the tapemark to be overwritten.

BSR (Backspace Record). Use this mnemonic if you want to backspace a tape file by one block. When a CNTRL macro instruction with BSR as the second operand is executed, the IOCS causes the tape to be stopped at the block just backspaced in the proper position for re-reading during the next read-forward operation. The IOCS immediately branches to your end-of-file routine if the operand BSR refers to an input file and a tapemark is detected when the macro instruction has been executed.

ERG (Erase Gap): Use this mnemonic if you want to erase all signals that may be recorded on a section of tape; i.e., a length of blank tape (approximately 3 1/2 inches) is created.

FSF (Forward Space File). Use this mnemonic if you want to skip the remaining part or all of a tape input file. When a CNTRL macro instruction with FSF as the second operand is executed, the IOCS causes the tape to be stopped immediately beyond the tapemark that follows the trailer label set

(if any). In the case of a file without labels or with non-standard labels, the tape is stopped immediately beyond the tapemark that follows the last block of data. The IOCS branches to your end-of-file routine when the tapemark following the last data record has been encountered.

FSR (Forward Space Record). Use this mnemonic if you want to skip one block. When a CNTRL macro instruction with FSR as the second operand is executed, the IOCS causes the tape to be stopped at the beginning of the block following the one just skipped. This is the proper position for reading during the next read-forward operation. The IOCS immediately branches to your end-of-file routine if the operand FSR refers to an input file and a tapemark is detected when the CNTRL macro instruction has been executed.

REW (Rewind Tape). Use this mnemonic if you want to rewind a tape. When a CNTRL macro instruction with REW as the second operand is executed, the IOCS causes the tape to be stopped at the first record on the tape. This is the proper position for reading during a read-forward operation. The record may be (1) a volume label if standard labels have been specified, (2) a tapemark or a data record if no labels have been specified, or (3) a non-standard label if non-standard labels have been specified.

RUN (Rewind and Unload Tape): Use this mnemonic if you want to rewind and unload a tape.

WTM (Write Tapemark): Use this mnemonic if you want a tapemark to be written.

Special Considerations for BSR and FSR

When you issue a CNTRL macro instruction for a tape input file with BSR or FSR as the second operand, you must consider the relative position of the tape to the record being processed.

Unblocked Records and No Work Area. When a CNTRL macro instruction with BSR as the second operand refers to

(1) a file that is read forward, the tape is positioned so that the record being processed is in the proper position to be re-read during the next read-forward operation;

(2) a file that is read backward, the tape is positioned so that the second record stored on the tape behind the one being processed is in the proper position to be read during the next read-backward operation.

When a CNTRL macro instruction with FSR as the second operand refers to

(1) a file that is read forward, the tape is positioned so that the second record following the one being processed is in the proper position to be read during the next read-forward operation;

(2) a file that is read backward, the tape is positioned so that the record being processed is in the proper position to be re-read during the next read-backward operation.

Unblocked Records and a Work Area. When a CNTRL macro instruction with BSR or FSR as the second operand is executed, the tape is in the same position as if no work area were used.

Blocked Records and No Work Area. When a CNTRL macro instruction with BSR as the second operand refers to

(1) a file that is read forward, the tape is positioned so that the block in the input area is in the proper position to be re-read during the next read-forward operation;

(2) a file that is read backward, the tape is positioned so that the second block stored on the tape behind the one currently in the input area is in the proper position to be read during the next read-backward operation.

When a CNTRL macro instruction with FSR as the second operand refers to

(1) a file that is read forward, the tape is positioned so that the second block following the one currently contained in the input area is in the proper position to be read during the next read-forward operation;

(2) a file that is read backward, the tape is positioned so that the block in the input area is in the proper position to be re-read during the next read-backward operation.

Blocked Records and a Work Area. A CNTRL macro instruction with BSR or FSR as the second operand causes the tape to be positioned as if no work area were used, except when the last record of a block is being processed. In this case, the tape is positioned as described below.

1. When a CNTRL macro instruction with BSR as the second operand refers to

   (a) a file that is read forward, the tape is positioned so that the block following the block whose last record is currently being processed is read during the next read-forward operation;

(b) a file that is read backward, the tape is positioned so that the third block stored on the tape behind the block whose last record is currently being processed is read during the next read-backward operation.

2. When a CNTRL macro instruction with FSR as the second operand refers to

   (a) a file that is read forward, the tape is positioned so that the third block following the block whose last record is currently being processed is read during the next read-forward operation;

   (b) a file that is read backward, the tape is positioned so that the block stored on the tape behind the block whose last record is currently being processed is read during the next read-backward operation.

For tape output files with blocked records, you should issue a TRUNC macro instruction if a partially filled block of records is to be written on magnetic tape before a CNTRL macro instruction for the file is issued.

Effect of CNTRL on Block Count

When a CNTRL macro instruction with BSF, BSR, FSF, or FSR as the second operand is issued, the block count written or checked when using standard labels may be wrong. The control routine does not update the block count. If a tape input file with standard labels is specified and the block count is incorrect at the end of the volume or file, a programmed halt occurs.

TRUNC MACRO INSTRUCTION

Name	Operation	Operand
[name]	TRUNC	filename

The name of the file to which this macro instruction (TRUNCate) refers is the only operand required.

Use this macro instruction when blocked output records are to be written onto magnetic tape. It may be issued for either fixed- or variable-length blocked records. When you issue a TRUNC macro instruction, the output area being used to build output records is considered full. The block of records in the output area is then written onto magnetic tape (as a short block) and the output area is made available for building the next block of records.

The last record included in the short block is the record that was built before the last PUT instruction preceding TRUNC was executed. Therefore, if you build records in a work area and you determine in the problem program that a record belongs to a new block, issue a TRUNC macro instruction followed by a PUT macro instruction for this particular record. However, if you build the records in the output area, determine whether or not a record belongs to a new block and, if so, issue a TRUNC macro instruction before you build the record.

Whenever variable-length blocked records are built directly in the output area, you must use the TRUNC macro instruction to write a completed block of records. When you issue the PUT macro instruction after each variable-length record is built, the output routines supply the number of bytes remaining in the output area. From this, you can determine if the next variable-length record will fit in the block. If not, issue the TRUNC macro instruction to write out the block and make the entire output area available for building the record. The amount of remaining space is supplied in the register specified in the VARBLD entry (see VARBLD=(register) in the description of the DTFMT Statement).

A TRUNC macro instruction causes no operation if the preceding PUT

- is issued after the last record of a block has been built in the output area;
- causes the last record of a block to be moved from a work area to the output area for inclusion in the block.

In either case the entire block is written onto magnetic tape by the PUT macro instruction.

RELSE MACRO INSTRUCTION

Name	Operation	Operand
[name]	RELSE	filename

The name of the file to which this macro instruction (RELeaSE) refers is the only operand required.

You may use this macro instruction when reading blocked input records from magnetic tape. RELSE allows you to skip the remaining records in a block and continue processing with the first record of the next block, which is read when the next GET macro instruction is executed.

The RELSE macro instruction can be used, for instance, in a job in which only the first three records of each block on magnetic tape are to be processed. In this case you must issue three successive GET macro instructions followed by a RELSE macro instruction.

Another example of using the RELSE macro instruction is a job in which records on magnetic tape are categorized, and each category (perhaps a major grouping) begins with the first record of a block. Categories can be located readily by checking only the first record of each block.

The RELSE macro instruction discontinues deblocking of the current block of records, which may be either of fixed or variable length. RELSE causes the next GET macro instruction to transfer a new block to the input area and make the first record available for processing. This GET macro instruction initializes the I/O register or moves the first record to a work area.

LBRET MACRO INSTRUCTION

Name	Operation	Operand
	LBRET	1
	LBRET	2

This macro instruction (LaBel RETurn) applies only to magnetic tape files containing standard user-labels (UHL and/or UTL) that are to be checked or written. You must issue a LBRET macro instruction at the end of your label routine (specified by the LABADDR entry in the DTFMT statement for the file) to return to the IOCS after label processing.

Specify the operand 1 to return to the IOCS if:

1. an input file with user labels is being processed and control is to be returned to the IOCS to eliminate the checking of one or more user labels. The IOCS then skips the remaining labels in the set and processing continues.

2. an output file with user labels is being processed and control is to be returned to the IOCS when the last user label has been built. The IOCS writes the last label (from the label output area) and processing continues. A LBRET macro instruction with a 1 in the operand field is always required to terminate an output label set.

Specify the operand 2 to cause further label processing for:

1. an input file with standard user labels to return to the IOCS after each label has been checked. Then the IOCS makes the next label, if any, available for checking in the label input area. When the IOCS reads the tapemark following the label set, it terminates label processing.

2. output files with user labels to return to the IOCS after each label, except the last, has been built. The IOCS causes the writing of the label contained in the label output area. The IOCS then returns to the LABADDR label routine to allow you to build the next label. The label set is terminated by issuing a LBRET macro instruction with the operand 1. For details on writing standard labels under control of the IOCS, refer to the description of the LABADDR entry under DTFMT Statement.

The IOCS requires the values it places into registers 14 and 15 before transferring control to the LABADDR routine. Hence, if you want to use one or both of these registers in the LABADDR routine, save their contents before you begin using them. In addition, you must restore these contents before issuing the LBRET macro instruction.

FEOV MACRO INSTRUCTION

This macro instruction (Force End Of Volume) is used for tape input or output files to force an end-of-volume condition at a point other than the normal tapemark (input) or the reflective marker (output). This indicates that the processing of records on one volume is considered finished, but that more records for the same logical file are to be read from or written into the following volume.

Name	Operation	Operand
[name]	FEOV	filename

The operand contains the name of the file to which this macro instruction pertains; the name must be the same as the one specified in the header entry of the DTFMT statement for the file.

When you issue this macro instruction for an input tape, the IOCS

1. causes the execution of the operation specified in the REWIND entry,

2. switches to the next reel on another drive in accordance with the ALTTAPE detail entry in the DTFMT statement for the file; and

3. processes the header label (or labels) as required.

When you issue the FEOV macro instruction for an output tape, the IOCS causes the last block of records to be written, if necessary, and writes a tapemark. Then the IOCS

1. causes the writing of the standard trailer label including the accumulated block count, and branches to the LABADDR routine if this is specified;

2. switches to the next reel in accordance with the ALTTAPE detail entry in the DTFMT statement for the file; and

3. processes the header label (or labels) as required.

The following example illustrates the use of the FEOV macro instruction.

If FILABL=NSTD or FILABL=NO has been specified for a multi-volume input file, the IOCS cannot detect an end-of-volume condition. When a tapemark is detected, the IOCS transfers control to the EOFADDR routine, in which you must determine whether or not an end-of-volume condition exists. If so, issue a FEOV macro instruction to have the IOCS perform the end-of-volume functions in accordance with the detail entries.

The IOCS provides routines for processing records of sequential disk files. To utilize the IOCS functions, you must describe your sequential disk file in the problem program using the DTFSD file definition statement, and issue the appropriate imperative macro instructions to perform the desired I/O operations.

All records in sequential disk files must be blocked or unblocked format-F records.

DTFSD Statement

This file definition statement applies to sequential disk files only. The name field of the header entry must contain the name of the file and the operation field must contain DTFSD. For ease of reference, the detail entries to be made in the operand field are described below in alphabetical order.

BLKSIZE=n

The operand n specifies the length of the blocks (in number of bytes) that the IOCS reads or writes. The maximum BLKSIZE you are allowed to specify is 27000 for blocked records and 4096 for unblocked records.

The specified length must be a multiple of the record length. Since the IBM 2311 Disk Storage Drive uses a fixed sector length of 270 bytes, the IOCS adjusts the specified block length to the next higher integer multiple of 270. You must take this into consideration when defining the I/O area in the problem program. In the format-1 file label, however, the originally specified block length, not the adjusted block length, is entered.

Some block sizes could cause a cylinder overflow, i.e., not all sectors belonging to one block would be written on the same cylinder. The IOCS automatically avoids this situation. A block that would cause a cylinder overflow is written as the first block of the next cylinder. One or more sectors of the full cylinder may remain unused. If the writing of a block would cause an extent overflow, the same technique is used, i.e., the block is written as the first block of the next extent (if available).

The block size of input and corresponding output files should not be different.

COMROUT=YES

This optional entry indicates that a common IOCS routine is to be generated for several files of the same type processed by the same problem program. I/O routines that fall into this category are (1) input routines, (2) update routines, and (3) output routines.

By generating a common routine for several files it is possible to reduce the amount of main storage required for the simultaneous processing of three or more files of the same type e.g., three output files. If only two files are processed simultaneously, the amount of main storage required is not reduced. In some cases, it is even increased.

CONTROL=YES

This entry is required if a CNTRL macro instruction is issued for the file. The CNTRL macro instruction can be used to initiate a seek operation. Note that this entry is ignored if you specify two I/O areas for the file.

DEVICE=DISK11F

This entry specifies that a fixed-sector IBM 2311 Disk Storage Drive (Model 11 or 12) is used as I/O device for the sequential-access file. If this entry is omitted or mispunched, the IOCS assumes the correct device and issues a warning to the operator.

DSKXTNT=n

The number n specifies the maximum number of extents in any one of the volumes for the file. If the entry is omitted, the IOCS assumes that there are three extents for the file. The IOCS uses this information to reserve the storage area in which the addresses of the extent boundaries are saved. The maximum number of extents permitted is 99 per file.

DTAREX=name

This entry applies to output files only. If the last extent of the last volume is filled with data, control is transferred to the routine specified in the optional entry DTAREX=name. If you have not specified a routine, the job is discontinued. The

routine specified in the DTAREX entry must contain the necessary CLOSE macro instruction(s). For a mainline program this includes the closing of all disk files.

EOFADDR=name

The EOFADDR entry is mandatory for all input files. This entry specifies the name of the routine in the problem program to which the IOCS branches when an end-of-file condition occurs. In that routine, you can perform any operation required for the end of the file. Usually a CLOSE macro instruction is issued.

If you issue a GET or PUT macro instruction in your end-of-file routine, a halt occurs. This halt permits no restart unless you process an update file. In the case of an update file, you can make a restart and continue processing. This allows you to update, i.e., extend, the file beyond the EOF record by issuing a GET macro instruction to read a pseudo record. Replace this pseudo record by the record to be added to the file. A subsequent PUT macro instruction writes the added record onto disk. After all records have been added to the file, you must simulate an end-of-file condition.

An end-of-file condition is detected when the end-of-file record containing /*b (where b = blank) in the first three bytes is read. An end-of-file condition in an input file can also be detected by an extent overflow, in case there was not sufficient space for the end-of-file record within the extent when the file was written (see the detail entry DTAREX).

Note: If, in the end-of-file routine, you want to discontinue/terminate the processing of a program that is a mainline program, you must first close all disk files.

ERRIO=name

This entry specifies the symbolic name of a two-byte area in which the IOCS places the address of the I/O area containing a block that caused an irrecoverable read or write error. Use this entry only if you also specify ERROPT=name and two I/O areas.

ERROPT=code

Use this entry if you do not want to discontinue a job in case a disk read or write error cannot be corrected by the error routine.

Code	Explanation
SKIP	The record block containing the error is to be skipped, i.e., an input block should not be made available for processing. This specification is not permitted for output files, nor for input files that are to be updated.
name	This is the name of a routine to which the IOCS should branch on a disk error. An 8-byte error block that contains information on the type of error is made available to the problem program. In the error routine, do not issue any GET or PUT macro instructions for records in the block that caused the read or write error, because a subsequent GET or PUT macro instruction referring to the file will skip that block and process the next block. To branch back to the next sequential instruction following the GET, PUT, or CNTRL macro instruction which recognized the error, use the return address in register 14. The IOCS provides the address of the 8-byte error block in a storage area to which you can refer by using a symbolic address consisting of the name of the file plus an A-suffix. If, for example, the name of the file involved is PAYROLL, the address of the error block is made available to the problem program in the storage area PAYROLLA. If you did not specify ERRIO=name, the IOCS places the address of the error block into the storage area PAYROLLA-4. Note: If, in the error routine, you want to discontinue the processing of a program that is a mainline program, you must first close all disk files.

The 8-byte error block contains information on the type of error, the disk address where the error occurred, etc. The structure of the error block is as follows:

```
Byte 0  -- first byte of sense information
           (status byte); indicates the
           following if the corresponding
           bit is on:
           Bit 0: not used -- set to zero
               1: intervention required
               2: end of cylinder
               3: equipment check
               4: data check
               5: seek check
               6: no record found
               7: track-condition check
```

Byte 1 -- residual sector count

This byte indicates the number of sectors that could not be processed by a multi-sector I/O operation

Byte 2 -- unit status of Channel Status Word (CSW); indicates the following if the corresponding bit is on:

Bit 0: not-equal scan
1: status modifier
2: not used -- set to zero
3: busy
4: channel end
5: device end
6: unit check
7: not used -- set to zero

Byte 3 -- used by the IOCS

Byte 4 -- displacement of Channel Command Word (CCW)

This number is added to the CCW address (bytes 6-7) to compute the actual address of the CCW for this operation.

Byte 5 -- Logical Unit Block (LUB) displacement

Byte 6-7 - CCW address

Additional information about the operation in error can be derived from the CCW itself.

The sum of the CCW address (bytes 6-7) and the CCW displacement (byte 4) is the actual address of the CCW.

CCW	Byte 0:	command code
	Byte 1:	bit 0 = CCW chaining indicator bits 1-7 = sector count
	Bytes 2-3:	data address (I/O area)
	Bytes 4-5:	count-area address
Count Area	Byte 0:	not used
	Bytes 1-2:	cylinder address
	Bytes 3-4:	head address
	Byte 5:	sector (record) number

For a more detailed description of this error information refer to the SRL publication IBM System/360 Model 20, Functional Characteristics, Form A26-5847. Save the error information (contained in the count-area), the disk address of the block that caused the error, and the contents of the I/O area, because this information is essential when an alternate track must be assigned to replace a defective track. In some cases, a recovery and restart procedure may not be possible. For this reason, you should always maintain a duplicate of every file used in the installation.

If you omit the entry ERROPT=code, the job will be discontinued automatically when a permanent disk error is detected.

IOAREA1=name

This entry specifies the name of the I/O area to be used by the file. The name must be the symbol you specified in the DS or DC statement defining the area in the source program. For additional information regarding the use of the I/O area, see I/O Areas under Overlapping and Storage Areas.

The length of the area must be equal to 270 bytes or an integer multiple thereof. If the calculated length is not equal to 270 or an integer multiple thereof, use the next larger value allowed.

IOAREA2=name

Two input or output areas can be specified for a file, to permit overlapping of data transfer with processing operations in a Model 20, Submodel 5. The IOAREA2 entry must specify a symbolic name identical with the one used in the DS or DC statement that sets up the second I/O area. Both I/O areas must be of the same length.

The length of the area must be 270 bytes or an integer multiple thereof.

If two I/O areas are used for disk files, you must also specify a work area or include the entry IOREG=(register).

IOREG=(register)

This entry specifies a register which contains either the address of an input record that is available for processing or the address of an output area in which you can build your next record. As operand, you may specify any of the registers 8 through 13 (enclosed in parentheses) or a symbolic name (in parentheses) that has been equated to one of the registers 8 through 13. You can use the same register for several files.

You must include an IOREG entry when blocked input or output records are processed in the I/O area.

For a file with unblocked records, you can omit the IOREG entry, unless you use two I/O areas. In that case you must specify either WORKA=YES or IOREG. However, do not issue IOREG together with WORKA=YES.

RECFORM=code

This entry specifies the record format of the file. The IOCS can handle both possible formats of records in the same program. However, all records in a given file must be of the same format. If this entry is omitted, the IOCS assumes unblocked format-F records.

Code	Record Format
FIXUNB	Unblocked format-F records
FIXBLK	Blocked format-F records

RECSIZE=n

This entry specifies the number of bytes in an individual record. The maximum record length is 4096 bytes. If records are unblocked, this entry may be omitted. In this case, the record length is assumed to equal the block length specified in the detail entry BLKSIZE.

TYPEFLE=code

This entry is used to specify the type of file.

Code	Type of File
INPUT	An input file
OUTPUT	An output file

UPDATE=YES

This entry specifies that an input file is to be updated. When this entry is included, a PUT macro instruction will replace, on disk, the record retrieved by the preceding GET macro instruction.

VERIFY=NO

You can use this entry only for output files. When it is specified, the output records are not checked. When it is omitted, all records written onto disk are verified; if a write error is detected, the IOCS attempts to recover the error as described in the section Device Error Recovery.

Although more processing time is needed when the records are verified, you are strongly recommended to omit the entry VERIFY=NO unless you require maximum throughput.

WORKA=YES

This entry specifies that records are to be processed in a work area rather than in an I/O area. The symbolic name used in the DS or DC statement that reserves the work area must be used as the second operand of each GET or PUT macro instruction for the file.

Whenever two I/O areas are used for unblocked records, either the WORKA entry or the IOREG entry must be used. For additional information regarding the use and length of a work area, see the section Overlapping and Storage Areas.

Imperative Macro Instructions

The imperative macro instructions for sequential disk files are described in the following order: PUT, GET, CNTRL. For a description of OPEN and CLOSE refer to the section Instructions for Opening and Closing Files.

PUT MACRO INSTRUCTION

This instruction writes logical records that have been built directly in an output area or in a specified work area.

Name	Operation	Operands
[name]	PUT	filename[,workname]

You can write the PUT macro instruction with one or two operands, depending on the area in which the records are built.

1. One Operand Specified. Use this format if records are to be processed directly in an I/O area. The operand specifies the name of the file in which you want to place the record. The file name must be the same as the one used in the header entry of the DTFSD file definition statement.

When blocked records are processed directly in an output area, the use of an I/O register is required (refer to the description of the IOREG=(register) detail entry under DTFSD Statement). For a file with blocked records, a PUT macro instruction causes the IOCS to (1) include the affected record in a record block and (2) change the address in the I/O register to identify the next available position of the output area in which you can build the next record. After an OPEN macro instruction the I/O register contains the first available position of the output area.

For a file with unblocked records, a PUT macro instruction causes the IOCS to write the affected record onto disk. If two I/O areas are specified an I/O register or the WORKA=YES entry is required.

Note: The IOCS does not clear the output area after a PUT macro instruction. To avoid having interspersed characters from preceding records in your output record, use every position of the output area.

2. Two Operands Specified. Use this format if records are processed in a work area. The first operand specifies the name of the file. The second operand specifies the name of the work area from which records are moved to the output area. When an output area is full, the IOCS takes the data in that output area and writes them onto disk. You can use more than one work area if you issue separate PUT macro instructions for writing records from each work area.

Do not specify IOREG=(register) if you use a work area (refer to the description of the WORKA=YES entry under DTFSD Statement).

Note that the IOCS does not clear the work area after a PUT macro instruction. You should ensure that no characters from preceding records remain in the area either by clearing the work area after each PUT macro instruction or by using every position of the work area.

GET MACRO INSTRUCTION

This macro instruction makes the next sequential logical record from an input file available for processing in either an input area or a specified work area.

When an end-of-file condition or an extent end condition is detected, the IOCS transfers control to the end-of-file routine specified by the EOFADDR entry in the DTFSD statement.

Name	Operation	Operands
[name]	GET	filename[,workname]

You can write the GET macro instruction with either one or two operands, depending on the area in which the records are processed.

1. One Operand Specified. Use this format if records are to be processed directly in an input area. The operand specifies the name of the file from which the record is to be read. This name must be the same as the one you used in the header entry of the DTFSD statement for the file.

When blocked records are processed in an input area, you must specify an I/O register that the IOCS needs to identify the next record to be processed. (Refer to the description of the IOREG=(register) entry in the section DTFSD Statement). This I/O register always contains the address of the currently available record. The GET routine places the proper address into the I/O register.

No I/O register is needed for a file with unblocked records, unless two I/O areas are specified.

2. Two Operands Specified. Use this format if records are to be to be processed in a work area. The first operand specifies the name of the file. The second operand specifies the symbolic name of the work area to be used (refer to the description of the WORKA=YES entry under DTFSD Statement). You can specify more than one work area for a file if you issue separate GET macro instructions to move the records to the different work areas. It might be advantageous to plan two work areas, and to specify each area in alternate GET macro instructions. This allows you, for instance, to determine a possible change in the control level by comparing each record with the preceding one.

Do not specify IOREG=(register) together with WORKA=YES.

CNTRL MACRO INSTRUCTION

The CNTRL macro instruction is used to initiate a seek operation. You must include a CONTROL=YES entry in the DTFSD file definition statement if you issue a CNTRL macro instruction for the file.

Name	Operation	Operands
[name]	CNTRL	filename,SEEK

The control macro instruction contains CNTRL in the operation field, and the name of the disk file for which the operation is specified as the first operand. The second operand is SEEK.

The CNTRL macro instruction initiates the access movement for the next GET or PUT macro instruction for a file. While the access arm is moving, you may process data and/or request I/O operations on other devices. The CNTRL macro instruction causes the IOCS to seek the track that contains (or should contain) the next block of the file. However, the CNTRL macro instruction does not prevent the execution of the seek operation initiated by the GET/PUT routine.

Note: If two I/O areas are used, a CNTRL macro instruction is treated as a no-operation instruction.

The direct-access method of file organization allows you to process records in random and sequential order. However, when processing a direct-access file sequentially you must observe certain restrictions (see Note under the detail entry BLKSIZE=n of the DTFDA statement).

The IOCS locates the records to be processed by referring to their physical disk addresses, which you must supply in the problem program. Direct-access files apply to disk only. You must describe your direct-access file using the DTFDA file definition statement, and issue the proper imperative macro instructions to perform the desired I/O operations.

Records in direct-access files must be unblocked format-F records, one or more sectors in length. If you want to apply the direct-access method to a file containing blocked format-F records (more than one logical record in one physical record) you must provide for the blocking and deblocking of records in the problem program.

Supply the disk storage location from which a record is to be read, or into which it is to be written, by specifying track and record references as described under Cylinder, Track and Record References.

DTFDA Statement

This file definition statement applies only to direct-access files. The name field of the header entry must contain the name of the file and the operation field must contain DTFDA. For ease of reference, the detail entries to be entered in the operand field are described below in alphabetical order.

ADRTEST=NO

When ADRTEST=NO is specified, the IOCS does not check whether the address specified in the SEEKADR entry is valid or whether the address is within the limits defined in the XTENT statements. As a result, bits 0 and 1 of the first error byte are not set. (Refer to ERRBYTE=name).

If this entry is omitted, the IOCS automatically performs these checks and sets the appropriate bits in the first error byte.

Even though the length of the program is increased when the addresses are checked, you are recommended to omit the entry ADRTEST=NO.

BLKSIZE=n

This entry is mandatory and specifies the length of the blocks (in bytes) the IOCS is to read or write. The maximum block length you may specify is 16200. Since unblocked format-F is the only type of record permitted for direct-access files, you must provide for any blocking or deblocking of records in your problem program.

In defining the I/O area in the problem program, keep in mind that the IBM 2311 uses a fixed sector length of 270 bytes.

Some block sizes cause a cylinder overflow, i.e., not all sectors belonging to one block can be written on the same cylinder. The IOCS automatically continues this block on the next cylinder, provided this does not lead to an extent overflow.

Note: Since the actions the IOCS performs for direct-access files and for sequential files differ in case of a cylinder overflow, you must ensure that no cylinder overflow occurs if you process a direct-access file sequentially. You can avoid a cylinder overflow if the number of sectors contained in a block is either 1, 2, 5 or 10. If the file starts at the beginning of a cylinder, the number of sectors can also be 4, 20, 25, or 50.

If the reading or writing of a block causes an extent overflow, the READ or WRITE macro instruction is not executed and bit 0 of the first error byte (see ERRBYTE entry) is set to indicate this condition.

CONTROL=YES

This entry is required if a CNTRL macro instruction is to be issued for the file. The CNTRL macro instruction can be used to initiate a seek operation.

DEVICE=DISK11F

This entry specifies that a fixed-sector IBM 2311 (Model 11 or 12) is used for the direct-access file. If this entry is missing or mispunched, the IOCS assumes the correct device and issues a warning to the operator.

DSKXTNT=n

The value n specifies the number of extents. If the entry is omitted, the IOCS assumes n=3. The IOCS uses this information to reserve the main-storage area in which the addresses of the extent boundaries are saved. The maximum number of extents permitted is 99 per file.

ERRBYTE=name

This entry is mandatory. It specifies the symbolic address of a two-byte field in which the IOCS will indicate exceptional conditions. The indications will be available after a WAITF macro instruction has been executed. You may test for the following conditions:

Bit Positions	Exceptional Condition If Bit Is 1
Byte 1 : Bit 0	invalid address
Bit 1	address outside extent
Bit 2-7	not used
Byte 2*: Bit 0	not used--set to zero
Bit 1	intervention required
Bit 2	end of cylinder
Bit 3	equipment check
Bit 4	data check
Bit 5	seek check
Bit 6	no record found
Bit 7	track condition check

*First byte of sense field

IOAREA1=name

This entry is mandatory and specifies the name of the I/O area to be used for the file. The name must be the symbol used in defining the area in the source program. You can specify only one I/O area for each direct-access file.

The size of the area must be equal to 270 bytes or an integer multiple of 270. If the calculated length is not equal to 270 or an integer multiple thereof, the next larger value allowed must be used for the I/O area.

READID=YES

This entry is required if records are to be retrieved from the direct-access file, i.e., if a READ macro instruction in the problem program refers to the file.

SEEKADR=name

This mandatory entry specifies the name of the 8-byte area that contains the disk address of the record to be read or written. The area must be defined in the problem program. Its format and contents are described under Cylinder, Track and Record References.

The problem program must place the address of the desired disk record in the specified location before issuing a READ or WRITE macro instruction. If the disk address is invalid, bit 0 of the first error byte is set to 1. If the address is outside the extents specified in the XTENT statements, bit 1 of the error byte is set to 1.

TYPEFLE=code

This entry specifies the type of file and how the disk labels are to be processed. The entry must not be omitted.

Code	Explanation
INPUT	The labels of an input file are to be read and checked.
OUTPUT	The labels of an output file are to be written.

VERIFY=NO

When this entry is specified, the record just written is not verified. If you omit this entry, all records written onto disk are verified. If a write error is detected, the IOCS attempts to recover the error as described in the section Device Error Recovery. If a permanent write error occurs, the IOCS inserts into the ERRBYTE the first byte of the sense field (For a detailed description of the sense field, refer to the SRL publication IBM System/360 Model 20 Functional Characteristics, Form A26-5847).

Although processing time is increased when the records are verified, you are strongly recommended to omit the entry VERIFY=NO unless you require maximum throughput.

WRITEID=YES

This entry is required if records are to be stored in the direct-access file, i.e., if a WRITE macro instruction in the problem program refers to the file. The entry is also used for input files if they are to be updated by the problem program.

Imperative Macro Instructions

The imperative macro instructions for direct-access files are described in the following order: WRITE, READ, WAITF, CNTRL, CNVRT. For a description of OPEN and CLOSE refer to the section Instructions for Opening and Closing Files.

WRITE MACRO INSTRUCTION

This macro instruction causes a record that has been built in an output area in main storage to be written onto disk.

Name	Operation	Operands
[name]	WRITE	filename,ID

The first operand specifies the name of the disk file into which the record is to be placed. The file name must be the same as the one specified in the header entry of the DTFDA statement for the file. The second operand must be ID as shown.

Before this macro instruction is executed, the problem program must provide the track and record reference of the disk location in which the record is to be stored (refer to Cylinder, Track and Record References below). When the WRITE macro instruction is executed, the IOCS searches the specified track for the desired location on the track. When the correct location has been found, the data record is written from the output area in main storage. A data record may require one or more sectors of disk storage; the length of the data record is specified in the BLKSIZE=n entry in the DTFDA statement for the file. If an I/O error occurs, the appropriate bits are set as described for the entry ERRBYTE=name of the DTFDA statement.

If you use the WRITE macro instruction in a program, you must include the WRITEID=YES entry in the DTFDA statement for the file.

READ MACRO INSTRUCTION

This macro instruction causes a record on disk to be read into main storage.

Name	Operation	Operands
[name]	READ	filename,ID

The first operand specifies the name of the disk file from which the record is to be retrieved. The file name must be the same as the one specified in the header entry of the DTFDA statement for the file. The second operand must be ID as shown.

Before this macro instruction is executed, the problem program must provide the track and record reference of the desired record (see Cylinder, Track and Record References below). When the READ macro instruction is executed, the IOCS searches the specified track for the particular record. When the correct record has been located, it is read into the input area in main storage. If an I/O error occurs, the appropriate bits are set as described for the entry ERRBYTE=name of the DTFDA statement.

If you use the READ macro instruction in a program, you must include the READID=YES entry in the DTFDA statement for the file.

WAITF MACRO INSTRUCTION

The WAITF macro instruction ensures that the reading or writing of a record into or from the I/O area for the file has been completed before further processing in the same I/O area is performed. This macro instruction enables you to overlap seek operations with processing in a Submodel 2 or 4, or a Submodel 5 not utilizing the read/compute, write/compute (RWC) feature. The RWC feature of a Submodel 5, on the other hand, provides full overlapping of I/O operations.

Name	Operation	Operand
[name]	WAITF	filename

The file name in the operand field must be the same as the one specified in the header entry of the DTFDA statement for the file.

After an I/O operation has been started, the IOCS immediately returns control to the problem program. Therefore, when the program is ready to process the input record or build the succeeding output record for the same file, make a test to ensure that the previous I/O operation has been completed. To do this, issue a WAITF macro instruction in the problem program. If the I/O operation has not been completed, the program enters a waiting loop and remains there until the entire record has been read or written. In the problem program, a WAITF macro instruction should be issued after each READ or WRITE macro instruction.

The following example shows one possible placement of the WAITF macro instruction:

```
-------------
READ (WRITE) AAA,ID
-------------     processing that does
-------------     not refer to the
-------------     I/O area of file AAA
WAITF AAA
-------------     processing that may
-------------     refer to the
-------------     I/O area of file AAA
WRITE (READ) AAA,ID
-------------
```

The WAITF macro instruction also makes the error-status information available in the second byte of the two-byte field specified by the entry ERRBYTE=name of the DTFDA statement. If the entry ADRTEST=NO is not specified, you should check the error information in the ERRBYTE entry after every WAITF macro instruction.

CNTRL MACRO INSTRUCTION

This macro instruction is used to position the access mechanism for the reading or writing of a disk record. The CNTRL macro instruction can be used to improve the performance of the program.

Name	Operation	Operands
[name]	CNTRL	filename,SEEK

The first operand specifies the name of the disk file for which the seek operation is desired. The file name must be the same as the one specified in the header entry of the DTFDA definition statement for the file. The second operand must be SEEK.

The CNTRL macro instruction causes the access mechanism to be moved to the disk address contained in the field specified in the entry SEEKADR=name of the DTFDA statement. After the CNTRL macro instruction has started the movement of the access mechanism towards the specified disk location, control is returned to the problem program, which may process data and/or request I/O operations for files on other I/O devices.

Each CNTRL macro instruction must be preceded by a WAITF macro instruction.

CNVRT MACRO INSTRUCTION

This macro instruction is used to convert a packed decimal address of three bytes into the format of the seek field as used in direct-access processing.

Name	Operation	Operands
[name]	CNVRT	seekfield,packedfield

The first operand specifies the name of the 8-byte field to which the converted address is to be moved. This field has the same format as, and can be identical with, the seek field MBBCCHHR specified for the entry SEEKADR=name of the DTFDA statement. Note that only the four low-order bytes representing the disk address CHHR are altered by this macro instruction. You must define the eight-byte seek field in the problem program by first choosing the appropriate entry for the first byte in a DC statement and then specifying a DC statement (e.g., DC XL7'00') to fill the remaining seven bytes with zeros.

The entries for the first byte are as follows:

X'00' to refer to the first volume
X'01' to refer to the second volume
X'02' to refer to the third volume
X'03' to refer to the fourth volume.

These entries must agree with the definitions in the VOL, DLAB, and XTENT control statements. The entries X'02' and X'03' apply to the Model 20, Submodel 5 only.

The second operand specifies the name of a 3-byte field in packed decimal format (in the problem program) that contains the disk address to be converted. The format of this field must be as follows:

CC	CH	R+

Byte 1 2 3

The meaning of the symbols CCCHR is explained under Cylinder, Track and Record References.

The packed decimal address is not checked for validity. Therefore, it is your responsibility to supply the correct cylinder, head, and record values. The highest address that may be specified is 20299.

Since the CNVRT macro instruction uses register 14, the contents of this register are overwritten.

Cylinder, Track and Record References

To refer to a specific location in disk storage, you must provide the cylinder, track and record references of the location. You can derive these cylinder, track and record references, which constitute the

sector address, by using a randomizing formula. Selecting the best formula for a given file may require some consideration, since it is desirable to minimize (1) the number of records for which the same disk address is derived, and (2) the amount of wasted storage space, i.e., the number of unused record locations between the records of the file.

The cylinder, track and record references consist of eight bytes of information in the following form:

MBBCCHHR

Note that the sector address is for compatibility purposes in the form of the standard IBM System/360 random-access address. Make the sector address available to the IOCS in an 8-byte field before issuing a READ or WRITE macro instruction. The use of the CNVRT macro instruction is strongly recommended for this purpose. The symbolic address of the 8-byte field must be specified in the SEEKADR entry of the DTFDA statement for the file.

The contents of these eight bytes are as shown below. Provide all numbers in binary notation.

Symbol	Byte(s)	Name and Contents
M	0	Pack Number For Model 20 Submodels 2 and 4, this is either 0 or 1; for Submodel 5 it is any number from 0 to 3. The entry indicates the specific disk pack. M must be 0 if only one pack (volume) is used for a file, regardless of the drive on which it is mounted. If more than one pack (volume) is used for a file, M=0 addresses the first volume, M=1 the second, and so on, as defined by the VOL, DLAB and XTENT control statements. The symbolic device addresses for disk drives can be assigned physical device addresses as desired.
BB	1-2	Reserved Zeros
CC	3-4	Cylinder Number For Model 12 any value from 000 to 102, for Model 11 any value from 000 to 202, indicating the number of the cylinder in which the record is located. Note that the cylinders 1-3 are used for alternate tracks on both models, and that byte 3 will always contain zeros.
HH	5-6	Head Number A number from 0 to 9 indicating the read/write head to be used for reading or writing the record. Each head reads from, or writes on, one disk surface. Head 0 is assigned to disk surface 0. Likewise, heads 1 through 9 are assigned to disk surfaces 1 through 9, respectively. Note that byte 5 will always contain zeros.
R	7	Record Reference A number from 0 to 9 indicating a specific record (sector) on a track.

Identifier (ID): Bytes 3 through 7 (CCHHR) of the cylinder, track and record reference are referred to as the ID or identifier. Each disk record (sector) is preceded by a count area which contains the ID and other data. When a READ or WRITE macro instruction is executed, the computer compares the ID in the count area with the corresponding part of the cylinder, track and record reference. An equal comparison indicates that the desired record has been found.

When a READ (WRITE) macro instruction for the file is executed, the IOCS uses the cylinder, track and record reference to (1) select the specific track on the appropriate disk pack and (2) locate the specified record location (sector).

The organization of files according to the Indexed-Sequential File Management System (ISFMS) permits disk records to be processed in random order or in sequential order by control information. For random processing, supply the control information (key) of the desired record to the IOCS and issue a READ or WRITE macro instruction to transfer the specified record. For sequential processing by control information (key), specify the first record to be processed and then issue GET or PUT macro instructions until all desired sequential records have been processed. The successive records are made available in sequential order by their keys. Variations in macro instructions permit:

- a logical file of records to be loaded onto disk (created);
- individual records to be read from, added to, or updated in the file.

The logical records must be of fixed length, and the length must be specified in the RECSIZE entry of the DTFIS statement. Logical records may be either blocked (two or more logical records in one block) or unblocked (one logical record per block).

Whenever you use an indexed-sequential file, you must describe the file and the main-storage areas allotted to the file in the DTFIS file definition statement.

DTFIS Statement

This file definition statement applies only to indexed-sequential files. The name field of the header entry must contain the name of the file and the operation field must contain DTFIS.

For ease of reference, the detail entries to be entered in the operand field are described below in alphabetical order.

ADAREX=name

This entry is mandatory if IOROUT=ADD or IOROUT=ADDRTR is specified.

Each new record to be inserted in an organized file is entered in an overflow area. If the specified overflow area is full and more records are yet to be added, the IOCS branches to the symbolic address specified in the ADAREX entry. In this routine you should terminate the job since the Model 20 IOCS does not permit the extents of a previously organized file to be changed, and the file must therefore be wholly reorganized. In addition, the ADAREX routine must contain the CLOSE macro instructions for all files involved. In a mainline program, all disk files must be closed in the ADAREX routine. Note that an ADAREX entry must not be used if IOROUT=LOAD has been specified.

ALTREX=name

This entry is mandatory if IOROUT=LOAD is specified. It is optional if IOROUT=ADD or IOROUT=ADDRTR is specified.

If a file is to be extended (IOROUT=LOAD) after records have been added to the last track of a file during a previous add operation, the IOCS branches to your ALTREX routine. You must reorganize the file before it can be extended. The branch condition is detected during the execution of the SETFL macro instruction and before the LOAD operation is initiated. Therefore, you must not issue an ENDFL macro instruction in your ALTREX routine. This instruction would cause the last prime data record to be overwritten by an EOF record and thus destroy your file. However, you must issue a CLOSE macro instruction for your file.

You can avoid that records are added to the last track of a file if you specify the ALTREX detail entry for ADD and ADDRTR files. If the program tries to add a record to the last track of a file, the IOCS branches to your ALTREX routine. In this routine you can determine whether the record can be added to the file or whether it is to be inserted at a later time. If the record can be added, i.e., the key of the record to be inserted is higher than the key of the last record in the file, perform a LOAD/Extension run to insert the record. If the key is not higher, you may continue processing and add the record

- after another track has become the last track of the file during a succeeding LOAD/Extension run, or
- by another program in which ALTREX is not specified. However, this makes further extensions of the file impossible because the EOF record is transferred to the overflow area.

CYLOFL=n

Include this entry if cylinder overflow areas are to be reserved for a logical file. A cylinder overflow area is located on each cylinder within the prime data area of the data file. It contains records that overflow from tracks in that cylinder.

To reserve the areas for cylinder overflow, this entry is required when the particular file is to be loaded onto disk and records are to be added at a later time. The specification n (where n is any integer from 1 through 9) is the number of tracks to be reserved on each cylinder. Note that the actual size of the cylinder overflow area may be less than the specified number of tracks. If the block size of prime data blocks is such that track overflow blocks occur (i.e., not all sectors of a block are on the same track), the last prime data block of a cylinder uses one or more sectors of the cylinder overflow area. The maximum number of cylinder overflow sectors that may be used by the last prime data block is equal to the number of sectors per block minus 1.

If you specify an independent overflow area (by an XTENT statement) in addition to the CYLOFL entry, overflow records are written in the independent overflow area after a cylinder overflow area has been filled.

CYNDEX=name

This entry is mandatory if IOROUT=LOAD has been specified for the file. The entry CYNDEX=name specifies the symbolic address to which the IOCS branches if the cylinder-index area becomes full while a file is being loaded or extended. You must reorganize the file and specify a larger disk area for the cylinder index in an XTENT statement. The CYNDEX routine should also contain the necessary ENDFL and CLOSE macro instructions.

Note: If this routine discontinues the processing of a program that is a mainline program, the CYNDEX routine must contain CLOSE macro instructions for all disk files.

DERREX=name

This entry is mandatory for all indexed-sequential files. It specifies the symbolic address to which the IOCS branches if an irrecoverable disk error has occurred. If, in this routine, you want to discontinue the processing of a program that is a mainline program, you must first close all disk files.

Certain error information is made available in the problem program if the detail entry ERRINF=YES is specified. The first byte of this error information can be addressed by using a symbolic address consisting of the file name with the suffix A. For example, if the name of the file is PAYROLL, the address of the error information will be PAYROLLA.

The first two bytes of this information contain the address of an error block as already described for sequential disk files. (Refer to the detail entry ERROPT=code in the section Instructions for Processing Sequential Disk Files).

The next two bytes (beginning at PAYROLLA+2) contain logical error information as described under the detail entry ERRINF=YES.

Register 14 contains the return address, i.e., the address of the next sequential instruction following the macro instruction that detected the error. Since the IOCS macro instructions use register 14, you must save the contents of this register before you issue a macro instruction in your error routines.

You may provide individual error procedures for the various IOCS functions such as loading, adding, sequential or random retrieving, updating, etc. However, you must save the error information and the contents of the I/O area. This information is essential in the event that an alternate track must be assigned to replace a track that is found to be defective. It must be emphasized that in some cases a recovery and restart procedure may not be possible. For this reason you should always maintain a duplicate of every file you use in the installation.

Note: For sequential retrieval, you must first issue an ESETL and then a SETL macro instruction before you issue the next GET macro instruction in your error routine.

DEVICE=DISK11F

This entry specifies that a fixed-sector IBM 2311 (Model 11 or 12) is used as I/O device for the indexed-sequential file. If this entry is omitted or mispunched, the IOCS assumes the correct device and issues a warning to the operator.

DPCRCD=YES

You may specify this entry for ADD and ADDRTR files to avoid having disk storage locations reserved in the overflow area for duplicate records. You should specify this entry if your ADD or ADDRTR file is likely to contain duplicate records.

DSKXTNT=n

This entry is mandatory for all indexed-sequential files. It specifies the maximum number of extents used for the file. The number must include (1) all the prime data area extents, (2) the cylinder index area, and (3) the independent overflow area (if used), all of which are specified by XTENT statements. Thus, the minimum number specified by this entry is 2: one extent for a prime data area and one extent for a cylinder index. The maximum number of extents permitted is 99 per file.

DTAREX=name

This entry is mandatory if IOROUT=LOAD has been specified for the file. The entry DTAREX=name specifies the symbolic address to which the IOCS branches if the prime data area becomes full while a file is being loaded. You may issue an ENDFL macro instruction in your routine to prepare the newly organized file for closing. This permits the remaining records to be treated as additions to the file.

DUPREX=name

This entry specifies the symbolic address to which the IOCS branches if a duplicate record (equal keys) is detected while loading or extending a file, or when adding a record to a file. This entry is mandatory if the specification for IOROUT is either LOAD, or ADD, or ADDRTR. You can avoid having disk storage locations reserved for duplicate records if you specify the detail entry DPCRCD=YES for ADD or ADDRTR files.

EOFADDR=name

Include this entry when records of the file are to be retrieved sequentially (i.e., if either TYPEFLE=SEQNTL or TYPEFLE=RANSEQ has been specified for the file). The entry EOFADDR=name specifies the symbolic address of the routine to which the IOCS branches when an end-of-file condition occurs. An end-of-file condition is detected by reading the end-of-file record. In your end-of-file routine, you may perform any operations required for the end of the job. Generally, a CLOSE macro instruction is issued for the file.

Note: If, in the end-of-file routine, you want to discontinue/terminate the processing of a program that is a mainline program, close all disk files first.

ERRINF=YES

This optional detail entry indicates whether you require logical error information or not.

If you specify ERRINF=YES, logical information about each disk error that may occur is provided in a special field. You can address this field using a symbolic name that consists of the file name with the suffix A+2. For example, if the name of the file is PAYROLL, the logical error information is contained in PAYROLLA+2. The information indicates the disk area in which the error occurred (the corresponding bit is on).

Byte	Bit	Area
PAYROLLA+2	0	prime data area
PAYROLLA+2	1	cylinder overflow area
PAYROLLA+2	2	independent overflow area
PAYROLLA+2	3	cylinder index
PAYROLLA+2	4	track index
PAYROLLA+2	5-7	not used

PAYROLLA+3 is not used. If the entry ERRINF=YES is omitted, no logical error information is supplied.

IOAREAL=name

This entry is mandatory if the specification for IOROUT is LOAD, ADD, or ADDRTR. The entry IOAREAL=name specifies the symbolic address of the output area to be used when loading or extending a file, or when adding records to a file. The specified name must be the same as the one that defines the area in the problem program.

For unblocked records of ADD and ADDRTR files, the output area must be large enough to contain one record plus the link field, i.e., the size of IOAREAL must be equal to or greater than the record size specified in the RECSIZE=n entry plus 6, (see Note below). For blocked records, the output area must be large enough to ccntain a complete block. If records are to be added to a file of blocked records, the output area must be preceded by a work area that is large enough to contain one record. This work area is used by the IOCS during the shift procedure which is necessary when a record is inserted into the prime data area (see the description of WORKA=name below).

Note: The total length of IOAREAL must be equal to 270 bytes or an integer multiple of 270. If the calculated length is not 270 or an integer multiple thereof, the next larger allowed value must be used for the output area.

IOAREAR=name

This entry is mandatory if TYPEFLE=RANDOM or TYPEFLE=RANSEQ has been specified for the file. The entry IOAREAR=name specifies the symbolic address of the I/O area to be

used for random retrieval and/or updating operations. The specified name must be the same as the one that defines the area in the problem program. The size of the area must be an integer multiple of 270 and large enough to contain a block or, for unblocked files, a record of the file. No work area is required.

IOAREAS=name

This entry is mandatory if TYPEFLE=SEQNTL or TYPEFLE=RANSEQ has been specified for the file. The entry IOAREAS=name specifies the symbolic address of the I/O area to be used for sequential retrieval and/or updating operations. The specified name must be the same as the one that defines the area in the problem program. The size of the area must be the same as described for the entry IOAREAR=name. No work area is required.

IOREG=(register)

This entry is required only for RETRVE or ADDRTR files containing blocked records that are to be processed in the I/O area.

As operand, specify in parentheses any of the registers 8 through 13 or a symbolic name that has been equated to one of the registers 8 through 13. The specified register contains the address of a logical input record that is available for processing. The IOCS places this address into the register each time a READ or GET macro instruction is executed.

IOROUT=code

This entry is mandatory for all indexed-sequential files. It specifies the type of function to be performed.

Code	Function
LOAD	Building a logical file on disk or extending a file beyond the highest record currently contained in an organized file.
ADD	Inserting new records into an organized file.
RETRVE	Retrieving records from a file for either random or sequential processing and/or updating.
ADDRTR	Both inserting new records into a file (ADD) and retrieving records for processing and/or updating (RTR).

KEYARG=name

Include this entry for

1. random retrieval (TYPEFLE=RANDOM or TYPEFLE=RANSEQ has been specified), and
2. sequential retrieval (TYPEFLE=SEQNTL or TYPEFLE=RANSEQ has been specified), if this type of retrieval operation is to begin with a specific key, i.e., the operand of the SETL macro instruction is either KEY or GKEY.

The entry KEYARG=name specifies the symbolic address of the location that contains the key of either the record that is to be retrieved or the record with which sequential retrieval is to begin.

The problem program must place the key into the specified location before issuing the macro instruction that requires the key.

KEYLEN=n

This entry is mandatory for all indexed-sequential files. It specifies the length of the key in number of bytes (maximum length is 60 bytes).

KEYLOC=n

This entry is required for all indexed-sequential files. It specifies the position of the leftmost byte of the key relative to the beginning of the record. For example, if the key is recorded in positions 21-25 of each record in the file, KEYLOC=21 must be specified. To determine the leftmost byte of the key, the first byte of the record is counted as byte 1. The key may be defined as a field anywhere within a record, however, the rightmost three bytes of the record must not be included.

NRECDS=n

This entry is always required. It specifies the number of records in a block. For unblocked records, n will be 1.

Because the IBM 2311 uses a fixed sector length of 270 bytes, the IOCS transfers from and to disk an area whose length is equal to the specified number of records times the number of bytes contained in a record, provided the product of the two is an integer multiple of 270. If the product is not an integer multiple of 270, the IOCS transfers an area whose length is equal to the next higher integer multiple of 270. Take this into consideration when defining the I/O area in your program.

RECFORM=code

This mandatory entry specifies the record format of the file. The IOCS can handle both of the permissible record formats in one program. However, all records in a given file must be of the same format.

Code	Record Format
FIXUNB	Unblocked format-F records
FIXBLK	Blocked format-F records

The specification that is used when the logical file is loaded onto disk must again be included whenever the file is processed.

RECSIZE=n

This entry is mandatory for all indexed-sequential files. It specifies the number of bytes in a logical record. All logical records must be of the same length. The maximum length permitted for records of an indexed-sequential file is 4096 bytes.

RTRVEX=name

This entry specifies the symbolic address to which the IOCS branches if the record with the described key cannot be found in the file. This entry is always required if TYPEFLE=RANDOM or TYPEFLE=RANSEQ has been specified for the file. If TYPEFLE=SEQNTL has been specified the entry RTRVEX is not required; however, when you omit this entry, you may specify only the operand BOF or GKEY in a SETL macro instruction referring to the file.

If the IOCS branches to the specified routine during the execution of a SETL macro instruction, you must write an ESETL macro instruction in your routine before any other macro instruction referring to the same file is issued.

Note: If the RTRVEX routine discontinues the processing of a program that is a mainline program, it must include CLOSE macro instructions for all disk files.

SQCHEX=name

This entry specifies the symbolic address to which the IOCS branches if a record is out of collating sequence while a file is being loaded or extended.

Note: If the SQCHEX routine discontinues the processing of a program that is a mainline program, it must include CLOSE macro instructions for all disk files.

TYPEFLE=code

This entry is required when a retrieval function is to be performed (i.e., IOROUT=RETRVE or IOROUT=ADDRTR has been specified). The TYPEFLE entry specifies the type of processing to be used for the file.

Code	Type of Processing
RANDOM	Random Processing: Records are retrieved from the file in random order by key. READ macro instructions are used to transfer records from disk to main storage.
SEQNTL	Sequential Processing: The IOCS retrieves records in sequential order by key. The first record retrieved may be the first record of the file or a record specified by the key (see SETL Macro Instruction). The problem program specifies the first key to be retrieved. GET macro instructions are used to transfer records from disk to main storage.
RANSEQ	Random and Sequential Processing: READ and/or GET macro instructions are used to transfer records from disk to main storage: READ for random retrieval and GET for sequential retrieval.

UPDATE=code

This entry is required if disk records are to be updated. It specifies the type of processing used to update records.

Code	Type of Processing
RANDOM	Random Processing with Updating: A WRITE macro instruction causes a record to be written onto disk at the same location from which this record was retrieved by a preceding READ macro instruction. UPDATE=RANDOM can only be specified in conjunction with either TYPEFLE=RANDOM or TYPEFLE=RANSEQ.
SEQNTL	Sequential Processing with Updating: A PUT macro instruction causes a record to be written onto disk at the same location from which this record was retrieved by a preceding GET macro instruction. UPDATE=SEQNTL can only be specified in conjunction with either TYPEFLE=SEQNTL or TYPEFLE=RANSEQ.

RANSEQ Random and/or Sequential Processing with Updating:
A WRITE macro instruction causes a record to be written onto disk at the same location from which it was retrieved by a preceding READ macro instruction. A record that was retrieved by a GET macro instruction is written onto disk by a subsequent PUT macro instruction. UPDATE=RANSEQ can only be specified in conjunction with the TYPEFLE=RANSEQ detail entry.

VERIFY=NO

When this entry is specified, the records are not checked after they have been written onto disk. If you omit this entry all records written onto disk are verified. If an irrecoverable disk-write error occurs, the IOCS branches to your DERREX routine. Although processing time increases when the records are verified, you are strongly recommended to omit the entry VERIFY=NO unless you require maximum throughput.

WORKA=name

This entry is required when records are to be added to a file containing blocked records. The entry specifies the symbolic address of a work area that must immediately precede the I/O area specified in the IOAREAL entry. This work area must be exactly as long as one logical record. The specified name must be the same as the one that defines the area in the problem program.

WORKL=name

This entry must be included whenever a file is to be loaded or extended, or records are to be added to a file. The entry specifies the symbolic address of the work area to which you must move the data records so that the IOCS can load them or add them to the file. The specified name must be the same as the one that defines the area in the problem program.

If the file is to contain blocked records, this work area must be large enough to accommodate one logical record. If the file is to contain unblocked records, six additional bytes must be provided for ADD and ADDRTR files so that the pertinent sequence-link field can be suffixed to each record. If unblocked records are added to a file, the contents of the area defined by WORKL is destroyed by a WRITE NEWKEY macro instruction.

WORKR=name

Include this entry when (1) the records of the file are processed in random order and (2) the individual records are to be processed in a work area rather than in the I/O area. The entry specifies the symbolic address of the work area, and the specified name must be the same as the one that defines the area in the problem program. This work area must be large enough to accommodate one logical record.

When this entry is included and a READ or WRITE instruction referring to the file is executed, the IOCS moves the individual record to or from this area.

When this entry is included for a file, the IOREG detail entry must be omitted unless the entry TYPEFLE=RANSEQ is included in the problem program.

WORKS=YES

This entry is required if (1) records are to be processed in sequential order and (2) the individual records are to be processed in work areas rather than in the I/O area. Each GET or PUT macro instruction must specify the symbolic address of the work area to or from which the IOCS is to move the record. The work area must be large enough to accommodate one logical record.

When this entry is included for a file, the IOREG detail entry must be omitted unless the entry TYPEFLE=RANSEQ is included in the problem program.

Loading or Extending Indexed-Sequential Files

Three different macro instructions are required in the problem program to load the records of an indexed-sequential file onto disk: SETFL, WRITE, and ENDFL.

The function of originally loading an indexed-sequential file onto disk and the function of extending this file by adding new presorted records beyond the record with the highest key are essentially the same. Both are considered a load operation (specified by the IOROUT=LOAD entry in the DTFIS statement for the file), and they both use the same macro instructions in the problem program.

If you wish to load a file with a file name for which an unexpired file label exists, a halt occurs. The operator then has the choice to extend the existing file or to load the new file. In the latter case, the file with the unexpired file label is destroyed. The IOCS automatically performs an original load function if it

encounters an expired file label with the same file name. If you want to extend your file, you must therefore ensure that the file is not expired, i.e., the date entered on the DATE card for the job must not be higher than the expiration date for the file.

You must specify the areas of the volumes in which you want to load your file by means of job control XTENT statements. The areas are:

1. the prime data area(s) (on one or more volumes) where the data records are written,
2. a cylinder index area where you want the IOCS to build the cylinder index, and
3. an (optional) independent overflow area.

Note: All extents that are required for the LOAD function and all extents that might be required later in order to add new records must be specified for the initial loading of the file.

During the load operation, the IOCS builds the track and cylinder indexes.

SETFL MACRO INSTRUCTION

This macro instruction (SET File Load mode) prepares the IOCS for a file loading or extending operation.

Name	Operation	Operand
[name]	SETFL	filename

The operand contains the name of the disk file to be loaded or extended. The file name must be the same as the one specified in the header entry of the DTFIS statement for the file.

The SETFL macro instruction must be issued whenever a new file is to be loaded or a loaded file is to be extended.

WRITE MACRO INSTRUCTION

This macro instruction loads one record into a disk file.

Name	Operation	Operands
[name]	WRITE	filename,NEWKEY

The first operand specifies the name of the file into which the record is to be loaded. The file name must be the same as the one specified in the header entry of the DTFIS statement for the file. The second operand must be NEWKEY.

Before issuing the WRITE macro instruction, the problem program must place the record for the file into the work area whose symbolic address you must provide in the WORKL=name entry of the DTFIS statement. The WRITE macro instruction causes the record in the work area to be moved into the output area specified in the IOAREAL=name entry of the DTFIS statement.

If records are to be blocked, the block will be built in the output area and placed onto disk when it is full; otherwise, one record is placed onto disk each time a WRITE macro instruction is executed.

Before the record is moved to the output area, the IOCS performs a sequence check by key. The key may be a part number, an employee serial number, or any other identifying information that is contained in every record of a file.

All keys of a file must have the same length. A key should not consist entirely of a string of hexadecimal zeros nor should a key contain any byte in which all the bits are on (hexadecimal FF).

The key may be positioned anywhere within the record, with the following restrictions:

1. The key position must be the same for all records of the file.
2. At least three bytes must be available between the last byte of the key and the end of the record.

Specify the location of the key in the KEYLOC and the length of the key in the KEYLEN entry of the DTFIS statement for the file. This enables the IOCS to check the sequence of records and to ensure that each record is in proper sequence by its key before the record is moved from the work area into the output area. The IOCS also checks for duplicate records, i.e., records with the same keys. If a record is out of sequence or a duplicate record is found, the IOCS branches to the appropriate routine(s) as specified in the SQCHEX and/or DUPREX entries of the DTFIS statement for the file.

ENDFL MACRO INSTRUCTION

Issue this macro instruction (END File Load mode) if you want to terminate the loading or extending of your file.

Name	Operation	Operand
[name]	ENDFL	filename

The operand contains the name of the disk file that has been loaded or extended. The file name must be the same as the one specified in the operand field of the SETFL macro instruction that initiated the operation.

The ENDFL macro instruction must be issued to complete the loading or extending of a file.

Adding Records to Indexed-Sequential Files

The macro instructions used in the problem program for the insertion of new records are WRITE and WAITF.

After an indexed-sequential file has been loaded onto disk, new records can be inserted into their proper places in the file. The location of the key in the new records must be the same as that in the records already in the file. The functions required for adding records are provided by specifying ADD or ADDRTR in the IOROUT entry of the DTFIS statement for the file.

WRITE MACRO INSTRUCTION

This macro instruction inserts one record into an indexed-sequential file.

Name	Operation	Operands
[name]	WRITE	filename,NEWKEY

Before the WRITE macro instruction is issued, the problem program must store the record to be added into a work area specified in the WORKL entry of the appropriate DTFIS statement.

Note that an additional work area is required when records are to be added to a file containing blocked records. Specify this additional work area in the WORKA entry of the DTFIS statement and define it so that it immediately precedes the area IOAREAL. The operations caused by a WRITE macro instruction are described in detail under Inserting Records in the section Processing Indexed-Sequential Files.

WAITF MACRO INSTRUCTION

This macro instruction ensures that the execution of the preceding WRITE macro instruction has been completed before further processing is done that involves the work areas specified for the file. The WAITF macro instruction permits the seek operation for seeking the track index to be overlapped with processing.

Name	Operation	Operand
[name]	WAITF	filename

The file name must be the same as the one specified in the header entry of the DTFIS statement for the file.

The WAITF instruction must follow a WRITE macro instruction. It does not have to be the next instruction after the WRITE macro instruction. However, it must be written before the next macro instruction for the same file is issued.

If an error condition is detected during the execution of the WRITE functions, the IOCS branches to the appropriate error routine.

Random Retrieval and Updating

The following three macro instructions are available for use in the problem program to retrieve and update records randomly: READ, WRITE, and WAITF.

If you want to retrieve the records of an indexed-sequential file in random order for processing and/or updating, enter RETRVE or ADDRTR in the IOROUT entry of the appropriate DTFIS statement. Random processing must be specified in the TYPEFLE entry of the file definition statement, and updating (if used) must be specified in the UPDATE entry.

Because random reference to the file is made by record keys, the problem program must supply the key of the desired record to the IOCS. To do this, store the key in the key area specified by the KEYARG entry of the appropriate DTFIS statement. The key that is placed into the key area specified in the KEYARG entry designates both the record to be retrieved and (if updating is specified) the record location on disk into which the updated record is to be written.

READ MACRO INSTRUCTION

This macro instruction causes a specified record to be retrieved from an indexed-sequential file.

Name	Operation	Operands
[name]	READ	filename,KEY

The first operand specifies the name of the file from which a record is to be retrieved and placed into main storage. The file name must be the same as the one specified in the header entry of the DTFIS statement for the file. The second operand must be KEY. Place the key of the record into the field designated by the KEYARG entry, before issuing the READ macro instruction.

If a work area is used in addition to the I/O area (DTFIS entry WORKR), the record is also moved into the specified work area.

If the records of the file are blocked, the IOCS causes the entire block containing the desired record to be read into the I/O area. The record is made available for processing either in the I/O area or in the work area specified in the WORKR entry of the DTFIS statement. For processing in the I/O area, the IOCS supplies the record address in the register specified directly or symbolically in the IOREG entry of the DTFIS statement.

If the IOCS does not find the specified record, control is transferred to the routine specified in the RTRVEX entry of the appropriate DTFIS statement.

WRITE MACRO INSTRUCTION

This macro instruction causes the record retrieved by the preceding READ macro instruction (for the same file) to be written onto disk at the location from which it was retrieved.

Name	Operation	Operands
[name]	WRITE	filename,KEY

The first operand specifies the name of the file to which a record is to be returned. The file name must be the same as the one specified as the first operand of the READ macro instruction that retrieved the record. The second operand must be KEY.

The WRITE macro instruction causes the record to be written onto disk at the location specified by the key used by the preceding READ macro instruction. Therefore, if you do not change the contents of KEYARG, the problem program need not supply the key again.

WAITF MACRO INSTRUCTION

This macro instruction ensures that the reading of a record into, or the writing of a record from, the I/O area for the file has been completed before further processing is done that involves the same I/O area. The WAITF macro instruction also permits the seek operation for seeking the track index to be overlapped with processing.

Name	Operation	Operand
[name]	WAITF	filename

The file name must be the same as the one specified in the header entry of the DTFIS statement for the file.

A WAITF macro instruction must be issued after every READ or WRITE macro instruction referring to a file for which random retrieval and updating has been specified. After a READ macro instruction, issue a WAITF macro instruction before you start processing the record retrieved by that READ macro instruction. After a WRITE macro instruction, issue a WAITF macro instruction either

1. before you issue another READ macro instruction referring to the same file, or
2. before you start using the I/O area for the file in the problem program, or
3. before you issue any other macro instruction that refers to the same file (including CLOSE),

whichever is earlier.

Sequential Retrieval and Updating

Four macro instructions are available for use in the problem program to cause the records of an indexed-sequential file to be retrieved and updated sequentially: SETL, GET, PUT, and ESETL. If you want to retrieve and update the records of the file in both ways, randomly and sequentially, the macro instructions READ, WRITE, and WAITF are used for random retrieval as explained in the section Random Retrieval and Updating.

From an indexed-sequential file, records can be retrieved in sequential order by key for processing and/or updating. To cause the IOCS to perform sequential retrieval and updating, specify the following in the detail entries of the appropriate DTFIS statement:

1. Retrieval of records (RETRVE or ADDRTR in the IOROUT entry).

2. Sequential processing (SEQNTL or RANSEQ in the TYPEFLE entry).

3. Updating of records (SEQNTL or RANSEQ in the UPDATE entry).

If the specifications ADDRTR and RANSEQ are used in these three detail entries, the records can be retrieved randomly or sequentially, and new records can be added.

Sequential retrieval can begin either at a record identified by key or at the beginning of the logical file. Specify the starting reference as the second operand in the SETL macro instruction for the file.

The key of the first record must be moved to the main-storage field specified in the KEYARG=name entry of the DTFIS statement for the file. The IOCS derives the position of the key from the KEYLOC entry of the DTFIS statement. When searching for the specified record, the IOCS first locates the correct track and then examines the key area within each record on the track to find the specified record.

SETL MACRO INSTRUCTION

This macro instruction (SET Limits) prepares the IOCS for sequential processing and determines the point at which processing is to begin.

Name	Operation	Operands
[name]	SETL	filename,BOF
[name]	SETL	filename,KEY
[name]	SETL	filename,GKEY

The SETL macro instruction can be written in one of three forms depending on the specified starting reference. In all forms, the first operand specifies the name of the file for which sequential retrieval and updating is desired. The file name must be the same as the one specified in the header entry of the DTFIS statement for the file.

The meaning of the second operand is as follows:

1. If it is BOF, sequential retrieval begins with the first record in the file, i.e., the next GET macro instruction for the file retrieves the first logical record of the file.

2. If it is KEY, sequential retrieval begins with the record whose key is contained in the field specified in the KEYARG=name entry. If the record is not found, control is transferred to the routine specified in the RTRVEX=name entry of the DTFIS statement.

3. If it is GKEY, sequential retrieval begins with the record whose key is contained in the field specified in the KEYARG=name entry or, if this record is not found, with the record having the next greater key. If no greater key is available, the first GET causes the IOCS to branch to the address of the end-of-file routine.

After the SETL macro instruction has been executed the disk-storage access mechanism is positioned for retrieval of the first record; therefore, if the second operand is not BOF, the problem program must supply the key before the SETL macro instruction is executed.

Note: Neither BOF nor GKEY require the RTRVEX entry. However, if RTRVEX is not specified when KEY is used as the second operand, an error condition occurs. The Assembler prints a diagnostic message stating that KEY is treated as an undefined symbol.

GET MACRO INSTRUCTION

This macro instruction causes the next sequential record (according to key) to be retrieved from an indexed-sequential file.

Name	Operation	Operands
[name]	GET	filename
[name]	GET	filename,workname

The GET macro instruction can be written in either of two forms, depending on the area in which the records will be processed. In both forms, the first operand specifies the name of the file from which a record is to be retrieved. The file name must be the same as the one specified in the header entry of the DTFIS statement for the file.

If records are to be processed in the I/O area, no second operand is required. If records are to be processed in a work area, the second operand specifies the name of the work area to which each record is to be moved.

Because the IOCS waits for the completion of each GET macro instruction, the retrieved record is available when the next sequential instruction in the problem program is executed.

PUT MACRO INSTRUCTION

This macro instruction causes the record retrieved by the preceding GET macro instruction (for the same file) to be replaced in the location from which it was retrieved.

Name	Operation	Operands
[name]	PUT	filename
[name]	PUT	filename,workname

The PUT macro instruction can be written in either of two forms, depending on the area in which the records will be built. In both forms, the first operand specifies the name of the file to which a record is to be returned. The file name must be the same as the one specified as the first operand of the GET macro instruction that retrieved the record.

If records are processed in the I/O area, no second operand is required. If records are processed in a work area, the second operand specifies the name of the work area from which records will be moved to the I/O area. Note that the work area referred to in the PUT macro instruction will frequently be the same as the one in the GET macro instruction. However, any work area may be referred to in the PUT macro instruction.

A PUT macro instruction is required only when a record in the file has been changed (updated).

If a PUT macro instruction is not issued for any record in the block, the subsequent GET for the file will not cause the writing of the block. At the end of the file, the ESETL macro instruction causes the last block processed to be written, if necessary.

ESETL MACRO INSTRUCTION

This macro instruction (End SET Limit) indicates the end of a sequential retrieval and updating operation that was initiated by a SETL macro instruction.

Name	Operation	Operand
[name]	ESETL	filename

The operand contains the name of the file that has been sequentially retrieved and updated. The file name must be the same as the one specified in the operand of the SETL macro instruction that initiated sequential processing.

When, in a program, sequential retrieval is to be followed by the addition of records to the file (i.e., IOROUT=ADDRTR has been specified) or by random retrieval, write the ESETL macro instruction at the end of sequential retrieval and before you issue a WRITE instruction for the first addition or a READ instruction for the first record to be randomly retrieved. If sequential retrieval is to be resumed when random retrieval and additions have been completed, issue another SETL macro instruction. Figure 16 illustrates the proper use of SETL and ESETL macro instructions.

```
SETL   filename,BOF
 .
 .
GET    filename
 .
 .
PUT    filename
 .
GET    filename
 .
PUT    filename
 .               save key of last record
 .               processed sequentially if
 .               that key is to be used in
 .               the next SETL instruction
ESETL  filename
 .
WRITE  filename,NEWKEY
 .
WAITF  filename
READ   filename,KEY
 .
WAITF  filename
 .
WRITE  filename,KEY
 .               provide proper key for
 .               the SETL instruction
WAITF  filename
 .
SETL   filename,GKEY
```

Figure 16. Use of SETL and ESETL Macro Instructions for IOROUT=ADDRTR

Organizing and Processing Indexed-Sequential Files

The first part of this section provides information on the organization of indexed-sequential files, i.e., on cylinder and track indexes and overflow areas. The second part tells you how an indexed-sequential file is processed by the IOCS. This information is not required for coding a program. However, you might find it useful reading since some of the information is liable to help you to improve your coding.

ORGANIZING AN INDEXED-SEQUENTIAL FILE

When a logical file of presorted records is loaded onto disk, the IOCS organizes the file in such a manner that you have direct access to any record.

Reference can be made to records at random throughout the logical file, or to a series of records in the file in their presorted sequence (collating sequence). The IOCS routines also provide for additions to the file at a later time, still maintaining both the random and the sequential access capabilities.

The IOCS loads the records one after the other into a specified area of the disk volume. This area is called the prime data area. It may consist of one or more disk extents. To define the prime data area, you must specify the starting and ending limits of its extent(s) in job control XTENT statements (one for each extent). The limits of prime data extents must be on cylinder boundaries.

Indexes

As the IOCS loads a file of records sorted by control information, it builds two indexes for the file - a track index and a cylinder index. These indexes are utilized for both random and sequential access to records.

Once a file has been loaded and the related indexes have been built, the IOCS routines search for specified records by referring to the indexes. When a particular record (specified by the key) is requested for processing, the IOCS searches the cylinder index, then the track index, and finally the individual track.

The indexes are made up of a series of entries, each of which includes the address of a track and a key as follows:

1. For the track index, this key is that of the last record on a specific track. Note: If the length of the last block on a given track exceeds the available space, the portion in excess is written on the subsequent track. Thus, the 'last record' of a track may, in effect, be located on the next higher track.
2. For the cylinder index, this key is the highest key within each cylinder.

The entries are normally blocked, i.e., one disk sector contains more than one entry. The exact number of entries per sector and the number of sectors required for an index depend on the keylength and on the number of prime data tracks (10 minus number of cylinder-overflow tracks). The track address requires six bytes.

Track Index. The track index is the low-level index for the logical file. A separate track index is built for each prime data cylinder used by the file; it contains index entries for that cylinder only. Each track index is located on the cylinder that it indexes. It always begins on the first sector of that cylinder.

When the track indexes are originally built, they contain two identical entries (normal and overflow) for each track utilized on the cylinder. The use of two index entries for each track is required because of overflow records that will occur if more records are inserted in the file at a later time. (Refer to Overflow Areas and Addition of Records.) When overflow records for a track exist, the second (overflow) index entry contains the key of the highest record in the overflow chain and the address of the lowest record in the overflow chain for the track. For example, if the prime data area of the logical file utilizes eight tracks on a cylinder when the file is built and two tracks are overflow tracks, the track index (on track 0) might contain the entries shown in Figure 17. Any subsequent records on track 0 are logical-file data records. The first data record following the track index always begins with a new sector.

Cylinder Index. The cylinder index is the high-level index for the logical file. This index contains one entry for each prime data cylinder occupied by the file. The cylinder index is built in a separate extent, which you specify in a job control XTENT statement. It must be on cylinder boundaries. The index must be built on a cylinder that does not contain data records

for the file (but it may contain the independent overflow area). This cylinder may be on a separate volume provided this volume is on-line whenever the logical file is processed.

The cylinder index contains one entry for each cylinder occupied by the data file. Each entry contains the highest key associated with the cylinder and the address of the track index for that cylinder. For example, if a file requires six cylinders, the cylinder index might contain the entries shown in Figure 18 . The dummy entry indicates the end of the cylinder index.

Overflow Areas and Addition of Records

After a logical file has been loaded onto disk, it may subsequently become necessary to add records to the file. The records to be added may

1. contain keys that are above the highest key currently in the file (in this case, the records constitute an extension of the file), or

2. contain keys that are either lower than the lowest key currently in the file or fall between keys already in the file (in this case, the records are to be inserted in proper sequence in the organized file).

If all records to be added have keys that are higher than the highest key in the present file, the new records, which must be presorted, can be added by loading them into the file. No overflow area is required.

If new records are to be inserted among those already organized, an overflow area is required. The IOCS uses overflow areas to permit the insertion of records without a complete reorganization of the established file. The random and sequential retrieval of records is maintained by inserting references to the overflow chains in the track indexes and by using a chaining technique in the overflow records. For chaining, a sequence-link field is suffixed to the data record in the overflow area. The sequence-link field is a six-byte area. It contains the address of the record in the overflow area that has the next higher key. Thus, a chain of sequential records can be followed in a search for a particular record. The sequence-link field of the highest record in the chain indicates the end of the chain.

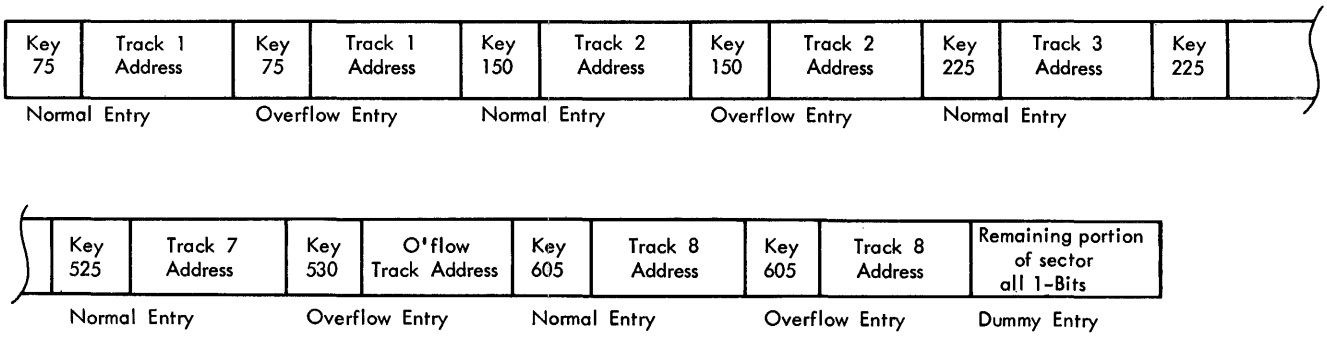
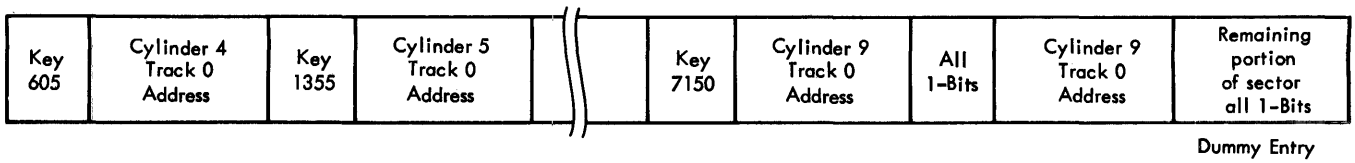


Figure 17. Schematic Example of a Track Index



•Figure 18. Schematic Example of a Cylinder Index

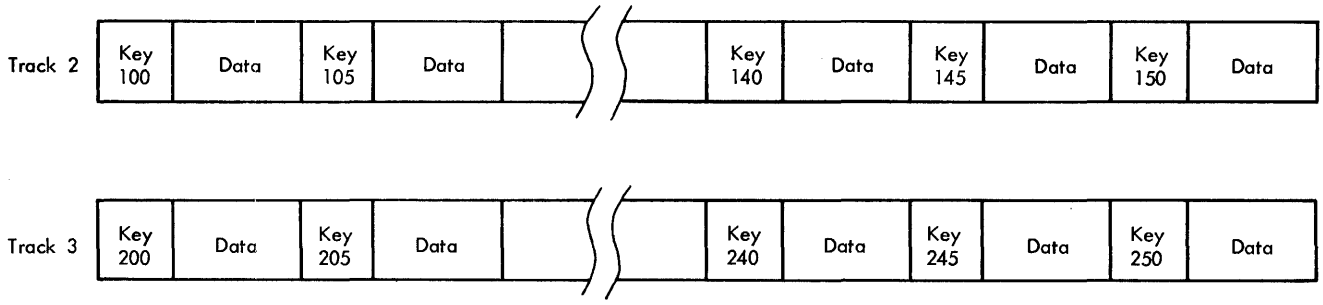


Figure 19. Data Records as Originally Organized on Tracks 2 and 3

To add a record, the IOCS searches the established indexes first to determine on which track the record must be inserted. The keys of the last records on the tracks in the present file determine the track where an inserted record belongs. A record is always inserted on the track where:

1. the last key is higher than the key of the insertion record, and

2. the last key of the preceding track (if any) is lower than the key of the insertion record.

For example, assume tracks 2 and 3 are organized with the record keys shown in Figure 19. In this case, records with keys such as 151, 175, 199, 215, and 239 are inserted on Track 3 or in the related overflow chain that has developed. Any key lower than 150 is added to either Track 1 or Track 2; any key higher than 250 belongs to Track 4 or above. The track index overflow entries always contain that key which was the highest on a particular track at the time the disk file was originally organized.

Updating of the Track Index. The IOCS updates the track index to reflect the changes caused by the addition of records. The first index entry for the track has the key field changed to indicate the new last record located on the track. The second index entry for the track has the track address changed to point to the address of the lowest overflow record of the chain. If a record with key 102, for example, is added to a file organized as shown in Figure 19 and if the overflow area is located on Track 9, the track index records contain the information shown in Figure 20.

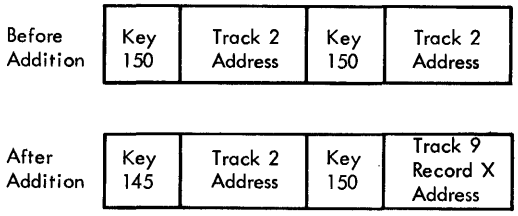


Figure 20. Example of Track Index Entries Before and After Addition of a Record on Track 2

Chaining by Sequence Link Field. If a record is to be placed between the last record currently on the track and the last record originally on the track it belongs in the overflow area. The IOCS writes the record in the overflow area following the last record previously written. The IOCS searches through the chain of records associated with the corresponding track for this record and identifies the sequential position the record should take. Then the sequence-link fields of the new record, and the record preceding it by sequential key, are adjusted to point to the proper records.

If, for example, records with the keys 150, 145, and 140 are already in the overflow area and record 142 is to be added, the sequence-link fields of records 140 and 142 must be adjusted (see Figure 21).

RECORD	SEQUENCE-LINK FIELD	
	Before Addition	After Addition
150	*	*
145	150	150
140	145	142
142+	-	145

* end of chain
\+ added record

Figure 21. Example of Sequence-Link Fields Adjusted for Addition of a Record (142)

If a record is inserted into the last track of a file, the EOF record is transferred to the overflow area in the same manner as the last data record on any other track.

Overflow-Area Options: You may specify the location of the overflow area(s) for a logical file. The overflow areas may be built by one of three methods:

1. The overflow area for records may be located on each cylinder within the prime data area that is specified by one or more job control XTENT statements for the data file. In this case, you must specify the number of tracks to be reserved for overflow records on each cylinder occupied by the file. The overflow records that occur within a particular cylinder are written in the cylinder overflow area for that cylinder.

   Specify the number of tracks to be reserved for each cylinder overflow area in the CYLOFL entry of the DTFIS statement when the records of the particular file are to be loaded or when records are to be added to an organized file.

2. You can specify an independent overflow area for storing all overflow records of the logical file. In this case, include a job control XTENT statement when the program is executed to specify the disk extent to be used as overflow area. This area may be on the same volume with the data records, or on a different volume that is on-line. However, it must be contained within one volume.

3. You may use both cylinder overflow areas (method 1) and an independent overflow area (method 2). In this case, overflow records are first placed in the cylinder overflow areas within the data file. When any cylinder overflow area is full, the additional overflow records from that cylinder are written into the independent overflow area.

   Method 3 (described above) is preferable to methods 1 and 2, because it provides faster retrieval and avoids frequent file re-organizations. Faster retrieval is achieved because most of the overflow records are located in the cylinder-overflow area. File reorganization is avoided because the overflow records can be stored in the independent overflow area if the cylinder-overflow area is full, so that the program need not enter the ADAREX routine.

The block length (depending on the number and length of the records) for the file applies to both the prime data area and the overflow area(s). Since a sequence-link field (a 6-byte area) is suffixed to each individual record in an overflow area, the number of records to a block in that area may be less than the number of records to a block in the corresponding prime data area. You need not be concerned about this because the IOCS computes the blocking factor according to the block length.

Note: The Model 20 IOCS does not permit the extents of an organized file to be changed. Therefore, if the available overflow areas are full and records are still to be added, the file must be completely reorganized.

When an indexed-sequential file is processed, a field is made available for updating the number of tagged deletion records. You can address this two-byte field in the problem program. When processing is completed, i.e., when a CLOSE macro instruction is issued, the updated field is rewritten into the format-2 label of the pertinent file.

You may address the first byte of this area in main storage by defining an address constant that contains the file name of the pertinent file plus a displacement of 120.

Example: If the file name is PAYROLL, the address constant is:

```
DC Y(PAYROLL+120)
```

Storage Areas

Records in one logical file are transferred to or from one or more I/O areas in main storage. The areas must always be large enough to contain a block of records or a single record if unblocked records are specified. For the functions of adding or retrieving records, the I/O area must also

provide space for a sequence-link field that is used in conjunction with overflow records when the entry RECFORM=FIXUNB is specified (see Overflow Areas and Addition of Records) above. The I/O area requirements are illustrated in Figure 22 and described in detail under IOAREAL, IOAREAR, and IOAREAS in the discussion of the DTFIS Statement.

Records may be processed directly in the I/O area or in a work area. If blocked records are to be processed in the I/O area, specify a register directly or symbolically in the IOREG entry of the DTFIS statement. The specified register is used for indexing, i.e., to point to the beginning of each record and thus locate the record for processing.

If the records are to be processed in a work area, specify one of the DTFIS entries WORKR or WORKS in the file definition statement. The IOCS moves each individual input record from the input area to the work area where it is available to the problem program for processing. Similarly, the IOCS moves a completed record from the work area o the output area. The work area must be large enough to accommodate one data record. Whenever a work area is used, an I/O register is not required.

Multi-File Processing: If multi-file processing with the same physical file is desired (i.e., if two file names - A and B - have been specified in two file definition statements, but the associated DLAB cards contain identical disk extents and file identifiers), issue a CLOSE macro instruction for the file with the filename A before opening the file with the filename B.

PROCESSING AN INDEXED-SEQUENTIAL FILE

This section informs you about the functions the IOCS performs when loading or adding records, or when retrieving and updating records randomly or sequentially.

Loading Records

As records are loaded onto disk, the IOCS causes the writing of a track-index entry each time a track is full, and the writing of a cylinder-index entry each time a cylinder is full. When a track index is completed, the IOCS fills the remaining portion of the sector with 1-bits.

The ENDFL macro instruction causes the writing of the last block of data records, which contains the end-of-file record. It also causes any required index entries to be written, including the dummy entry for the cylinder index.

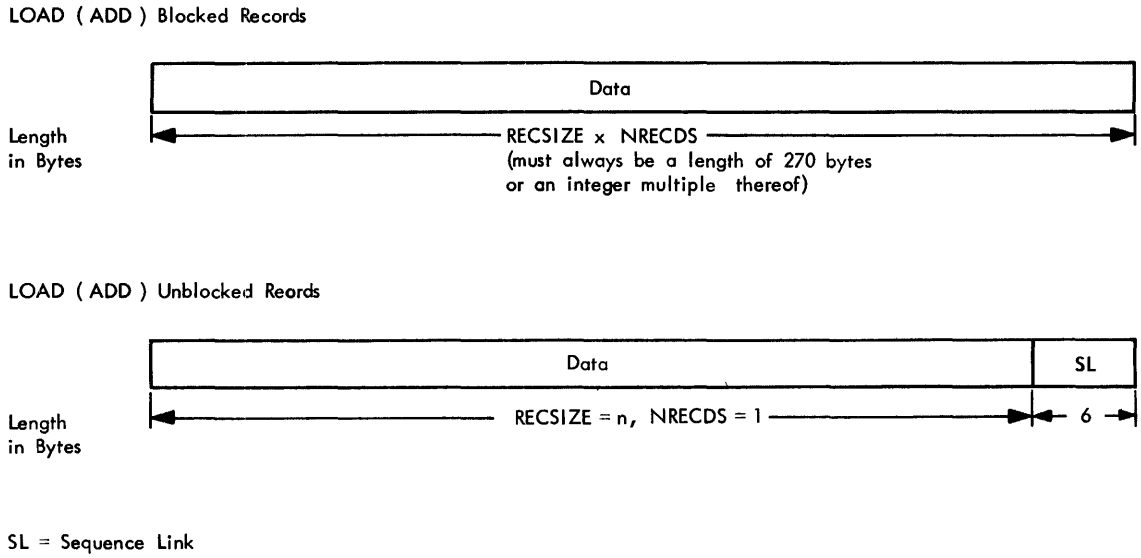


Figure 22. I/O Areas Required for Loading or Adding Function

Adding Records

Before a WRITE macro instruction can be executed, the access mechanism must locate the track on which the record is to be inserted. Normally, this is done by first searching the cylinder index and then the track index. However, before the access mechanism is moved to check the cylinder index, the IOCS checks the I/O area in main storage containing the track index to determine if the proper track-index sector is already in main storage from a previous add-record operation. If so, access movement to the cylinder index is not necessary and the appropriate track-index entry is used to obtain the proper insertion address. This method of checking the track-index sector in main storage before initiating access movement may reduce the average access time considerably if the input records are pre-sorted. In this case, the proper track index is likely to be in main storage for the majority of the insertion records.

If the proper track index is not in main storage, the IOCS causes the following: (1) a seek to the cylinder index, (2) a search for the entry pointing to the proper track index, and (3) a seek to this track index. The IOCS then transfers control to the problem program to permit processing while the access mechanism is moved to the track index location.

The WAITF macro instruction causes the remaining functions of the preceding WRITE macro instruction, i.e., search of the track index, insertion of the record in the prime data or overflow area, and updating of track index entries, to be performed.

When all the functions of the preceding WRITE macro instruction are completed, the IOCS returns control to the instruction immediately following the WAITF macro instruction.

Inserting Records into a File of Unblocked Records. The IOCS searches the indexes to locate the correct track for the record. If the correct track is not an overflow track, the IOCS performs a scan if a prime data block does not occupy more than one sector. Otherwise, it performs an equal/high search by comparing the keys of the records on the track with the key of the record to be inserted. When a record is found whose key is equal to or higher than the key of the insertion record, that record is read from the track and placed into main storage (I/O area). The two keys are compared to determine if a duplicate record was found. If a duplicate record was found, the IOCS branches to the routine specified in the DUPREX entry of the DTFIS statement. If no duplicate key was found, the IOCS causes (1) the insertion record (in the WORKL area) to be written on the track and (2) the record that was read from the track into the I/O area to be moved into the WORKL area.

The next record on the track is read into the I/O area. Then the record in the work area is written on the track. This sequence of operations is repeated until the last record on the track has been read into the I/O area. This last record is then written into the appropriate overflow area, and the appropriate track-index entries are updated. The IOCS uses the cylinder overflow area, provided this area has been specified in the CYLOFL entry of the DTFIS statement and the area is not yet full. Figure 23 illustrates the status of the areas in main storage (IOAREAL and WORKL) during an add-record operation as described above.

If the cylinder overflow area is full, or if you specify only an independent overflow area by means of a job control XTENT statement, the last record is transferred to the independent overflow area. If an independent overflow area has not been specified (or is full) and the cylinder overflow area is full, there is no place to store the overflow record. The IOCS then branches to the routine specified in the ADAREX entry of the DTFIS statement.

If the proper track for the insertion record is an overflow track, the IOCS searches the overflow chain and checks for a duplicate record. If a duplicate record is found, the IOCS branches to the DUPREX routine. If no duplication is found, the IOCS causes (1) the record to be written in the next available location within the overflow area, including the 6-byte sequence-link field and (2) the appropriate linkages to be adjusted to maintain sequential order by key. The new record is written in either the cylinder overflow area or in the independent overflow area. If both these areas are full, the IOCS branches to the ADAREX routine.

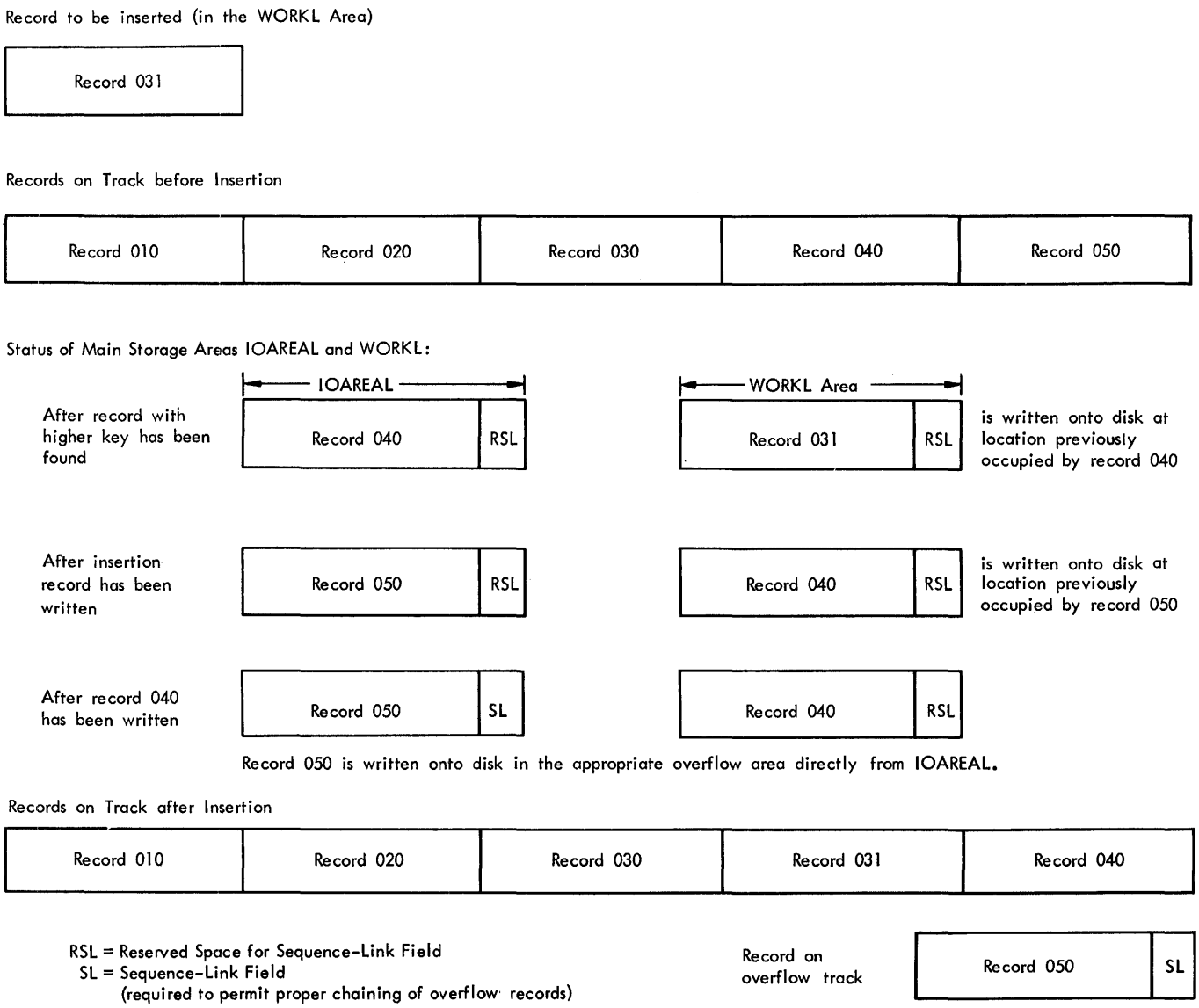


Figure 23. Status of Main Storage Areas IOAREAL and WORKL During an Add-Record Operation for a File Containing Unblocked Records

Inserting Records into a File of Blocked Records. The IOCS searches the indexes to locate the track that contains the begin address of the block into which the new record is to be inserted. This track may be an overflow track.

If the track that contains the begin address of the block is not an overflow track, the IOCS searches the key areas of the records on the track to locate the desired block. When located, the block is read into the I/O area.

The IOCS then examines the key areas within each logical record to find the exact position to insert the new record. The IOCS checks for duplication of records and, if a duplication exists, branches to the DUPREX routine. If there is no duplicate record, all records below the insert position are shifted to the left by one record to make room for the new record that is moved from the WORKL area into the I/O area. As a result, the first record of the block is now contained in the WORKA area.

The block is then written back onto disk beginning with the leftmost byte of the WORKA area. However, the record that originally was the last record of the block is not written at this time. This record is

moved into the WORKA area after the completion of the WRITE operation. A subsequent read instruction reads the next block from disk into the I/O area. This is followed by a write instruction that begins with the WORKA area again. This sequence of operation is repeated until all blocks on the track have been processed.

After the last block of the track has been written, one record is left at the end of the I/O area. This record is then set up as an overflow record with the proper sequence-link field and written into the overflow area. The indexes are updated and the IOCS returns control to the problem program for the next record to be added. If no overflow area is available, the IOCS branches to the ADAREX routine.

Figure 24 illustrates the status of the main storage areas IOAREAL, WORKL, and WORKA during an add-record operation for a file containing blocked records.

If the point of insertion is on an overflow track, the functions are the same as described above under Inserting Records into a File of Unblocked Records.

Random Processing

The READ macro instruction causes the IOCS to search the indexes to determine the track that contains the desired record. The search for the appropriate record is performed in two different manners.

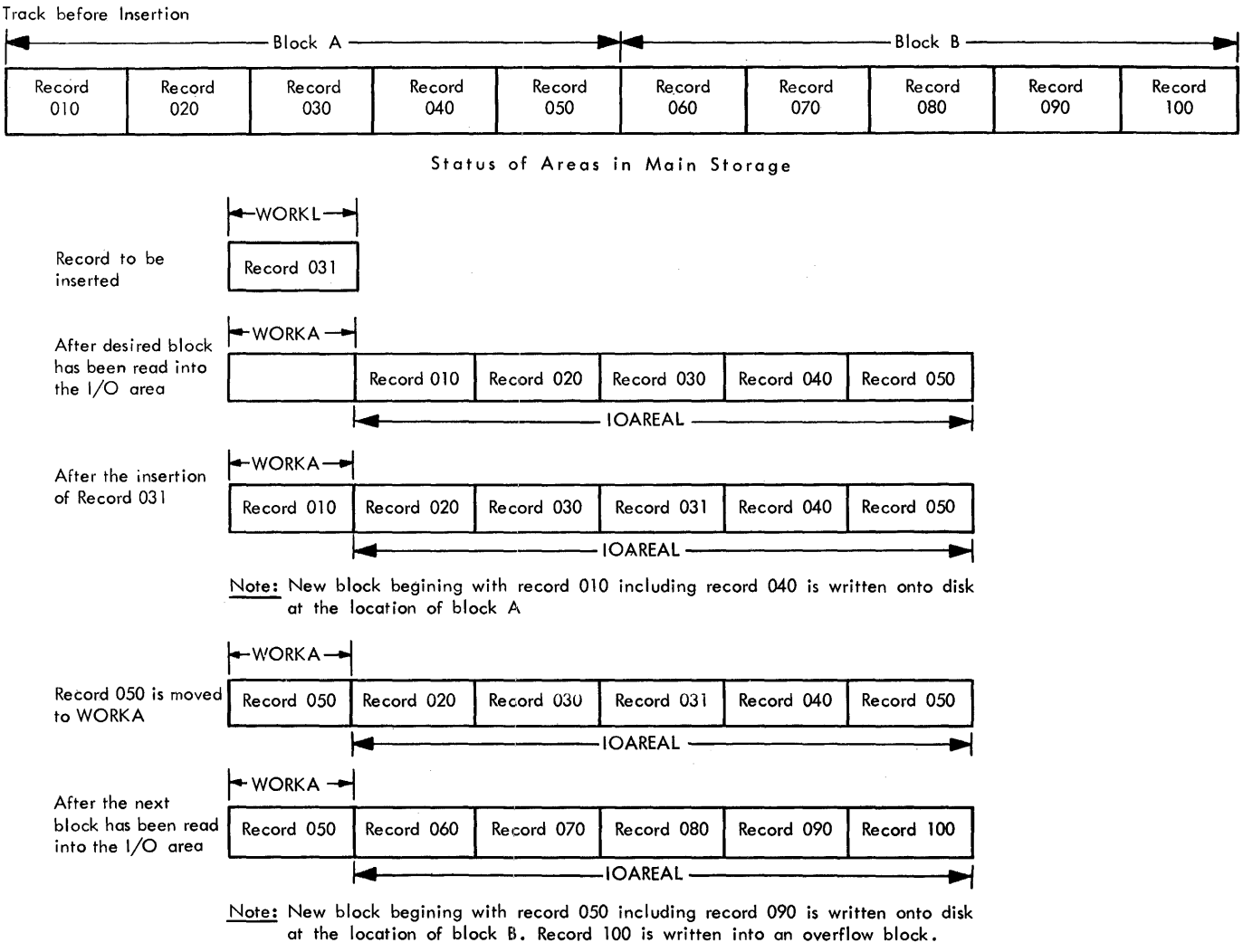


Figure 24. Status of Main Storage Areas IOAREAL, WORKL, and WORKA During an Add-Record Operation for a File Containing Blocked Records

a) If the block length (i.e., the number of records times therecord length) is less than 270 bytes, the program simply scans the track concerned until the appropriate record is found. Then the record is read into the I/O area (IOAREAR).

b) If the block length exceeds 270 bytes, all records on the given track are successively read into the I/O area until the correct record is encountered.

The WRITE macro instruction causes the record retrieved by the preceding READ macro instruction and processed by the problem program to be rewritten onto disk at the location from which it was retrieved.

The WAITF macro instruction causes the remaining functions of the preceding READ(WRITE) macro instruction (e.g., search of the track index, insertion of the record) to be performed. When all functions are completed, the IOCS returns control to the instruction immediately following the WAITF macro instruction.

Sequential Processing

If a GET macro instruction with the file name as the only operand is used and the file contains unblocked records, one logical record is retrieved and made available for processing in the I/O area. If the file contains blocked records, this GET causes the entire block to be read into the I/O area when no further records are available from the block that is already in main storage. The IOCS then makes the addresses of the records available one at a time in the register specified directly or symbolically in the detail entry IOREG of the appropriate DTFIS statement. Each subsequent GET causes the contents of the I/O register to be increased by the number of bytes contained in one record.

If a GET macro instruction with two operands (filename and workname) is used and the file contains unblocked records, the record read from disk into the I/O area is made available for processing in the specified work area. If the file contains blocked records, this GET causes the entire block to be read into the I/O area when no further records are available from the block that is already in main storage. The IOCS then makes the first record of the block available for processing in the specified work area. Each subsequent GET causes the next record to be made available for processing.

If a file containing blocked records is retrieved and updated, the GET macro instruction determines whether or not an output operation is necessary. Because a PUT macro instruction need only be executed for those records that have been updated, the PUT macro instruction merely indicates whether or not a block of records should be written onto disk after all records in that block have been processed. When a GET macro instruction that refers to the first record in a new block is executed, the IOCS determines whether or not a PUT macro instruction was issued for a record in the block just completed. If there was, that block is returned to disk before the new block is read. If no PUT macro instruction was issued, the output operation is omitted because no record in the block was changed.

For unblocked records, the execution of a PUT macro instruction causes a record to be written into the disk storage location from which a record was retrieved by the preceding GET macro instruction for that file.

For blocked records, the execution of a PUT macro instruction does not initiate the writing of records onto disk. Instead, it only indicates that a block should be written before the next one is read. If a work area is specified, the execution of a PUT macro instruction will also move the record from the work area to its proper position within the block. The writing of the block is controlled as described for the GET macro instruction above.

For blocked records, the ESETL macro instruction causes the last block of records to be returned to disk if a PUT macro instruction was issued for any record in that block.

In Assembler language programs, you can issue five macro instructions to communicate with the Monitor. Two refer to the communication region (COMRG and MVCOM), two request functions of the Fetch routine (FETCH and EOJ) and one refers to the printer-keyboard input area (IQIPT). Appendix C is a summary of these Monitor macro instructions.

COMRG MACRO INSTRUCTION

The COMRG macro instruction is used to refer to the communication region. It causes the address of the first byte of the region, which is on a halfword boundary, to be placed into register 8. Then, any field in the region can be referred to by relative addressing.

The COMRG macro instruction has the following format:

Name	Operation	Operand
	COMRG	

Example: The following sequence of instructions places the address of the beginning of the communication region into register 8, loads the contents of bytes 20 and 21 of the communication region (User Area II) into register 12, and then stores the contents of register 12 in a location whose symbolic address is TEST.

```
COMRG
LH      12,20(0,8)
STH     12,TEST
```

MVCOM MACRO INSTRUCTION

The MVCOM macro instruction is used to modify information in the user area of the communication region (bytes 12 to 23). The format of the MVCOM macro instruction is:

Name	Operation	Operands
	MVCOM	first-byte,number,address

first-byte
The relative address of the first byte to be modified in the user area (can be any of the decimal values 12-23).

number
The number of bytes to be modified in the area (can be any of the decimal values 1-12).

address
The name of the storage location that contains the modifying bytes. (A relocatable expression may be used instead of a name.)

Note: After the execution of a MVCOM macro instruction, register 8 contains the address of the communication region.

Example: If MOD is the symbolic address of a location containing the binary number 10001111, the statement:

```
MVCOM  15,1,MOD
```

causes the number 10001111 to be moved to byte 15 of the communication region.

FETCH MACRO INSTRUCTION

The formats of the FETCH macro instruction are:

Name	Operation	Operand
	FETCH	phasename
	FETCH	

phasename
In a disk-resident system, the name of the phase that is to be loaded from the core-image library into main storage. In a card-resident system, the phase that physically follows the phase containing FETCH is to be loaded (phase name is ignored).

No Operand
The FETCH macro instruction may be used without an operand. In a disk-resident system, the omission of the operand causes the next subphase of the last phase specified to be loaded. A subphase is a separately executable routine within a phase of a problem program. It may be overlaid after execution. In a card-resident system, the phase that physically follows the phase containing FETCH is to be loaded.

The Monitor uses registers 8, 14, and 15 in executing the FETCH macro instruction and does not restore their contents. If

you use any of these registers in your program, you should store their contents before issuing a FETCH macro instruction.

When you use card and printer files, issue a WAITC macro instruction prior to issuing a FETCH macro instruction to ensure that all pending card and printer interrupts are handled properly. (Refer to Loading a Program Phase under Programming Considerations).

EOJ MACRO INSTRUCTION

The EOJ (end of job) macro instruction is used at the end of the last phase of a program. It indicates to the disk-resident Monitor that the Job Control program is to be called to prepare the next job for processing.

When program execution is controlled by the card-resident Monitor, an EOJ macro instruction causes a system halt. Loading of the next physical program that follows (the Job Control deck that precedes the next object program deck) is initiated by pressing START.

The format of the EOJ macro instruction is:

Name	Operation	Operand
	EOJ	

Note: If you use your own input/output routines instead of the macro instructions provided by the IOCS, the new PSW may be used to point to the interrupt routine. Ensure that all pending interrupts are cleared and the original contents of the new PSW -- the begin address of the Monitor interrupt routine -- is restored before you issue a FETCH or EOJ macro instruction.

IQIPT MACRO INSTRUCTION

With the IQIPT macro instruction you can refer to the printer-keyboard input area defined at Monitor generation time. This macro instruction loads the begin address of the input area into register 8. Thus, you can refer to any field in the area by means of this base address and a displacement. The format of the IQIPT macro instruction is:

Name	Operation	Operand
	IQIPT	

Unlike the macro instructions for printer-keyboard input and output files, the IQIPT macro instruction is generated in-line in the program and does not require linkage to any DTF routines. It can be used only in programs executed under the control of a Monitor with a printer-keyboard input area.

Programming Considerations

This section informs you about the restrictions you have to observe when writing a program. It tells you which registers you may use and how you can reduce main storage requirements by applying the overlay programming technique.

RESTRICTIONS

When writing a problem program, observe the following:

1. To avoid multiple definition of symbols, do not use any symbols starting with the letter I because all symbols used by the IOCS start with this letter.

2. Do not use file names that are longer than seven characters because the IOCS uses the eighth character position.

3. To avoid multiple-definition of symbols, do not use a file name followed by an additional character as a symbol in your program because the Assembler derives entry points to the IOCS by adding a character to the file name. For example, if READCRD has been assigned as the name of an input file, do not to use symbols such as READCRDA, READCRDB, etc., in the source program.

4. When you use the IOCS in your program, you are not allowed to issue an XIO instruction because this would cause an unexpected interrupt and thus interfere with automatic scheduling of I/O operations by the IOCS.

Loading a Program Phase Including File Definition Statements

When a new program phase including file definition statements is loaded by means of a FETCH macro instruction, the permanent link data area of the Monitor program is modified to point to the interrupt routines in the new program phase. As a result, any I/O macro instruction referring to a file defined in the preceding program phase can no longer be executed. You must also ensure that all pending card and printer interrupts from the preceding program phase have been handled properly. Do this by issuing a WAITC macro instruction before you issue the FETCH for loading the next phase. Figure 25 shows the use of the WAITC macro instruction when another program phase is to be loaded.

Note that a second or subsequent program phase cannot be loaded from a card input device in which data cards were read during any of the preceding program phases.

When, in an inquiry or in a mainline program, you issue a FETCH macro instruction for a separately assembled program that contains file definition statements, you must ensure that this program was also assembled as inquiry or mainline program, respectively. Otherwise, no file protection is provided for this program during inquiries.

OVERLAY PROGRAMMING FOR OPEN AND CLOSE

When your program processes one or more tape and disk files, you can use the overlay programming technique to reduce the number of main-storage positions required. This programming technique allows you to have part or all of the OPEN I/O routines for tape and disk files overlaid the problem program and to have part or all of the problem program overlaid by the CLOSE routines for tape and disk files.

When you specify OVLAY in the DTFEN statement, the OPEN and CLOSE routines for tape and disk files are not generated as part of the DTF routines. Instead, they are generated in-line, i.e., when the Assembler encounters the first (or only) OPEN (and CLOSE) macro instruction for a tape or disk file.

It is not sufficient, however, to specify OVLAY in the DTFEN statement in order to have the OVLAY function performed. In addition, you must observe the following (refer to Figure 25):

1. Write your own tape label handling routines (if any), including those needed when closing a file (or files), ahead of the first OPEN macro instruction.

2. Position all literals required by these label routines ahead of the first OPEN macro instruction (use an LTORG Assembler instruction).

3. Open all tape and disk files before the OPEN routines are overlaid by the problem program.

   Note: If a program utilizing the overlay programming technique is loaded from cards and the loading device is also used as input device for a card file, make sure that the first card of the data file is in proper position to be fed from the hopper of the reading device at the time the file is opened by means of an OPEN macro instruction. (All program cards must have been read when the OPEN macro instruction for the card file is executed.)

The routines used to open files (and additional volumes of multi-volume files) are not available after they have been overlaid. Therefore, OVLAY cannot be specified in programs that process:

- multi-volume tape files,
- multi-file tape reels if more than one file on the tape is used, and/or
- multi-volume sequential disk files.

4. Initiate execution of the OPEN macro instruction by a subsequent XFR statement (XFR BEGIN in Figure 25) which may or may not immediately follow the OPEN macro instruction. A FETCH macro instruction (the first FETCH in Figure 25) must be issued following the OPEN macro instruction. This FETCH causes part or all of the problem program to be loaded. Note that the FETCH macro instruction destroys the contents of register 8.

5. Issue an ORG statement (ORG BEGIN in Figure 25) after the XFR statement. The operand of this ORG statement specifies the address where the overlay is to start and may be the same as the name of the OPEN macro instruction. For details concerning the ACTION, REPRO, XFR, and ORG statements, refer to the SRL publications IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006, and IBM System/360 Model 20, Disk and Tape Programming Systems, Assembler Language, Form C24-9002.

6. Use XFR and ORG statements (XFR and ORG xxx in Figure 25) prior to the CLOSE macro instruction. The operand of the ORG statement specifies an address in the preceding problem program. Only one CLOSE macro instruction should be issued for all files.

7. Issue a FETCH macro instruction (last FETCH in Figure 25) for another program subphase (i.e., another part of the program). This subphase would include the routines that have been generated for the CLOSE and EOJ macro instructions. The loading of this subphase begins at the address specified as the operand of the ORG statement preceding the CLOSE macro instruction.

8. If the overlay programming technique is employed, the Open routine, in some cases, already inserts the first record into the I/O area. Therefore, the I/O areas should be defined immediately preceding the load address of the individual program phases so that they are not overwritten.

```
        START
        DTF
         .
         .
        DTF
         .
         .
        DTFEN  OVLAY
         .       Generated EOF and EOV
         .       routines
LABADR -----
        -----    Tape label handling
        -----    routines
BEGIN   -----    Problem program
        -----    initialization
        USING 12
        USING 13
        -----
        DROP 12
        DROP 13
        OPEN   diskfle, tapefle
         .
         .       Generated OPEN routines
         .       for disk and tape files
         .
         .
        -----
        USING 12
        USING 13
        FETCH
        REPRO
        ACTION DUP
        XFR    BEGIN
        REPRO
        ACTION NODUP
-------------------------------------------
        ORG    BEGIN
ROUTIN -----
        OPEN   cardfle
         .       Generated linkage (to
         .       DTFSR routine)
        -----
        -----
        WAITC
        FETCH
        -----
        REPRO
        ACTION DUP
        XFR    ROUTIN
        REPRO
        ACTION NODUP
-------------------------------------------
        ORG    xxx
        -----
        DROP 12
        DROP 13
FINIS   CLOSE  diskfle, tapefle, cardfle
        -----
        EOJ
        END    FINIS
```

Figure 25. Coding for File Processing Using the Overlay Technique

Note: All I/O areas have to be available during OPEN, Processing, and CLOSE time.

Steps 1 through 5 cause some or all of the coding between the location indicated by the operand of the first ORG statement (BEGIN) and the next XFR statement (XFR BEGIN) to be overlaid by the problem program. Steps 6 and 7 cause the overlaying of part or all of the problem program, for instance, with the CLOSE routine and the end-of-job routines.

User-Written Macro Definitions

When you use the overlay programming technique, the program must not include any of your own macro definitions that contain any of the global SETB symbols &BG0-&BG19, &BG21, &BG27-&BG28, &BG69, and &BG80-&BG87. (Note that you should not use the global SETB symbols &BG13, &BG27, and &BG28 even when OVLAY is not specified.) You must either change the global SETB symbols in your macro definitions or write the program without the use of the overlay programming technique.

Assignment of Base Registers

Since the OPEN and CLOSE routines are generated in-line, consider their approximate sizes when assigning and loading the base registers for the program. For information on the sizes of these routines, refer to the SRL publication IBM System/360 Model 20, Disk Programming System, Performance Estimates, Form C33-6003.

When DTFEN OVLAY is specified, the routines for the processing of the IBM standard labels are generated as part of the OPEN routines and not as part of the DTF routines. These routines use registers 9 and 10 as base registers.

You must drop your base registers prior to issuing the first OPEN or CLOSE macro instruction. Immediately after an OPEN macro instruction, the USING instructions must be repeated. It is not necessary to reload the registers, however, because their contents are restored to the value they contained before the OPEN macro instruction was executed (see Figure 25).

REGISTER USAGE

Registers 11, 12 and 13 have special restrictions on their use in programs containing 1419/1259 Magnetic Character Reader IOCS macro instructions. Refer to the SRL publication, IBM System/360 Model 20, Disk and Tape Programming Systems, Input/Output Control System for the 1419 and 1259 Magnetic Character Readers; Form C33-6001.

Register 14 and 15 have special restrictions on their use in programs containing BSCA IOCS macro instructions. Refer to the SRL publication, IBM System/360 Model 20, Binary Synchronous Communications Adapter, Form C33-4001.

You may freely use any or all of the registers from 10 through 13. Registers 8, 9, 14, and 15 are not readily available for reasons explained below.

Register 8 is used by the Monitor macro instructions FETCH, IQIPT, COMRG, and MVCOM. Save the contents of register 8 before issuing any of these macro instructions.

Registers 8 and 9 are used as operand registers in the LABADDR routine. In this routine, the two registers do not contain the values that were placed into them in the problem program. The two registers are restored to their original values if you return to the IOCS by issuing an LBRET macro instruction with a 1 as operand.

Registers 9 and 10 are not readily available when OVLAY is specified in the DTFEN statement. If you issued USING instructions for register 9 and 10 at the beginning of the program, you must drop these registers prior to issuing the first OPEN macro instruction. You must repeat the USING instructions immediately after the OPEN macro instruction. Reloading the register is not required because its contents are restored to the value that was contained in the register before the execution of the OPEN macro instruction.

Registers 14 and 15 are used by the FETCH macro instruction and by the IOCS imperative macro instructions (GET, PUT, etc.) If you use one or both of these registers in the problem program, make sure that their contents are no longer required before you issue an imperative macro instruction or a FETCH macro instruction, or save the contents of these registers if you need them at a later time.

If you anticipate transition to a larger System/360 model, be aware that the Basic Programming Support and the Basic Operating System do not allow you to use registers 12 and 13.

Registers Required by the IOCS

The record format and the combination of I/O and work areas used in the problem program determine the number of registers (none, one, or two) that must be specified. The summary in Figure 26 indicates when it is required to specify registers IOREG and/or VARBLD.

Record Format	Number of I/O Areas	Work Area specified	IOREG required?	VARBLD required?
Fixed Blocked	1	No	Yes	No
Fixed Blocked	1	Yes	No	No
Fixed Blocked	2	Yes	No	No
Fixed Blocked	2	No	Yes	No
Fixed Unblocked	1	No	No	No
Fixed Unblocked	1	Yes	No	No
Fixed Unblocked	2	No	Yes	No
Fixed Unblocked	2	Yes	No	No
Variable Blocked	1	No	Yes	Yes*
Variable Blocked	1	Yes	No	No
Variable Unblocked	1	No	No**	No
Variable Unblocked	1	Yes	No	No
Undefined	1	No	No	No
Undefined	1	Yes	No	No

*Output files only.
**Required if read-backward is specified.

●Figure 26. Summary of Index Register Requirements

You can use the inquiry-request functions of the printer-keyboard to temporarily suspend the processing of a job in order to load and execute a program cataloged in the core-image library. Such a program is referred to as an inquiry program. You call an inquiry program into main storage by pressing the printer-keyboard Request key and entering the program name on the keyboard. An installation is not limited to a single inquiry program; any number of programs can be written and stored in the core-image library.

For a program that is to be interrupted by an inquiry request, you must enter the specification MAINPRG=YES in the DTFBG statement. If you want to execute a program as an inquiry program, you must specify INQPRG=YES in the DTFBG statement, and you must use a Monitor that contains routines for inquiry interrupts. (For a description of this Monitor, refer to the SRL publication IBM System/360 Model 20, Disk Programming System, System Generation and Maintenance, Form C33-6006.)

The inquiry program specifies the operations to be performed, using the inquiry record (if any) and any other files that are necessary. You can request an inquiry during the processing of a wide variety of programs. Note that if mainline programs are processing magnetic tape or card files that are also accessed by the inquiry program, it is probable that the interrupted program cannot be restored to its original status. You cannot process tape files with standard labels in an inquiry program.

Special file protection routines are included in the IOCS Open and Close routines for disk files. These routines are described under File Protection below.

A recommended technique for disk files is to reserve at least two logical unit blocks (LUBs) for the exclusive use of inquiry programs. Since they are not used by the mainline programs, their contents are not altered by job control statements submitted during a job stream.

The inquiry program, like any other program, requires that label information be supplied in order to open disk files to be used in the program. The disk files, of course, must be on-line at the time the inquiry is requested. As an inquiry program is called via the printer-keyboard, i.e., it is not preceded by a separate job control run, permanent labels should be used for inquiry programs. If you want to use temporary labels for an inquiry program, you must provide the required job control information with the job control information for the mainline program that is to be interrupted by the inquiry program. (For details concerning the use of permanent labels, refer to the SRL publication IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006.)

Inquiry Record

When the operator wishes to enter an inquiry, he presses the Request key on the printer-keyboard. If the current program can be interrupted, the message 'ENTER PROGNAME' is printed unless a Monitor with the option INQMSG=NO is used. The operator must enter the name of the desired inquiry program and press EOT. If an inquiry record is processed by the program, i.e., a Monitor with the INQIPT option is used, the operator must then enter the data and again press EOT. If the program does not use an inquiry record, the EOT key must nevertheless be pressed to signify the completion of input.

Since the inquiry record is read into the input area INQIPT in the Monitor at the time an inquiry request is submitted, you do not have to include a READ macro instruction for this record in a program written in Assembler/IOCS language. Nor do you specify a DTFPK file definition statement for the record. Instead, you must issue an IQIPT macro instruction to determine the location of the input area in the Monitor. The IQIPT macro instruction loads the address of the input area INQIPT into register 8, refer to the IQIPT Macro Instruction) You can then access the inquiry record contained in that area. The length of the inquiry record is limited to the length of the input area INQIPT. If you want to process additional printer-keyboard files in an inquiry program, you must define those files with a DTFPK file definition statement and must issue the necessary imperative macro instructions as described in the section Instructions for Processing Printer-Keyboard Files.

The advantage of submitting data as an inquiry record is that no printer-keyboard IOCS is required and that the processing of the mainline program continues while the operator enters the data.

Monitor I/O Areas

If the INQIPT option is used, the Monitor must contain an input area to accommodate the inquiry record in main storage and make it available to the inquiry program. The output area for printer-keyboard records in an inquiry program may be located in the problem program itself, or in an area immediately following the Monitor in main storage. The latter area (INQOPT) is allocated at the time the Monitor is generated. (Refer to the SRL publication IBM System/360 Model 20, Disk Programming System, System Generation and Maintenance, Form C33-6006).

The Monitor output area for the printer-keyboard is of advantage only in a Model 20, Submodel 5. By specifying an output area outside the problem program, it is possible to overlap the printing of the last output record on the printer-keyboard with the roll-in and processing of the mainline program. Note that overlapping is achieved only if no part of the printer-keyboard output area is overlaid by the inquiry program or the interrupted mainline program. Both should be loaded above this area.

In a Model 20 system that has no overlap feature a printer-keyboard output area located outside the problem program is not advantageous. In this environment, fewer bytes of main storage are required if each inquiry program contains its own printer-keyboard output area, and if the mainline programs are loaded immediately behind the Monitor.

Opening Disk Files

When disk files are opened in an inquiry program

- all OPEN macro instructions for disk files must be given together (preferably at the beginning) in an inquiry program, and
- all OPEN macro instructions for disk input files must be given before the OPEN macro instructions for disk output files.

Error in Inquiry Program

If an irrecoverable error occurs in an inquiry program, the mainline program may be re-entered by transferring control to EOJ. This is accomplished by entering (on the CPU console) the main-storage address X'00C2' into register 3.

FILE PROTECTION

For disk files, special routines generated for the OPEN and CLOSE macro instructions provide for protection of files. These routines are only included in the Open and Close routines of programs that are assembled with a DTFBG statement and the operands MAINPRG=YES and/or INQPRG=YES. If a program can be run as either a mainline program or an inquiry program, the Open/Close routines for disk files are prepared/changed at object time to process disk files according to the requirements of a mainline or inquiry program.

The protection for disk files during inquiries is also provided for by RPG. A mainline program written in Assembler/IOCS language can be interrupted by an inquiry program written in RPG, and vice versa.

No protection is provided for card and magnetic tape files. Therefore, you must carefully evaluate the use of card and magnetic tape files in an inquiry program if the same files are being processed in the interrupted mainline program. If standard labels are used for magnetic tape files, you cannot process these files in an inquiry program.

Mainline Open and Close Routines

If a program is run as a mainline program that is to permit interrupts by inquiry programs, the Open and Close routines for all disk input and output files are expanded to perform the following functions:

1. Open routines:
   a. postpone any interruptions (of the mainline program) initiated by inquiry requests until the end of the Open routines has been reached, i.e., until the label processing performed by the Open routines is completed.
   b. move the file-protection switches from the DTF block of the disk file being opened to the format-1 label.

File protection switches are initially set in the DTF blocks for all disk files defined in a program assembled with MAINPRG=YES in the DTFBG statement. These switches are moved to the format-1 label to indicate the status of the file being processed in a mainline program. Whether the file can be accessed by an inquiry program that interrupts mainline processing, depends on the conditions shown in Figure 27.

2. Close routines:

   a. prevent any mainline-program interruption (initiated by an inquiry request) during execution of the Close routines, i.e., during label processing.

   b. turn off the file-protection switches in the format-1 label to indicate that the mainline program has finished processing the file.

Note: A mainline program that opens a disk file must also close it in order to turn off the protection switches, i.e., to ensure that it is not protected from access by later inquiry programs. If disk files opened in a mainline program have not been closed (i.e., an error halt occurred and the mainline program was discontinued), you should prepare a dummy mainline program which opens and closes the affected disk files.

Inquiry Open Routines

If a program is executed as an inquiry program, the Open routines for all disk input and output files are expanded to perform the following functions:

1. Open routines:

   a. test the file protection switches in the format-1 label.

      If these switches indicate that the file is being processed by a mainline program, the inquiry program may still access the file. However, if the inquiry program specifies operations (e.g., UPDATE, ADD, or LOAD) that may conflict with the operations being performed on the same file in the mainline program, the inquiry program is discontinued and processing of the mainline program is resumed.

      The conditions under which a file being processed in a mainline program is protected are shown in Figure 27.

   b. check the extent limits of output files being opened against the extent limits of all disk files processed by the interrupted mainline program. If extent overlay is detected, the inquiry program is discontinued and processing of the mainline program is resumed.

Inquiry-Mainline Program - Same File

Indexed-Sequential Files

	Inquiry Program				
Mainline Program	IOROUT=LOAD	IOROUT=ADD/ADDRTR without UPDATE	IOROUT=RETRVE with UPDATE	IOROUT=ADDRTR with UPDATE	IOROUT=RETRVE without UPDATE
IOROUT=LOAD	P	P	P	P	P
IOROUT=ADD/ ADDRTR without UPDATE	P	P	A	P	A
IOROUT=RETRVE with UPDATE	P	P	P	P	A
IOROUT=ADDRTR with UPDATE	P	P	P	P	A
IOROUT=RETRVE without UPDATE	P	P	A	P	A

Sequential Disk or Direct-Access Files

	Inquiry Program		
Mainline Program	TYPEFLE=OUTPUT	TYPEFLE=UPDATE	TYPEFLE=INPUT
TYPEFLE=OUTPUT	P	P	P
TYPEFLE=UPDATE	P	P	A
TYPEFLE=INPUT	P	A	A

P - The file that is being processed in the mainline program is protected from being accessed in the inquiry program.
A - Inquiries are allowed.

Figure 27. Protection of Files During Inquiry Operations

If a printer-keyboard is attached to your system, you can request an interrupt that allows you, by means of an ATENT routine, to modify the main routine, set indicators and switches, and retrieve information from, or supply information to, the main routine. You interrupt the program phase that is currently executed by pressing the Request key on the printer-keyboard. Control is then transferred to your ATENT routine whose entry point is provided by the ATENT macro instruction. You must issue an ATENT macro instruction as the first and a RETRN macro instruction as the last statement in your ATENT routine.

All files used by the ATENT routine must be defined together with the other files at the beginning of the program. When using IOCS imperative macro instructions in the ATENT routine you must observe the following restriction: You are not allowed to use macro instructions pertaining to a device for which macro instructions have already been issued in the main routine. For instance, if the main routine contains card routines, you must not issue card macro instructions in the ATENT routine. The same rule applies to the printer, magnetic tape and disk. An exception is the printer-keyboard. If you have printer-keyboard output macro instructions in the main routine, you may issue macro instructions for printer-keyboard input in the ATENT routine; and vice versa.

If you include an ATENT routine in your program, you can execute the program only as a mainline program that does not allow inquiry interrupts, i.e., you must specify ATENT=YES in the DTFBG statement.

Note that the registers are saved and restored by the ATENT and RETRN macro instructions. However, the contents of the registers are not known at the time the interrupt occurs.

ATENT MACRO INSTRUCTION

The ATENT macro instruction has the following format:

Name	Operation	Operand
[name]	ATENT	

The operand field must be blank. Enter ATENT in the operation field.

The ATENT macro instruction must precede your ATENT routine to which it provides the entry point. The ATENT macro instruction exchanges PSW addresses and saves all registers. You can issue the ATENT macro instruction only in a mainline program that does not allow inquiry interrupts. Therefore, you must specify ATENT=YES in the DTFBG statement (see DTFBG statement under the section Begin and End Definitions.)

RETRN MACRO INSTRUCTION

The RETRN macro instruction has the following format:

Name	Operation	Operand
[name]	RETRN	

The operand field must be blank. Enter RETRN in the operation field.

Issue the RETRN macro instruction as the last statement in your ATENT routine. The RETRN macro instruction returns control to the point of invocation after restoring all registers and the PSW address. When you use the RETRN macro instruction you must specify ATENT=YES in the DTFBG statement to indicate that the program is to be run as a mainline program that does not allow inquiry interrupts.

Control Statements

Depending upon the types of files to be processed, you may have to supply control statements that provide information to the IOCS. Control statements will be read by the Job Control program before the object program is loaded. These statements provide the IOCS with information that is necessary to (1) check the label(s) of an input file, (2) create the label(s) for an output file, or (3) define the limits of the disk storage area(s) for a disk file.

Two control statements must be supplied for each labeled tape file. At least three control statements must be supplied for each disk file (a minimum of four for indexed-sequential files). The types of control statements are:

Volume Control Statement. This statement specifies the symbolic unit to be used and the name of the file.

Tape Label Control Statement. This statement provides information for checking and/or creating tape labels for a file.

Disk Label Control Statement. This statement provides information for checking and/or creating disk labels for a file.

Extent Control Statement. This statement provides information about the disk extents to be used and specifies the symbolic unit to be used. You must provide one statement for each extent.

Each magnetic tape or disk file requires one Volume control statement and one Tape (Disk) Label control statement. Extent control statements are required for disk files only. The format and contents of each of these control statements are described in the SRL publication IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006.

The IOCS provides an error recovery routine for each I/O device. The actions taken when an error occurs are described below.

Card, Printer, and Printer-Keyboard Equipment Errors

When errors, such as feed checks, occur on card, printer, and printer-keyboard equipment, the IOCS stops the execution of the program to allow the machine operator to take corrective action. An error indication is displayed on the console to identify the type of error and to indicate the required operator action.

Tape Error Routines

If a tape read error occurs, the physical IOCS (PIOCS) routines cause the tape to be backspaced and reread 100 times before the block is considered to be incorrect. If an error cannot be corrected, the PIOCS routines indicate this fact either to the IOCS or to the operator depending on the specifications in, or omission of, the ERROPT detail entry. (Refer to the description of the detail entries ERROPT and ERRIO). Indication to the operator is made by a display of an error code on the CPU console. This display indicates the type of error and the associated device address.

If a tape write error occurs, the error tape is backspaced to the beginning of the block, a gap is skipped, and the block is rewritten. If necessary, this procedure is repeated by backspacing to end of last gap, skipping another gap, and writing. The block is rewritten 9 times before a tape write error is indicated.

Disk Error Routines

When a disk read or write error occurs, the IOCS rereads or rewrites the block a standard number of times before the block is considered as incorrect. If a read or write error cannot be corrected by the error routines, this is indicated to the problem program. Refer to the ERROPT, ERRBYTE, ERRINF, ERRIO, and DERREX entries in the appropriate file definition statement.

Note. Normally, the records are checked after they have been written. However, if VERIFY=NO is specified in the definition statement for a particular file, checking is omitted.

Language Compatibility

The DPS IOCS for the System/360 Model 20 is closely patterned after the Basic Programming Support IOCS and the Basic Operating System IOCS. Because the DPS IOCS is designed to support card, printer, printer-keyboard, magnetic tape and disk I/O devices that are unique to the Model 20 and in order to achieve optimum performance of all devices, some macro instructions and file definition entries are not identical to those of the other systems. Therefore, if you anticipate transition from Model 20 to other models of System/360, you should be aware that programs using the DPS IOCS require some modification before they can be processed by the other System/360 Assemblers.

All tape data sets created under control of the DPS IOCS are fully upward compatible. Disk data sets that have been created on the IBM 2311, Model 11 and 12, cannot be processed on the IBM 2311, Model 1, and vice versa.

Appendix A. Summary of File Definition Statements

The tables in Figures 28 through 36 show all of the file definition statements and the detail entries available.

The tables are in the same order as the discussions of the file definition statements:

DTFBG - Figure 28; DTFEN - Figure 29; DTFSR - Figure 30; DTFPK - Figure 31;
DTFLC - Figure 32; DTFMT - Figure 33; DTFSD - Figure 34; DTFDA - Figure 35;
DTFIS - Figure 36.

	Operand		
Operation	Keyword	Allowable Specification	Remarks
DTFBG			Applies to all file types.
	ATENT	YES	Required if program includes ATENT routine.
	INQPRG	YES	Required for inquiry programs.
	MAINPRG	YES	Required if program is executed as mainline program that permits interrupts by inquiry requests.

•Figure 28. DTFBG Statement and Associated Detail Entries

	Operand		
Operation	Keyword	Allowable Specification	Remarks
DTFEN			Mandatory for all files.
	OVLAY		Required if overlay of Open and Close routines for magnetic tape and disk files is desired.

•Figure 29. DTFEN Statement and Associated Detail Entry

Operation	Operand Keyword	Operand Allowable Specification	Applies to 2560	2520 Card Read-Punch	2520 Card Punch	1442 Mod.5 Punch	2501	2203	1403	Remarks
DTFSR			M	M	M	M	M	M	M	Always first card, may include detail entries from column 16 to column 71.
	BINARY	YES	O*	O*			O			*Only for simple files.
		INPUT	O	O						Only for combined files.
	BLKSIZE	length of simple file I/O area in bytes	O	O	M	M	M	M	M	Indicates length of area specified by IOAREA1 and IOAREA2 entries.
	CONTROL	YES	O	O	O			O	O	Required if CNTRL is issued for a file.
	CRDPRA	name of card-print area	O							
	CRDPRLn	length of card-print area in bytes	O							n in the keyword is a print head number.
	DEVICE	MFCM1	M*							Mandatory detail entry for card and printer files.
		MFCM2								
		CRP20		M						
		PUNCH20			M					*One of the possible specifications must be entered.
		PUNCH42				M				
		READ01					M			
		PRINTER							M	
		PRINTLF						M*		
		PRINTUF								
	EOFADDR	name of end-of-file routine	M*	M*			M			*Only for input and combined files.
	INAREA	name of combined file input area	M*	M*						*Combined files only.

M = Mandatory O = Optional

Figure 30. Summary of the DTFSR Statement and the Associated Detail Entries, Part 1 of 3

Operation	Operand: Keyword	Operand: Allowable Specification	Applies to: 2560	Applies to: 2520 Card Read-Punch	Applies to: 2520 Card Punch	Applies to: 1442 Mod.5 Punch	Applies to: 2501	Applies to: 2203	Applies to: 1403	Remarks
DTFSR	INBLKSZ	length of combined file input area in bytes	M*	M*						*Combined files only
	IOAREA1	name of first I/O area	M	M	M	M	M	O*		*Entry required for 2203 only when dual-feed carriage is used.
	IOAREA2	name of second I/O area					O			Can be used if 2501, Model A2, is used in overlap mode.
	OUAREA	name of combined file output area	M*	M*						*Combined files only.
	OUBLKSZ	length of combined file output area in bytes	M*	M*						*Combined files only.
	OVERLAP	NO	O	O	O	O	O			If omitted, file is processed in overlap mode.
	PFORMTn	xxyy	O	O						Indicates that the field is to be checked for blanks from columns xx to yy prior to punching.
	PFXIT	name of routine used when PFORMTn test fails	O	O						
	PRINTOV	YES						O	O	Required if PRTOV is issued for file.
	RFORMTn	xxyyz	O	O			O			Indicates that input cards are to be checked for numerics or blanks from columns xx to yy

M = Mandatory O = Optional

Figure 30. Summary of the DTFSR Statement and the Associated Detail Entries, Part 2 of 3

Operation	Operand: Keyword	Operand: Allowable Specification	Applies to: 2560	2520 Card Read-Punch	2520 Card Punch	1442 Mod.5 Punch	2501	2203	1403	Remarks
DTFSR	RFXIT	name of routine used when RFORMTn test fails	O	O			O			
	SEQNCE	xxyy	O	O			O			Indicates sequence check of input cards desired from col. xx to yy.
	SEQXIT	name of routine used when SEQNCE test fails	O	O			O			Must be specified when SEQNCE is specified.
	TYPEFLE	INPUT					M			*One of the three specifications must be entered.
		OUTPUT	M*	M*	M	M		M	M	
		CMBND								
	WORKA	YES	M	M	M	M	M	M	M	Mandatory for all card and printer files.

M = Mandatory O = Optional

Figure 30. Summary of the DTFSR Statement and the Associated Detail Entries, Part 3 of 3

Name	Operation	Operand		Remarks
		Keyword	Allowable Specification	
Filename	DTFPK			Applies to printer-keyboard files only.
		BLKSIZE	Length of largest record: Input 2 to 511 Output - 1 to 511	Mandatory for input and output files. Must be large enough to accommodate maximum-size record. If line-counter table is used for an output file, length must not exceed 125.
		CONTROL	YES	Optional for output files. Required if CNTRL macro instruction is issued for the file.
		EOFADDR	Name of end-of-file routine	Mandatory for input files. Name of your routine to which control is given when end-of-file indicator /* is entered (detected by WAITF macro).
		IOAREA	Name of I/O area in the problem program	Name of your I/O area. If omitted for output file, output area of Monitor is used. Must be as large as BLKSIZE.
		LCTABLE	YES	Optional for output files. Required if output control by skipping is desired. If used, line-counter table must be defined by a DTFLC.
		PRINTOV	YES	Optional for output files. Required if PRTOV macro instruction is issued for the file. If used, line-counter table must be defined by a DTFLC.
		RECSIZE	Register (8-13) or symbolic name of a register (in parentheses)	In the specified register, provide the number of characters to be printed in the next record.
		TYPEFLE	INPUT or OUTPUT	Designates that this DTFPK is for an input or an output file on the printer-keyboard.
		WORKA	YES	Mandatory for output files. Length equal to BLKSIZE. If entry omitted a warning is given.

•Figure 31. Summary of the DTFPK Statement and the Associated Detail Entries

Name	Operation	Operand	Remarks
	DTFLC	Formsize,E1,E2,...,E48	This statement is required to define a line-counter table for simulating carriage control. The entries represent the length of the form, and channels associated with particular lines. It may be omitted if no skipping is used for the output file.

Figure 32. Summary of DTFLC Statement

Operation	Operand		Remarks
DTFMT			Applies to magnetic tape files only.
	ALTTAPE	SYSIPT SYSOPT SYSnnn	Required for multi-volume files using 2 tape drives.
	BLKSIZE	length of file I/O area in bytes	Mandatory for all magnetic tape files. Indicates length of area specified by the IOAREA1 entry.
	CKPTREC	YES	Required to read tapes containing interspersed checkpoint records.
	CONTROL	YES	Required if a CNTRL macro is issued for the file.
	DEVADDR	SYSPT SYSOPT SYSnnn	Mandatory for all magnetic tape files. SYSIPT, SYSOPT, or SYSnnn are the symbolic addresses to be used when processing a magnetic tape file.
	EOFADDR	name of end-of-file routine	Mandatory for input files only. Not used for output files.
	ERRIO	name of user-defined 2-byte area	Use only if ERROPT=name and/or WLRERR=name and two I/O areas specified.
	ERROPT	IGNORE	Ignored.
		SKIP	Error block skipped.
		name of error routine	Return to IOCS via register 14.
	FILABL	STD	Standard labels.
		NSTD	Non-standard labels.
		NO	No labels.

Figure 33. Summary of the DTFMT Statement and the Associated Detail Entries, Part 1 of 2

Operation	Operand		Remarks
	Keyword	Allowable Specification	
DTFMT	IOAREA1	name of user-defined area	Mandatory for all magnetic tape files.
	IOAREA2	name of user-defined area	Requires IOREG or WORKA entry. Optional.
	IOREG	register number from 8 to 13 or symbolic name of register (both in parentheses)	Required when blocked records are processed in the I/O area. Required if IOAREAR but not work area specified.
	LABADDR	name of user routine	Return to main routine by issuing a LBRET macro.
	READ	FORWARD	If omitted, IOCS assumes forward reading.
		BACK	
	RECFORM	FIXUNB FIXBLK VARUNB VARBLK UNDEF	Entry may be omitted if record format is fixed unblocked.
	RECSIZE	number of bytes in one record or number of register indicating record length in number of bytes or symbolic name of register	Required if fixed length blocked or undefined record format is specified (number of bytes in one record).
	REWIND	UNLOAD	If omitted, the tape is rewound but not unloaded on OPEN or CLOSE, end-of-volume, or end-of-file condition.
		NORWD	
	TPMARK	NO	Optional. Applies to unlabeled tape output files.
	TYPEFLE	INPUT	Mandatory for all magnetic tape files.
		OUTPUT	
	VARBLD	number or symbolic name of register (in parentheses) indicating available bytes	Required if variable-length blocked records are built in the output area.
	WLRERR	name of user routine	
	WORKA	YES	Required for blocked records, or if ICAREA2 without IOREG specified.

Figure 33. Summary of the DTFMT Statement and the Associated Detail Entries, Part 2 of 2

Operation	Operand		Remarks
	Keyword	Allowable Specification	
DTFSD			Applies to sequential disk files only.
	BLKSIZE	length of block	Mandatory entry. Indicates length of one block of records. Length must be an integer multiple of RECSIZE.
	COMROUT	YES	One common routine generated for all input files for which entry is specified. Same holds analogously for update and output files.
	CONTROL	YES	Required if a CNTRL macro instruction is issued for the file.
	DEVICE	DISK11F	Specifies IBM 2311, Model 11 or 12. Warning issued in case of error.
	DSKXTNT	maximum number of extents in any one volume for the file	If omitted, the IOCS assumes three extents for the file. Maximum is 99.
	DTAREX	name of user's routine	Optional entry for output files. If not specified, job is discontinued.
	EOFADDR	name of user's end-of-file routine	Mandatory for input files only. Not required for output files.
	ERRIO	name of user-defined 2-byte area	Use only if ERROPT=name and two I/O areas specified.
	ERROPT	SKIP	Error block is to be skipped.
		name of error routine	Return to program via register 14.
	IOAREA1	name of a user-defined area	Mandatory for all disk files. Length of area must be 270 bytes or integer multiple thereof. Equal or greater than BLKSIZE.
	IOAREA2	name of user-defined area	Requires IOREG or WORKA entry. Optional. Length must be 270 or integer multiple thereof.
	IOREG	number or symbolic name of a register (in parentheses)	Required when blocked records are processed in the I/O area. Required if IOAREA2 but no WORKA specified.
	RECFORM	FIXUNB	If not specified, FIXUNB is assumed.
		FIXBLK	
	RECSIZE	number of bytes in one record	Mandatory only for files containing blocked records. If not specified for unblocked records, RECSIZE=BLKSIZE. Maximum record size is 4096 bytes.

Figure 34. Summary of DTFSD Statement and the Associated Detail Entries, Part 1 of 2

Operation	Operand		Remarks
	Keyword	Allowable Specification	
DTFSD	TYPEFLE	INPUT	Mandatory for all disk files.
		OUTPUT	
	UPDATE	YES	Required if records are to be updated.
	VERIFY	NO	Required if write checking not desired.
	WORKA	YES	Required if a record is to be processed in a work area. Required if IOAREA2 but no IOREG specified.

Figure 34. Summary of the DTFSD Statement and the Associated Detail Entries, Part 2 of 2

Operation	Operand		Remarks
	Keyword	Allowable Specification	
DTFDA			Applies to direct-access disk files only.
	ADRTEST	NO	Address is not checked for validity nor against extent limits specified in XTENT statement.
	BLKSIZE	length of I/O area	Mandatory detail entry. Indicates length of one block. Maximum block size is 16,200 bytes.
	CONTROL	YES	Required if a CNTRL macro instruction is issued for the file.
	DEVICE	DISK11F	Specifies IBM 2311, Model 11 or 12. Warning issued in case of error.
	DSKXTNT	maximum number of extents in any one volume for the file.	If omitted, the IOCS assumes three extents for the file.
	ERRBYTE	name of a user-defined area	Mandatory detail entry. The IOCS will make error indications available in these bytes.
	IOAREA1	name of a user-defined area	Mandatory detail entry. Length of area must be 270 bytes or integer multiple thereof.
	READID	YES	Required if problem program uses READ.
	SEEKADR	name of a user-defined area	Mandatory detail entry. The IOCS obtains address of desired disk location from this area.
	TYPEFLE	INPUT	Mandatory for all direct-access files.
		OUTPUT	Standard labels are checked.
	VERIFY	NO	Required if write-checking is not desired.
	WRITEID	YES	Required if problem program uses WRITE.

Figure 35. Summary of the DTFDA Statement and the Associated Detail Entries

Operation	Operand		Type of Processing								Remarks
					Retrieve			Add-Retrieve			
	Keyword	Allowable Specifications	Load and Extend	Add	Sequential	Random	Random-Sequential	Sequential	Random	Random-Sequential	
DTFIS			M	M	M	M	M	M	M	M	Applies to indexed sequential files only.
	ADAREX	name of routine		M				M	M	M	Used when overflow area is full.
	ALTREX	name of routine	M	O				O	O	O	Used to perform or avoid additions to the last track of a file.
	CYLOFL	number of cylinder overflow tracks	O	O				O	O	O	Required if cylinder overflow areas are used.
	CYNDEX	name of routine	M								Used when cylinder index area is full.
	DERREX	name of routine	M	M	M	M	M	M	M	M	Used when a disk error occurs.
	DEVICE	DISK11F	O	O	O	O	O	O	O	O	Warning issued in case of error or if omitted.
	DPCRCD	YES		O				O	O	O	Used for duplicate records.
	DSKXTNT	maximum number of extents to be used	M	M	M	M	M	M	M	M	The maximum number is 99.
	DTAREX	name of routine	M								Used when the prime data area is full.
	DUPREX	name of routine	M	M				M	M	M	Used when duplicate key is detected.
	EOFADDR	name of end-of-file routine			M		M	M		M	Mandatory for input files. Not required for output files.
	ERRINF	YES	O	O	O	O	O	O	O	O	If not specified, no logical error information is supplied.

M = Mandatory O = Optional

•Figure 36. Summary of the DTFIS Statement and the Associated Detail Entries, Part 1 of 3

Operation	Operand: Keyword	Operand: Allowable Specifications	Type of Processing: Load and Extend	Type of Processing: Add	Type of Processing: Retrieve: Sequential	Type of Processing: Retrieve: Random	Type of Processing: Retrieve: Random-Sequential	Type of Processing: Add-Retrieve: Sequential	Type of Processing: Add-Retrieve: Random	Type of Processing: Add-Retrieve: Random-Sequential	Remarks
DTFIS	IOAREAL	name of output area	M	M				M	M	M	Length of area must be 270 bytes or integer multiple thereof.
	IOAREAR	name of I/O area				M	M		M	M	Length of area must be 270 bytes or integer multiple thereof.
	IOAREAS	name of I/O area			M		M	M		M	Length of area must be 270 bytes or integer multiple thereof.
	IOREG	number of any register from 8 to 13 in parentheses or symbolic name equated to register			O	O	O	O	O	O	Required for processing blocked records in the I/O area.
	IOROUT	LOAD	M								
		ADD		M							
		RETRVE			M	M	M				
		ADDRTR						M	M	M	
	KEYARG	name of user-defined area			O	M	M	O	M	M	IOCS obtains key of desired record from this area.
	KEYLEN	number of bytes in the key	M	M	M	M	M	M	M	M	Maximum length is 60 bytes.
	KEYLOC	high-order position of key field	M	M	M	M	M	M	M	M	First byte of record is counted as byte 1.
	NRECDS	number of records in one block	M	M	M	M	M	M	M	M	For unblocked records, use a 1.
	RECFORM	FIXUNB	O	O	O	O	O	O	O	O	Specification of RECFORM is compulsory. Only one of the operands is permitted at a time.
		FIXBLK	O	O	O	O	O	O	O	O	

M = Mandatory O = Optional

Figure 36. Summary of the DTFIS Statement and the Associated Detail Entries, Part 2 of 3

Operation	Operand		Type of Processing								Remarks
					Retrieve			Add-Retrieve			
	Keyword	Allowable Specifications	Load and Extend	Add	Sequential	Random	Random-Sequential	Sequential	Random	Random-Sequential	
DTFIS	RECSIZE	length of one record in bytes	M	M	M	M	M	M	M	M	
	RTRVEX	name of routine			O	M	M	O	M	M	Used when desired record is not found.
	SQCHEX	name of routine	M								Used when a record is out of sequence.
	TYPEFLE	RANDOM				M			M		Indicates type of retrieval. IOROUT specifies RETRVE or ADDRTR.
		SEQNTL			M			M			
		RANSEQ					M			M	
	UPDATE	RANDOM				O	O		O	O	Indicates type of updating.
		SEQNTL			O		O	O		O	
		RANSEQ					O			O	
	VERIFY	NO	O	O	O	O	O	O	O	O	Required if write checking is not desired.
	WORKA	name of of work area		O				O	O	O	Required if FIXBLK is specified. Must precede the area specified by IOAREAL.
	WORKL	name of a work area	M	M				M	M	M	
	WORKR	name of a work area				O	O		O	O	
	WORKS	YES			O		O	O		O	

M = Mandatory O = Optional

Figure 36. Summary of the DTFIS Statement and the Associated Detail Entries, Part 3 of 3

Appendix B. Summary of Imperative Macro Instructions

The table in Figure 37 shows all IOCS macro instructions, including the possible operands, that are available.

The table in Figure 38 shows all imperative macro instructions used in conjunction with indexed-sequential files. Each macro instruction is marked with (M) and (O) to indicate whether it is mandatory or optional for the various types of file processing.

Operation	Operand	Remarks
CLOSE	file1,file2,...,file16	Up to 16 files may be closed with one CLOSE macro instruction.
CNTRL	filename,BSF	Backspace file.
	filename,BSR	Backspace record.
	filename,ERG	Erase gap.
	filename,FSF	Forward space file.
	filename,FSR	Forward space record.
	filename,REW	Rewind tape.
	filename,RUN	Rewind and unload tape.
	filename,SEEK	Search for specified disk location.
	filename,SK,n,m (n = 1,2,..., or 12) (m = 1,2,..., or 12)	n causes immediate skip to the specified channel. m causes skip to specified channel after printing.
	filename,SP,n,m (n = 0,1,2, or 3) (m = 0,1,2, or 3)	n causes the specified number of lines to be spaced immediately. m causes the specified number of lines to be spaced after printing.
	filename,SS,n (n = stacker number)	Select stacker of a multi-stacker I/O device.
CNVRT	seekfield,packedfield	Converts packed decimal address into format of seek field used in direct access.
CRDPR	,workarea,printarea	Card print. DTFSR statement must include CRDPRA and CRDPRLn detail entries.
ENDFL	filename	End loading or extending of an indexed-sequential file.

Figure 37. Summary of Imperative Macro Instructions, Part 1 of 3

Operation	Operand	Remarks
EOM	filename	Enter overlap mode. Applies to combined files for which a previous LOM has been given.
ESETL	filename	End sequential processing of an indexed-sequential file.
FEOV	filename	Force end of volume. Applies to multi-volume tape files.
GET	filename,workarea	The second operand, including the comma preceding it, must be omitted if no work area has been specified.
LBRET	n (n = 1 or 2)	Label return. Required for return to the IOCS from LABADDR routines.
LOM	filename	Leave overlap mode. Applies to combined files for which overlap mode has been specified.
OPEN	file1,file2,...,file16	Up to 16 files may be opened with one OPEN macro instruction. (However, only one printer-keyboard input and one printer-keyboard output file can be specified in a single program.)
PRTOV	filename,n,address (n = 9 or 12)	Branch on print overflow. n specifies the channel indicator to be tested. An automatic skip to channel 1 occurs if the last operand (address of user routine) is omitted.
PUT	filename,workarea	The second operand, including the comma preceding it, must be omitted if no work area has been specified.
READ	filename	Applies to printer-keyboard files only.
	filename,ID	Applies to direct-access files only.
	filename,KEY	Applies to indexed-sequential files only.
RELSE	filename	Release current block of tape input file.
SETFL	filename	Prepare for loading or extending an indexed-sequential file.

Figure 37. Summary of Imperative Macro Instructions, Part 2 of 3

Operation	Operand	Remarks
SETL	filename,BOF	Prepare for sequential processing of an indexed-sequential file beginning with the first record.
	filename,KEY	Prepare for sequential processing of an indexed-sequential file beginning with the record having the specified key.
	filename,GKEY	Prepare for sequential processing of an indexed-sequential file beginning with the record whose key is equal to or greater than the specified key.
TRUNC	filename	Truncate current block of tape output file.
WAITC		Wait for the completion of pending card and printer I/O operations.
WAITF	filename	Wait for a printer-keyboard input operation or a disk operation to end.
WRITE	filename,ID	Applies to direct-access files only.
	filename,KEY	Replace a record retrieved from an indexed-sequential file.
	filename,NEWKEY	Place a new record into an indexed-sequential file.

Figure 37. Summary of Imperative Macro Instructions, Part 3 of 3

Indexed-Sequential Processing Macros										
Operation	Operand	Type of Processing								Remarks
		Load	Add	Retrieve			Add-Retrieve			
				SEQNTL	RANDOM	RANSEQ	SEQNTL	RANDOM	RANSEQ	
CLOSE	filename1, filenamen	M	M	M	M	M	M	M	M	max. of 16 operands
ENDFL	filename	M								same name as in SETFL
ESETL	filename			M		M	M		M[2]	same name as in SETL
GET	filename			M		O	O		O	
	filen,workn			M		O	O		O	
OPEN	filename1, filenamen	M	M	M	M	M	M	M	M	max. of 16 operands
PUT	filename			O		O	O		O	
	filen,workn			O		O	O		O	
READ	filen,KEY				M	O		O	O	
SETFL	filename	M								
SETL	filen,BOF filen,KEY filen,GKEY			M		O[1]	O[1]		O[1,2]	
WAITF	filename		M		M	O		M	O	
WRITE	filen,NEWKEY	M	M				O	O	O	
	filen,KEY				O	O		O	O	

M = Mandatory
O = Optional
[1] Mandatory if sequential processing is used. [2] A READ or WRITE macro instruction must not be issued between SETL and ESETL referring to the same file.

Figure 38. Summary of Imperative Macro Instructions for Indexed-Sequential Files

Appendix C. Summary of Monitor Macro Instructions

Five macro instructions can be used in Assembler language programs to communicate with the Monitor program. Two of them (COMRG and MVCOM) refer to the communication region.

Two (FETCH and EOJ) request functions of the Fetch routine. One refers to the printer-keyboard input area. Figure 39 shows a summary of the formats and functions of the Monitor macro instructions.

Name	Operation	Operands	Functions
	COMRG		Places the address of the first byte of the communication region in register 8. Enables the communication region to be referred to by relative addressing.
	MVCOM	first-byte, number, address	Modifies information in the user areas of the communication region (bytes 12-23).
	FETCH FETCH	phase name	Disk-resident system: requests the loading of another phase or subphase into main storage from the core-image library. Card-resident system: loads the phase that physically follows the phase containing FETCH.
	EOJ		Disk-resident system: indicates to the Monitor program that a job has been completed and that the Job Control program must be called to prepare for the next job. Card-resident system: causes system halt. To load the next Job Control deck, the operator must press START.
	IQIPT		Loads the begin address of the printer-keyboard input area into register 8. You can thus refer to the area through relative addressing. Required if you use inquiry record.

Figure 39. Summary of Monitor Macro Instructions

Appendix D. Programming Examples

This appendix provides you with a number of examples that illustrate the use of the IOCS for the various types of files.

Example	Type of File
1 and 1.1	Card files (2520 Card Read-Punch), Overlap
2	Card and printer files (2560 MFCM, 2501 Card Reader, Printer)
3	Printer-keyboard input file and indexed-sequential file
4	Printer-keyboard output file and sequential disk file
5 and 5.1	Tape output file with user labels and card file (2560 MFCM)
6	Tape update file, card file (2501) and printer file
7	Sequential disk output file and card file
8	Sequential disk update file and printer file
9	Sequential disk file
10	Direct-access file (read/write) and card file (2501)
11	Indexed-sequential load file and card file (2501)
12	Indexed-sequential file (reorganization) and sequential disk file
13	Indexed-sequential add file and card file (2560 MFCM)
14	Indexed-sequential file (random/sequential ADDRTR and UPDATE) and card file (2501)
15	Indexed-sequential file and printer-keyboard file in inquiry program
16	Sequential disk, printer, and printer-keyboard input/output files - program includes ATENT routine

All examples are divided into the sections

- file definition,
- processing routine,
- exit routines, and
- definition of constants.

A description of the processing and exit routines is given separately for each example. The numbers in parentheses refer to the numbers in the left-hand margin of the individual examples. For information on the file definition statements and the definition of constants, which includes the definition of I/O and work areas, refer to the relevant sections in this publication.

Example 1 and 1.1 - Card Files (2520 Card Read-Punch)

In this example, a combined input/output file is processed on a 2520 Card Read-Punch. The card input is checked for blanks (2) in columns 10 through 15 to determine whether the record is to be updated. If no update is required, i.e., the check for all blanks failed, the next card is read. If the test is true, the record is updated (3) and the character string 'UPDATE' is punched (4) into columns 10 through 15 of the record.

It is assumed that most of the cards are to be updated. Therefore, the OVERLAP=NO entry has been omitted. However, a LOM macro instruction is issued (1) between OPEN and the first GET to cause the GET routine to work in non-overlap mode.

The IOCS performs a sequence check in columns 73 through 80 and a read-format check in columns 1 and 2. If a sequence error occurs or if columns 1 and 2 are not blank, the card is selected (5) into stacker 2. When the end-of-file card is detected, the file is closed (6) and the job is terminated.

This example requires 5940 bytes of main storage.

Example 1.1 is similar to example 1. However, this time it is assumed that only a few cards of the combined file are to be updated. Therefore, the entry OVERLAP=NO (1) is included in the file definition statement. The program is executed in non-overlap mode. This means that throughput is slower while storage requirements are reduced.

This example requires 5710 bytes of main storage.

```
                TITLE 'IOCS EXAMPLE NO 1 - CARD FILE ON 2520'
          *
          *     FILE DEFINITION
          *     ---------------
          *
                START 0000
          START EQU   *
                ORG   *+4608
                PRINT NOGEN
          *
          CARD1 DTFSR CONTROL=YES,                                      -
                      EOFADDR=EOFC1,                                    -
                      DEVICE=CRP20,                                     -
                      INAREA=AREA1,                                     -
                      INBLKSZ=80,                                       -
                      OUAREA=AREA2,                                     -
                      OUBLKSZ=80,                                       -
                      RFORMT1=01020,                                    -
                      RFXIT=RFEX1,                                      -
                      SEQNCE=7380,                                      -
                      SEQXIT=SQEX1,                                     -
                      TYPEFLE=CMBND,                                    -
                      WORKA=YES
                DTFEN
                EJECT
          *
          *     PROCESSING ROUTINE
          *     ------------------
          *
                USING START,0,1,2,3
          BEGIN EQU   *
                OPEN  CARD1
1 ------------> LOM   CARD1                 PREFERABLE IF NEARLY ALL CARDS ARE
          *                                 TO BE PUNCHED I.E. TO BE UPDATED
          LOOP1 EQU   *
                GET   CARD1,AREA3           GET IS WORKING IN NON OVERLAP MODE
                CLC   AREA3+09(6),=C'       ' CHECK IF UPDATE POSSIBLE
2 ------------> BNE   LOOP1
          *
3 ------------> MVC   AREA3+09(6),=C'UPDATE' COLUMNS 10-15 ARE UPDATED BY
          *                                 CHARACTER STRING 'UPDATE'
4 ------------> PUT   CARD1,AREA3           PUT IS WORKING IN OVERLAP MODE
                B     LOOP1
          *
          *     EXIT ROUTINES
          *     -------------
          *
          SQEX1 EQU   *                     EXIT IF SEQUENCE CHECK OCCURS
                HPR   X'F01',0              HALT AND CONTINUE AS FOR RFEX1
          RFEX1 EQU   *                     EXIT  IF TEST FOR BLANKS IN COLUMNS
          *                                 1-2 FAILS
          *                                 NO UPDATE REQUIRED
5 ------------> CNTRL CARD1,SS,2            SELECT ERROR CARD INTO HOPPER 2
                B     LOOP1
          *
          EOFC1 EQU   *                     EOF-CONDITION DETECTED
6 ------------> CLOSE CARD1
                EOJ
          *                                 TERMINATION OF JOB
          *
          *     DEFINITION OF CONSTANTS
          *     -----------------------
          *
          AREA1 DC    80C' '                INAREA FOR 2520
          AREA2 DC    80C' '                OUAREA FOR 2520
          AREA3 DC    80C' '                WORKAREA FOR 2520
          *
                LTORG
          *
                END   BEGIN
```

Example 1. Card File (2520 Card Read-Punch), Overlap

```
                TITLE 'IOCS EXAMPLE NO 1.1 - CARD FILE ON 2520'
        *
        *       FILE DEFINITION
        *       ---------------
        *
                START 0000
        START   EQU   *
                ORG   *+4608
                PRINT NOGEN
        CARD1   DTFSR CONTROL=YES,                                          -
                      DEVICE=CRP20,                                         -
                      EOFADDR=EOFC1,                                        -
                      INAREA=AREA1,                                         -
                      INBLKSZ=80,                                           -
                      OUAREA=AREA2,                                         -
                      OUBLKSZ=80,                                           -
1 ------------------> OVERLAP=NO,                                          -
                      REFORMT1=01020,                                       -
                      REXIT=REEX1,                                          -
                      SEQNCE=7380,                                          -
                      SEQXIT=SQEX1,                                         -
                      TYPEFLE=CMBND,                                        -
                      WORKA=YES
                DTFEN
                EJECT
        *
        *       PROCESSING ROUTINE
        *       ------------------
        *
                USING START,0,1,2,3
        BEGIN   EQU   *
                OPEN  CARD1
        LOOP1   EQU   *
                GET   CARD1,AREA3          GET IS WORKING IN NON OVERLAP MODE
                CLC   AREA3+09(6),=C'      ' CHECK IF UPDATE POSSIBLE
                BNE   LOOP2
        *
                MVC   AREA3+09(6),=C'UPDATE' COLUMNS 10-15 ARE UPDATED BY
        *                                  CHARACTER STRING 'UPDATE'
                PUT   CARD1,AREA3          PUT IS WORKING IN NON.OVERLAP MODE
                B     LOOP1
        LOOP2   EQU   *                    EXIT IF TEST FOR BLANKS IN COLUMNS
        *                                  10-15 FAILS I.E. THE CARD IS ALREADY
        *                                  UPDATED
                CNTRL CARD1,SS,2           SELECT CARD FOR HOPPER 2
                B     LOOP1
        *
        *       EXIT ROUTINES
        *       -------------
        *
        SQEX1   EQU   *                    EXIT IF SEQUENCE CHECK OCCURS
                HPR   X'F01',0             HALT AND CONTINUE AS FOR REEX1
        REEX1   EQU   *                    EXIT  IF TEST FOR BLANKS IN COLUMNS
        *                                  1-2 FAILS
        *                                  NO UPDATE REQUIRED
                CNTRL CARD1,SS,2           SELECT ERROR CARD INTO HOPPER 2
                B     LOOP1
        EOFC1   EQU   *                    EOF-CONDITION DETECTED
                CLOSE CARD1
                EOJ
        *                                  TERMINATION OF JOB
        *
        *       DEFINITION OF CONSTANTS
        *       -----------------------
        *
        AREA1   DC    80C' '               INAREA FOR 2520
        AREA2   DC    80C' '               OUAREA FOR 2520
        AREA3   DC    80C' '               WORKAREA FOR 2520
        *
                LTORG
                END   BEGIN
```

Example 1.1. Card File (2520 Card Read-Punch), Non-Overlap

```
                TITLE 'IOCS EXAMPLE NO 2 - FOR CARD/PRINTER FILE'
        *
        *
                START 4608
                PRINT NOGEN
        *
        *       FILE DEFINITIONS
        *       ----------------
        *
        CARD1   DTFSR BLKSIZE=80,                                              -
                      DEVICE=READ01,                                           -
                      EOFADDR=EOFC1,                                           -
                      IOAREA1=AREA1,                                           -
                      IOAREA2=AREA2,                                           -
                      TYPEFLE=INPUT,                                           -
                      WORKA=YES
        CARD2   DTFSR BLKSIZE=80,                                              -
                      CONTROL=YES,                                             -
                      DEVICE=MFCM1,                                            -
                      EOFADDR=EOFC2,                                           -
                      IOAREA1=AREA3,                                           -
                      TYPEFLE=INPUT,                                           -
                      WORKA=YES
        CARD3   DTFSR BLKSIZE=80,                                              -
                      CONTROL=YES,                                             -
                      CRDPRA=AREA4,                                            -
                      CRDPRL1=64,                                              -
                      CRDPRL2=64,                                              -
                      DEVICE=MFCM2,                                            -
                      IOAREA1=AREA5,                                           -
                      TYPEFLE=OUTPUT,                                          -
                      WORKA=YES
        PRINT   DTFSR BLKSIZE=120,                                             -
                      CONTROL=YES,                                             -
                      DEVICE=PRINTER,                                          -
                      PRINTOV=YES,                                             -
                      TYPEFLE=OUTPUT,                                          -
                      WORKA=YES
                DTFEN
                EJECT
        *
        *       PROCESSING ROUTINE
        *       ------------------
        *
        BEGIN   EQU   *
                BASR  8,0                 LOAD BASE REGISTER 8
                USING *,8,9
        BSREG   LH    9,BSADD             LOAD BASE REGISTER 9
        *
                OPEN  CARD1,CARD2,CARD3,PRINT
        *
                CNTRL PRINT,SK,1          SKIP
                MVC   WORKP+DSPL1(32),=C'LIST COMPARE RUN ON 2501 / MFCM1'
        *
        LOOP1   EQU   *
    1 ------>   PUT   PRINT,WORKP         PRINT-HEADER LINE
                PRTOV PRINT,12
        SWIT1   NOP   LOOP2               FIRST SWITCH, INITIALLY NO BRANCH
    2 ------>   MVI   SWIT1+1,X'F0'       SET FIRST SWITCH TO BRANCH
                CNTRL PRINT,SP,3          SPACE 3 LINES
        *
        LOOP2   EQU   *
                MVC   WORKP+1(120),WORKP  CLEAR PRINT WORK AREA
                GET   CARD1,WORK1
                GET   CARD2,WORK2
    3 ------>   MVC   WORKP+DSPL3(80),WORK2 PREPARE FOR LISTING
    4 ------>   CLC   WORK1(80),WORK2     CHECK IF CARDS EQUAL
                BE    LOOP1
```

Example 2. Card and Printer Files, Part 1 of 2

```
                MVC   WORKP+DSPL2(3),=C'OLD' INDICATE RECORD AS OLD
5 ------------> PUT   PRINT,WORKP          PRINT THE RECORD IN ERROR
                PRTOV PRINT,12
                CNTRL ,SS,4                SELECT ERROR CARD
                MVC   WORKP+DSPL3(80),WORK1  PREPARE NEW RECORD
6 ------------> MVC   WORKP+DSPL2(3),=C'NEW' INDICATE RECORD AS NEW
                MVC   WORK3(80),WORK1
7 ------------> PUT   PRINT,WORKP          PRINT NEW RECORD
                PRTOV PRINT,12
                CNTRL CARD3,SS,1           SELECT NEW CARD TO BE PUNCHED INTO
      *                                    STACKER1 I.E. MERGE INTO CARD2 FILE
                PUT   CARD3,WORK3          PUNCH AND PRINT NEW RECORD
                MVC   WORK4(64),WORK3      PREPARE FIRST 64 BYTES FOR HEAD 1
                MVC   WORK4+112(16),WORK3+64
8 ------------> CRDPR ,WORK4,AREA4
                CRDPR ,WORK4+64,AREA4+64
                CNTRL PRINT,SP,1           SPACE 1 LINE
                B     LOOP2
      *
      *         EXIT ROUTINES
      *         -------------
      *
      EOFC1     EQU   *                    EOF CONDITION DETECTED
      EOFC2     EQU   *
                CNTRL PRINT,SK,1           SKIP
                CLOSE CARD1,CARD2,CARD3,PRINT
                EOJ
      *                                    TERMINATION OF JOB
      *         DEFINITION OF CONSTANTS
      *         -----------------------
      *
      AREA1     DC    80C' '               IOAREA1 FOR 2501
      AREA2     DC    80C' '               IOAREA2 FOR 2501
      AREA3     DC    80C' '               IOAREA1 FOR MFCM1
      AREA4     DC    128C' '              CARD PRINT AREA FOR MFCM2
      AREA5     DC    80C' '               IOAREA1 FOR MFCM1
      *
      WORK1     DC    80C' '               WORAREA FOR 2501
      WORK2     DC    80C' '               WORKAREA FOR MFCM1
      WORK3     DC    80C' '               WORKAREA FOR MFCM2
      WORK4     DC    128C' '              WORK AREA FOR CARD PRINT
      WORKP     DC    120C' '              WORK AREA FOR PRINTER
      *
      BSADD     DC    Y(BSREG+4096)
      *
      DSPL1     EQU   10
      DSPL2     EQU   25
      DSPL3     EQU   30
                LTORG
                END   BEGIN
```

Example 2. Card and Printer Files, Part 2 of 2

Example 2 - Card and Printer Files (2560 MFCM, 2501 Card Reader, Printer)

A card input file read on a 2501 is compared with an input file read on an MFCM1. A card output file and a printer file are created.

After the files have been opened, a header line is prepared and printed (1), and three lines are spaced. The switch is then modified (2) so that the space instruction is by-passed after subsequent print operations.

A card is read from each input file and the record read on the MFCM1 is moved (3) into the work area of the printer file. The records are compared (4). If they are equal, the record in the print area is printed, and the next two cards are read. If the records are not equal, the record read from the MFCM1 is indicated as 'OLD' and printed (5). The characters 'NEW' are inserted (6) in the record read from the 2501, and the record is printed (7). The card output file is punched and card-printed (8).

This example requires 3130 bytes of main storage.

```
                TITLE 'IOCS EXAMPLE NO 3 - FOR PRINTER KEYBOARD INPUT FILE'
        *
        *
                START 4608
                PRINT NOGEN
        *
        *       FILE DEFINITIONS
        *       ----------------
        *
                DTFBG MAINPRG=YES
        DATA1   DTFIS DERREX=ERRC1,                                            -
                      DEVICE=DISK11F,                                          -
                      DSKXTNT=4,                                               -
                      IOAREAR=AREA1,                                           -
                      IOROUT=RETRVE,                                           -
                      KEYARG=KEYA1,                                            -
                      KEYLEN=10,                                               -
                      KEYLOC=1,                                                -
                      NRECDS=16,                                               -
                      RECFORM=FIXBLK,                                          -
                      RECSIZE=80,                                              -
                      RTRVEX=RTREX,                                            -
                      TYPEFLE=RANDOM,                                          -
                      UPDATE=RANDOM,                                           -
                      WORKR=WORK1
        PUTIN   DTFPK BLKSIZE=15,                                              -
                      EOFADDR=EOFC1,                                           -
                      IOAREA=AREA2,                                            -
                      TYPEFLE=INPUT
                DTFEN
                EJECT
        *
        *       PROCESSING ROUTINE
        *       ------------------
        *
        BEGIN   EQU   *
                BASR  8,0                 LOAD BASE REGISTER 8
                USING *,8
        *
                OPEN  DATA1,PUTIN
        *
        LOOP1   EQU   *
                READ  PUTIN               READ PRINTER KEYBOARD RECORD I.E.
        *                                 10 BYTES KEY INFORMATION AND
        *                                  5 BYTES UPDATE INFORMATION
    1 ------->  WAITF PUTIN               WAIT FOR COMPLETION OF READ
    2 ------->  MVC   KEYA1,AREA2         INSERT KEY INTO KEYARG FIELD
    3 ------->  READ  DATA1,KEY           RETRIEVE RECORD
                WAITF DATA1               WAIT FOR COMPLETION OF READ
        *
                MVC   WORK1+15(5),AREA2+10      INSERT UPDATE INTO RECORD
        *
    4 ------->  WRITE DATA1,KEY           RETURN RECORD TO DISK
                WAITF DATA1               WAIT FOR COMPLETION OF WRITE
                B     LOOP1
        *
        *       EXIT ROUTINES
        *       -------------
        *
        ERRC1   EQU   *                   DISK ERROR CONDITION
                HPR   X'F01'(1),0         HALT IF IRRECOVERABLE DISK ERROR
                B     EOFC1
        *
        RTREX   EQU   *
                HPR   X'F01',0            REQUIRED RECORD NOT FOUND
        EOFC1   EQU   *                   /* TYPED ON PRINTER KEYBOARD
                CLOSE DATA1,PUTIN
                EOJ
        *                                 TERMINATION OF JOB
```

Example 3. Printer-Keyboard Input File and Indexed-Sequential File, Part 1 of 2

```
*        DEFINITIONS OF CONSTANTS
*        ------------------------
*
AREA1    DC    135CL10' '          IOAREA FOR DISK DATA1
AREA2    DC    15C' '              IOAREA FOR PRINTER KEYBOARD INPUT
*
WORK1    DC    80C' '              WORKAREA FOR DISK DATA1
KEYA1    DC    C'          '       KEYARG FIELD FOR KEY ON DISK DATA1
*
         END BEGIN
```

Example 3. Printer-Keyboard Input File and Indexed-Sequential Disk File, Part 2 of 2

Example 3 - Printer-Keyboard Input File and Indexed-Sequential File

A record from an indexed-sequential file is retrieved randomly and updated according to control information typed in on the printer-keyboard. The control information consists of ten bytes of key information and five bytes of update information.

After the record has been completely transferred to the input area (1), the key information is inserted into the KEYARG field (2). Then, the record is retrieved (3), updated, and written back on disk (4).

This example requires 4450 bytes of main storage.

```
          TITLE 'IOCS EXAMPLE NO 4 - FOR PRINTER KEYBOARD OUTPUT FILE'
*
          START X'1200'
          PRINT NOGEN
*
*         FILE DEFINITIONS
*         ----------------
*
          DTFBG ,                        NO MAINLINE AND NO INQUIRY PROGRAM
DATA1     DTFSD BLKSIZE=800,                                              -
                DEVICE=DISK11F,                                           -
                EOFADDR=EOFC1,                                            -
                ERROPT=SKIP,                                              -
                IOAREA1=AREA1,                                            -
                IOAREA2=AREA2,                                            -
                IOREG=(13),                                               -
                RECFORM=FIXBLK,                                           -
                RECSIZE=80,                                               -
                TYPEFLE=INPUT
OUTPT     DTFPK CONTROL=YES,                                              -
                BLKSIZE=30,                                               -
                IOAREA=AREA3,                                             -
                LCTABLE=YES,                                              -
                PRINTOV=YES,                                              -
                TYPEFLE=OUTPUT,                                           -
                WORKA=YES
          DTFLC 45,00301,04012
          DTFEN
          EJECT
*
*         PROCESSING ROUTINE
*         ------------------
*
BEGIN     EQU   *
          BASR  10,0                     LOAD BASE REGISTER 10
          USING *,10
*
          OPEN  DATA1,OUTPT
*
          COMRG ,                        LOAD ADDRESS OF COMMUNICATION
*                                        REGION INTO REGISTER 8
          MVC   HEAD1+22(2),0(8)         PREPARE DATE IN HEAD1 (MONTH)
1 ------> MVC   HEAD1+25(2),2(8)                              (DAY)
          MVC   HEAD1+28(2),4(8)                               (YEAR)
          PUT   OUTPT,HEAD1              TYPE HEAD1
          PUT   OUTPT,HEAD2              TYPE HEAD2
          CNTRL OUTPT,SP,3               SPACE 3 LINES
LOOP1     EQU   *
          GET   DATA1
2 ------> CLC   0(6,13),=C'            TEST FOR BLANKS IN COLUMNS 1 - 6
          BE    LOOP1
*
3 ------> MVC   WORK1(30),0(13)         IF NOT, MOVE FIRST 30 BYTES TO WORK1
          PRTOV OUTPT,12                 TEST FOR END OF FORM ,IF YES
*                                        SKIP TO NEXT PAGE
          PUT   OUTPT,WORK1              TYPE RECORD
          B     LOOP1
*
*         EXIT ROUTINES
*         -------------
*
EOFC1     EQU   *
          CNTRL OUTPT,SP,1               SPACE 1 LINE
          MVI   WORK1,X'40'              CLEAR AREA WORK1
          MVC   WORK1(30),WORK1
          MVC   WORK1(11),=C'END OF FILE'
4 ------> PUT   OUTPT,WORK1              TYPE EOF CONDITION
          CLOSE DATA1,OUTPT
          EOJ
*                                        TERMINATION OF JOB
```

Example 4. Printer-Keyboard Output File and Sequential Disk File, Part 1 of 2

```
*        DEFINITIONS OF CONSTANTS
*        ------------------------
*
AREA1    DC    81CL10' '           IOAREAS FOR DATAS
AREA2    DC    81CL10' '
AREA3    DC    30C' '              IOAREA FOR OUTPT
*
HEAD1    DC    C'FILE = DATA1,  DATE =   /  /  '
HEAD2    DC    C'IDENTIFIED RECORDS RETRIEVED  '
*
WORK1    EQU   HEAD1               WORKAREA FOR OUTPT
*
         LTORG
*
         END   BEGIN
```

Example 4. Printer-Keyboard Output File and Sequential Disk File, Part 2 of 2

Example 4 - Printer-Keyboard Output and Sequential Disk File

A sequential disk file is checked for records containing blanks in columns 1 through 6. Records that do not contain blanks in these columns are printed on the printer-keyboard. The current date in the communication region is fetched and printed on the printer keyboard (1). A record of the sequential file is read and checked (2). If the record does not contain blanks in columns 1 through 6, it is moved to the printer-keyboard work area and printed (3). When the end of the file is reached, the message 'END OF FILE' is printed (4).

This example requires 4270 bytes of main storage.

```
                TITLE 'IOCS EXAMPLE NO 5 - TAPE OUTPUT FILE WITH USER LABELS'
*
*
                START 4608
                USING *-4608,0,1,2,3
                PRINT NOGEN
*
*               FILE DEFINITION
*               ---------------
*
DATA1           DTFMT BLKSIZE=360,                                      -
                      DEVADDR=SYS010,                                   -
                      FILABL=STD,                                       -
                      IOAREA1=AREA1,                                    -
                      LABADDR=LABEL,                                    -
                      READ=FORWARD,                                     -
                      RECFORM=FIXBLK,                                   -
                      RECSIZE=90,                                       -
                      TYPEFLE=OUTPUT,                                   -
                      WORKA=YES
CARD1           DTFSR BLKSIZE=80,                                       -
                      CONTROL=YES,                                      -
                      DEVICE=MFCM1,                                     -
                      EOFADDR=EOFC1,                                    -
                      IOAREA1=AREA2,                                    -
                      TYPEFLE=INPUT,                                    -
                      WORKA=YES
                DTFEN
                EJECT
*
*               PROCESSING ROUTINE
*               ------------------
*
BEGIN           EQU   *
                OPEN  DATA1,CARD1
*
LOOP1           EQU   *
                GET   CARD1,WORK2
1 ------------> AP    WORK1,CONT1       UPDATE RECORD COUNT
                PUT   DATA1,WORK1
                B     LOOP1
*
*               EXIT ROUTINES
*               -------------
*
EOFC1           EQU   *                 EXIT FOR EOF CONDITION
                CLOSE DATA1,CARD1
                EOJ                     TERMINATION OF JOB
*
*
*               USER LABEL HANDLING
*
LABEL           EQU   *
*                                       DO NOT USE REGISTERS 14 AND 15
2 ------------> CH    8,TYPE            CHECK IF HEADER LABEL REQUIRED
                BE    LABL1             BRANCH IF HEADER LABEL REQUIRED
                LH    REG2,LABL8        LOAD REG2 WITH TRAILER-LABEL ADDR.
                B     LABL2             IF NO HEADER LABEL INDICATION,
*                                       RETURN TO IOCS
LABL1           LH    REG2,LABL6
                AH    REG2,LABLN        UPDATE LABEL POINTER
3 ------------> CH    REG2,LABL7        CHECK BRANCH TO LBRET 1
                BE    LABL2             IF EQUAL,BRANCH TO LBRET1
4 ------------> MVC   0(80,9),0(REG2)   BUILD ADDITIONAL LABEL INFORMATION
                STH   REG2,LABL6
5 ------------> LBRET2                  RETURN TO IOCS AFTER BUILDING USER
*                                       LABEL EXCEPT THE LAST ONE
LABL2           MVC   0(80,9),0(REG2)
6 ------------> LBRET1                  RETURN TO IOCS AFTER BUILDING LAST
*                                       I.E. FOURTH USER LABEL
```

Example 5. Tape Output File with User Labels and Card File, Part 1 of 2

```
*         DEFINITION OF CONSTANTS FOR USER LABEL
*
7 ->LABL5 EQU   *                    DEFINITION OF USER LABEL INFORMATION
          DC    CL24'UHL1   FIRST USER LABEL'
          DC    56C' '
          DC    CL24'UHL2  SECOND USER LABEL'
          DC    56C' '
          DC    CL24'UHL3  THIRD USER LABEL'
          DC    56C' '
          DC    CL24'UHL4  LAST USER LABEL'
          DC    56C' '
LABL6     DC    Y(LABL5-80)          BEGIN POINTER OF LABL5 AND SAVEFIELD
LABL7     DC    Y(LABL5+3*80)        END POINTER OF LABL5
*
LABL8     DC    Y(*+2)
          DC    CL24'UTL1  USER TRAILER LABEL'
          DC    56C' '
*
*         DEFINITION OF CONSTANTS
*         -----------------------
*
AREA1     DC    36CL10' '            IOAREA FOR DATA1
AREA2     DC    80C' '               IOAREA FOR CARD1
WORK1     DC    PL10'+0'             WORK AREA FOR DATA1
8 ->WORK2 DC    08CL10' '            WORK AREA FOR CARD + DATA1
          DS    0H
TYPE      DC    C' 0'
*
LABLN     DC    H'80'                LENGTH OF USER INFORMATION
CONT1     DC    PL10'+1'
*
REG2      EQU   13
*
          END   BEGIN
```

Example 5. Tape Output File with User Labels and Card File, Part 2 of 2

Example 5 - Tape Output File with User Labels and Card File (2560 MFCM)

A tape output file is created from a card input file read on the 2560 MFCM1. The output file has four additional header labels and one additional trailer label.

The processing loop reads the cards and writes the records on magnetic tape. Common overlapping work areas are used (8) for this purpose. A record count is prepared in the first ten bytes of the tape records (1). The LABADDR routine checks (2) if header labels are required. If no labels are to be written, control is returned to the IOCS. If labels are to be built, the label constants (7) are moved to the label area used by the open routine (4). The label information pointer is checked (3). If further labels are to be written, LBRET 2 is executed (5). Otherwise a branch to LBRET 1 is taken (6).

This example requires 3230 bytes of main storage.

```
                TITLE 'IOCS EXAMPLE NO 5.1 - TAPE OUTPUT WITH USER LABELS'
        *
        *
                START 4608
                USING *-4608,0,1,2,3
                PRINT NOGEN
        *
        *       FILE DEFINITION
        *       ---------------
        *
        DATA1   DTFMT BLKSIZE=380,                                          -
                      DEVADDR=SYS010,                                       -
                      FILABL=STD,                                           -
                      IOAREA1=AREA1,                                        -
                      IOREG=(REG2),                                         -
                      LABADDR=LABEL,                                        -
                      READ=FORWARD,                                         -
                      REWIND=NORWD,                                         -
                      RECFORM=VARBLK,                                       -
                      TYPEFLE=OUTPUT,                                       -
                      VARBLD=(REG1)
        CARD1   DTFSR BLKSIZE=80,                                           -
                      CONTROL=YES,                                          -
                      DEVICE=MFCM1,                                         -
                      EOFADDR=EOFC1,                                        -
                      IOAREA1=AREA2,                                        -
                      TYPEFLE=INPUT,                                        -
                      WORKA=YES
                DTFEN
                EJECT
        *
        *       PROCESSING ROUTINE
        *       ------------------
        *
        BEGIN   EQU   *
                OPEN  DATA1,CARD1
                SR    REG1,REG1
        *
        LOOP1   EQU   *
                GET   CARD1,WORK2+10
1 ------------> CH    REG1,RECLN        CHECK IF AREA OVERFLOW
                BNL   LOOP2             IF YES TRUNCATE BLOCK
2 ------------> TRUNC DATA1
        *
3 ----> LOOP2   MVC   WORK1(2),RECLN    MOVE RECORD LENGTH INTO WORK AREA
                LH    REG1,RECLN        PREPARE MOVE INSTRUCTION
4 ------------> SH    REG1,=H'1'
                STH   REG1,MOVE1        INSERT LENGTH INTO INSTRUCTION
                MVI   MOVE1,X'D2'       RESTORE OP-CODE
5 ----> MOVE1   MVC   0(84,REG2),WORK1  MOVE RECORD INTO IOAREA
                AP    WORK2,CONT1       UPDATE RECORD COUNT
                PUT   DATA1
                B     LOOP1
        *
        *       EXIT ROUTINES
        *       -------------
        *
        EOFC1   CLOSE DATA1,CARD1
                EOJ
        *                               TERMINATION OF JOB
        *
        *       USER LABEL HANDLING
        *
        LABEL   EQU   *
        *                               DO NOT USE REGISTERS 14 AND 15
                CH    8,TYPE
                BE    LABL1             BRANCH IF HEADER LABEL INDICATION
                LH    REG2,LABL8        LOAD REG2 WITH TRAILER-LABEL ADDR.
                B     LABL2             IF NO HEADER LABEL INDICATION,
        *                               RETURN TO IOCS
```

Example 5.1. Tape Output File with User Labels and Card File, Part 1 of 2

```
LABL1    LH    REG2,LABL6
         AH    REG2,LABLN          UPDATE LABEL POINTER
         CH    REG2,LABL7          CHECK BRANCH TO LBRET1
         BE    LABL2               IF EQUAL,BRANCH TO LBRET1
         MVC   0(80,9),0(REG2)     BUILD ADDITIONAL LABEL INFORMATION
         STH   REG2,LABL6
         LBRET2                    RETURN TO IOCS AFTER BUILDING USER
*                                  LABEL EXCEPT THE LAST ONE
LABL2    MVC   0(80,9),0(REG2)
         LBRET1                    RETURN TO IOCS AFTER BUILDING LAST
*                                  I.E. FOURTH USER LABEL
*        DEFINITION OF CONSTANTS FOR USER LABEL
*
LABL5    EQU   *                   DEFINITION OF USER LABEL INFORMATION
         DC    CL24'UHL1   FIRST USER LABEL'
         DC    56C' '
         DC    CL24'UHL2  SECOND USER LABEL'
         DC    56C' '
         DC    CL24'UHL3  THIRD USER LABEL'
         DC    56C' '
         DC    CL24'UHL4  LAST USER LABEL'
         DC    56C' '
LABL6    DC    Y(LABL5-80)         BEGIN POINTER OF LABL5 AND SAVEFIELD
LABL7    DC    Y(LABL5+3*80)       END POINTER OF LABL5
LABL8    DC    Y(*+2)
         DC    CL24'UTL1  USER TRAILER LABEL'
         DC    56C' '
*
*        DEFINITION OF CONSTANTS
*        -----------------------
*
AREA1    DC    H'3400'             IOAREA FOR DATA1
         DC    H'0'
         DC    208C' '
         DC    208C' '
AREA2    DC    80C' '              IOAREA FOR CARD1
WORK1    DC    H'84'               WORKAREA FOR DATA1
         DC    H'0'
WORK2    DC    PL10'+0'            DATA AREA FOR DATA1.
         DC    08CL10' '
         DS    0H
TYPE     DC    C' 0'
6 -> RECLN DC  H'84'               FIELD FOR CURRENT RECORD LENGTH
CONT1    DC    PL10'+1'
LABLN    DC    H'80'
*
REG1     EQU   13
REG2     EQU   12
         END   BEGIN
```

Example 5.1. Tape Output File with User Labels and Card File, Part 2 of 2

Example 5.1 - Tape Output File with User Labels and Card File (2560 MFCM)

This example is very similar to example 5. However, (pseudo) variable records are built for the tape output file.

It is assumed that the field RECLN (6) contains the record length. Before the record is moved to the I/O area (5), a check (1) is performed to determine whether the block has to be truncated (2) or whether the record fits into the area. The number of bytes that are still available in the I/O area are contained in REG2, i.e., in the VARBLD register. The record length is inserted into the first two bytes of the work area (3). The required length is inserted into the MVC instruction (4), i.e., record length minus one. Then, the record is moved to the I/O area (5) and written onto tape.

This example requires 3340 bytes of main storage.

```
            TITLE 'IOCS EXAMPLE NO 6 - TAPE UPDATE WITH CARD INPUT'
      *
            START 4608
            USING *-4608,0,1,2,3
            PRINT NOGEN
      *     FILE DEFINITIONS
      *     ----------------
      *
      PUTIN DTFMT ALTTAPE=SYS008,                                          -
                  BLKSIZE=360,                                             -
                  DEVADDR=SYS007,                                          -
                  EOFADDR=EOFC1,                                           -
                  ERRID=ERRID,                                             -
                  ERROPT=ERRC1,                                            -
                  FILABL=STD,                                              -
                  IOAREA1=AREA1,                                           -
                  IOAREA2=AREA2,                                           -
                  IOREG=(REG1),                                            -
                  READ=FORWARD,                                            -
                  RECFORM=FIXBLK,                                          -
                  RECSIZE=90,                                              -
                  REWIND=UNLOAD,                                           -
                  TYPEFLE=INPUT
      OUTPT DTFMT ALTTAPE=SYS011,                                          -
                  BLKSIZE=180,                                             -
                  DEVADDR=SYS010,                                          -
                  FILABL=STD,                                              -
                  IOAREA1=AREA3,                                           -
                  IOAREA2=AREA4,                                           -
                  IOREG=(REG2),                                            -
                  READ=FORWARD,                                            -
                  RECFORM=FIXBLK,                                          -
                  RECSIZE=90,                                              -
                  REWIND=UNLOAD,                                           -
                  TYPEFLE=OUTPUT
      CARD1 DTFSR BLKSIZE=20,                                              -
                  DEVICE=READ01,                                           -
                  EOFADDR=EOFC2,                                           -
                  IOAREA1=AREA5,                                           -
                  IOAREA2=AREA6,                                           -
                  SEQNCE=0110,                                             -
                  SEQXIT=SEQEX,                                            -
                  TYPEFLE=INPUT,                                           -
                  WORKA=YES
      PRINT DTFSR BLKSIZE=90,                                              -
                  DEVICE=PRINTER,                                          -
                  TYPEFLE=OUTPUT,                                          -
                  WORKA=YES
            DTFEN
            EJECT
      *     PROCESSING ROUTINE
      *     ------------------
            USING DSECT,REG2
      BEGIN EQU   *
            OPEN  PUTIN,OUTPT,CARD1,PRINT
      *
      LOOP1 EQU   *
1 ------> MVI   SWIT2+1,X'00'       CHANGE SWITCH 2 TO NOP
            GET   CARD1,WORK1        GET NEXT CARD
            PACK  UPDT1(10),UPDT1(10)
2 ----> LOOP2 GET   PUTIN            GET TAPE RECORD
3 ------> MVC   0(90,REG2),0(REG1)
4 ------> CP    RCRD1(10),UPDT1(10) CHECK IF UPDATE REQUIRED
      SWIT1 BNE   LOOP3              IF NOT, PUT RECORD IMMEDIATELY
      *                              ON TAPE
5 ------> MVC   RCRD2(3),UPDT2       INSERT UPDATE INFORMATION FROM CARD
          MVC   RCRD4(7),UPDT3       INTO TAPE RECORD
6 ------> MVI   SWIT2+1,X'F0'       CHANGE SWITCH 2 TO BRANCH
7 ----> LOOP3 PUT   OUTPT
      SWIT2 NOP   LOOP1              BRANCH TO GET NEXT CARD
8 ------> B     LOOP2                BRANCH TO GET NEXT TAPE RECORD
```

Example 6. Tape Update File, Card File and Printer File, Part 1 of 2

```
*        EXIT ROUTINES
*        -------------
*
EOFC1    EQU   *                    END OF TAPE INPUT FILE
         CLOSE PUTIN,OUTPT,CARD1,PRINT
         EOJ
*                                   TERMINATION OF JOB
9 ->EOFC2    EQU   *                END OF CARD FILE CONDITION
         MVI   SWIT2+1,X'00'        DEACTIVATE CARD READING
         B     LOOP2                TRANSFER REST OF TAPE FILE
ERRC1    EQU   *                    TAPE ERROR FOR INPUT FILE
10->PUTPR    PUT   PRINT,*          PRINT ERROR RECORD
11->ERRIO    EQU   PUTPR+4          IOAREA IN ERROR ADDRESS WILL BE
*                                   INSERTED AS PRINTAREA ADDRESS
         B     SWIT2                RETURN TO LOOP
SEQEX    EQU   *
         WAITC
         HPR   X'E01',0             HALT INDICATES CARD OUT OF SEQUENCE
*                                   THE CARD WILL BE SKIPPED ON RESTART
         GET   CARD1,WORK1          ONE DUMMY GET REQUIRED
         B     LOOP1                RETURN TO LOOP
*
*        DEFINITION OF CONSTANTS
*        -----------------------
         DS    0H
AREA1    DC    36CL10' '            IOAREAS FOR INPUT
AREA2    DC    36CL10' '
AREA3    DC    18CL10' '            IOAREAS FOR OUTPT
AREA4    DC    18CL10' '
AREA5    DC    02CL10' '            IOAREAS FOR CARD1
AREA6    DC    02CL10' '
WORK1    EQU   *                    WORKAREA FOR CARD1, CONTAINING
UPDT1    DC    10C' '               FIELD1
UPDT2    DC    03C' '               FIELD2
UPDT3    DC    07C' '               FIELD3
*
12->DSECT    DSECT
RCRD1    DS    CL10                 FIELD 1 IN INPUT RECORD
         DS    CL8
RCRD2    DS    CL3                  FIELD 2 IN INPUT RECORD
RCRD3    DS    CL2
RCRD4    DS    CL7                  FIELD 3 IN INPUT RECORD
*        NO FURTHER DEFINITION MUST BE GIVEN, IF NOT REQUIRED IN DSECT
REG1     EQU   08                   EQUATING OF REGISTERS
REG2     EQU   09
REG3     EQU   10
         END   BEGIN
```

Example 6. Tape Update File, Card File and Printer File, Part 2 of 2

Example 6 - Tape Update File, Card File (2501) and Printer File

A tape input file is updated by a presorted card input file, and an output file is created on tape.

A card is read and the branch switch, which causes the next card to be read, is set to no-operation (1). Tape records are read (2) and moved from the input to the output area (3). The card record and the tape record are compared (4) to determine whether they have the same identifier. If they do, the record is updated (5) and the read switch is changed (6) so that the next card can be read. If no update is required, the card is immediately written onto tape (7) and the next tape record is read (8).

In the card end-of-file routine (9) the switch for card reading is turned off and the rest of the tape file is copied. If an error occurs in the tape input file, the

error routine prints (10) the first record of the block preceding the block that contains the error record. ERRIO (11) is equated to PUTPR+4 to cause the address of the I/O area containing the error record to be inserted as work area address in the PUT macro instruction. For ease of reference, a dummy section is generated for the tape output area (12). The referenced base register REG2 is the IOREG of the tape output file.

This example requires 3680 bytes of main storage.

Example 7 - Sequential Disk Output File and Card File

This example illustrates the creation of a sequential disk output file from a card input file. A card is read, moved to the I/O area (2) and written onto disk (3). IOREG must be specified (1) because blocked records are processed in the I/O area and no work area is specified.

This example requires 4050 bytes of main storage.

```
                TITLE 'IOCS EXAMPLE NO 7 - SEQUENTIAL DISK OUTPUT FILE'
        *
        *
                START 4608
                USING *-4608,0,1,2,3
                PRINT NOGEN
        *
        *       FILE DEFINITIONS
        *       ----------------
        *
        CARD1   DTFSR BLKSIZE=80,                                       -
                      DEVICE=READ01,                                    -
                      EOFADDR=EOFC1,                                    -
                      IOAREA1=AREA1,                                    -
                      IOAREA2=AREA2,                                    -
                      TYPEFLE=INPUT,                                    -
                      WORKA=YES
        DATA1   DTFSD BLKSIZE=800,                                      -
                      DEVICE=DISK11F,                                   -
                      DTAREX=DTAEX,                                     -
                      ERROPT=ERRC1,                                     -
                      IOAREA1=AREA3,                                    -
1 ------------------> IOREG=(REG1),                                    -
                      RECFORM=FIXBLK,                                   -
                      RECSIZE=80,                                       -
                      TYPEFLE=OUTPUT
                DTFEN
                EJECT
        *
        *       PROCESSING ROUTINE
        *       ------------------
        *
        BEGIN   EQU   *
                OPEN  CARD1,DATA1
        *
        LOOP1   EQU   *
                GET   CARD1,WORK1
2 ------------> MVC   0(80,REG1),WORK1     MOVE ONE RECORD
        *
3 ----> LOOP2   PUT   DATA1
                B     LOOP1
        *
        *       EXIT ROUTINES
        *       -------------
        *
        DTAEX   EQU   *                   EXIT ON EXTENT OVERFLOW ON DISK
                HPR   X'F01',0            HALT AND CONTINUE AS FOR EOFC1
        *
        EOFC1   EQU   *                   EXIT ON EOF CONDITION ON 2501
                CLOSE CARD1,DATA1
                EOJ
        *                                 TERMINATION OF JOB
        *
        ERRC1   EQU   *                   EXIT ON PERMANENT DISK ERROR
                HPR   X'F01'(1),0         HALT REQUIRES ALTERNATE TRACK ASSGN.
                B     EOFC1
        *
        *
        *       DEFINITION OF CONSTANTS
        *       -----------------------
        *
        AREA1   DC    80C' '              INPUT AREAS FOR 2501
        AREA2   DC    80C' '
        AREA3   DC    81CL10' '           OUTPUT AREA FOR DISK
        *
        WORK1   DC    80C' '              WORK AREA FOR 2501
        *
        REG1    EQU   8
        *
                END   BEGIN
```

Example 7. Sequential Disk Output File and Card File

```
                TITLE 'IOCS EXAMPLE NO 8 - SEQUENTIAL DISK UPDATE FILE'
        *
                PRINT NOGEN
                START 4608
                USING *-4608,0,1,2,3
        *
        *       FILE DEFINITIONS
        *       ----------------
        *
        DATA1   DTFSD BLKSIZE=800,                                          -
                      DEVICE=DISK11F,                                       -
                      EOFADDR=EOFC1,                                        -
                      ERROPT=ERRC1,                                         -
                      IOAREA1=AREA1,                                        -
                      IOAREA2=AREA2,                                        -
                      IOREG=(REG1),                                         -
                      RECFORM=FIXBLK,                                       -
                      RECSIZE=80,                                           -
                      TYPEFLE=INPUT,                                        -
                      UPDATE=YES
        PRINT   DTFSR BLKSIZE=80,                                           -
                      DEVICE=PRINTER,                                       -
                      PRINTOV=YES,                                          -
                      TYPEFLE=OUTPUT,                                       -
                      CONTROL=YES,                                          -
                      WORKA=YES
                DTFEN
                EJECT
        *
        *       PROCESSING ROUTINE
        *       ------------------
        *
        BEGIN   EQU   *
                OPEN  DATA1,PRINT
                CNTRL PRINT,SK,1          SKIP
                PUT   PRINT,WORK1         PRINT HEADING
                CNTRL PRINT,SP,3          SPACE 3 LINES
        *
        LOOP1   EQU   *
                GET   DATA1
1 ------------> CLC   0(6,REG1),=C'      ' CHECK FOR UPDATE INDICATION
                BE    LOOP2               BRANCH IF UPDATE IS REQUIRED
                B     LOOP1
        LOOP2   EQU   *
                MVC   0(6,REG1),=C'UPDATE' INSERT UPDATE INFORMATION
2 ------------> MVC   WORK1(80),0(REG1)   MOVE RECORD INTO WORKAREA FOR PRINT
                PRTOV PRINT,12
3 ------------> PUT   PRINT,WORK1         PRINT UPDATED RECORD
4 ------------> MVI   SWCH1+1,X'F0'       CHANGE SWITCH1 TO BRANCH
5 ------------> PUT   DATA1               REWRITE UPDATED RECORD
                B     LOOP1
        *       EXIT ROUTINES
        *       -------------
        *
        EOFC1   EQU   *                   EXIT ON EOF CONDITION
        SWCH1   NOP   CLOSE
                MVC   WORK1(10),NOTE1
6 ------------> PUT   PRINT,WORK1         PRINT NOTE 'NO RECORDS UPDATED'
        CLOSE   EQU   *
                CLOSE DATA1,PRINT
                EOJ
        *                                 TERMINATION OF JOB
        ERRC1   EQU   *                   EXIT ON DISK ERROR
                CNTRL PRINT,SP,1
                MVC   WORK1(80),0(REG1)   RECORDS OF NEXT BLOCK BEHIND THE
                PUT   PRINT,WORK1         BLOCK CONTAINING THE PRINTED RECORD
        *                                 IS UNREADABLE
                CNTRL PRINT,SP,1
                MVI   SWCH1+1,X'F0'
                B     LOOP1               SKIP THE ERROR BLOCK
```

Example 8. Sequential Disk Update File and Printer File, Part 1 of 2

```
*        DEFINITION OF CONSTANTS
*        -----------------------
*
AREA1    DC    81CL10' '           INPUT AREAS FOR DATA1
AREA2    DC    81CL10' '
*
WORK1    DC    CL30'LISTING OF RECORDS UPDATED'
         DC    CL10' ON DATA1 '
         DC    40C' '
NOTE1    DC    CL10'        NO'
*
REG1     EQU   8
*
         LTORG
*
         END   BEGIN
```

Example 8. Sequential Disk Update File and Printer File, Part 2 of 2

Example 8 - Sequential Disk Update File and Printer File

A sequential disk file is to be updated by inserting the characters 'UPDATE'. Each record is read and checked (1) if update is required. If so, the update information is moved to the I/O area and the record is made available for printing (2). The record is printed (3) and written back onto disk (5). Switch 1 is changed to by-pass the printing of the message that no records are updated (4,6). If an error is detected, the error routine prints the first record of the block preceding the block that contains the error record.

This example requires 4730 bytes of main storage.

```
                TITLE 'IOCS EXAMPLE NO 9 - EXTENSION OF SEQUENTIAL DISK FILE'
         *
         *
                START 4608
                USING *-4608,0,1,2,3
                PRINT NOGEN
         *
         *       FILE DEFINITION
         *       ---------------
         *
         DATA1  DTFSD BLKSIZE=800,                                          -
                      DEVICE=DISK11F,                                       -
                      EOFADDR=LOOP2,                                        -
                      IOAREA1=AREA1,                                        -
                      IOREG=(REG1),                                         -
                      RECFORM=FIXBLK,                                       -
                      RECSIZE=80,                                           -
                      TYPEFLE=INPUT,                                        -
                      UPDATE=YES
         PRINT  DTFSR BLKSIZE=80,                                           -
                      CONTROL=YES,                                          -
                      DEVICE=PRINTER,                                       -
                      PRINTOV=YES,                                          -
                      TYPEFLE=OUTPUT,                                       -
                      WORKA=YES
                DTFEN
                EJECT
         *
         *       PROCESSING ROUTINE
         *       ------------------
         *
         BEGIN  EQU   *
                OPEN  DATA1,PRINT          OPEN FILES
                CNTRL PRINT,SK,1
                MVI   X'00CE',X'00'        CLEAR 'CE'-BYTE
         LOOP1  EQU   *
    1 ------>   GET   DATA1
                MVC   WORK1+9(70),9(REG1)
                B     LOOP1
    2 -->LOOP2  NOP   CLOSE                ROUTINE TO EXTEND THE FILE
                MVI   LOOP2+1,X'F0'        ACTIVATE BRANCH TO CLOSE
         LOOP3  EQU   *
                MVC   0(80,REG1),WORK1     MOVE RECORD TO IOAREA
    3 ---------->  PUT   DATA1
                PRTOV PRINT,12
    4 ---------->  PUT   PRINT,WORK1
    5 ---------->  MVI   80(REG1),X'00'    CLEAR FURTHER POSSIBLE /* INDICATOR
                GET   DATA1                UPDATE RECORD POINTER
    6 ---------->  CLI   X'00CE',X'EF'     CHECK CE-BYTE FOR EOF INDICATION
         SWCH1  BNE   LOOP3                IF NOT CONTINUE EXTENSION
                MVI   SWCH1+1,X'00'        SET BRANCH OFF
    7 ---------->  MVC   WORK1(3),=C'/* '  INSERT /* AS EOF CONDITION
                CH    REG1,=Y(AREA1+9*80) ENSURE PROPER WRITING OF EOF CONDIT.
                BL    LOOP3
                MVC   0(80,REG1),WORK1
    8 ---------->  PUT   DATA1             WRITE LAST BLOCK
                CLOSE DATA1,PRINT          CLOSE FILES
                EOJ                        TERMINATION OF JOB
         *
         *       DEFINITION OF CONSTANTS
         *       -----------------------
         *
         AREA1  DC    81CL10' '             IOAREA FOR INPUT
         WORK1  DS    0CL80
                DC    CL10'EXTENSION '
                DC    7CL10' '
         REG1   EQU   8
                END   BEGIN
```

Example 9. Extension of a Sequential Disk File

Example 9 - Extension of a Sequential Disk File

This example shows how a sequential disk update file may be extended. The file(1) is read until the EOF condition is encountered and the EOF routine (2) is entered. In the end-of-file routine, a switch is set in order to transfer control to EOJ if the EOF routine is entered a second time, i.e., if the end of an extent is reached or if a /* is detected which does not belong to the current file. Therefore, all disk extents should be cleared or an apporpriate restart should be provided for those cases.

The record to be added to the file is moved to the I/O area, written onto disk (3) and printed (4). Possible EOF indicators in subsequent records are replaced by binary zeros (5). When all records have been added to the file, X'EF' must be entered into the CE-byte to simulate the end-of-file condition (6). The characters /*b are inserted (7) in each record of the last block, which is written onto disk by another PUT (8).

When /* is encountered the first time, the execution of the PUT macro instruction (3) leads to a halt. A restart is required. However, X'EF' must not be entered into the CE-byte at this time.

This example requires 3640 bytes of main storage.

```
                TITLE 'IOCS EXAMPLE NO 10 - DIRECT ACCESS ONTO DISK'
          *
                START 4608
                USING *-4608,0,1,2,3
                PRINT NOGEN
          *
          *     FILE DEFINITIONS
          *     ----------------
          *
          CARD1 DTFSR BLKSIZE=80,                                        -
                      DEVICE=READ01,                                     -
                      EOFADDR=EOFC1,                                     -
                      IOAREA1=AREA1,                                     -
                      IOAREA2=AREA2,                                     -
                      TYPEFLE=INPUT,                                     -
                      WORKA=YES
          DATA1 DTFDA BLKSIZE=240,                                       -
                      DEVICE=DISK11F,                                    -
                      ERRBYTE=ERRBT,                                     -
                      IOAREA1=AREA3,                                     -
                      READID=YES,                                        -
                      SEEKADR=SEEKA,                                     -
                      TYPEFLE=INPUT,                                     -
                      WRITEID=YES
                DTFEN
                EJECT
          *
          *     PROCESSING ROUTINE
          *     ------------------
          *
          BEGIN EQU   *
                OPEN  CARD1,DATA1
          LOOP1 EQU   *
                GET   CARD1,WORK1
                MVN   SEEKA(1),WORK1       INSERT VOLUME NUMBER 'N'
     1 ---------> PACK  PACKF(3),WORK1+1(5) PACK THE DISK ADDRESS 'CCCHR'
     2 ---------> CVRT  SEEKA,PACKF         CONVERT DEC. TO HEX. DISK ADDRESS
          *
     3 ---------> CLI   INDIC,C' '          CHECK IF UPDATE REQUIRED
                BE    LOOP3                IF NOT, GO TO WRITE
          LOOP2 READ  DATA1,ID
                WAITF DATA1
     4 ---------> BAS   14,ERRC1            CHECK IF ERROR OCCURS
          *
     5 ---------> MVC   AREA3(10),TEXT1     UPDATE RECORD
                B     LOOP4
          LOOP3 EQU   *
     6 ---------> MVC   AREA3(70),TEXT1     BUILD NEW RECORD
     7 --------->LOOP4 WRITE DATA1,ID
                WAITF DATA1
                BAS   14,ERRC2             CHECK IF DISK ERROR OCCURS
                B     LOOP1
          ERRC1 EQU   *
     8 ---------> TM    ERRBT,B'11000000'   TEST IF ADDR. IS INVALID OR
          *                                OUTSIDE EXTENTS
                BZ    ERRC2
                HPR   X'F01',0             HALT INDICATES USER ERROR
                B     LOOP1                READ NEXT CONTROL CARD
          ERRC2 EQU   *
                TM    ERRBT+1,B'01010000'  TEST INTERVENTION REQ./EQUIPM. CHECK
                BZ    ERRC3
     9 ---------> MVI   X'00CE',X'00'
                HPR   X'F02'(1),0          HALT INDICATES EQUIPMENT CHECK
          *                                SWITCH DISK DRIVE ON/OFF
                CLI   X'00CE',X'FF'        IF X'FF' IS ENTERED GO TO EOJ
                BE    EOFC1
          ERRC3 TM    ERRBT+1,B'00001110'  TEST DATA/SEEK CHECK OR NO REC.FOUND
                BZ    0(0,14)
                HPR   X'F03'(1),0          HALT INDICATES DISK ERROR
    10 ---------> B     LOOP1                READ NEXT CONTROL CARD
```

Example 10. Direct-Access File and Card File, Part 1 of 2

```
*        EXIT ROUTINE
*        ------------
*
EOFC1    EQU   *
         CLOSE CARD1,DATA1
         EOJ
*                                  TERMINATION OF JOB
*
*        DEFINITION OF CONSTANTS
*        -----------------------
*
AREA1    DC    80C' '               INPUT AREAS FOR 2501
AREA2    DC    80C' '
AREA3    DC    27CL10' '            OUTPUT AREA FOR DISK
*
WORK1    DC    CL7'NCCCHR'          DRIVE AND DISK ADDRESS - PACKED
INDIC    DC    CL3'I'               UPDATE INDICATOR
TEXT1    DC    CL10' '              UPDATE INFORMATION AND
         DC    60C' '               NEW RECORD INFORMATION
*
ERRBT    DC    H'0'                 RESERVATION FOR ERROR INFORMATION
SEEKA    DC    XL8'00'              SEEK ADDR.FIELD 'MBBCCHHR'
PACKF    DC    XL3'00'              PACK FIELD X'CCCHR+'
*
         END   BEGIN
```

Example 10. Direct-Access File and Card File, Part 2 of 2

Example 10 - Direct-Access File and Card File

A direct-access file is processed by control information read on the 2501 Card Reader.

The cards contain the following information:

cols. 1-6	the disk address NCCCHR
col. 8	I for update, or b indicating that a new record is to be created
cols. 10-80	data.

The disk address is converted (2) if it is in packed format (1). A test is performed (3) to see whether an update is required. If it is, the disk address is checked for validity (4, 8). The next card is read if the address is invalid. Otherwise, the update information is inserted (5), and the record is written onto disk (7). If column 8 contains a blank, a new record is created (6) and written onto disk (7).

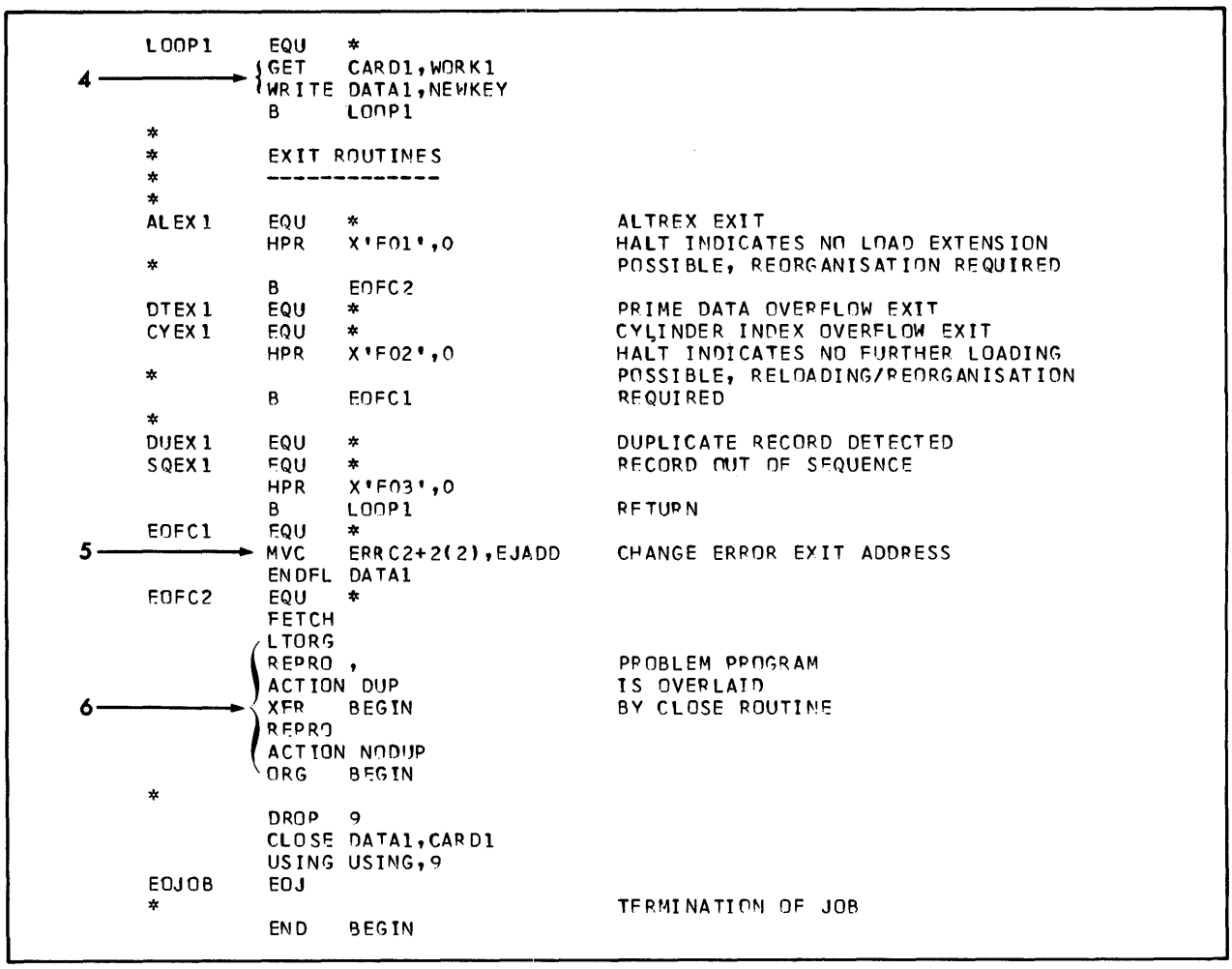
In case of an equipment error (9), the job is aborted. If no record is found or the data-seek check fails, the next card is read (10).

In this example it is assumed that the file has already been loaded. If this is not the case, TYPEFLE=OUTPUT must be specified to create a label in the VTOC.

This example requires 3420 bytes of main storage.

```
                TITLE 'IOCS EXAMPLE NO 11 - LOADING AN ISFMS FILE'
        *
        *
                START 4608
                PRINT NOGEN
        *
        *       FILE DEFINITIONS
        *       ----------------
        *
                DTFBG
        DATA1   DTFIS ALTREX=ALFX1,                                    -
                      CYLOFL=3,                                        -
                      CYNDEX=CYEX1,                                    -
                      DERREX=ERRC1,                                    -
                      DEVICE=DISK11F,                                  -
                      DSKXTNT=4,                                       -
                      DTAREX=DTEX1,                                    -
                      DUPREX=DUEX1,                                    -
                      IOAREAL=AREA1,                                   -
                      IOROUT=LOAD,                                     -
                      KEYLEN=10,                                       -
                      KEYLOC=1,                                        -
                      NRECDS=16,                                       -
                      RECFORM=FIXBLK,                                  -
                      RECSIZE=80,                                      -
                      SQCHEX=SQEX1,                                    -
                      WORKL=WORK1
        CARD1   DTFSR BLKSIZE=80,                                      -
                      DEVICE=READ01,                                   -
                      EOFADDR=EOFC1,                                   -
                      IOAREA1=AREA2,                                   -
                      IOAREA2=AREA3,                                   -
                      TYPEFLE=INPUT,                                   -
                      WORKA=YES
                DTFEN OVLAY
                EJECT
        *
        *       DEFINITION OF CONSTANTS
        *       -----------------------
        *
                USING USING,9
        USING   EQU   *
        AREA1   DC    135CL10' '
1 →     AREA2   DC    80C' '
        AREA3   DC    80C' '
        WORK1   DC    80C' '
        USADD   DC    Y(USING)
        EJADD   DC    Y(EOJOB)
        *
        *       PROCESSING ROUTINE
        *       ------------------
        *
        ERRC1   EQU   *                   DISK ERROR EXIT
2 →     HPR     X'F01'(1),0               HALT REQUIRES ALTERNATE TRACK ASSGN.
        ERRC2   B     EOFC1
        BEGIN   EQU   *
                DC    X'4890'             LH  9,USADD  I.E.
                DC    Y(USADD)            LOAD BASE REGISTER 9
        *
                DROP  9
                OPEN  DATA1,CARD1
                USING USING,9
                SETFL DATA1
                FETCH
                REPRO
                ACTION DUP                OPEN ROUTINE IS
3 ——————→       XFR   BEGIN               OVERLAID BY
                REPRO ,                   PROBLEM PROGRAM
                ACTION NODUP
                ORG   BEGIN
```

Example 11. Indexed-Sequential Load File and Card File, Part 1 of 2



```
         LOOP1    EQU   *
4 ------------>   GET   CARD1,WORK1
         |        WRITE DATA1,NEWKEY
         |        B     LOOP1
         *
         *        EXIT ROUTINES
         *        -------------
         *
         ALEX1    EQU   *                  ALTREX EXIT
                  HPR   X'F01',0           HALT INDICATES NO LOAD EXTENSION
         *                                 POSSIBLE, REORGANISATION REQUIRED
                  B     EOFC2
         DTEX1    EQU   *                  PRIME DATA OVERFLOW EXIT
         CYEX1    EQU   *                  CYLINDER INDEX OVERFLOW EXIT
                  HPR   X'F02',0           HALT INDICATES NO FURTHER LOADING
         *                                 POSSIBLE, RELOADING/REORGANISATION
                  B     EOFC1              REQUIRED
         *
         DUEX1    EQU   *                  DUPLICATE RECORD DETECTED
         SQEX1    EQU   *                  RECORD OUT OF SEQUENCE
                  HPR   X'F03',0
                  B     LOOP1              RETURN
         EOFC1    EQU   *
5 ------------>   MVC   ERRC2+2(2),EJADD   CHANGE ERROR EXIT ADDRESS
         |        ENDFL DATA1
         |EOFC2   EQU   *
         |        FETCH
         |        LTORG
         |        REPRO ,                  PROBLEM PROGRAM
         |        ACTION DUP               IS OVERLAID
6 ------------>   XFR   BEGIN              BY CLOSE ROUTINE
         |        REPRO
         |        ACTION NODUP
         |        ORG   BEGIN
         *
                  DROP  9
                  CLOSE DATA1,CARD1
                  USING USING,9
         EOJOB    EOJ
         *                                 TERMINATION OF JOB
                  END   BEGIN
```

Example 11. Indexed-Sequential Load File and Card File, Part 2 of 2

Example 11 - Indexed-Sequential Load File and Card File (2501)

Records contained in punched cards are loaded onto disk as an indexed-sequential file. The overlay technique is used for the Open (3) and Close (6) routines. All I/O and work areas (1) must precede the overlay address BEGIN. The same applies to the error routines (2) which may be entered during the Close routine. In order to avoid a program loop, the address of the error routine is changed (5) to the EOJ address before the Close routine is entered. Processing consists of reading a card and writing the record onto disk (4).

This example requires 5570 bytes of main storage.

```
                TITLE 'IOCS EXAMPLE NO 12 - REORGANISATION OF AN ISFMS FILE'
        *
        *
                REPRO
                PHASE REORG1,A,4608
        *
                PRINT NOGEN
        START   START 0
                ORG   *+4608
        *
        *       FILE DEFINITIONS
        *       ----------------
        *
        DATA1   DTFIS DERREX=ERRC1,                                          -
                      DEVICE=DISK11F,                                        -
                      DSKXTNT=4,                                             -
                      EOFADDR=EOFC1,                                         -
                      IOAREAS=AREA1,                                         -
                      IOREG=(8),                                             -
                      IOROUT=RETRVE,                                         -
                      KEYARG=KEYA1,                                          -
                      KEYLEN=10,                                             -
                      KEYLOC=1,                                              -
                      NRECDS=16,                                             -
                      RECFORM=FIXBLK,                                        -
                      RECSIZE=80,                                            -
                      TYPEFLE=SEQNTL
        SQSAV   DTFSD BLKSIZE=1600,                                          -
                      DEVICE=DISK11F,                                        -
                      IOAREA1=AREA2,                                         -
                      IOREG=(9),                                             -
                      RECFORM=FIXBLK,                                        -
                      RECSIZE=80,                                            -
                      TYPEFLE=OUTPUT
                DTFEN
                EJECT
        *
        *       PROCESSING ROUTINE
        *       ------------------
        *
                USING START,0,1,2,3
        BEGIN   EQU   *
                OPEN  DATA1,SQSAV
                SETL  DATA1,BOF
        *
        LOOP1   EQU   *
                GET   DATA1
1 ------------> PACK  KEYPK(6),KEYA1(10)  SAVE THE KEY FOR LATER USE
        *                                  I.E. IF ERROR OCCURS
                MVC   0(80,9),0(8)         TRANSFER RECORD
2 ------------> PUT   SQSAV
                B     LOOP1
        *
        *       EXIT ROUTINES
        *       -------------
        *
        ERRC1   EQU   *                    READ ERROR ON DATA1
                HPR   X'F01'(1),0
        *
                ESETL DATA1
3 ------------> AP    KEYPK(6),=P'1'       INCREASE KEY BY ONE
                UNPK  KEYA1(10),KEYPK(6)   RESTORE THE KEY
                SETL  DATA1,GKEY
                B     LOOP1
        *
        EOFC1   EQU   *
                ESETL DATA1
                CLOSE DATA1,SQSAV
        *
                FETCH REORG2
```

Example 12. Indexed-Sequential File (Reorganization) and Sequential File, Part 1 of 3

```
*         DEFINITION OF CONSTANTS
*         -----------------------
*
AREA1     DS    10CL135
AREA2     DS    10CL160
*
KEYPK     DC    6C' '
KEYA1     DC    10C' '
*
          END   BEGIN
*
*
          TITLE 'IOCS EXAMPLE NO 12 - REORGANISATION OF AN ISFMS FILE'
*
*       I-----------------------------------------------------I
*       I THE FOLLOWING SECOND PHASE MUST BE SEPARATELY ASSEMBLED I
*       I-----------------------------------------------------I
*
          REPRO
          PHASE REORG2,A,4608
*
START     START 0
          ORG   *+4608
          PRINT NOGEN
*
*         FILE DEFINITIONS
*         ----------------
*
DATA2     DTFIS ALTREX=ALEX1,                                    -
                CYLOFL=3,                                        -
                CYNDEX=CYEX1,                                    -
                DERREX=ERRC1,                                    -
                DEVICE=DISK11F,                                  -
                DSKXTNT=5,                                       -
                DTAREX=DTEX1,                                    -
                DUPREX=DUEX1,                                    -
                IOAREAL=AREA1,                                   -
                IOROUT=LOAD,                                     -
                KEYLEN=10,                                       -
                KEYLOC=1,                                        -
                NRECDS=16,                                       -
                RECFORM=FIXBLK,                                  -
                RECSIZE=80,                                      -
                SQCHEX=SQEX1,                                    -
                WORKL=WORK1
SQSAV     DTFSD BLKSIZE=1600,                                    -
                DEVICE=DISK11F,                                  -
                EOFADDR=EOFC1,                                   -
                ERROPT=SKIP,                                     -
                IOAREA1=AREA2,                                   -
                RECFORM=FIXBLK,                                  -
                RECSIZE=80,                                      -
                TYPEFLE=INPUT,                                   -
                WORKA=YES
          DTFEN
          EJECT
*
*         PROCESSING ROUTINE
*         ------------------
*
          USING START,0,1,2,3
BEGIN     EQU   *
          OPEN  DATA2,SQSAV
          SETFL DATA2
*
LOOP1     EQU   *
4 ------> GET   SQSAV,WORK1
          WRITE DATA2,NEWKEY
          B     LOOP1
```

Example 12. Indexed-Sequential File (Reorganization) and Sequential File, Part 2 of 3

```
*
*        EXIT ROUTINES
*        -------------
*
CYEX1    EQU   *                    CYLINDER INDEX AREA OVERFLOW EXIT
DTEX1    EQU   *                    PRIME DATA AREA OVERFLOW EXIT
         HPR   X'F01',0             CHANGE EXTENT CARDS
*
ENDC1    EQU   *
         ENDFL DATA2
         CLOSE DATA2,SOSAV
         EOJ
*                                   TERMINATION OF JOB
*
ERRC1    EQU   *                    ERROR EXIT
         HPR   X'F01'(1),0          WRITE ERROR ON DISK
         B     ENDC1
*
ALEX1    EQU   *                    EXIT ON OCCUPIED LAST TRACK
5 -->SQEX1   EQU   *                EXIT ON SEQUENCE ERROR
DUEX1    EQU   *                    EXIT ON DUPLICATE RECORD
         HPR   X'F02'(1),0          ERROR HALT
         B     LOOP1
*
*        DEFINITIONS OF CONSTANTS
*        ------------------------
*
AREA1    DS    10CL135
AREA2    DS    10CL160
*
WORK1    DS    80C
*
         END   BEGIN
```

Example 12. Indexed-Sequential File (Reorganization) and Sequential File, Part 3 of 3

Example 12 - Indexed-Sequential File (Reorganization) and Sequential Disk File

This example illustrates the reorganization of an indexed-sequential file. The program consists of two phases which must be assembled separately. Records of the indexed-sequential file are retrieved and written back onto disk in sequential order (2). Then, in the second program phase, the sequential file is read and loaded onto disk in indexed-sequential order (4). To allow maximum time performance, the I/O areas must be as large as possible.

The error routine for the indexed-sequential input file skips all error records. When an error occurs, the last key saved is updated by one (3). The key is assumed to contain an unpacked decimal value. It is retrieved with the option GKEY. If another error occurs, the key is again retrieved and updated until the GET is executed without an error. The key save-field KEYPK is supplied with the key each time a GET has been executed (1).

The exits (5) are abnormal, i.e., they should only occur if there is an error in the system or in the IOCS routines.

This example requires 11,700 bytes of main storage in phase 1 and 12,470 bytes in phase 2.

Example 13 - Indexed-Sequential ADD File and Card File

Records read on a 2560 MFCM are added to an indexed-sequential file.

The processing routine reads the cards and adds the records to the disk file (1). If a duplicate record occurs no add operation is performed, and the card is selected (2) into stacker 4. If records are to be added to the last track of the prime data area, those cards are not added but selected (3) into stacker 5. They can be included in the file during a subsequent load/extension run. If all overflow areas are full (4), the job is terminated. The file must be reorganized before further records can be added. The error routine EREX1 checks for errors in the track or cylinder index (5). If an error occurs, the job is aborted.

This example requires 10,430 bytes of main storage.

```
          TITLE 'IOCS EXAMPLE NO 13 - ADDING RECORDS TO AN ISFMS FILE'
*
*
          START 0
          USING *,0,1,2,3
          ORG   *+4608
          PRINT NOGEN
*
*         FILE DEFINITIONS
*         ----------------
*
DATA1     DTFIS ADAREX=ADEX1,                                            -
                ALTREX=ALEX1,                                            -
                CYLOFL=3,                                                -
                DERREX=EREX1,                                            -
                DEVICE=DISK11F,                                          -
                DSKXTNT=4,                                               -
                DUPREX=DUEX1,                                            -
                ERRINF=YES,                                              -
                IOAREAL=AREA1,                                           -
                IOROUT=ADD,                                              -
                KEYLEN=10,                                               -
                KEYLOC=1,                                                -
                NRECDS=16,                                               -
                RECFORM=FIXBLK,                                          -
                RECSIZE=80,                                              -
                WORKA=AREA0,                                             -
                WORKL=WORK1
CARD1     DTFSR BLKSIZE=80,                                              -
                CONTROL=YES,                                             -
                DEVICE=MFCM1,                                            -
                EOFADDR=EOFC1,                                           -
                IOAREA1=AREA2,                                           -
                TYPEFLE=INPUT,                                           -
                WORKA=YES
*
*
          DTFEN OVLAY
          EJECT
*
*         DEFINITION OF CONSTANTS
*         -----------------------
*
AREA0     DS    80C                 IOAREAS FOR ADD FILE (MANDATORY SE-
AREA1     DS    10CL135             QUENCE)
AREA2     DS    80C                 INPUT AREA FOR CARD
*
WORK1     DS    80C                 WORK AREA
*
*
*
*         PROCESSING ROUTINE
*         ------------------
          DS    0H
BEGIN     EQU   *
          OPEN  DATA1,CARD1
*
          FETCH
          REPRO
          ACTION DUP                OPEN ROUTINE IS
          XFR   BEGIN               OVERLAID BY
          REPRO ,                   PROBLEM PROGRAM
          ACTION NODUP
          ORG   BEGIN
*
LOOP1     EQU   *
          GET   CARD1,WORK1
1-------> WRITE DATA1,NEWKEY
          WAITF DATA1
          B     LOOP1
```

Example 13. Indexed-Sequential ADD File and Card File, Part 1 of 2

```
          *
          *         EXIT ROUTINES
          *         -------------
          *
          DUEX1     EQU   *                    EXIT ON DUPLICATE RECORD
2 ------------->    CNTRL ,SS,4                SELECT INTO STACKER 4
                    B     LOOP1
          *
          ALEX1     EQU   *                    EXIT ON RECORDS FOR LAST TRACK
3 ------------->    CNTRL ,SS,5                SELECT INTO STACKER 5
                    B     LOOP1
          *
4 ------> ADEX1     EQU   *                    EXIT ON OVERFLOW AREA FULL
                    HPR   X'E01',0
                    B     EOFC1
          *
          EREX1     EQU   *                    EXIT ON PERMANENT DISK ERROR
                    HPR   X'E01'(1),0
                    CNTRL ,SS,5                SELECT INTO STACKER5 THOSE
          *                                    RECORDS THAT ARE NOT ADDED
5 ------------->    TM    DATA1A+2,B'00011000' TEST IF ERROR IN INDEX
                    BO    EOFC1                IF YES, ABORT THE JOB
          *
                    B     LOOP1
          *
          EOFC1     EQU   *                    EXIT ON EOF CONDITION
                    CLOSE DATA1,CARD1
                    EOJ
          *                                    TERMINATION OF JOB
                    END   BEGIN
```

Example 13. Indexed-Sequential ADD File and Card File, Part 2 of 2

Example 14 - Indexed-Sequential File (Random/Sequential ADDRTR) and Card File

Records of an indexed-sequential file are retrieved randomly or sequentially, and updated. The card input file is read on the 2501 Card Reader.

The cards contain the following information

cols.	1-10	key
col.	11	blank
cols.	12-13	number of records to be retrieved
cols.	15-20	update information
cols.	15-80	add information

The overlay technique for Open (1) and Close (19) is used. After reading (2) a card, a check is performed to determine whether the number of records to be retrieved is a valid decimal number. If it is invalid, it is set to C'00'. If only one record is to be retrieved (3), random processing is performed (LOOP3), i.e., the record is read (12), updated (13), printed (14), and, if desired, written back onto disk (15). If more than one record is to be retrieved, the number is packed (4) and the first record is retrieved sequentially with the option KEY. Columns 15 through 20 are checked for blanks (6). If they are not blank, the record is updated and the switch for writing (9) is changed (7) so that the record can be put onto disk. The switch is reset (5) before the next record is checked for update. Each record retrieved is printed (8). If the record count becomes zero (10), the next card is read (11, 2).

If a record could not be found, a new record is built and added to the file (16). This record is indicated as an addition (17) and printed. In the end-of-file routine for the indexed-sequential file (18) the next card is read. At the end of the card file, all files must be closed.

This example requires 8870 bytes of main storage.

```
         TITLE 'IOCS EXAMPLE NO 14 - RANSEQ ADD-RETRIEVE PROCESSING'
*
*
         PRINT NOGEN
         START 4608
         USING *-4608,0,1,2,3
         PRINT NOGEN
*
*        FILE DEFINITIONS
*        ----------------
*
DATA1    DTFIS ADAREX=ADEX1,                                            -
               CYLOFL=3,                                                -
               DERREX=EREX1,                                            -
               DEVICE=DISK11F,                                          -
               DSKXTNT=4,                                               -
               DUPREX=DUEX1,                                            -
               EOFADDR=EOFC1,                                           -
               IOAREAL=AREA1,                                           -
               IOAREAR=AREA2,                                           -
               IOAREAS=AREA3,                                           -
               IOREG=(8),                                               -
               IOROUT=ADDRTR,                                           -
               KEYARG=KEYA1,                                            -
               KEYLEN=10,                                               -
               KEYLOC=1,                                                -
               NRECDS=16,                                               -
               RECFORM=FIXBLK,                                          -
               RECSIZE=80,                                              -
               RTRVEX=RTEX1,                                            -
               TYPEFLE=RANSEQ,                                          -
               UPDATE=RANSEQ,                                           -
               WORKA=AREA0,                                             -
               WORKL=WORK1
CARD1    DTFSR BLKSIZE=80,                                              -
               DEVICE=READ01,                                           -
               EOFADDR=EOFC2,                                           -
               IOAREA1=AREA4,                                           -
               IOAREA2=AREA5,                                           -
               TYPEFLE=INPUT,                                           -
               WORKA=YES
PRINT    DTFSR BLKSIZE=100,                                             -
               CONTROL=YES,                                             -
               DEVICE=PRINTER,                                          -
               PRINTOV=YES,                                             -
               TYPEFLE=OUTPUT,                                          -
               WORKA=YES
         DTFEN OVLAY
         EJECT
*
*        DEFINITIONS OF CONSTANTS
*        ------------------------
*
AREA0    DS    80C                  IOAREAS FOR ADD
AREA1    DS    135CL10
AREA2    DS    135CL10              IOAREA FOR RANDOM RETRIEVAL
AREA3    DS    135CL10              IOAREA FOR SEQUENTIAL RETRIEVAL
AREA4    DS    80C                  IOAREAS FOR 2501
AREA5    DS    80C
*
WORK1    DC    80C' '               WORK AREA FOR CARD/DISK
WORK2    DC    C'DATA1 PROC'        WORK AREA FOR PRINTER
WORK3    DC    CL30'ESSED BY RANSEQ ADD-RETRIEVE W'
         DC    CL10'ITH UPDATE'
         DC    50C' '
KEYA1    EQU   WORK1
*
COUNT    DC    PL2'0'
*
         LTORG
```

Example 14. Indexed-Sequential File (Random/Sequential ADDRTR and UPDATED), Part 1 of 3

```
*
*        PROCESSING ROUTINE
*        ------------------
*
         DS    0H
BEGIN    EQU   *
         OPEN  DATA1,CARD1
*
         FETCH
         REPRO
         ACTION DUP                 OPEN ROUTINE IS
1 ------>XFR   BEGIN                OVERLAID BY
         REPRO ,                    PROBLEM PROGRAM
         ACTION NODUP
         ORG   BEGIN
*
         CNTRL PRINT,SK,1           SKIP
         PUT   PRINT,WORK2          PRINT HEADER
         CNTRL PRINT,SP,3
         MVC   WORK2(10),=CL10' '   CLEAR PART OF HEADER
LOOP1    EQU   *
         CNTRL PRINT,SP,1           SPACE ONE LINE
2 ------>GET   CARD1,WORK1
         CLI   WORK1+11,C'0'        CHECK FOR CORRECT TWO DIGIT DECIMAL
         BL    LOP12                NUMBER
         CLI   WORK1+11,C'9'
         BNH   LOP13
LOP12    MVI   WORK1+11,C'0'        IF FIRST DIGIT INCORRECT, SET IT 0
LOP13    CLI   WORK1+12,C'0'
         BL    LOP14
         CLI   WORK1+12,C'9'
         BNH   LOP15
LOP14    MVI   WORK1+12,C'0'        IF SECOND DIGIT INCORRECT, SET IT 0
LOP15    EQU   *
*
3 ------>CLC   WORK1+11(2),=C'01'  CHECK IF RANDOM PROCESSING REQUIRED
         BNH   LOOP3
*
*                                   SEQUENTIAL PROCESSING
*
         MVI   RTEX1+1,X'00'        SET SWITCH TO NOP
4 ------>PACK  COUNT(2),WORK1+11(2) PACK NO. OF RECORDS TO BE UPDATED
         SETL  DATA1,KEY
*
5 ------>MVI   SWCH1+1,X'F0'       SET SWITCH TO BRANCH
LOOP2    EQU   *
         GET   DATA1
6 ------>CLC   WORK1+14(5),=5C' '  CHECK IF UPDATE REQUIRED
         BE    LOP21
         MVC   45(5,8),WORK1+14     UPDATE RECORD
7 ------>MVI   SWCH1+1,X'00'       SET SWITCH TO NOP
LOP21    EQU   *
         MVC   WORK3(80),0(8)
         PRTOV PRINT,12
8 ------>PUT   PRINT,WORK2         PRINT RETRIEVED RECORD
*
9 ------>SWCH1 B     LOP22
         PUT   DATA1
LOP22    EQU   *
         SP    COUNT(2),=PL2'1'     CHECK IF NEXT RECORD IS TO BE
10 ----->CP    COUNT(2),=PL2'0'     UPDATED, IF NOT READ NEXT CARD
         BH    LOOP2
*
         ESETL DATA1
11 ----->B     LOOP1
```

Example 14. Indexed-Sequential File (Random/Sequential ADDRTR and UPDATE), Part 2 of 3

```
*
*                                       RANDOM PROCESSING
*
LOOP3    EQU   *
         MVI   RTEX1+1,X'F0'            SET SWITCH TO BRANCH
         MVI   SWCH2+1,X'F0'            RESET SWITCH TO BRANCH
12 ----> READ  DATA1,KEY
         WAITF DATA1
         CLC   WORK1+14(5),=C'  '       CHECK IF UPDATE REQUIRED
         BE    LOP31
13 ----> MVC   45(5,8),WORK1+14         UPDATE RECORD
         MVI   SWCH2+1,X'00'            SET SWITCH TO NOP I.E. UPDATE REQ.
LOP31    EQU   *
         MVC   WORK3(80),0(8)
         PRTOV PRINT,12
14 ----> PUT   PRINT,WORK2
*
         WAITF DATA1
SWCH2    B     LOOP1
15 ----> WRITE DATA1,KEY                WRITE UPDATED RECORD ON DISK
         WAITF DATA1
         B     LOOP1
*
*        EXIT ROUTINES
*        -------------
*
RTEX1    NOP   RTEX2                    EXIT IF NO RECORD HAS BEEN FOUND
         ESETL DATA1
RTEX2    EQU   *
16 ----> WRITE DATA1,NEWKEY
         MVC   WORK3(80),WORK1          MOVE RECORD TO PRINT AREA
17 ----> MVC   WORK2+5(3),=C'ADD'       MOVE ADD IDENTIFIER
         PRTOV PRINT,12
         PUT   PRINT,WORK2
         MVC   WORK2+5(3),=C'   '       CLEAR ADD IDENTIFIER
         WAITF DATA1
         B     LOOP1
*
18 ----> EOFC1 B     LOOP1              EXIT ON EOF CONDITION
*
ADEX1    EQU   *                        EXIT ON OVERFLOW AREA FULL
         HPR   X'F01',0                 HALT INDICATES NO ADD POSSIBLE
         B     LOOP1
*
DUEX1    EQU   *                        EXIT ON DUPLICATE RECORD
         HPR   X'F02',0                 ABNORMAL HALT
         B     EOJOB
*
EREX1    EQU   *                        PERMANENT DISK ERROR
         HPR   X'F01'(1),0              ERROR HALT
         B     EOJOB
*
EOJOB    EQU   *                        END OF JOB HANDLING
EOFC2    EQU   *
*
         FETCH
         LTORG
         REPRO ,                        PROBLEM PROGRAM
         ACTION DUP                     IS OVERLAID
19 ----> XFR   BEGIN                    BY CLOSE ROUTINE
         REPRO
         ACTION NODUP
         ORG   BEGIN
*
         CLOSE DATA1,CARD1,PRINT
         EOJ
*                                       TERMINATION OF JOB
         END   BEGIN
```

Example 14. Indexed-Sequential File (Random/Sequential ADDRTR and UPDATE), Part 3 of 3

```
                TITLE 'IOCS SAMPLE NO 15 - INQUIRY PROGRAM'
*
                START 4608
                PRINT NOGEN
*
*               FILE  DEFINITIONS
*               ----------------
                DTFBG INQPRG=YES
DATA1           DTFIS DERREX=ERRC1,                                    -
                      DEVICE=DISK11F,                                  -
                      DSKXTNT=4,                                       -
                      EOFADDR=EOFC1,                                   -
                      IOAREAS=AREA1,                                   -
                      IOROUT=RETRVE,                                   -
                      KEYARG=KEYA1,                                    -
                      KEYLEN=10,                                       -
                      KEYLOC=1,                                        -
                      NRECDS=16,                                       -
                      RECFORM=FIXBLK,                                  -
                      RECSIZE=80,                                      -
                      RTRVEX=RTEX1,                                    -
                      TYPEFLE=SEQNTL,                                  -
                      WORKS=YES
OUTPT           DTFPK TYPEFLE=OUTPUT,                                  -
                      BLKSIZE=80,                                      -
                      WORKA=YES
                DTFEN
                EJECT
*               PROCESSING ROUTINE
*               ------------------
BEGIN           EQU   *
                BASR  9,0
                USING *,9
                OPEN  DATA1,OUTPT
1 ------------> IOIPT ,                      LOAD ADDRESS OF INQUIRY RECORD
*                                             INPUT AREA INTO REGISTER 8
2 ------------> MVC   KEYA1(10),0(8)         INSERT KEY INTO KEYARG FIELD
                CLC   11(3,8),=C'   '         CHECK IF NO. OF RECORDS IS INDICATED
                BE    LOOP1
3 ------------> PACK  COUNT(2),11(3,8)       PACK NUMBER INTO RECORD COUNT
LOOP1           SETL  DATA1,GKEY
LOOP2           EQU   *
4 ------------> GET   DATA1,WORK1
                PUT   OUTPT,WORK1
                SP    COUNT(2),=PL2'+1'     CHECK IF ANOTHER RECORD IS REQUIRED
5 ------------> CP    COUNT(2),=PL2'0'
                BH    LOOP2                   IF YES RETURN INTO LOOP2
CLOSE           EQU   *
                ESETL DATA1
                CLOSE DATA1,OUTPT
                EOJ
*                                             TERMINATION OF JOB AND RETURN TO
*                                             MAINLINE PROGRAM
*               EXIT ROUTINES
*               -------------
ERRC1           EQU   *                       EXIT FOR DISK ERROR
                HPR   X'E01'(1),0             HALT INDICATES ALTERNATE DISK
*                                             ASSIGNMENT IS REQUIRED
EOFC1           EQU   *                       END OF FILE EXIT
RTEX1           EQU   *                       NO RECORD FOUND EXIT
                B     CLOSE
*
*               DEFINITION OF CONSTANTS
*               -----------------------
AREA1           DC    135CL10' '
WORK1           DC    80C' '
KEYA1           DC    10C' '                  KEYARG FIELD
COUNT           DC    PL2'+1'
                LTORG
                END   BEGIN
```

●Example 15. Indexed-Sequential File and Printer Keyboard File in Inquiry Program

Example 15 - Indexed-Sequential File and Printer-Keyboard File in Inquiry Program

The inquiry program retrieves records of an indexed-sequential file by key.

The key has the following format:

10 bytes	key information
1 byte	blank
3 bytes	number of records to be retrieved.

The printer-keyboard input area (INQIPT) in the Monitor must be at least 14 bytes long. The IQIPT macro instruction (1) places the address of the input area that is to contain the inquiry record into register 8. The key is moved to the KEYARG field (2) and the record count is initialized (3). Records are retrieved (4) and printed until the count is zero (5). If it is zero, EOJ is entered and control is returned to the mainline program.

This example requires 4660 bytes of main storage.

```
                TITLE 'IOCS EXAMPLE NO 16 - AN ATENT ROUTINE'
*
*
                START 4608
                USING *-4608,0,1,2,3
                PRINT NOGEN
*
*               FILE DEFINITIONS
*               ----------------
*
                DTFBG ATENT=YES
DATA1           DTFSD BLKSIZE=800,                                     -
                      DEVICE=DISK11F,                                  -
                      EOFADDR=EOFC1,                                   -
                      IOAREA1=AREA1,                                   -
                      RECFORM=FIXBLK,                                  -
                      RECSIZE=80,                                      -
                      TYPEFLE=INPUT,                                   -
                      UPDATE=YES,                                      -
                      WORKA=YES
PRINT           DTFSR BLKSIZE=80,                                      -
                      CONTROL=YES,                                     -
                      DEVICE=PRINTER,                                  -
                      PRINTOV=YES,                                     -
                      TYPEFLE=OUTPUT,                                  -
                      WORKA=YES
PUTIN           DTFPK BLKSIZE=81,                                      -
                      EOFADDR=EOFC2,                                   -
                      IOAREA=AREA2,                                    -
                      TYPEFLE=INPUT
TYPEN           DTFPK BLKSIZE=80,                                      -
                      IOAREA=AREA3,                                    -
                      TYPEFLE=OUTPUT,                                  -
                      WORKA=YES
                DTFEN
                EJECT
```

Example 16. Sequential Disk, Printer, and Printer-Keyboard Files - ATENT Routine, 1 of 3

```
         *          PROCESSING ROUTINE
         *          ------------------
         *
         BEGIN      EQU   *
                    OPEN  PRINT,PUTIN,TYPEN
                    CNTRL PRINT,SK,1
         *
    1--> OPEN1      OPEN  DATA1
         SWCH0      MVI   SWCH3+1,X'F0'       MODIFY SWITCH FOR UPDATE
                    MVI   SXCH0+1,X'F0'       RESET SWITCH MODIFIER
         *
         LOOP1      EQU   *
                    GET   DATA1,WORK1
         *
    2--> SWCH1      B     SWCH2               SWITCH FOR TYPING
                    PUT   TYPEN,WORK1
         *
    3--> SWCH2      B     SWCH3               SWITCH FOR PRINTING
                    PRTOV PRINT,12
                    PUT   PRINT,WORK1
         *
    4--> SWCH3      B     LOOP2               SWITCH FOR UPDATE
                    PUT   DATA1,WORK2
         LOOP2      EQU   *
                    B     LOOP1
         *          ATENT ROUTINE
         *          -------------
         *
                    ATENT
         *
    5--------------> READ  PUTIN               READ CONTROL INFORMATION
                    WAITF PUTIN
                    LH    POINT,=Y(AREA2)     LOAD POINTER WITH AREA2 ADDRESS
         ATE00      EQU   *
    6--------------> CLC   0(3,POINT),=C'EOJ'  DECISION AND MODIFING ROUTINE
                    BNE   ATE02
    7--------------> MVI   SWCH4+1,X'F0'       ACTIVATE EOJ FOR NEXT EOF CONDITION
    8--> ATE02      CLC   0(4,POINT),=C'TYPE'
                    BNE   ATE04
                    MVI   SWCH1+1,X'00'       ACTIVATE TYPING
    9--> ATE04      CLC   0(6,POINT),=C'NOTYPE'
                    BNE   ATE06
                    MVI   SWCH1+1,X'F0'       DEACTIVATE TYPING
   10--> ATE06      CLC   0(5,POINT),=C'PRINT'
                    BNE   ATE08
                    MVI   SWCH2+1,X'00'       ACTIVATE PRINTING
   11--> ATE08      CLC   0(7,POINT),=C'NOPRINT'
                    BNE   ATE10
                    MVI   SWCH2+1,X'F0'       DEACTIVATE PRINTING
         *
   12--> ATE10      CLI   AREA2,C'='          CHECK IF UPDATE REQUIRED
                    BNE   ATE20
   13--------------> MVC   WORK2(80),AREA2+1   MOVE UPDATE TEXT
   14--------------> MVI   SWCH0+1,X'00'       ACTIVATE UPDATE FOR NEXT REOPEN
         *
         ATE20      AH    POINT,=H'1'         UPDATE AREA2 POINTER
   15--------------> CH    POINT,=Y(AREA2+81)
                    BNL   ATE30
                    CLI   0(POINT),C','
                    BNE   ATE20
                    AH    POINT,=H'1'
                    B     ATE00
         *
   16--> ATE30      RETRN
         *
         *          END OF ATENT ROUTINE
```

Example 16. Sequential Disk, Printer, and Printer-Keyboard Files - ATENT Routine, 2 of 3

```
         *        EXIT ROUTINES
         *        ------------
         EOFC1    EQU   *                    EOF CONDITION FOR SEQUENTIAL DISK
17 ---> SWCH4    NOP   EOFC2
                 CLOSE DATA1
                 B     OPEN1
         EOFC2    EQU   *
18 --->          CLOSE DATA1,PRINT,PUTIN,TYPEN
                 EOJ
         *                                   TERMINATION OF JOB
         *
         *
         *        DEFINITION OF CONSTANTS
         *        -----------------------
         *
         AREA1    DC    80CL10' '           INPUT AREA FOR DATA1
         AREA2    DC    82C' '              INPUT AREA FOR CONTROL INFORMATION
         AREA3    DC    80C' '              OUTPUT AREA PRINTER KEYBOARD
         *
         WORK1    DC    80C' '              WORK AREA FOR DATA1
         WORK2    DC    80C' '              WORK AREA FOR UPDATE DATA1
         *
         POINT    EQU   10                  POINTER FOR AREA2
         *
                  LTORG
         *
                  END   BEGIN
```

Example 16. Sequential Disk, Printer, and Printer-Keyboard Files - ATENT Routine, 3 of 3

Example 16 - Sequential Disk, Printer, and Printer-Keyboard Input/Output Files - Program Includes ATENT Routine

An inquiry Monitor is required for this example, which demonstrates the use of the ATENT routine. The program reads a sequential disk file and is modified by control information supplied in the ATENT routine. Valid control information is:

```
/*      - Close files and terminate job.
EOJ     - Control is transferred to EOJ
          with the next CLOSE.
TYPE    - All records are typed on the
          printer-keyboard.
NOTYPE  - Typing of records is suppressed.
PRINT   - All records are printed.
NOPRINT - Printing of records is suppressed.
```

Incorrect control information is ignored. More than one option can be given at a time, if separated by a comma, e.g., NOPRINT, EOJ.

In the mainline program, a sequential disk file is read. On end-of-file, the disk file is closed (17), reopened (1), and read again. Print, type, EOJ, and update options are activated and deactivated by the ATENT routine. SWCH1 controls typing on the printer-keyboard (2), SWCH2 provides for printing of records (3), and SWCH0 and SWCH3 control updating of records (4). SWCH4 determines (17) whether or not the job is to be terminated if the next end-of-file condition for DATA1 (disk file) is encountered. After the ATENT routine has been entered, control information is entered on the printer-keyboard (5).

Checking and modifying is performed according to this control information (6) - (14). If, for instance, the check for EOJ (6) is true, SWCH4 (17) is set to branch (7), i.e., all files are closed and the job is terminated (18) the next time an end-of-file condition is detected. Likewise, tests for typing (8), suppression of typing (9), printing (10), and suppression of printing (11) are performed. If updating is required (12), i.e., if an equal sign is typed in, the update information is entered immediately after the equal sign and moved to the work area (13). SWCH0 is modified to change (14) SWCH3 to NOP. The next time the file is reopened, the whole file will be updated. A register (POINT) is used to check (15) whether all options have been processed. If this is the case, the RETRN macro instruction (16) is executed. If /* is typed in on the printer-keyboard, all files are closed (18) and the job is terminated.

This example requires 4230 bytes of main storage.

Glossary

Access Method: Any of the data management techniques available for transferring data between main storage and an input/output device.

Access Time: (1) The time interval between the instant at which data is called for from a storage device and the instant delivery is completed, i.e., the read time. (2) The time interval between the instant at which data is requested to be stored and the instant at which storage is completed, i.e., the write time.

Allocate: To assign storage locations or areas of storage for specific routines, portions of routines, constants, data, etc.

Alternate Drive: When two drives are given for one multi-volume file, the first drive is the primary drive and the second drive is the alternate drive. Tape reels or disk packs are mounted such that the first is on the primary drive, the second on the alternate drive, the third on the primary drive, etc.

Alternate Track Area: An area of three cylinders on the disk pack in which tracks may be used as alternatives to defective tracks occurring elsewhere on the disk pack.

Assemble: To prepare a machine-language program from a symbolic-language program by substituting absolute operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses.

Assembler: A program that prepares an object language program by producing absolute or relocatable machine code from a machine-oriented source program of statements containing symbolic operation codes and symbolic operands.

Assembler Language: A symbolic language (used to write source programs) which enables the programmer to use all machine functions as if he were coding in machine language.

b: The symbol for a blank space.

Binary Synchronous Communications Adapter (BSCA): A feature that may be built into the Central Processing Unit of a Submodel 2, 4, or 5. It permits the system to function on a switched or leased communications network as a processor terminal.

Block:
1. To group records for the purpose of conserving storage space or increasing the efficiency of access or processing.
2. A physical record on tape or disk.

Blocking Factor: The number of logical records in a physical record.

Buffer (Program Input/Output): A portion of main storage into which data is read, or from which it is written.

Checkpoint Records: Records that contain the status of the job and the system at the time the records are written by the checkpoint routine. These records provide the necessary information for restarting a job without having to return to the beginning of the job.

Communication Region: An area of the Monitor. Contains date, storage-capacity specification, UPSI byte, user areas 1 and 2, program-name area, and various control bits used by the system. Provides for communication within a program and between programs.

Data File: A collection of related records treated as a unit and consisting of data in one of several prescribed arrangements and described by control information to which the system has access.

Data Management: See File Management.

Data Set: See Data File.

Deblock: To change the format of a file so that a physical record comprises only one logical record.

Direct Access: Retrieval or storage of data by a reference to its location on a volume, rather than relative to the previously retrieved or stored data.

Disk Label: A physical identification record on disk which identifies the volume or file.

EOF Record: End-of-file record which terminates a logical set of input records (/*b in columns 1 through 3).

Extent: Area of a disk file specified by an upper limit and a lower limit.

File Label: Label containing information applicable to a given data file or portion

of a data file stored on a particular volume.

File Management: A general term that collectively describes those functions of the control program that provide access to files, enforce data storage conventions, and regulate the use of input/output devices.

File Organization: Refers to the method of arranging data records on an external storage device.

File Processing: The method of retrieving records from, adding records to, or updating records in a file.

File Reorganization: A term used to describe the process of writing a new file from an indexed-sequential file, purging records that are tagged for deletion, and placing records in the overflow area into their sequential positions in the prime data area.

Fixed-Length Record: A record having the same length as all other records with which it is logically or physically associated.

Index (Data Management):
1. A table in the catalog structure used to locate files.
2. A table used to locate the records of an indexed-sequential file.

Inquiry Programs: Inquiry programs are initiated by pressing the Request key on the printer-keyboard and typing in the name of the program. The current contents of main storage (excluding the Monitor) are rolled out on the system disk pack; then the inquiry program is loaded and processed; after execution is completed, the old status is restored and execution of the mainline program resumes. Inquiry programs can be executed only under control of a Monitor that supports inquiry facilities.

Inter-Block Gap: A blank space on magnetic tape that separates physical records.

I/O Area: An area (portion) of main storage into which data is read or from which data is written.

Logical File: Used to describe a file that shares a reel of tape or a disk with other files.

Logical Record: A record identified from the standpoint of its content, function, and use rather than its physical attributes. It is meaningful with respect to the information it contains.

Logical Unit Block (LUB): An entry in the Logical Unit Table.

Logical Unit Table: A part of the Monitor. It has logical unit blocks, each of which refers to one specific symbolic I/O address. These symbolic addresses are related to actual I/O device addresses by means of ASSGN control statements.

Macro Definition: A set of statements in the macro library used by the DPS/TPS Assembler program to expand a macro instruction specified in the source program into a series of machine instructions.

Macro Instruction: A macro instruction is a statement that is used in a source program and replaced by a specific sequence of machine instructions in the associated object program.

Mnemonic: A contraction or abbreviation whose characters are suggestive of the full expression.

Monitor: The main control program in DPS. Resident in main storage throughout a system run. The IBM distribution package contains the standard Monitor and several Monitor macro definitions. Instead of employing the standard Monitor, you can tailor a Monitor according to the system requirements by specifying certain macro instructions, and generate it by means of an assembly run.

Monitor I/O Area. An area of main storage within the Monitor used as a buffer by the Fetch routine when loading problem programs.

Object Program: The output of a single execution of an assembler or compiler.

Operand: The representation of a value that must be supplied to define a selective function to the program.

Overlay: To place a phase or subphase into main storage locations occupied by another phase or subphase that has already been processed.

Phase: (1) A portion of a program executed as one main-storage load. (2) The smallest addressable unit in the core-image library of a tape or disk-resident system.

Physical Disk and Tape I/O Routines: A set of routines that is contained in the Monitor program and performs tape and disk I/O operations for the Monitor and problem programs.

Physical Unit Block (PUB): An entry in the Physical Unit Table.

Physical Unit Table: A feature of the Monitor program. It has up to ten physical unit blocks, each of which contains a phy-

sical device address. Pointers to these entries are inserted into the logical unit table by means of ASSGN control statements.

Physical Record: A record identified from the standpoint of the manner or form in which it is stored and retrieved; that is, one that is meaningful with respect to access. (Contrasted with Logical Record.)

Problem Program: A general term for any program that is not a control program.

Read/Compute, Write/Compute Overlap Feature: A feature of the IBM System/360 Model 20, Submodel 5 that permits data transfer from or to I/O units to be overlapped with processing.

Reblock: To change the format of a file so that a different number of logical records comprises one physical record. See Block.

Record: A general term for any unit of data that is distinct from all others when considered in a particular context.

Restart: To re-establish the status of a job using the information recorded at a checkpoint.

RWC feature: See Read/Compute, Write/Compute Overlap Feature.

Seek: To position the access mechanism of a direct-access device at a specified location.

Source Program: A series of statements in the symbolic language of an assembler or compiler, which constitutes the entire input to a single execution of the assembler or compiler.

Stacked Job Processing: A technique that permits multiple job definitions to be grouped (stacked) for presentation to the system, which automatically recognizes the jobs, one after the other.

Statement: A meaningful expression or generalized instruction in a source language.

Subphase: A separately assembled routine within a phase of a problem program. It may be overlaid after execution. The method of building a program from subphases is used when a large problem program is to be executed.

Symbolic I/O Address: A symbol used in IBM-supplied and user-written programs to refer to an I/O device (e.g., SYSRES, SYSIPT, SYS005). This address is related to an actual address by means of the logical unit table.

SYSIPT: See System Input Unit.

System Disk Pack: The disk pack on which your disk-resident system is stored.

System Input Unit: A device specified as a source of an input job stream.

System Tape: The reel of magnetic tape on which the tape-resident system is located.

Tape Labels: Special records at the beginning and end of tape files. There are volume, header, and trailer labels. They are used to identify the reel of tape and the file they precede. They also contain certain housekeeping information.

Tapemark: A special symbol that can be read from, or written on, magnetic tape. Used to distinguish the end of a file or file segment, and to segregate the labels from data.

Throughput: A measure of system efficiency: the rate at which work can be handled by a computing system.

Unblock: To change the format of a file so that a physical record comprises only one logical record. See Block.

Variable-Length Records: Logical records in a file in which the number of bytes in each record is not a fixed value, but may vary within prescribed limits.

Volume: That portion of a single unit of storage media that is accessible to a single read/write mechanism. For example, a reel of magnetic tape for a 2415 magnetic tape drive, or one 1316 Disk Pack for a 2311 Disk Storage Drive.

Volume Label: A label which uniquely identifies a volume.

Volume Table of Contents (VTOC): A table associated with a direct-access volume, which describes each data set on the volume.

Index

IBM System/360 Printed in U.S.A. C24-9007-5

IBM®

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

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

READER'S COMMENT FORM

IBM System/360 Model 20
Disk Programming System
Input/Output Control System

Form C24-9007-5

- How did you use this publication?

  As a reference source ..... ☐
  As a classroom text ..... ☐
  As a self-study text ..... ☐

- Based on your own experience, rate this publication . . .

	.....	.....	.....	.....	.....
As a reference source:	Very Good	Good	Fair	Poor	Very Poor
As a text:	Very Good	Good	Fair	Poor	Very Poor

- What is your occupation? .....

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

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

YOUR COMMENTS, PLEASE . . .

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

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

Fold

Fold

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

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation

112 East Post Road

White Plains, N. Y. 10601

Attention: Department 813 BP

Fold

Fold

IBM

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

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

CUT ALONG THIS LINE

IBM System/360 Printed in U.S.A. C24-9007-5