File S360(Mod 20)-30
Form C24-9003-4

TPS

IBM Systems Reference Library

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

This publication describes the functions, principal features, and use of the Input/Output Control System supplied by IBM as part of the Model 20 Tape Programming System (TPS). This Input/Output Control System (IOCS) facilitates the programming of input/output operations.

The following subjects are covered: (1) the definition statements that describe the files to be processed, (2) the initialization macro instruction that makes the files available to the system for data input or data output, (3) the processing macro instructions that cause input/output operations, and (4) the completion macro instructions that terminate processing of data for one or more of the files used in an application.

Also included is a section containing general programming considerations, e.g., information about blocking and deblocking of records, combinations of input/output and work areas, tape error routines, and register usage. Programming examples are also given.

Readers of this publication should be thoroughly familiar with the contents of the SRL publication IBM System/360 Model 20, Functional Characteristics, Form A26-5847.

Titles and abstracts of other related publications are given in the publication IBM System/360 Model 20 Bibliography, Form A 26-3565.

This publication is intended for Model 20 programmers using the TPS Input/Output Control System.

The reader should be familiar with basic programming concepts and with the operating principles of his system as described in the following SRL publications:

IBM System/360 Model 20

- Functional Characteristics, Form A26-5847;
- Tape Programming System, Control and; Service Programs, Form C24-9000;
- Disk and Tape Programming Systems, Assembler Language, Form C24-9002.

Depending on the equipment and programs used, the following publications are also required:

IBM System/360 Model 20

- Disk and Tape Programming Systems, Input/Output Control System for the IBM 1419 and 1259 Magnetic Character Readers, Form C33-6001;
- Input/Output Control System for the Binary Synchronous Communications Adapter, Form C33-4001;

Titles and abstracts of other related publications are given in the publication IBM System/360 Model 20 Bibliography, Form A26-3565.

Fifth Edition (March, 1969)

This is a major revision of, and obsoletes, C24-9003-3, and Technical Newsletter N33-8552. The changes are associated mainly with the introduction of Submodel 5 of the Model 20. 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.

This edition applies to program version 4, modification level 0 of IBM System/360 Model 20 TPS IOCS, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM Systems, consult the latest SRL Newsletter, Form N20-0361, for the edition that is 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 is provided at the back of this publication for readers' comments. If the form has been removed, comments may be addressed to IBM Laboratory, Publications Dept., P.O.Box 24, Uithoorn/Netherlands.

This publication describes the functions, principal features, and use of the Model 20 Input/Output Control System for punched-card equipment and magnetic tape.

The Model 20 TPS Input/Output Control System (IOCS) is a set of tested routines, in Assembler language, provided by IBM. The IOCS routines are part of the Model 20 Tape Assembler macro library. The programmer can utilize these routines by simply issuing appropriate macro instructions in his program.

A major part of most programs written in Assembler language consists of routines required to read data into the system and to print (punch or write on tape) the results of the processing performed on the input data. By utilizing the IOCS routines, the programmer can save programming time because he can concentrate on solving his problems and let the IOCS handle data input and output operations.

Also, the IOCS routines take advantage of the time sharing feature (this is a means of overlapping input/output operations with each other and with processing) of the Model 20, thereby optimizing throughput. When a Model 20 Submodel 5 is used, the IOCS can make use of the read/compute, write/compute overlap feature, which allows tape-data transfer to be overlapped with processing.

The IOCS provides routines for:

- transfer of data from input/output devices to main storage and vice versa,
- checking and writing of labels (if any),
- blocking and deblocking of records,
- switching between input/output areas under certain conditions, and
- performing input/output control functions such as card stacking, tape rewind, etc.

USE AND FUNCTIONS OF IOCS

When a program utilizing the IOCS is assembled, the macro instructions specify which of the IOCS routines are to be called from the macro library. The routines are extracted, tailored according to the operands in the macro instructions, and inserted into the source program. The complete program now consists of both source program statements and tailored routines from the macro library in Assembler language. In subsequent phases of the assembly, the entire object program is assembled.

Two types of macro instructions are required to cause the desired input/output functions: declarative macro instructions, which are referred to as definition statements, and imperative macro instructions.

The programmer uses definition statements to specify the input/output routines he requires for his particular application. Based on the information provided, these routines are selected and developed at assembly time.

A linkage to the selected input/output routines is required at each point in the program where an input/output operation is to occur. The user need not provide these linkages. He only writes an imperative macro instruction in his source program at the point he desires the input/output operation to occur. When an imperative macro instruction is read during assembly of the source program, the Assembler automatically inserts the required linkages to the selected input/output routines.

Other Programs Used by the IOCS

When a program using the IOCS is executed, some of the Basic Monitor routines are used. Therefore, programs using the IOCS require the Basic Monitor program to be in main storage when they are being executed. The Job Control program is required if the user's program requires job control cards to be read at the beginning of program execution. The Job Control program is also required if the IOCS is to make use of the read/compute, write/compute overlap feature.

For further information regarding the Basic Monitor and the Job Control programs, refer to the SRL publication IBM System/360 Model 20, Tape Programming System, Control and Service Programs Form C24-9000.

MACHINE REQUIREMENTS

This section describes the minimum and maximum configurations for assembling and executing IOCS routines of the tape-resident system.

Minimum System Configuration

Submodel 2

- An IBM 2020 Central Processing Unit, Model C2 (8192 bytes of main storage);
- an IBM 2415 Magnetic Tape Unit, Model 2, 3, 5, or 6;
- 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, Model A1;
- One of the following printers:
  IBM 1403 Printer, Model N1, 2, or 7,
  IBM 2203 Printer, Model A1.

Submodel 5

- An IBM 2020 Central Processing Unit, Model C5 (8192 bytes of main storage);
- the same input/output units as described above for Submodel 2.

Maximum System Configuration

Submodel 2

- An IBM 2020 Central Processing Unit, Model D2 (16,384 bytes of main storage);
- an IBM 2415 Magnetic Tape Unit, Model 1 through 6;
- an IBM 1442 Card Punch, Model 5;
- an IBM 2501 Card Reader, Model A1 or A2;
- one of the following card units:
  IBM 2520 Card Read Punch, Model A1,
  IBM 2520 Card Punch, Model A2 or A3,
  IBM 2560 Multi-Function Card Machine, Model A1;
- one of the following printers:
  IBM 1403 Printer, Model N1, 2, or 7,
  IBM 2203 Printer, Model A1;
- a Binary Synchronous Communications Adapter (Feature No.2074);
- one of the following magnetic character readers:
  IBM 1419 Magnetic Character Reader, Model 1 or 31,
  IBM 1259 Magnetic Character Reader, Model 1, 31, or 32.

Submodel 5

- An IBM 2020 Central Processing Unit, Model E5 (32,768 bytes of main storage);
- the same input/output units as described above for Submodel 2.

Notes:

1. Only three tape drives are required for assembly if no literals are used in the source program and no object-program output on tape is required.
2. At least one 9-track tape drive is required. If 7-track tapes are used, the data conversion feature is required if:
   - EBCDIC characters other than those included in the ECD character set are to be written, or
   - the tape file to be read has been created using the data conversion feature.

   The ECD character set consists of the digits 0 through 9, the alphabet, and 28 special characters, all of which can be represented by the six rightmost bits of a byte.

   The translate feature is required if 7-track tapes are used and standard labels must be read from or written on these tapes.

MACHINE FEATURES SUPPORTED

The following input/output devices are supported by the IOCS:

1403	Printer
1442-5	Card Punch
2203	Printer, standard and dual feed carriage
2501	Card Reader
2520	Card Punch
2520	Card Read-Punch
2560	Multi-Function Card Machine (MFCM), including Card Print feature
2415	Magnetic Tape Unit and Control, Models 1 through 6
1259	Magnetic Character Reader
1419	Magnetic Character Reader, Models 1 and 31.

The IOCS also supports:

- Binary Synchronous Communications Adapter (Feature No. 2074).
- Additional main storage up to 16,384 (Submodel 2), or 32,768 bytes (Submodel 5).

Note 1: The Data Conversion feature is required if 7-track tapes are to be used and either

a. EBCDIC characters other than those included in the BCD character set are to be written, or
b. the tape file to be read has been created using the Data Conversion feature.

The BCD character set consists of the digits 0 through 9, the alphabet, and 28 special characters, all of which can be represented by the six rightmost bits of a byte.

Note 2: The Translate feature is required if 7-track tapes are used and standard labels must be read from or written on these tapes.

Note 3: The object program can make use of the Read/Compute, Write/Compute Overlap feature if a Submodel 5 is used.

DEFINITIONS

RECORD

A record is a unit of information comprised of one or more alphameric and/or special characters. For the purpose of the IOCS, such a unit of information is referred to as a logical record. For transfer from the CPU to magnetic tape and vice versa, data is compiled in physical units of information that are referred to as blocks. A block may consist of one logical record only or an integer multiple thereof.

The IOCS accepts three record formats. They are: (1) fixed length, (2) variable length, and (3) undefined. Fixed and variable length records may be blocked or unblocked. The record formats and the allowable record types are described below:

Fixed-Length Records

Fixed-length records are logical records within a set of records that are all of the same length. They may be blocked or unblocked. If these records are unblocked, the IOCS handles each logical record as a block. If they are blocked (applies only to records on magnetic tape), one or more logical records of fixed length comprise one block. Figure 1 shows an example of fixed-length records on tape.

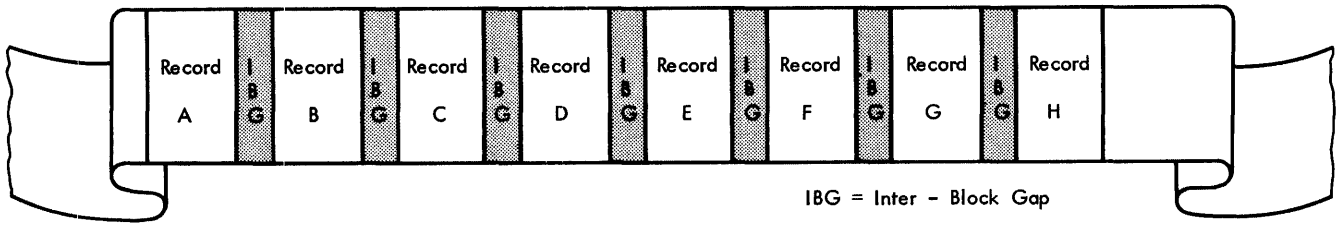
Variable-Length Records

Variable-length records are logical records of a set of records that vary in length (applies only to records on magnetic tape). They may be blocked or unblocked. If these records are unblocked, the IOCS handles each logical record the same way as a block. If they are blocked, one or more records of variable length make up one block.

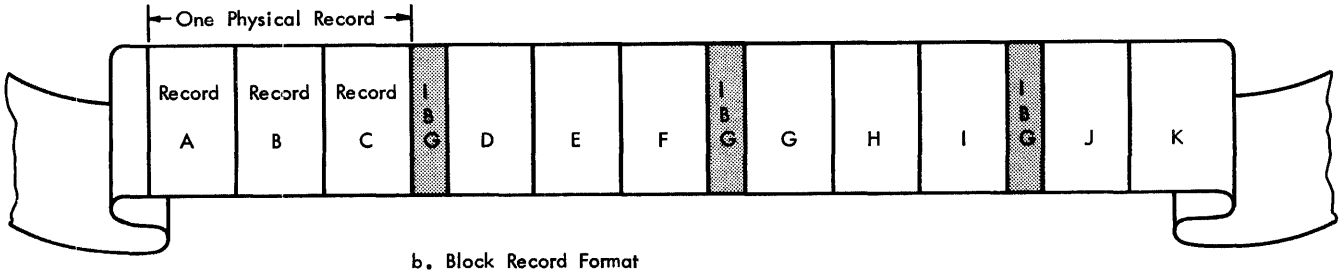
A variable-length record must contain a record-length indication. The first four bytes of a variable-length record are used for this purpose. The first two of these four bytes show the number of bytes contained in the record in binary notation. The remaining two bytes contain binary zeros.

The record-length indication must be provided by the user whenever he is creating a variable-length record. The four bytes required for the length indication are included in the byte count for the record.

A block-length indication is required for each block. This block-length indication consists of four bytes that precede the record-length indication for the first (or only) logical record of the block.



a. Unblocked Record Format



b. Block Record Format

Figure 1. Example of Fixed-Length Records on Tape

The first two of these four bytes indicate, in binary notation, the number of bytes contained in the block. The remaining two bytes contain binary zeros. The block-length includes the four bytes for the block-length field itself.

Although the block-length indication does not appear in the record that is furnished to or made available by the problem program, the programmer must define input/output areas that are large enough to accommodate the four bytes required for the block-length indication.

Figure 2 is a schematic representation of variable-length records on tape.

Undefined Format

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 under control of the problem program.

Record Formats Permitted

The formats that can be used depend on the type of input/output file as follows:

Card and Printer Files: Only unblocked records of fixed length are allowed.

Tape Files: Both blocked and unblocked records of fixed and variable length and records of undefined format are allowed, except when a file is to be read backward. When a file is to be read backward, variable-length blocked records are not allowed.

FILE

A file is a set of records that contains related information, e.g. an inventory file of part numbers, an employee file, or a customer file. Such a set of records may be punched into cards (card file), printed on forms (printer file), or written on tape(s) (tape file). For the purpose of the Model 20 IOCS, files can be of two types: simple and combined.

1 Simple File
A set of records that are all either read, printed, punched, card printed, or written on tape during one pass through the system.
Exception: A simple file that is either read or punched on the 2560 MFCM may also be card printed during the same pass.

2. Combined File
A set of card records, some or all of which will be read and/or punched during one pass through the system. The records comprising a combined file must be fed from one hopper. A combined file that is processed on the 2560 MFCM may also be card printed.

VOLUME

A volume is a tape reel of data. A volume may contain only part of a file (a multi-reel file, for example) or many files.

LABELS

A label is a tape record used to identify either a volume or a file. A label may be standard or non-standard. It is considered standard if it meets the format requirements for standard tape labels.

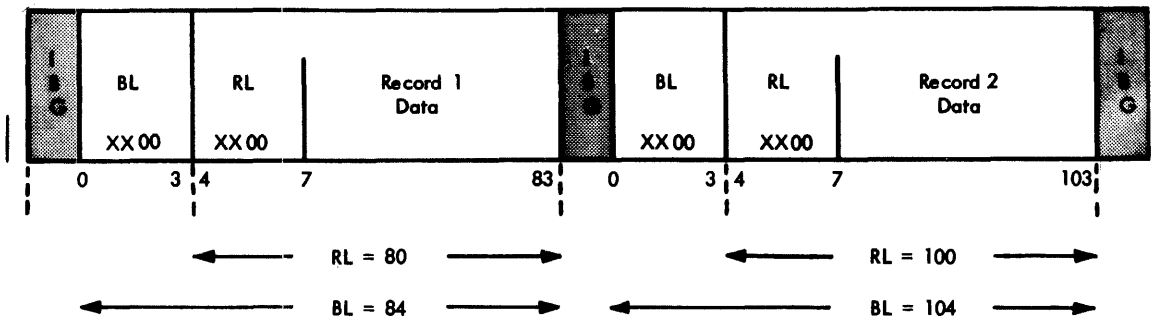
Label processing and formats of labels are described in detail in the SRL publication IBM System/360 Model 20, Tape Programming System, Control and Service Programs, Form C24-9000.

CARD/PRINTER OVERLAP MODE

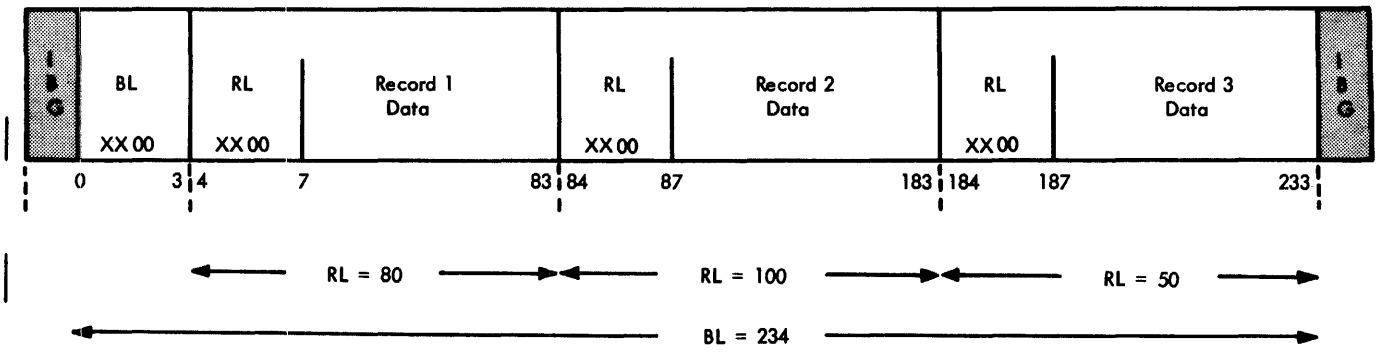
This is a mode of operation that allows the execution of card and printer I/O operations and processing to be performed simultaneously. A mode of operation that does not permit I/O operations and processing to be executed simultaneously is referred to as non-overlap mode.

READ/COMPUTE, WRITE/COMPUTE OVERLAP FEATURE

When a Submodel 5 is used, the read/compute, write/compute (RWC) overlap feature is available. When the CPU is running in the RWC overlap mode (under control of the overlap monitor), data transfer to or from tape units is overlapped with processing.



a. Variable Length - Unblocked Record Format



b. Variable Length - Blocked Format

● Figure 2. Example of Variable-Length Records on Tape

Macro instructions are provided to reduce the amount of repetitive coding. A macro instruction given by the programmer causes the generation of a set of individual machine instructions at the time of assembly. The generated instructions cause the desired machine function to be performed when the program is executed.

In this publication, the following conventions apply to the description of the macro instructions:

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 must be supplied by the programmer.
3. Information that is contained within brackets [ ] represents an option that can be included or omitted, depending on the requirements of the program.
4. A series of three periods enclosed by commas indicates that a variable number of items may be included.

The programmer writes his macro instructions on the standard coding form, X28-6509, provided by IBM. For details regarding the use of this form and how these instructions are entered on the form, refer to the SRL publication IBM System/360 Model 20, Disk and Tape Programming Systems, Assembler Language, Form C24-9002.

Two types of macro instructions are required for the processing of the records in a logical file: one declarative macro instruction (referred to as file definition statement in this publication) and one or more imperative macro instructions.

All imperative macro instructions consist of a mnemonic in the operation field and of one or more operands in the operand field. The precise format of the imperative macro instructions is shown separately for each of them later in this publication. The format of the definition statements is described under the heading Definition Statements below.

The input/output macro instructions provided by IBM are presented in this section in the following groups:

1. Definition Statements: DTFSR and DTFMT (both including detail entries), DTFEN and DTFBG.
2. Initialization Macro Instruction: OPEN.
3. Processing Macro Instructions: GET, PUT, CNTRL, CRDPR, EOM, LOM, PRTOV, WAITC, LBRET, RELSE, and TRUNC.
4. Completion Macro Instructions: CLOSE and FEOV.

Figure 8 shows a summary of all the DTFSR detail entries available to the user of the Model 20 IOCS. This summary shows the allowable entries and for which input/output device(s) a specific entry may be required. Example for the use of this summary: If a file is to be read and/or punched on the IBM 2560 MFCM, the X's in the 2560 column indicate to the programmer which entries he may have to provide.

Figure 9 shows a summary of all DTFMT detail entries that are available to the user of the Model 20 IOCS.

Figure 13 is a summary of all imperative macro instructions. The chart in Figure 13 shows all allowable entries in the various fields of the IBM System/360 Assembler Coding Form.

DEFINITION STATEMENTS

The programmer must use definition statements to describe to the IOCS the characteristics of each file to be processed. Definition statements are used to assign a name to each of the user's input/output files, to describe the input/output device used for each file, to define the input/output areas required, etc. The macro phase of the Assembler selects the routines required by the user on the basis of the definition statements given by the programmer.

There are two types of file definition statements: DTFSR (Define The File in a SeRial type device) for card and printer files and DTFMT (Define The File on Magnetic Tape) for tape files. The user must write one DTFSR statement for each card and printer file to be used by the program. He must write one DTFMT statement for each tape file.

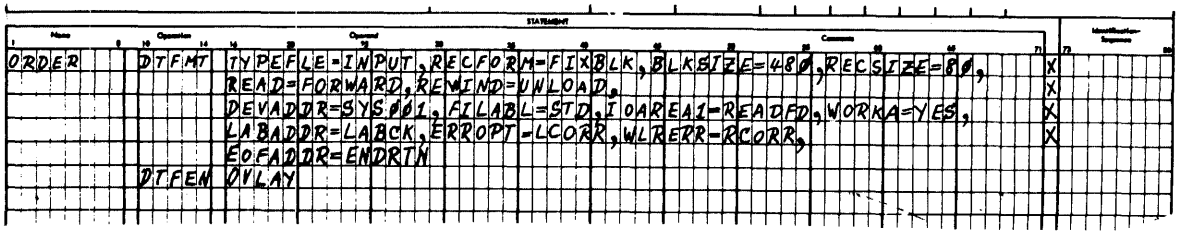


Figure 3. DTFMT Statement Followed by a DTFEN Statement

A DTFEN statement (Define The File END) must follow the last definition statement of a given program. A DTFBG statement (Define The File BeGin), if used, must be written as the first definition statement. If the Linkage Editor program is to be used, all the DTF statements must be contained in one and only one of the programs to be linked.

The user must write his definition statements ahead of his problem program. If his program includes more than one DTFSR statement, the user must write his DTFSR statements contiguously, i.e., he is not permitted to write a DTFSR statement for one file followed by a DTFMT statement and a DTFSR statement for another file.

Figure 3 is an example of a DTFMT statement followed by a DTFEN statement.

A DTFSR or a DTFMT statement consists of (1) a header entry that assigns a name to the file specified and (2) detail entries as required to define such information as the device to be used, the mode of processing, etc. All DTFSR (DTFMT) statement cards, except the last one, must have a continuation punch in column 72. This continuation punch may be any non-blank character. Punching in continuation cards must begin in column 16, except when the Assembler input format has been changed by means of an ICTL statement.

DTFBG STATEMENT

If DTFBG RWC=YES is specified, the IOCS provides the routines that make use of the read/compute, write/compute overlap feature. All tape files will then be processed in the overlap mode. The statement has no name in the name field and must be written as the first definition statement.

Note: This statement must not be used when no tape files are to be processed.

HEADER ENTRIES

A header entry consists of a file name in the name field (columns 1 through 8) and DTFSR (DTFMT) in the operation field (columns 10 through 14). For the name entered in the name field, the same rules as for the Assembler apply, except as follows: it must not exceed seven characters in length and the letter "I" as the first character is not permitted.

The file name is used in imperative macro instructions that refer to the file.

DETAIL ENTRIES

A detail entry is composed of a keyword immediately followed by an equal sign (=) which, in turn, is followed by one specification. The length of a specification is limited to eight characters, including expressions with their operators (if any). Expressions are permitted for all detail entries that require a symbolic address. A comma must immediately follow the specification of each detail entry, except the last one (see Figure 3).

CAUTION: A blank within a detail entry specification causes the Assembler to consider the remaining characters of the specification, and all subsequent detail entries of the DTFSR (DTFMT) statement, as comments.

Together with the header entry, the detail entries describe the file and specify symbolic addresses of routines and areas used when processing the file. This set of entries is used to generate the IOCS routines for the file during assembly.

Detail entries may be written (and punched) immediately after the header entry. They may appear in any order. The programmer must include only those entries that apply to a particular file.

The following sections describe all possible detail entries for the DTFSR and the DTFMT statements.

DTFSR DETAIL ENTRIES

The DTFSR detail entries are required to define card and printer files. They can be divided into five categories as follows:

1. entries applicable for most files,
2. additional entries for simple files,
3. additional entries for combined files,
4. additional entries for card printing, and
5. additional entries for certain checking functions.

Detail Entries for Most Files

The DTFSR detail entries applicable to most files are:

DEVICE	OVERLAP
TYPEFLE	CONTROL
WORKA	BINARY
PRINTOV	EOFADDR

The entries DEVICE, TYPEFLE, and WORKA must be provided for each card and printer file to be used by the program.

The entries PRINTOV, OVERLAP, CONTROL, BINARY, and EOFADDR must be provided only for certain card and printer files to be used by the program.

- The PRINTOV entry must be provided for a printer file if a PRTOV macro instruction referring to the file is used in the program.
- The OVERLAP entry must be provided for all files to be processed in non-overlap mode.
- The CONTROL entry must be provided for a file if a CNTRL macro instruction referring to that file is used in the program.
- The BINARY entry must be provided for input files that are to be read in the column binary mode.
- The EOFADDR entry must be provided for all input and combined files.

DEVICE=

This entry identifies the input/output device to be used to process the particular file. One of the following specifications must be entered immediately after the equal sign (=) following the keyword:

CRP20	The file is to be read and/or punched by the IBM 2520 Card Read-Punch.
MFCM1	The file is to be read and/or punched with or without card printing from the Primary Feed of the IBM 2560 Multi-Function Card Machine.
MFCM2	The file is to be read and/or punched with or without card printing from the Secondary Feed of the IBM 2560 Multi-Function Card Machine.
PRINTER	The file is to be printed by an IBM 2203 Printer with a standard carriage or by an IBM 1403 Printer (see Note below).
PRINTLF	The file is to be printed on the lower carriage of an IBM 2203 with the dual feed carriage (see Note below).
PRINTUF	The file is to be printed on the upper carriage of an IBM 2203 with the dual feed carriage (see Note below).
PUNCH20	The file is to be punched by an IBM 2520 Card Punch.
PUNCH42	The file is to be punched by an IBM 1442 Card Punch, Model 5.
READ01	The file is to be read by an IBM 2501 Card Reader.

Note: If both feeds of an IBM 2203 printer with dual feed carriage are used, the programmer must write a DTFSR statement for the file printed by the lower carriage and a DTFSR statement for the file printed by the upper carriage. 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 and only one printer file DTFSR statement is permitted in a problem program.

TYPEFLE=

This entry defines the type of the file (i.e., input, output, or combined). One of the following specifications must be used:

INPUT for a simple input file

OUTPUT for a simple output file

CMBND for a combined file.

WORKA=YES

The WORKA=YES detail entry is mandatory for all card and printer files. The user must enter the name of his work area as the second operand in his GET, PUT, and CRDPR macro instructions for the particular file and not in the WORKA entry for that file.

PRINTOV=YES

This entry must be included for a printer file if a PRTOV macro instruction referring to this file is used in the program.

OVERLAP=NO

This entry indicates that the file is to be processed in non-overlap mode. If this entry is omitted, the file is processed in overlap mode. Since printer files are always processed in the overlap mode, this entry is not permitted for these files.

IOCS routines for overlapped processing require more (50-100 bytes) main storage space than the routines for non-overlapped processing.

CONTROL=YES

This detail entry is required if a CNTRL macro instruction will be issued for the file. A CNTRL macro instruction causes the input/output device to perform operations such as stacker select and form skip.

BINARY=

This entry is required if the cards are to be read in the column binary mode. The entry may be provided for both simple and combined files. The specifications are:

YES for simple files

INPUT for combined files.

The twelve punch positions of a card column read in column binary mode are stored in the 6 low-order bits of two adjacent bytes of the input area. Therefore, the input and work areas must be specified to contain a number of bytes that is equal to twice the number of columns to be read.

When the BINARY entry is used for a particular file, the entries SEQNCE and RFORMTn are not permitted for that file.

EOFADDR=name

This entry specifies the name of the routine in the source program to which the IOCS should branch on an end-of-file condition. In that routine, the user can perform any operation required for the end of job, and he generally issues a CLOSE macro instruction.

This entry is mandatory for input and combined files.

The IOCS detects end-of-file conditions by sensing an end-of-file card with /* punched in columns 1 and 2.

Additional Detail Entries for Simple Files

The entries described in this section are available for simple files only. One or more of these entries may be required for a given file. The detail entries are:

IOAREA1
IOAREA2
BLKSIZE

IOAREA1=name

This entry specifies the name of the input/output area to be used by a simple file. This name must be the symbol used by the programmer in defining the area in his program.

The IOAREA1 entry is not permitted for a printer file to be printed by the standard carriage. The printer uses the first 144 main storage positions as a print buffer and they cannot be used by the programmer.

Two files printed by 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 with the print area for the lower feed preceding the area for the upper feed (see Figure 4).

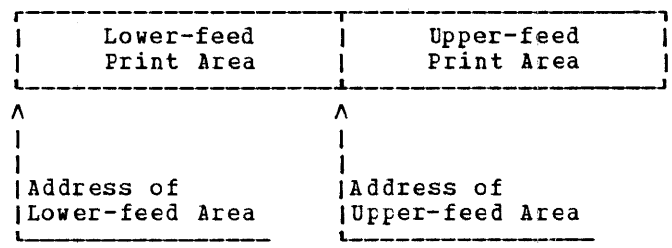


Figure 4. Print Area Format for Dual Feed Carriage

Note that for card and printer files a work area must be specified in addition to an input/output area. Refer to the description of the WORKA=YES detail entry.

IOAREA2=name

This entry can be used to indicate the name of a second input area when the IBM 2501 Card Reader is used in overlap mode. The name in the specification part of this entry must be identical with the symbol the programmer used in defining the area in his program. The area must be 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) is waiting to be moved into the work area. This may be of significance, for example, 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.

This entry must not be used 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.

BLKSIZE=n

This entry specifies the length of the input/output area(s) to be used by the file. The specification n must be equal to or less than the length in bytes of the reserved area. A BLKSIZE entry must be given for a printer file even though the IOAREA1 entry is not provided.

If two input/output areas are used for a file (IOAREA1 and IOAREA2), only one BIKSIZE detail entry is required in the DTFSR statement for the file, and this entry applies to both areas.

Maximum block lengths acceptable to the IOCS are as follows:

1. For card files: 80 bytes (160 bytes binary mode).

2. For printer files: 120, 132, or 144 bytes, depending on the number of print positions available. If a 2203 printer with the dual feed carriage feature is used, the total length of areas specified for both feeds must be equal to or less than 144 bytes.

The minimum block-length specifications are:

1. For input files: Two bytes (four bytes binary mode).

2. For output files: One byte.

Additional Detail Entries for Combined Files

The entries described in this section must be provided for each combined file. They are:

```
INAREA    CUAREA
INBLKSZ   OUBLKSZ
```

INAREA=name

This entry is used to specify the name of the input area to be used by the combined file. This name must be the symbol used by the programmer in defining the area in his program.

INBLKSZ=n

This entry specifies the length in bytes of the input area to be used by the combined file. The length applies to the area defined in the main program and referred to in the INAREA entry.

The maximum area length permitted is 80 bytes (160 bytes binary mode). The minimum length is two bytes (four bytes binary mode).

CUAREA=name

This entry specifies the name of the output area used by the combined file. The name must be the symbol used by the programmer in defining the area in his program.

OUBLKSZ=n

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

Additional Detail Entries for Card Printing

The following detail entries are only required if the card print feature of the IBM 2560 MFCM is to be used:

CRDPRA
CRDPRLn

The two entries apply only to simple or combined files to be processed by the IBM 2560 MFCM.

The CRDPRA and CRDPRLn entries are required in only one DTFSR statement of a program as they do not refer to a particular file. A CRDPR imperative macro instruction must be issued to cause the printing of data from the areas specified by the CRDPRA and CRDPRLn entries. Refer to the section CRDPR Macro Instruction.

CRDPRA=name

This entry is used in conjunction with CRDPRLn entries when printing on cards is desired.

The CRDPRA entry specifies the name of the area in main storage where the data to be printed by the lowest numbered MFCM print head is stored. The areas, from which the remaining print heads are to print, must be defined as contiguous 64-byte areas (refer to Figure 5). The print heads to be used must be defined by CRDPRLn entries in ascending order according to the print head numbers. One CRDPRLn entry must be included for each print head. Figure 6 shows the detail entries required to allow printing from the areas shown in Figure 5.

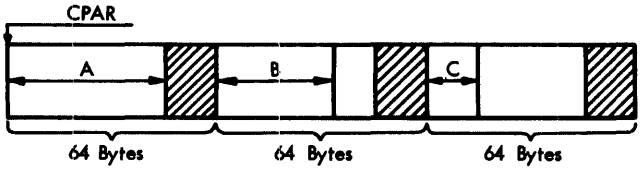


Figure 5. MFCM Card Print Areas

Figure 6. CRDPRA Detail Entry with CRDPRLn Entries

CRDPRLn=m

Entries of this type are used in conjunction with the CRDPRA=name entry to specify the print heads to be used. The keyword is CRDPRLn, where n is the number of the print head. The specification m indicates to the IOCS the number of bytes to be printed by this print head.

Refer to the example in Figures 5 and 6. In this example, print head 1 is to print the first 50 bytes of its 64-byte print area (part A), 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 will print the first 50 bytes of their 64-byte print areas. Therefore, the 64-byte print area for print head 2 in the example must contain blanks in bytes 41 through 50. Likewise, all bytes up to and including byte 50 of the 64-byte print area assigned to print head 5 would have to contain blanks if no printing were desired from print head 5 during a card print operation.

The programmer need not be concerned about filling the unused byte positions of a print area with blanks as this is an automatic function of the IOCS. If, as in our example, 50 bytes is the largest number of bytes specified for one particular print head, the IOCS clears all print areas up to and including byte 50 to blanks after every card print operation.

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.

The programmer may utilize any unused portions of the print areas. In the example, bytes 51 through 64 of all three 64-byte print areas could be used for other processing (shaded areas in Figure 5).

Additional Detail Entries for Checking Functions

The detail entries described below are available for card processing to enable the

user to specify certain checking functions.

SEQNCE	RFXIT
SEQXIT	PFORMTn
RFORMTn	PFXIT

SEQNCE=xxyy

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

If the input data is to be read in column binary mode, a SEQNCE entry must not be made for this file.

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

Only one SEQNCE entry is permitted for each file. Sequence checking is accomplished by a logical compare operation.

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 the user's SEQXIT routine causes the IOCS to change the processing mode (from overlap to non-overlap) for the GET that detected the error.

This change in the mode of operation enables the user to stacker-select the error card and/or to cause an error identification to be punched into this card.

Before branching to the user's routine, the IOCS places the record containing the field that led to the error condition into the work area. If the error card has been read by the IBM 2560 MFCM or the IBM 2520 Card Read-Punch, that card 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.

CAUTION: Do not destroy the contents of registers 14 and 15. Refer to the section Register Usage.

If a SEQNCE error and an RFORMT 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 indicates the name of the entry point of the user's routine to which control is to be transferred when a sequence error occurs. To return to the main program, the programmer must provide a branch to the address contained in register 14. After branching, the program executes the instruction following the GET that detected the sequence error.

RFORMTn=xxyyz

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

If the input data is to be read in column binary mode, an RFORMTn entry must not be made for this file.

The keyword of this entry is RFORMTn, where n is any number from 0 to 9. The n position allows the programmer to write up to ten different RFORMTn entries per file and thus have a maximum of ten fields checked.

The xx and yy specify the first and last card columns, respectively, 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 0. If the field is to be checked for numeric characters, z must be 1. When checking for numeric characters, the maximum field length is 16 columns.

When a field is tested for all blanks, the program branches to a user-written routine (or causes a system halt) if the test fails.

When a field is tested for numeric characters, the program branches (or causes a system halt) 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 the user's RFXIT routine causes the IOCS to change the processing mode (from overlap to non-overlap) for the GET that detected the error.

This change in the mode of operation enables the user to stacker-select the error card and/or to cause an error identification to be punched into this card.

Before branching to the user's routine, the IOCS places the record containing the

field that led to the error condition into the work area. If the error card has been read by the IBM 2560 MFCM or the IBM 2520 Card Read-Punch, that card is positioned at the pre-punch station. The next GET or ECM macro instruction causes the next record to be read.

CAUTION: Do not destroy the contents of registers 14 and 15. Refer to the section Register Usage.

The programmer 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 detail entry.

RFXIT=name

This entry is used in conjunction with the RFORMTn entry. It specifies the name of the entry point of the user's routine to which control is to be transferred if the test on the field specified in the RFORMTn entry is negative (i.e., the field tested contains characters other than those specified). To return to the main program, the programmer must provide a branch to the address contained in Register 14. After branching, the program executes the instruction following the GET that detected the RFORMTn error.

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

PFORMTn=xxyy

This entry enables the programmer to check those cards of a combined file that are not read into a work area by GET macro instructions to ensure that a specified card field (or fields) to be punched contains blanks.

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

If the field is found not to contain all blanks, the PUT macro instruction is not executed. Instead, either control is transferred to a user-written routine (provided the branch address has been furnished by a PFXIT detail entry), or a programmed halt occurs.

CAUTION: Do not destroy the contents of registers 14 and 15. Refer to the section Register Usage.

The specified input area must be large enough to permit the program to read the information in the columns specified in this entry into main storage.

The programmer 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 indicates the name of the entry point of the user's routine to which control is to be transferred if the test on the field specified by the PFORMTn entry fails. To return to the main program, the programmer must provide a branch to the address contained in register 14. After branching, the program executes the instruction following the PUT that detected the PFORMTn error.

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

If a PUT macro instruction is given 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 PFORMT 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, a halt occurs before punching is initiated. This enables the operator to replace the card that led to the error condition. That card is positioned at the pre-punch station.

DTFMT DETAIL ENTRIES

DTFMT detail entries apply to tape files only. Although many of them are identical with DTFSR detail entries, all possible

DTFMT detail entries are described in this section in this order:

TYPEFLE	IOAREA2
DEVADDR	IOREG
RECFORM	WORKA
RECSIZE	BLKSIZE
FILABL	CONTROL
LABADDR	VARBLD
ERROPT	ALTTAPE
WLRERR	READ
ERRIO	REWIND
EOFADDR	TPMARK
IOAREA1	CKPTREC

TYPEFLE=

This entry defines the type of the file (i.e., input or output). The allowable specifications are:

INPUT for an input file

OUTPUT for an output file.

DEVADDR=

This entry defines the symbolic address of a tape drive to be associated with the particular file. The following symbolic addresses are permitted:

SYSIPT
SYSOPT
SYSnnn

where nnn may be any number from 000 to 015.

An actual tape drive address is assigned to the symbolic address by means of an ASSGN control statement that is processed either (a) by the Job Control program before the problem program is executed, or (b) at the time of system generation. Refer to the SRL publication IBM System/360 Model 20, Tape Programming System, Control and Service Programs, Form C24-9000.

RECFORM=

This entry defines the record format of the file. The IOCS can handle different types of records in the same program. However, the records in a file must be of the same type. The following specifications are possible:

FIXUNB for fixed length unblocked records

FIXBLK for fixed length blocked records

VARUNB for variable length unblocked records

VARBLK for variable length blocked records

UNDEF for undefined records.

If the RECFORM entry is omitted, fixed-length unblocked records are assumed.

When variable-length records are specified for a tape output file, the user's input/output 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 develop the length indication for each record (as each record is handled the same way as a block). If these records are blocked, the four additional bytes are used to develop the length indication for the entire block.

RECSIZE=

This entry applies to tape files containing either fixed length blocked records or undefined format records.

n is specified to indicate the number of bytes in an individual record for a tape file containing fixed length records.

(n) is specified to indicate a register if the file contains record of undefined format.

The IOCS uses the specified register to (1) provide the record size in case of an input file or (2) derive from it the record size in case of an output file. For output files, it is the user's responsibility to place the number of bytes contained in a record into the specified register before this record is written.

If a file containing records of undefined format is to be read backward, the contents of the specified register must be used to determine the beginning of each individual record.

The specification is either the number of the desired register (any one of the numbers 8 through 13), or a symbol that stands for this register, in parentheses. (If the user's problem program contains IOCS macro instructions that refer to the IBM 1259 or 1419 Magnetic Character Readers, registers 11 and 12 must not be used.)

The minimum and maximum record lengths permitted are as shown below. Lengths are given in number of bytes.

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

* Excluding the four bytes required for record length indication.

FILABL=

This entry indicates the type of label processing to be performed. The allowable specifications are:

STD for a tape input file, if standard labels are to be checked; or for a tape output file, if IBM standard labels are to be written.

NSTD for input files only, non-standard labels are skipped. The labels must be terminated by a tape mark.

NSTD may also be specified for a tape input file with IBM standard labels if these labels are not to be checked.

NO is specified if no labels exist.

Note: If FILABL=NO is specified for an output file, any existing volume label on the output tape will be overwritten.

The FILABL entry may be omitted for an unlabeled tape.

LABADDR=name

The user may require the checking, or building up, of one, or up to nine, file labels in addition to the standard file header or trailer label. If so, he must provide a routine for this purpose. The name of his routine is specified in this entry. This routine is entered after the IOCS has processed the IBM standard label or a preceding user label. If this entry is omitted for an input file having additional labels, these additional labels are skipped.

For an input file, the user can determine the type of label that has been read by the identification in the label itself. For an output file, register 8 contains one of the following codes:

'bO'- for a header label (when a file is opened),

'bF'- for an end-of-file label (on an end-of-file condition),

'bV'- for an end-of-volume label (on an end-of-volume condition).

where b=blank.

Register 9 contains the address of the IOCS label area at the time the user's routine is being entered.

At the end of a LABADDR routine, the programmer must issue an LBRET macro instruction to return control to the IOCS. Refer to LBRET Macro Instruction.

CAUTION: Do not destroy the contents of registers 14 and 15. Refer to the section Register Usage.

ERROPT=

This entry applies to tape input files; it specifies functions that are to be performed when an error block is detected.

When the Basic Monitor program detects an error in a block of input records, the tape is backspaced and reread 100 times before the block is considered to be an error block. Unless the ERROPT entry is included, which specifies procedures to follow in the event of an error condition, a halt occurs and the job is terminated. Either IGNORE, SKIP, or the symbolic name of an error routine can be specified in this entry. One of these specifications is entered immediately after the = sign in the keyword. The functions of the three specifications are:

IGNORE The error condition is completely ignored, and the records are made available to the user for processing.

SKIP The block containing the error is skipped, i.e., it is not made available for processing. The next block is read from tape and processing continues with the first record of that block.

name The IOCS branches to a user-written error routine which can perform any chosen error procedure, for example, listing the error condition.

In his routine, the programmer must not issue any GET macro instructions for records in the error block. If he uses any other IOCS macros in his routine, he must save the contents of registers 14 and 15. At the end of his routine he must return to the IOCS by branching to the address in register 14. When control is returned to the problem program, the first record of the next block is available for processing in the main program. When two input/output areas are used, and ERRIO=name is specified, the address of the input/output area containing the error block is placed in the location indicated by the symbolic name in the ERRIO entry. Register 14 contains the return address.

Note: If FILABL=STD is specified, the error block is always counted in the block count.

The entry applies to wrong-length records if the DTFMT entry WLRERR is not included.

WLRERR=name

This entry applies only to tape input files that do not contain undefined records. It specifies the symbolic name of a user's routine to which the IOCS branches if a wrong-length record is read. In his routine the user may perform any operation he desires for wrong-length records. However, he must not issue any GET macro instructions for this file. If he uses any other IOCS macros in his routine, he must save the contents of registers 14 and 15.

At the end of his routine the user must return to the IOCS by branching to the address in register 14. When control is returned to the problem program, the first record of the next block is available for processing.

When two I/O areas are used, and ERRIO=name is specified, the address of the I/O area containing the wrong-length record is placed in the location indicated by the symbolic name in the ERRIO entry. Register 14 contains the return address.

Whenever fixed-length blocked records or variable-length records are specified (RECFORM=FIXBLK, =VARUNB, or =VARBLK), the machine check for wrong-length records is suppressed and the IOCS generates a programmed check for record length. For fixed-length blocked 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 the reading of short blocks of records without a wrong-length-record indication.

For variable-length records, record length is considered incorrect if the length of the physical record (block) is not the same as the block length specified in the first two bytes of the block.

If fixed-length unblocked records are specified (RECFORM=FIXUNB), the IOCS utilizes the machine check to determine whether a record is of correct length. Specifying RECFORM=FIXUNB causes the number of bytes specified in the BLKSIZE detail entry 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 to the user the number of bytes contained in the wrong-length record.

If the WLRERR entry is omitted from the set of DTFMT entries, but a wrong-length record is detected by the IOCS, one of the following results:

1. If the ERROPT entry is included for this file, the wrong-length record is treated as an error block and handled according to the user's specifications for an error (IGNORE, SKIP, or name of error routine).

2. If the ERROPT entry is not included, the job will be terminated since no error recovery procedure is available to handle the wrong-length record.

ERRIO=name

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

1. The input area containing the block that caused an irrecoverable read error (if the name of the user's error routine is specified in the ERROPT entry), or:

2. The input area containing the wrong-length record (if the name of the user's wrong-length record routine is specified in the WLRERR entry).

If READ=BACK is specified, the address of the input area plus the specified BLKSIZE minus 1 (in other words, the last byte of input the input area) is inserted in the two-byte area.

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

Note: Refer to the descriptions of the detail entries ERROPT= and WLRERR=name.

EOFADDR=name

This entry is mandatory for all input files. It specifies the name of the routine in the user's program that the IOCS should branch to on an end-of-file condition. In that routine, the user can perform any operation required for the end of job, and he generally issues a CLOSE macro instruction. However, the user must not issue a GET macro instruction in his EOFADDR routine since no further records are available for processing.

An end-of-file condition is detected by reading a tape mark and EOF in the trailer label when standard labels are specified. If standard labels are not specified, the Model 20 IOCS assumes an end-of-file condition when it reads a tape mark.

IOAREA1=name

This entry specifies the name of the input/output area to be used. This name must be the symbol used by the programmer in defining the area in his program.

If the record format is variable length, four bytes of the input/output area must be reserved for the block size field. In addition, the input/output area must be defined at a half-word boundary. Refer to Input/Output-Work Area Combinations.

IOAREA2=name

Two input, or output, areas can be specified for a tape file, to allow data transfer to be overlapped with processing. In such a case, IOAREA2=name must be included. This operand specifies the symbolic name of the second I/O area; this name must be identical to the name used in the DS or DC statement defining the area.

The length of the second I/O area must be equal to the length of the first I/O area. The second I/O area must be defined by the programmer at a half-word boundary.

A warning message (MNOTE) is given on the printer if IOAREA2 is specified without DTFBG RWC=YES being the first definition statement. (Refer to Input/Output-Work Area Combinations.)

IOREG=(n)

This entry specifies a register. The specification is either the number of the register (any one of the numbers 8 through 13) or a symbol that is equated to the register, between parentheses.

Note: If the user's problem program contains IOCS macro instructions that refer to the IBM 1259 or 1419 Magnetic Character Readers, registers 11 and 12 must not be used.

An IOREG entry is required when:

1. blocked input or output records are processed in the input/output area, or
2. variable-length unblocked records are read backward and are processed in the input area, or
3. two input, or output, areas are used and the records (either blocked or unblocked) are processed in the input/output area.

The specified IOREG register contains:

1. for an input file, the address of a logical record available for processing,
2. for an output file, the address of an area that is available to the user for building the next record.

WORKA=YES

If the user desires to process the records of a file in a work area rather than in the input/output area for the file, he must include the WORKA=YES entry and establish the work area(s) in main storage. The name of the work area is entered as the second operand in GET (PUT) macro instructions for the particular file. Input/Output areas for tape files must not be used as work areas. For further information about the use of a work area, refer to the section Input/Output-Work Area Combinations.

If WORKA=YES is specified for an output file containing variable-length records, the programmer must define the work area(s) at a half-word boundary.

BLKSIZE=n

This entry specifies the length of the input/output area. The specification n must be equal to the number of bytes of a block. If the record format is variable, the specified length must be equal to the number of bytes contained in the longest block of records.

The maximum block length acceptable to the IOCS is 4095 bytes which is equal to the maximum block length for IBM 2415 tapes. The minimum block length is 18

bytes, except for tape input files containing checkpoint records. For these tape files, the minimum area length is 20 bytes.

Note: A message (MNOTE) is given on the printer if the BLKSIZE entry in the DTF statement specifies a blocklength of less than 18 bytes, or if the RECSIZE entry specifies a record length of less than 18 bytes for output files, and less than 1 byte for input files (See also the table at the end of section RECSIZE=). Generation is then terminated.

If variable-length unblocked records or records of undefined format are to be processed in a work area, the programmer should consider the following: A GET causes the IOCS to move the number of bytes specified in the BLKSIZE detail entry from the input area into the work area; a PUT causes the IOCS to move this number of bytes from the work area into the output area. Therefore, the programmer must, for an output file, ensure that the address of the work area he uses is equal to or lower than the upper main storage limit minus the BLKSIZE value.

CONTROL=YES

This detail entry is required if a CNTRL macro instruction will be issued for the file. A CNTRL macro instruction causes the associated tape drive to perform operations such as tape rewind, rewind and unload, backspace, etc.

VARBLD=(n)

This entry is necessary if an output file with variable-length blocked records is being processed and no work area is specified. This entry specifies a register that indicates the number of bytes in the output area available for building the next record.

The specification is either the number of the desired register (any one of the numbers 8 through 13), or a symbol that stands for this register, in parentheses. (If the user's problem program contains IOCS macro instructions that refer to the IBM 1419 Magnetic Character Reader, registers 11 and 12 must not be used.)

After a PUT macro instruction is issued for a variable length record, the space still available is calculated and placed in the VARBLD register. The user then compares the length of his next record with the available space. If the record will not fit, the user must issue a TRUNC macro instruction to cause the completed block of records to be written on the tape file. Then the present 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 the TRUNC macro instructions, refer to the sections PUT Macro Instruction and TRUNC Macro Instruction.

ALTTAPE=

This entry specifies the symbolic address of a tape drive that is to be used as an alternate drive when a tape file is contained in two or more reels (volumes). The physical tape drive address can be assigned to the specified symbolic address either at the time of system generation or by means of an assign (ASSGN) statement that is used by the Job Control program. The user can specify one of the following:

SYSIPT
SYSOPT
SYSnnn

where nnn may be any number from 000 to 015.

If the physical tape-drive address is assigned to the specified symbolic address by means of an ASSGN statement, the second (fourth, sixth, etc.) reel of tape may be mounted on any one of the tape drives that are attached to the system and available. The selected tape drive is assigned to the specified symbolic address. The first (third, fifth, etc.) reel of tape would then be mounted on the tape drive specified in the DEVADDR entry of the DTFMT statement for the file.

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

The ALTTAPE detail entry may be specified for input and output files. If specified for an output file, the IOCS switches the tape drives in accordance with the ALTTAPE specification on detection of an end-of-volume condition, i.e., when the reflective marker at the end of the tape is sensed.

If the entry is specified for input files, the functions of the IOCS vary depending on the type of labels (if any) specified for the file.

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

2. Non-Standard and No Labels. The IOCS has no means of determining the end of

a volume. When a tape mark is sensed, the IOCS transfers control to the user's EOFADDR routine, where he can determine whether an end-of-file or an end-of-volume condition exists. In case of an end-of-volume condition, the user must issue an FEOV macro instruction. This causes the IOCS to switch the tape drives in accordance with the ALTTAPE specification, then IOCS returns control to the instruction following FEOV.

Note: ALTTAPE may not be specified when READ=BACK is specified.

READ=

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

FORWARD is specified for a tape to be read in the normal forward direction.

BACK is specified for a tape to be read backward. However, READ=BACK cannot be specified:
1. for tape input files containing variable-length blocked records,
2. when ALTTAPE is specified.

REWIND=

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

UNLOAD is specified to rewind the tape when an OPEN macro instruction is given and to rewind and unload the tape when a CLOSE macro instruction is given or an end-of-volume condition occurs.

NORWD is specified if no rewind is desired. This entry is mandatory if READ=BACK is specified for the file.

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

TPMARK=NO

This entry applies only to unlabeled tape output files (FILABL=NO). If included, this entry will prevent the writing of a tape mark as the first record on a tape. If this entry is not included, a tape mark will be written as the first record.

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 and bypasses checkpoint records.

Tape files created by the Model 20 IOCS programs will not contain any checkpoint records. Therefore, this entry is only required when it is desired to read from a magnetic tape that was written by use of another program and contains interspersed checkpoint records.

If the CKPTREC detail entry is specified for a tape input file, the programmer must specify a block size (BLKSIZE entry) of at least 20 bytes, which is the minimum length of a checkpoint record.

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

DTFEN STATEMENT

A DTFEN (Define The File END) statement must follow the last defining (DTFSR or DTFMT) statement. A DTFEN statement consists of DTFEN in the operation field, the name field is left blank. The operand field may be left blank or it may have OVLAY (overlay) specified to reduce the amount of main storage used for the program.

The overlay programming technique can be used successfully to reduce the number of storage positions required by the program when one or more tape files are involved. This programming technique permits the user to have part or all of the OPEN input/output routines for his tape file(s) overlaid by his problem program and to have part or all of his problem program overlaid by the CLOSE input/output routines for his tape file(s).

When OVLAY is specified, the OPEN and CLOSE input/output routines for the user's tape file(s) are not generated as part of the DTF routines. Instead, they are generated in-line, i.e., when the first (or only) OPEN (or CLOSE) macro instruction for a tape file is encountered by the Assembler.

It is not sufficient, however, to specify OVLAY in the DTFEN statement in order to have the OVLAY function performed. In

addition, the programmer must observe the following (refer to Figure 7):

1. Write all user-written label handling routines, 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. This is accomplished by means of an LTORG Assembler instruction.

3. Open all tape files before the OPEN routines are overlaid by the problem program.

   Note: If (1) a program utilizing the overlay programming technique is loaded from cards and (2) the loading device is also used as input device for a card file, the programmer must ensure 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 can not be specified in programs that process:

   a. multi-volume tape files, and

   b. multi-file tape reels if more than one file on the tape is used.

4. Initiate execution of the OPEN macro instruction by a subsequent XFR statement (XFR BEGIN in Figure 7) which may or may not immediately follow the OPEN macro instruction. A FETCH macro instruction (FETCH ROUTIN in Figure 7) must be given following the last OPEN macro instruction. This FETCH causes part or all of the user's program to be loaded.

5. Give an ORG statement (ORG BEGIN in Figure 7) immediately 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 FETCH, XFR, and ORG statements, refer to the SRL publications IBM System/360 Model 20, Tape Programming System, Control and Service Programs, Form C24-9000, and IBM System/360 Model 20, Disk and Tape Programming Systems, Assembler Language, Form C24-9002.

6. Use XFR and ORG-statements (XFR ROUTIN and ORG xxxx in Figure 7) just prior to the CLOSE macro instruction. The operand of the ORG statement specifies an address in the preceding problem program. As in the case of the OPEN macro instruction, only one CLOSE macro instruction for all files should be given.

7. Issue a FETCH macro instruction (FETCH FINIS in Figure 7) for another program segment (i.e. another part of the program). This segment would include the routines that have been generated for the CLOSE and EOJ macro instructions. The loading of this segment begins at the address specified as the operand of the ORG statement preceding the CLOSE macro instruction.

Steps 1 through 5, above, 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 by the CLOSE and end-of-job routines.

USER-WRITTEN MACRO DEFINITION. If the programmer writes his own macro definitions, the following restrictions apply:

- The global SETB symbol &BG69 must not be used if the program includes IOCS macro definitions.
- The global SETB symbols &BG72 through &BG77 must not be used if the program includes 1259 or 1419 IOCS macro definitions.
- The global SETB symbols &BG1 through &BG19 must not be used if the overlay programming technique is used.

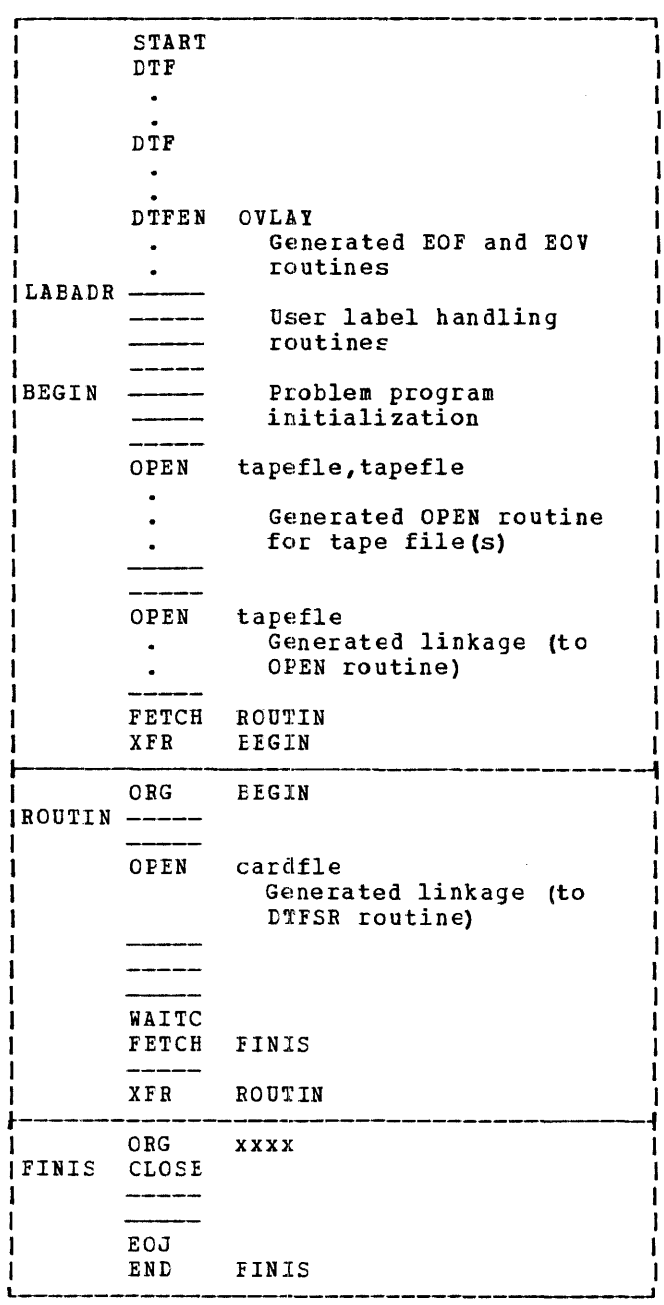


Figure 7. Coding for File Processing Using the Overlay Programming Technique

Note: At the end of generation of DTFEN, the global set symbols &BG1 through &BG68, &BG95 and &BG128 are set to zero by the ICCS (&BG1 through &BG19 and &BG69 are only reset if overlay has not been specified).

ASSIGNMENT OF BASE REGISTERS. Since the OPEN and CLOSE input/output routines are generated in line, the programmer must consider their approximate sizes when assigning and loading the base registers for his program. For information on the sizes of these routines, refer to the SRL publication IBM System/360 Model 20, Tape Programming System, Performance Estimates, Form C24-9010.

BASE REGISTER 9. When DTFEN OVLAY is specified the routines for the processing of the IBM standard labels VOL1 and HDR1 are generated as part of the OPEN routines instead of as part of the DTF routines. At the end of these label processing routines, the Assembler instruction DROP 9 is generated. If the user has given a USING instruction for register 9 at the beginning of his program, he must repeat this instruction immediately after the OPEN macro instruction. Reloading the register is not required because its contents is restored to the value that was contained in the register before the execution of the OPEN macro instruction.

Operation	Keyword	Operand: Allowable Specifications	Applies to: 2560	2520 Read-Punch	2520 Punch	1442 Mod 5 Punch	2501	2203	1403	Remarks
DTFSR			x	x	x	x	x	x	x	Always first card, may include detail entries from column 16 to column 71.
	BINARY	YES	x*	x*			x			* Only for simple files.
		INPUT	x	x						Only for combined files.
	BLKSIZE	length of simple file input/output area in bytes	x	x	x	x	x	x	x	Indicates length of area specified by ICAREA1 - ICAREA2 entries.
	CONTROL	YES	x	x	x			x	x	Required if a CNTRL macro is given for a file.

Figure 8. Definition Statement Summary for Card and Printer Files, Part 1 of 3

Operation	Operand: Keyword	Operand: Allowable Specifications	Applies to: 2560	2520 Read-Punch	2520 Punch	1442 Mod 5 Punch	2501	2203	1403	Remarks
	CRDPRA	name of user-defined card print area	x							
	CRDPRLn	length of card print area in bytes	x							n in the keyword is a printhead number.
	DEVICE	MFCM 1	x							
		MFCM 2	x							
		CRP20		x						
		PUNCH20			x					
		PUNCH42				x				
		READ01					x			
		PRINTER						x	x	
		PRINTLF						x		For 2203 with dual-feed-carriage.
		PRINTUF						x		
	EOFADDR	name of user's end-of-file routine	x*	x*			x			* Only for input and/or combined files.
	INAREA	name of combined file input area	x	x						Combined files only.
	INBLKSZ	length of combined file input area in bytes	x	x						Combined files only.
	IOAREA1	name of the user-defined area	x	x	x	x	x	x*		* Entry required for 2203 only when dual-feed-carriage used.
	IOAREA2	name of the user-defined area					x			Can be used if a 2501, Model A2, is used in overlap mode.
	OUAREA	name of combined file output area	x	x						Combined files only.
	OUBLKSZ	length of combined file output area in bytes	x	x						Combined files only.

Figure 8. Definition Statement Summary for Card and Printer Files, Part 2 of 3

Operation	Operand: Keyword	Operand: Allowable Specifications	Applies to: 2560	2520 Read-Punch	2520 Punch	1442 Mod 5 Punch	2501	2203	1403	Remarks
	OVERLAP	NO	x	x	x	x	x			If omitted, file is processed in overlap mode.
	PFORMTn	xxyy	x	x						Indicates that a check for blanks is to be made of field from col xx to yy prior to punching.
	PFXIT	name of user routine used when PFORMTn test fails	x	x						
	PRINTOV	YES						x	x	Required if a PRTOV macro is given for the file.
	RFORMTn	xxyyz	x	x			x			Indicates that a check for numerics or blanks is desired from col xx to yy in input cards.
	RFXIT	name of user routine used when RFORMn test fails	x	x			x			
	SEQNCE	xxyy	x	x			x			Indicates sequence check of input cards desired from col xx to yy.
	SEQXIT	name of user routine used when SEQNCE test fails	x	x			x			Must be specified when SEQNCE is specified.
	TYPEFLE	INPUT	x	x			x			
		OUTPUT	x	x	x	x		x	x	
		CMBND	x	x						
	WORKA	YES	x	x	x	x	x	x	x	Mandatory for all card and printer files.

Figure 8. Definition Statement Summary for Card and Printer Files, Part 3 of 3

Operation	Operand: Keyword	Operand: Allowable Specification	Remarks
DTFMT			Applies to tape files only.
	ALTTAPE	SYSIPT SYSOPT SYSnnn	Required for multi-volume files using two tape drives. SYSIPT, SYSOPT, and SYSnnn are symbolic addresses to be used when processing tape files.
	BLKSIZE	length of file input/output area in bytes	Length of IOAREA1 as defined in main program.
	CKPTREC	YES	Required to read tapes containing interspersed checkpoint records.
	CONTROL	YES	Required if a CNTRL macro is given for the file.
	DEVADDR	SYSIPT SYSOPT SYSnnn	SYSIPT, SYSOPT and SYSnnn are symbolic addresses to be used when processing tape files.
	EOFADDR	name of user end-of-file routine	For input files only.
	ERRIO	name of a two-byte area in which the IOCS places address of wrong-length record or of error block	May only be specified if the ERROPT entry specifies the name of the user's routine and/or if the WLRERR entry is included in the DTFMT definition, and if IOAREA2 is also included in this file definition.
	ERROPT	IGNORE SKIP name of user routine	If the ERROPT entry is omitted, a permanent read error causes the job to be terminated. When ERROPT=name is specified, the user must return to the IOCS via register 14.
	FILABL	STD	Standard labels.
		NSTD	Non-standard labels. Applies to input files only.
		NO	No labels. No labels is assumed if the FILABL entry is omitted.
	IOAREA1	name of the user-defined area	
	IOAREA2	name of the user-defined area	First DTF statement must be DTFEG RWC=YES.

● Figure 9. Definition Statement Summary for Tape Files, Part 1 of 2

Operation	Operand		Remarks
	Keyword	Allowable Specification	
	IOREG	number of any register from 8 to 13 in parentheses(n)	Required when either (1) blocked records are processed in the I/O area, or (2) variable length unblocked records are read backward and processed in the input area, or (3) records (either blocked or unblocked) are processed in the I/O area and IOAREA2 is specified. (If the user's problem program contains ICCS macro instructions that refer to the IBM 1259 or 1419 Magnetic Character Readers, registers 11 and 12 must not be used)
	LABADDR	name of user routine	User must return to main program by issuing a LBRET macro instruction.
	READ	FORWARD	If omitted, ICCS assumes forward reading.
		BACK	
	RECFORM	FIXUNB	Entry may be omitted if record format is fixed unblocked.
		FIXBLK	
		VARUNB	
		VARBLK	
		UNDEF	
	RECSIZE	number of bytes in one record or number of register indicating record length in number of bytes, in parentheses	Required if fixed-length blocked or undefined record format is specified. (If the user's problem program contains ICCS macro instructions that refer to the IBM 1259 or 1419 Magnetic Character Readers, registers 11 and 12 must not be used)
	REWIND	UNLOAD	If omitted, the tape is rewound, but not unloaded on OPEN, CLOSE and on end-of-volume condition.
		NORWD	
	TPMARK	NO	Applies to unlabeled tape output files.
	TYPEFLE	INPUT	
		OUTPUT	
	VARBLD	number of register in parentheses (n) for available-byte indication	Required if variable-length blocked records are built in the output area. (If the user's problem program contains ICCS macro instructions that refer to the IBM 1259 or 1419 Magnetic Character Readers, registers 11 and 12 must not be used.)
	WLRERR	name of user routine	The user must return to the ICCS via register 14.
	WORKA	YES	

Figure 9. Definition Statement Summary for Tape Files, Part 2 of 2

INITIALIZATION

Before the first record can be read from any input file or transferred to any output file by means of IOCS macro instructions, that file must be readied for use by issuing an OPEN macro instruction.

OPEN MACRO INSTRUCTION

The format of this macro instruction is:

Name	Operation	Operand
[name]	OPEN	file1,file2,...filen

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

Opening Card Files

For card and printer files, an OPEN macro instruction simply makes the file(s) concerned available for input and/or output.

Opening Tape Files

When a tape file with IBM 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 the programmer has specified no rewinding by including REWIND= NORWD in the DTFMT statement for the file. If the programmer has specified no rewinding and if a file beginning in the middle of the reel is opened, the user can position the tape by means of a FILES control statement for the Job Control program so that the first record read at OPEN time will be a label. If the first record is not a label the IOCS regards it 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 multifile tape volume are to be processed by one problem program, processing of each specified file must be completed before the file next in succession is opened.
Example: If the second, fourth, and sixth files of a multi-file tape volume are to be processed by one problem program, the programmer must write the OPEN macro instructions for these files in the following sequence:

```
OPEN second file
 .
 .
CLOSE second file
 .
 .
OPEN fourth file
 .
 .
CLOSE fourth file
 .
 .
OPEN sixth file
 .
 .
CLOSE sixth file
```

The concurrent processing of two or more files of a multifile tape volume is not possible.

Note that all files on a multifile volume must either contain the same type of labels (standard or non-standard) or contain no labels.

OPENING TAPE INPUT FILES: The processing done by the IOCS when an OPEN macro instruction is executed depends on whether the file has IBM standard labels, non-standard labels, or no labels. If the input file is to be read backward, the file must meet the requirements specified under Read-Backward Considerations below. An OPEN macro instruction causes the following:

1. If IBM 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 IBM standard file header label (HDR1);

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

   e. test the user labels (UHL1-UHL8), if a user's routine is specified, and make these labels available to the user's routine as they are read (refer to the Note below); and

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

   Note: If a user's label routine is not specified, user labels (if present) are skipped.

   If the file is to be read backward, steps e, d, and c are performed in this sequence; steps a and b are omitted

because the IOCS processes trailer labels instead of header labels.

2. If non-standard labels are specified, the file is spaced forward to the first record following the first tape mark. Therefore, the non-standard labels must be followed by a tape mark.

3. If no labels are specified, the first record on tape may be a data record or a tape mark followed by one or more tape marks. If the record is not a tape mark, it is assumed to be a data record, and the tape is backspaced by one record. If the first record is a tape mark, another record is read. If this record is a tape mark, the IOCS causes no further tape movement; otherwise, the IOCS assumes a data record and causes the tape to be 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 variable-length records have always been written using the Data Conversion feature and, therefore, can not be read backward. A file to be read backward is limited to one reel. Any tape mark 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 tape mark. The user is required to properly position files that are to be read backward prior to issuing an OPEN macro instruction. The proper positions are as follows:

IBM standard-labeled files should be positioned so that the first record read will be the tape mark following the trailer label set. Since the file trailer label is the first label to be checked when a file is to be read backward, this trailer label must be complete; it must 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 are complete.

Non-standard label files should also be positioned so that the tape mark following the trailer label set is the first record to be read. However, no label checking is performed.

Unlabeled files must be positioned so that the first record read is the tape mark following the last record of the file to be read.

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

The user must specify the NORWD (no rewind) option in his 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 IBM standard labels are specified, the IOCS will:

   a. check for a volume label if the file is positioned at loadpoint;

   b. read the file header label (if present) and check the expiration date to make sure the data on the tape is no longer active and may be destroyed;

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

   d. enter the user label routine, if this routine is specified, to allow the creation and writing of user header label(s) (UHL1-UHL8).

2. If no labels are specified, the IOCS will perform the rewind operation and write a tape mark 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 tape mark 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.

PROCESSING MACRO INSTRUCTIONS

These macro instructions cause input/output operations to be performed. If an operand of a processing macro instruction is the

symbolic address of an area or a routine, relative addressing is permitted. However, such an operand is limited in length to eight characters, including expressions with their operators.

The processing macro instructions common to all input/output devices are described first followed by a separate section each on (1) specific card and printer macro instructions and (2) specific tape macro instructions.

COMMON MACRO INSTRUCTIONS

In this section, processing macro instructions common to all input/output devices (GET, PUT, and CNTRL) are discussed.

GET MACRO INSTRUCTION

The format of this macro instruction is:

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

The GET macro instruction is written in one of two forms:

1. With one operand only. This format applies to tape files only. It is used if records are to be processed directly in the input area. The operand specifies the name of the file from which the record is to be read. The file name must be the same as the one specified in the DTFMT header entry for this file.

2. With two operands. This format is used if records are to be processed in a work area. The first operand specifies the name of the file. The second operand specifies the work area to be used. (Refer to the description of the WORKA=YES detail entry in the sections DTFSR Detail Entries and DTFMT Detail Entries.)

Processing in an Input Area: The first form of the GET macro instruction is used if records are to be processed directly in the input area(s). It requires only one operand. This operand specifies the name of the file from which the record is to be retrieved. The file name must be the same as that specified in the DTFMT header entry for the file.

The input area must be specified in the DTFMT entry IOAREA1. Two input areas may be used to permit an overlap of data-transfer and processing operations. The name of the second area is specified in the DTFMT entry IOAREA2. Whenever two input areas are specified, the IOCS routines transfer records alternately to each area. They handle this "flip-flop" so that the next consecutive record is always available to the program for processing.

When records are processed in the input area(s), a general purpose register must be specified in the DTFMT entry IOREG, if:

1. records are blocked,

2. variable-length unblocked tape records are read backward, or

3. two input areas are used, for either blocked or unblocked records.

This register always contains the absolute address of the leftmost position of the record currently available. The GET routine places this address in the register.

Processing in a Work Area: The second form of the GET macro instruction is used if records are to be processed in a work area. It causes the GET macro to move each individual record from the input area to a work area. In the case of variable-length records the record includes four bytes which hold the record length. As in the first form, the file name must be entered as the first operand. The name of the work area must be entered as the second operand, and YES must be specified in the WORKA entry of the DTFMT or DTFSR statement. The work-area name must be the same as that specified in the DS or DC instruction defining this area.

All records from a 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, for example, and to specify each area in alternate GET macro instructions. This would permit 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 variable-length unblocked records or records of undefined format are processed in a work area, a GET causes the IOCS to move the entire input area to the work area. (Refer to the description of the BLKSIZE detail entry in the section DTFMT Detail Entries.) If the record to be processed contains fewer bytes than the input area, undesired characters may be

moved into the work area along with the record.

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 is to be punched into the input cards, the programmer must use one of the programming methods described under Programming with LOM and EOM Macro Instructions in the section LOM Macro Instruction for Combined Files. Also refer to Programming Considerations -- Combined Files in the section PUT Macro Instruction below.

For a tape input file, a GET macro instruction may cause a read forward or a read backward operation. The type of read operation performed is determined by the READ= entry in the DTFMT statement.

PUT MACRO INSTRUCTION

The format of this macro instruction is:

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

This macro instruction is written in one of two forms:

1. With one operand only. This format applies to tape files only. It is used if records are to be processed directly in the input/output area. The operand specifies the name of the file for which the user wishes the PUT to be executed. The file name must be the same as the one specified in the DTFMT header entry for the file.

2. With two operands. This format is used if records are being processed in a work area. The first operand specifies the name of the file. The second operand specifies the work area from which the records are moved to the output area.

Building in an Output Area: The first form of the PUT macro instruction is used if records are to be built directly in the output area(s). It requires only one operand. This operand specifies the name of the file to which the record is to be transferred. The file name must be the same as that specified in the DTFMT header entry for the file.

The output area must be specified in the DTFMT entry IOAREA1. Two output areas may be used to permit an overlap of data transfer and processing operations. The name of the second area is specified in the DTFMT entry IOAREA2. Whenever two output areas are specified, the IOCS routines transfer records alternately from each area. They handle this "flip-flop" so that the proper output area is always available to the program for the next consecutive output record.

When records are built in the output area(s), a general purpose register must be specified in the DTFMT entry IOREG, if:

1. records are blocked, or

2. two output areas are used, for either blocked or unblocked records.

This register always contains the absolute begin address for building the next record in the output area.

Building in a Work Area: The second form of the PUT macro instruction is used if records are to be built in a work area. This form of the PUT macro instruction moves a record from a specified work area to the proper location in the output area specified in the DTFMT or DTFSR statement. As in the first form, the file name must be entered as the first operand. The name of the work area is entered as the second operand. YES must be specified in the WORKA entry. The name of the work area must be the same as that specified in the DS or DC instruction that defines the area in main storage. 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 where the completed record was built. However, only one work area can be specified in any one PUT macro instruction.

Unblocked Records

Records transferred to card or printer output files are always considered to be unblocked. Records transferred to magnetic tape output files may be blocked or unblocked. If they are to be treated as unblocked this must be specified in the DTFMT entry RECFORM.

Each PUT transfers a single unblocked (either fixed or variable-length) record from the output area (or input area if updating is specified) to the file. If a work area is specified in the PUT macro instruction, the record is first moved from the work area to the output area (or input area) and then to the file.

Blocked Records

When blocked records are to be written on tape (as specified in the DTFMT entry RECFORM), the individually built records must be formed into a block in the output area. The block of records is then transferred to the output file. The blocked records may be either fixed or variable length.

Fixed-length blocked records can be built directly in the output area or in a work area. Each PUT macro instruction for these records either adds an indexing factor to the register, or moves the completed record from the specified work area to the proper location in the output area. When an output block of records is complete, PUT causes the block to be transferred to the output file, and initializes the register if one is used.

Variable-length blocked records can also be built in either the output area or in a work area. The length of each variable-length record must be determined by the problem program and included in the output record as it is built. The record-length field must occupy the first four bytes of each record. The first two bytes specify the length of the record (including the four bytes for the record-length field itself), and the next two bytes are binary zeros. The user must define an output area that is large enough to accommodate the four bytes in which the IOCS places the block-length indication. The block-length includes the four bytes for the block-length field itself.

When variable-length blocked records are built in a work area, the PUT macro instruction performs approximately the same functions as it does for fixed-length blocked records. The PUT routines check the length of each output record to determine if the record will fit in the remaining portion of the output area. If the record will fit, PUT immediately moves the record. If it will not fit, PUT causes the completed block to be written and then moves the record. Thus, this record becomes the first record in a new block.

If variable-length blocked records are to be built directly in the output area, an additional DTFMT entry, a TRUNC macro, and additional user programming are required. The user's program must determine whether each record to be built will fit in the remaining portion of the output area. This must be known before processing of the record is started, so that, if the record will not fit, the completed block can be written and the record can be built at the beginning of a new block. Thus, the length of the record must be pre-calculated and compared with the amount of remaining space.

The amount of space available in the output area at any time can be supplied to the program (in a register) by the IOCS routines. For this the user must specify a register in the DTFMT entry VARBLD. This register is in addition to the register specified in the DTFMT entry IOREG. Each time a PUT macro instruction is executed, IOCS loads into this register the number of bytes remaining in the output area. The problem program uses this to determine whether the next variable-length record will fit. If it will not fit, a TRUNC macro instruction must be issued to transfer the block of records to the output file and make the entire output area available for building the next block.

Undefined Records

When undefined records are handled, PUT treats them as unblocked. The programmer must provide any blocking he wants. He must also determine the length of each record (in bytes) and load it in a register for IOCS use, before he issues the PUT macro instruction for that record. The register that will be used for this purpose must be specified in the DTFMT entry RECSIZE.

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 case of a combined file), and (3) transfers control to the main program when the punch operation has been completed.

When a card (or printer) 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.

Note: Printer files are always processed in the overlap mode.

Neither the output area nor the work area (if used) is cleared by the IOCS when a PUT macro instruction is executed. To avoid his output records containing inter-

spersed characters from preceding records, the user must ensure the following:

1. If the records are built in the output area

   a. that the record he builds uses every position of the output area, or

   b. that he clears the output area before he starts building his next record.

2. If the records are built in a work area

   a. that the record he builds uses every position of the work area, or

   b. that he clears the work area before he starts building his next record.

Programming Considerations -- 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,W2.

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. In other words, alternating GET and PUT statements for the file F1 cause the first (third, fifth, etc.) card to be read and the second (fourth, sixth, etc.) card to be punched.

CNTRL MACRO INSTRUCTION

The format of this instruction (control) is:

Name	Opera-tion	Operand
[name]	CNTRL	filename,mnemonic[,[n][,m]]

This macro instruction contains CNTRL in the operation field, and the name of the file for which the device operation is desired as the first operand in the operand field. As a second operand, the programmer must enter one of the mnemonics listed below to specify the desired operation. The third or fourth operand may or may not be required depending on the type of operation specified by the programmer.

The CNTRL macro instruction can be used by the programmer to cause such non-data transfer operations as form skipping, stacker selection, tape rewinding, etc., to be performed on the device associated with the file. A CONTROL=YES entry must be included in the DTFSR(DTFMT) statement for a particular file if a CNTRL macro instruction is given for the file.

The following is a description of available mnemonics to be used as the second operand, and of the contents of the third and fourth operands, when required.

STACKER SELECTION (SS) FOR THE IBM 2520, MODELS A1, A2, AND A3: Either of two stackers can be selected.

Operand			Function
Mnemonic	n	m	
SS	1	-	Select stacker 1
SS	2	-	Select stacker 2

In the IBM 2520, cards are normally stacked in stacker 1. The stacker selection mnemonic (SS) is used to select a card into the other stacker as specified by the third operand in this macro instruction.

When two stacker select CNTRL macro instructions are given for the same file and before the next GET or PUT macro instruction for that file, the second stacker select CNTRL macro instruction is effective; i.e., the second CNTRL macro instruction overrides the first. The following must be observed by the programmer when issuing a stacker select CNTRL to ensure that the instruction is in proper relationship to the GET, PUT, or ECM 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. (The second operand

required in GET (PUT) macro instructions referring to card or printer files is not shown.)

```
--------------
CNTRL AAA,SS,n
-------------- no GET or PUT
-------------- referring to file AAA
GET (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 ECM
-------------- 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 one of the five available stackers can be selected.

Operand			Function
Mnemonic	n	m	
SS	1	-	Select stacker 1
SS	2	-	Select stacker 2
SS	3	-	Select stacker 3
SS	4	-	Select stacker 4
SS	5	-	Select stacker 5

The CNTRL macro instruction for the IBM 2560 MFCM may have one of two formats:

1. CNTRL Filename,SS,n
2. CNTRL ,SS,n

If the first format is used, the functions are the same as described for the stacker select CNTRL macro instructions for the IBM 2520, Models A1, A2, and A3.

The second format, which is only possible for card files to be processed by an IBM 2560 MFCM, does not specify a file name. Absence of the file-name operand is indicated by a comma.

When two CNTRL macro instructions without a file name are given before the stacker select operation is performed, the second stacker selection is effective; i.e., the second CNTRL macro instruction overrides the first.

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

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

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, then the CNTRL macro instruction must be given prior to any subsequent PUT, GET, or EOM macro instruction addressing an MFCM file. This is illustrated by the coding example below. (The second operand required in GET (PUT) macro instructions referring to card or printer files is not shown.)

```
--------------
PUT (or GET) F1
-------------- 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, the CNTRL macro instruction must be given prior to any subsequent GET, PUT, or EOM macro instruction addressing an MFCM file (see the coding example below).

```
--------------
PUT F1
-------------- No PUT, GET or
-------------- EOM referring to
-------------- MFCM files
CNTRL ,SS,n
--------------
```

There is one exception to the above. Between the PUT for a card to be selected and the CNTRL for this card, a GET for the same file may be inserted (see the coding example below).

```
--------------
PUT F1
-------------- No PUT, GET, or
-------------- EOM referring to
-------------- MFCM files
GET F1
-------------- No PUT, GET, or
-------------- EOM referring to
-------------- MFCM files
CNTRL ,SS,n
--------------
```

When the card to be selected is read by a GET macro instruction, another GET, EOM, or PUT to the same file must be given prior to the CNTRL macro instruction for this card (see the coding example below).

```
--------------
GET F1
-------------- any combination of
-------------- macro instructions
-------------- referring to another
-------------- file.
GET F1 (or PUT F1)
(or EOM F1)
-------------- No PUT, GET, or
-------------- EOM referring to
-------------- MFCM files
CNTRL ,SS,n
--------------
```

FORM SPACING (SP) FOR PRINTERS: Line spacing on printers can be controlled.

Operand			Function
Mnemonic	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

This mnemonic is required to control line spacing. The programmer may omit either operand n or m. If operand n is omitted, the omission must be indicated by a comma (Example: CNTRL Filename, SP,,2).

If a delayed spacing CNTRL macro instruction is not given before the next PUT for the same file, the form is automatically spaced one line after printing.

When two delayed spacing CNTRL macro instructions are given before the next PUT for the same file, the second CNTRL is effective, i.e., the second CNTRL overrides the first. If both delayed spacing and skipping are specified before a PUT for the file, only the last specified operation will be performed.

Both the delayed-spacing and the immediate-spacing specifications of a CNTRL macro instruction can be given before a PUT for the particular file. As a result, the form is spaced by a number of lines that is equal to the total of lines specified in the two CNTRL macro instructions. It is immaterial, which of the two CNTRL macro instructions is issued first. Normally, however, the programmer would write only one CNTRL macro instruction, e.g., CNTRL filename,SP,1,2 (spacing one line immediately and two lines after printing).

In order to increase system throughput, delayed spacing should be used whenever possible.

FORM SKIPPING (SK) FOR PRINTERS: Skipping to a specific line on a printed form can be controlled.

Operand			Function
Mnemonic	n	m	
SK	n		Skip to punch in channel n(n=1,2,...,12) immediately
SK	n	m	Skip to punch in channel n(n=1,2,...,12) immediately and to channel m(m=1,2,...,12) after printing
SK		m	Skip to punch in channel m(m=1,2,...,12) after printing

The form-skipping mnemonic is used to specify the channel of the carriage control tape to which the form is to be skipped immediately and/or after the printing of a line. The programmer may omit either operand n or m. If operand n is omitted, the omission must be indicated by a comma. Example: CNTRL filename,SK,,12.

When two delayed skipping CNTRL macro instructions are given before the next PUT for the printer file, the skipping specified in the second CNTRL macro instruction is effective, i.e., the second CNTRL overrides the first.

In order to increase system throughput, delayed skipping should be used whenever possible.

TAPE UNIT CONTROL: For a tape file, a CNTRL macro instruction may be issued anywhere in the problem program.

The CNTRL macro instruction is used to control magnetic-tape functions that are not concerned with reading or writing data on the tape. The file name and the mnemonic specifying the desired operation are the only operands required in CNTRL macro instructions for tape files. Each mnemonic used is described separately below.

The FSR (or BSR) function permits the user to skip over a physical record (from one interrecord gap to the next). The record skipped is not read into main storage. The FSF (or BSF) operation permits the user to skip to the end of the logical file, which is identified by a tape mark.

Mnemonic	Function
BSF	Backspace tape to preceding tape mark.
BSR	Backspace tape for one block.
ERG	Erase tape to produce a gap.
FSF	Forward-space tape to next tape mark.
FSR	Forward-space tape to next inter-block gap.
REW	Rewind tape.
RUN	Rewind and unload tape.
WTM	Write a tape mark.

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

For all I/O area combinations, with the exception of one I/O area and no work area, IOCS reads the physical tape record following the one that is being processed at the time. Therefore, if a CNTRL FSR function is performed it is the second physical tape record following the one being processed, which will be skipped over.

One I/O Area and No Work Area. A CNTRL with BSR as the second operand causes:

1. For a file that is read forward -- the tape to be positioned so that the record being processed is in proper position to be read on the next read-forward operation.
2. For a file that is read backward -- the tape to be positioned so that the second record after the one being processed is in proper position to be read on the next read-backward operation.

A CNTRL with FSR as the second operation causes:

1. For a file that is read forward -- the tape to be positioned so that the second record after the one being processed is in proper position to be read on the next read-forward operation.
2. For a file that is read backward -- the tape to be positioned so that the record being processed is in proper position to be read on the next read-backward operation.

All I/O Combinations, with the exception of One I/O Area and No Work Area. A CNTRL with BSR as the second operand causes:

1. For a file that is read forward -- the tape to be positioned so that the block after the one whose last record is being processed is read on the next read-forward operation.
2. For a file that is read backward -- the tape to be positioned so that the third block after the one whose last record is being processed is read on the next read-backward operation.

A CNTRL with FSR as the second operand causes:

1. For a file that is read forward -- the tape to be positioned so that the third block after the one whose last record is being processed is read on the next read-forward operation.
2. For a file that is read backward -- the tape to be positioned so that the block after the one whose last record is being processed is read on the next read-backward operation.

Before forward or backward-spacing operations (FSR, FSF, BSR, or BSF), the magnetic tape is positioned at an interblock gap.

If blocked input records are being processed, and if the user does not want to process the remaining logical records in the block or one or more succeeding blocks, he must issue a RELSE macro instruction

before the control macro instruction. The next GET will then make the first record of the new block available for processing. If, for example, the CNTRL macro instruction with FSR is issued without a preceding RELSE, the tape is advanced, but the next GET will make the next record of the old block available for processing.

When a CNTRL macro instruction is issued for a tape output file, the programmer should issue a TRUNC macro instruction (TRUNC = truncate) if it is desired that a partially filled block of records be written on tape before the CNTRL macro instruction for the file is executed.

If an FSR function (or BSR for a file that is being read backward) encounters a tape mark, the IOCS branches to the user's end-of-file routine.

The functions of the individual mnemonic are described below following the section Effect of CNTRL on Block Count.

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 for 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 in error at end of volume or end of file, a programmed halt occurs.

BSF (Backspace to Tape Mark). This mnemonic is used if backspace to the first record of a tape file is desired. When a BSF operation is executed, the IOCS causes the tape to be stopped with the tape mark preceding the first data record of the file in proper position to be read on the next read forward operation. In case of an output file, the tape is positioned so that the next subsequent PUT for the file causes the tape mark to be overwritten. Refer to Effect of CNTRL on Block Count.

BSR (Backspace to Inter-Block Gap). This mnemonic is used if backspacing of a tape file for one block is desired. The IOCS branches immediately to the user's end-of-file routine if (1) a BSR operation is performed for an input file and (2) a tape mark is detected as a result of this BSR operation. When a BSR operation is executed, the IOCS causes the tape to be stopped with the block just backspaced in proper position to be re-read on a read forward operation. Refer to Effect of CNTRL on Block Count.

ERG (Erase Gap). This mnemonic is used to erase all signals that may be recorded on a section of tape; i.e., it creates a length of blank tape (approximately 3 1/2 inches).

FSF (Forward Space to Tape Mark). This mnemonic is used if the remaining part or all of a tape input file is to be skipped. When an FSF operation is executed, the IOCS causes the tape to be stopped immediately after the tape mark following the last data record of the file that has been read. In case of a file without labels or with non-standard labels, the tape is stopped immediately after the tape mark following the last block of data has been read. Refer to Effect of CNTRL on Block Count.

FSR (Forward Space to Inter-Block Gap). This mnemonic is used if one block is to be skipped. The IOCS branches immediately to the user's end-of-file routine if (1) an FSR operation is performed for an input file and (2) a tape mark is detected as a result of this FSR operation. When an FSR operation is executed, the IOCS causes the tape to be stopped with the block following the one just skipped in proper position to be read on the next read forward operation. Refer to Effect of CNTRL on Block Count.

Note: BSR or BSF always move the tape towards load point, regardless of read forward or read backward mode.
FSR or FSF always move the tape away from load point, regardless of read forward or read backward mode.

REW (Rewind Tape). This mnemonic is used if a tape rewind operation is desired. When a REW operation is executed, the IOCS causes the tape to be stopped with the first record on the tape in proper position to be read on a read forward operation. The record may be (1) a volume label if standard labels have been specified, (2) a tape mark 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). This mnemonic is used if the programmer desires a tape rewind operation to be followed by a tape unload operation.

WTM (Write Tape Mark). This mnemonic is used if a tape mark is to be written.

SPECIFIC CARD AND PRINTER MACRO INSTRUCTIONS

Macro instructions pertaining to card and printer files (CRDPR, EOM, LOM, and WAITC) are discussed in this section.

CRDPR MACRO INSTRUCTION

This macro instruction is only applicable if the user has an IBM 2560 MFCM equipped

with the card print feature. The format of this macro instruction (CaRD PRint) is:

Name	Operation	Operand
[name]	CRDPR	,workname,cardprintarea

Because this instruction does not refer to a specific file, it does not have a file-name operand. 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 of particular importance that the programmer writes his CRDPR statement in proper relationship to PUT, GET, or EOM macro instructions related to the same card. The same rules that apply to the stacker-select CNTRL macro instruction for the IBM 2560 MFCM without a file-name operand are also applicable to the CRDPR macro instruction.

CAUTION: When a CRDPR macro instruction is executed, the data that is contained in the specified work area is moved into the specified card print area. If the programmer desires to have the contents of the cards of a file printed on the same cards and the file is processed in non-overlap mode, he must consider the following: Two GET macro instructions (or one GET and one PUT macro instruction are required to move a card to and through the print station. If the two work areas specified in the GET macro instructions are the same, the contents of the card that was read by the second GET is card-printed on the card that was read by the first GET.

The programmer must write one CRDPR macro instruction for each line to be printed. If two CRDPR macro instructions are given for the same line, only the last of them will be executed. At the time of printing, all print lines specified for a particular card 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 or with all print heads during another print operation. If no data is to be printed on a line, the programmer simply does not enter any data into the associated print area or, if processing was performed in the area, he clears the area before printing takes place. Refer to CRDPRLn=m in the section Additional Detail Entries for Card Printing.

ECM MACRO INSTRUCTION

The format of this macro instruction (Enter Overlap Mode) is:

Name	Operation	Operand
[name]	EOM	filename

EOM is entered in the operation field and the name of the file to which the instruction refers is specified as the operand.

An ECM macro instruction applies only to combined files for which a previous LCM macro instruction has been given (see below). The EOM macro instruction causes (1) the next card to be read into the read area, and (2) subsequent GET macro instructions referring to the same file to be executed in overlap mode. The processing of the file in overlap mode begins immediately after the EOM macro instruction has been given. For further details regarding the use of the EOM macro instruction, refer to LCM Macro Instruction, below.

LOM MACRO INSTRUCTION

The format of this macro instruction (Leave Overlap Mode) is:

Name	Operation	Operand
[name]	LCM	filename

Except for the mnemonic in the operation field, the format of this macro instruction is the same as that of the ECM macro instruction. The LOM macro instruction applies to combined files for which overlap mode has been specified. The 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 into the same card of a combined file that is being processed in overlap mode. if an LCM macro instruction is given for a particular file, all subsequent GET instructions for that file are performed in non-overlap mode until an EOM macro instruction is given.

Programming Considerations - LOM and EOM Macro Instructions

A card of a combined file can be read and then punched only if the card is read by a GET macro instruction in non-overlap mode. There are three possible ways to cause the GET to operate in non-overlap mode during this reading and punching of the same card:

1. Provide an OVERLAP=NO detail entry 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 for the file and, in the source program, give an LOM macro instruction between the OPEN and the first GET macro instruction 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 for the file and, in the source program, precede each GET macro instruction with an 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 nearly every card 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 thus can be punched subsequently. However, when punching is not to be performed, the program immediately begins operation in the overlap mode. This method (third) 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 using the LOM and EOM macro instrcutions. 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.

PRTOV MACRO INSTRUCTION

This macro instruction (PRinT OVerflow) applies to printer files only. Its format is:

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

In the operand field, the programmer must specify the name of the file to which the instruction refers and the carriage tape channel indicator n to be tested (either 9 or 12). If the programmer provides his own routine to which the program should branch on an overflow condition, he must specify the name of the routine as the third operand. This macro instruction allows the programmer to check for printer overflow conditions by testing the channel 9 or the channel 12 indicator before:

1. the execution of the last PUT macro instruction referring to a printer with the standard carriage,

2. 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

3. 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 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 the programmer's routine if the tested indicator is on and a third operand has been specified. At this point, any IOCS macro (except PRTOV) may be issued, e.g., to print overflow page headings. At the end of the routine, return to IOCS by branching to the address in register 14. If IOCS macros are used, the contents of register 14 must be saved for this purpose. If a third operand has not been specified, an automatic skip to channel 1 is performed when the tested indicator is on. A PRINTCV=YES entry is required when a PRTOV macro instruction is issued for a file.

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 input/output operations before the next sequential instruction is executed. This macro instruction enables the programmer to establish uniform operating conditions for all card and printer input/output devices that are used in his program.

In a program using the IOCS, a WAITC macro instruction must be issued if one of the following three conditions exists:

1. A programmed stop is required to permit an error card to be replaced in a file whose cards are to be read in overlap mode.

2. A programmed stop is required to permit an error card to be replaced in a file whose cards are to be read on the IBM 2560 MFCM in non-overlap mode and a file in the other feed of the MFCM is to be processed in overlap mode.

3. In case of a multi-phase program, the next program phase is to be loaded by means of a FETCH macro instruction.

Except for the 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 input/output device involved. If the initiation of the input/output operation is delayed, the IOCS places the device request into a waiting list. The IOCS handles the device requests in this waiting list and executes the appropriate input/output operation as the requested input/output devices become available.

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, the programmer may want to provide a stop (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.

The programmer has no means to determine the status of the waiting list at the time the error is detected. Moreover, he is not able to determine the exact position of the error card in the input/output device. Therefore, the standard restart procedures cannot be applied.

Before he issues the HPR instruction, the programmer must issue a WAITC macro instruction to (1) establish uniform operating conditions for all card (and printer) input/output 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 input/output 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 input/output device used and, in case of an MFCM file, on the mode of operation. For details, refer to Figure 10, which is a summary of the stop and restart information.

DUMMY GET MACRO INSTRUCTIONS. To ensure proper program functions on restart, i.e., resume processing with the corrected card record, the programmer must issue either one or two dummy GET macro instructions as shown in Figure 10. 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 is already in the input/output 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 a card file and two input/output 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 given for the non-error file. The programmer 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 11.

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, the programmer must do either of the following in his restart routine:

1. Repeat the CNTRL macro instruction after the dummy GET macro instruction for the file in his 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.

Input/ Output 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 input/ output area	Yes	1	3	
	Overlap with two input/ output 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.

**Only required for the file containing the error card. A dummy GET is required for both files.

Figure 10. Programming with the WAITC Macro Instruction -- Halt and Restart Information

Figure 10 is provided to facilitate programming of restart routines and to furnish card-handling information that is not covered in the Model 20 IOCS Operating Procedures. The programmer 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.

An IOCS provided halt (due to a machine check) may occur during or immediately after the user-programmed restart routine and the number of cards in the input/output device may be less than stated in the appropriate standard procedure as provided in the SRL publication IBM System/360 Model 20 Card Programming Support, Input/Output Control System, Operating Procedures, Form C26-3803. In this case, only those cards must be stacked manually which were in the card feed of the input/output device at the time the halt occurred and do not have to be returned into the respective hopper.

The coding example in Figure 11 is provided to illustrate programming with the WAITC macro instruction. The example includes a simplified restart routine. For the purpose of this coding example, the following is assumed:

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 form 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.
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 11. These instructions are identified by sequence numbers in parentheses in the rightmost column of Figure 11. 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 11. 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 instrcution (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 were detected, four cards would have to be returned for file AAA and two cards for file BBB.

If the cards of the file BBB were to be read in overlap mode, instructions (2) and (11) would 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, the following must be considered by the programmer.

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, the programmer 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 12.

```
|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) |
|       |CRLPR    |                     | (9) |
|       |B        |REPT                 |(10) |
|       |.        |                     |     |
|       |.        |                     |     |
```

Figure 12. Coding Example -- Programming with the WAITC Macro Instruction Involving Card Printing

The coding example in Figure 12 is based on the assumption that:

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 without 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 12 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 the preceding card to be printed on. 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.

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.

LOADING THE NEXT PROGRAM PHASE. When another program phase is loaded by means of an XFR and a FETCH statement, the programmer must ensure that all pending card and printer interrupts that may have resulted from processing under control of the preceding program phase have been handled properly. This is accomplished by issuing a WAITC macro instruction prior to the FETCH statement causing the program-load operation. Figure 7 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.

SPECIFIC TAPE MACRO INSTRUCTIONS

Macro instructions pertaining to tape files (LBRET, RELSE, and TRUNC) are discussed, in this section.

LBRET MACRO INSTRUCTION

The format of this macro instruction (LaBel RETurn) is:

Name	Operation	Operand
[name]	LERET	1
[name]	LBRET	2

The LBRET (label return) macro instruction applies only to tape files that contain additional user labels, that the user wants to check or build/write. It must be issued at the end of the user's label routine (specified by the DTFMT entry LABADDR), to return to IOCS after header or trailer labels have been processed. This macro instruction requires one or both of the following operands:

OPERAND 1

Additional labels, Input file: To return to IOCS when the user wants to eliminate the checking of all remaining user labels, operand 1 is required. IOCS then skips the remaining labels in the set, and processing continues. If all labels are to be checked, operand 2 is used and IOCS terminates label processing when the tape mark following the last label is read.

Additional labels, Output file: To return to IOCS when the user determines that the last additional user label has been built, operand 1 is used. IOCS writes the last label (from the label output area) and processing continues. Operand 1 is always required to terminate the output label set.

OPERAND 2

Additional labels, Input file: Operand 2 is required to return to IOCS after each additional user label has been checked. IOCS makes the next label, if any, available for checking in the label input area. When IOCS senses the end of the label set (tape mark), it terminates label processing.

Additional labels, Output file: Operand 2 is required to return to IOCS after each additional user label except the last has been built. IOCS writes the label from the label output area and returns to the user's label routine to permit him to build his next label. The user must then use operand 1 to terminate the output label set.

The LBRET routine requires the values that the IOCS has placed into registers 14 and 15. Hence, if the user requires one or both of these registers in his routine, he must save the value placed into these registers by the IOCS before he starts using them. He must restore this value prior to issuing the LBRET macro instruction.

RELSE MACRO INSTRUCTION

The format of this macro instruction (RELeaSE) is:

Name	Operation	Operand
[name]	RELSE	filename

The name of the file to which this macro instruction refers (name field entry on the DTFMT header entry line) is the only operand required.

This macro instruction is used in conjunction with blocked input records read from tape. It allows the programmer to skip the remaining records in a block and continue processing with the first record of the next block read when the next GET macro instruction is issued.

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

Another example of using the RELSE macro instruction is a job in which records on tape are categorized, and each category (perhaps a major grouping) starts 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 present block of records which may be of either fixed or variable length. RELSE causes the transfer of a new block to the input area. The next GET makes the first record available for processing, either by the initialization of a register or by moving the record to a specified work area.

A RELSE macro instruction causes no operation to be performed if the preceding GET:

1. causes the last record of the block to be made available for processing in the input area; or

2. causes the last record of the block to be made available for processing in a work area.

TRUNC MACRO INSTRUCTION

The format of this macro instruction (TRUNCate) is:

Name	Operation	Operand
[name]	TRUNC	filename

The name of the file to which this macro instruction refers (name field entry on the DTFMT header entry line) is the only operand required.

This macro instruction is used in conjunction with blocked output records that are to be written on tape. It allows the programmer to write a short block of records (blocks do not include padded records). When a TRUNC macro instruction is issued, the output area being used to build output records is considered full. The block of records in the output area is then written on tape and the output area is made available to build the next block of records.

The last record included in the short block is the record that was built before the last PUT macro instruction preceding TRUNC was executed. Therefore, if records are built in a work area and the problem program determines that a record belongs in a new block, the TRUNC macro instruction should be issued first, followed by the PUT macro instruction for this particular record. If records are built in the output area, however, the programmer must determine if a record belongs in the block before he builds the record.

Whenever variable-length blocked records are built directly in the output area, this TRUNC macro instruction must be used to write a completed block of records. When the PUT macro instruction is issued after each variable-length record is built, the output routines supply the programmer with the space (number of bytes) remaining in the output area. From this, the programmer determines if his next variable-length record will fit in the block. If it will not fit, he issues the TRUNC macro instruction to write out the block and make the entire output area available to build the record. The amount of remaining space is supplied in the register specified in the DTFMT VARBLD entry (see VARBLD=(n) in the DTFMT Detail Entries section).

A TRUNC macro instruction causes no operation to be performed if the preceding PUT:

1. causes the last record of a block to be included in the output block; or

2. causes the last record of a block to be moved from a work area to the output area for inclusion in the block before it is written on the particular output tape.

In either case, the entire block is written on tape.

COMPLETION MACRO INSTRUCTIONS

After all records of a file have been processed, that file must be deactivated. The macro instructions CLOSE and FEOV (Force

End of Volume) are provided for this purpose. Note that the FEOV macro instruction actually does not deactivate the specified file, but merely forces an end-of-volume condition.

The need to 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 input/output 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.

2. Tape input files with standard labels. The record following the tape mark is a label whose first three characters are EOF.

3. Tape files with standard labels that are read backwards. The record following the tape mark is a label whose first three characters are HDR.

4. Tape input files without labels or with non-standard labels. A tape mark indicates an EOF condition. (The user's end-of-file routine must determine whether and end-of-file or an end-of-volume condition exists.)

5. All output files. The user's program determines the end of a file.

End-of-File Processing

When EOF occurs in a card input file, the IOCS branches to the programmer's end-of-file routine. The address of his 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 checks the EOF1 label and 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 user's LABADDR routine when the checking of the EOF1 label has been completed. (Refer to the description of the LABADDR detail entry in the section DTFMT Detail Entries and to the section LBRET Macro Instruction.) After the checking of trailer label(s) has been completed the IOCS branches to the user's EOFADDR routine.

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

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

In his end-of-file routine, the programmer may issue a CLOSE macro instruction to deactivate one or more files. The actions performed when a file is closed are described in the section CLOSE Macro Instruction.

End-of-Volume Processing

Some of the actions performed by the CLOSE macro instruction are also required when an end-of-volume (EOV) condition occurs while processing a tape file.

The IOCS detects an EOV condition for standard-label input files by means of the characters EOV in a trailer label. For output files, the IOCS detects an EOV condition by sensing the reflective marker at the end of the output tape. For all other types of files, the IOCS has no means of detecting an EOV condition.

During the processing of a tape file, an EOV condition can occur. This indicates that the next volume of the same file is required, either for reading more input records or for writing more output records.

If FILABL=STD has been specified for a file, the IOCS processes an EOV condition as follows:

1. For input files, the IOCS (1) checks the block count, (2) branches to the user's routine that processes additional user labels, if such processing has been specified, and (3) performs the rewind option. 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 that the programmer desires. The IOCS then processes the header label(s) of the next volume as described in the section

Opening Tape Output Files under Open Macro Instruction.

If no labels or non-standard labels have been specified for an input file, the user must determine an EOV condition and issue an FEOV macro instruction to have the IOCS perform the desired end-of-volume functions. To determine an EOV condition, the user must provide a subroutine in his EOFADDR routine, to which the IOCS branches on detection of a tape mark. For multivolume files, refer to the description of the ALTTAPE detail entry.

CLOSE Macro Instruction

The format of this macro instruction is

Name	Operation	Operand
[name]	CLOSE	file1,file2,...filen

The name of the file that is to be closed (assigned to it by the entry in the name field of a DTFSR or DTFMT header entry) must be specified as operand. Any number of files from one to sixteen may be closed with one CLOSE macro instruction.

The CLOSE macro instruction is used to deactivate any file that has previously been made available by an OPEN macro instruction. The CLOSE macro instruction ensures proper handling of the file after all records have been processed. The functions performed depend on (1) the type of file and the type of input/output device involved and (2) the labeling technique (if applicable).

A file may be closed at any time by issuing a CLOSE macro instruction.

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 that:

1. records remaining in the output area upon completion of processing are printed and/or punched,
2. all processed data cards remaining in the card feed path (not end-of-file cards) are selected into the appropriate stackers, and
3. all pending interrupts for the closed file(s) have been handled.

Closing Tape Files

The functions performed when a 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 according to the rewind option specified in 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. (When a file with short blocks is used as input, the IOCS handles these records properly.) No action by the programmer is required. A tape mark is written following the last record.

If labels have not been specified, a second tape mark is written and the tape is rewound as 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 tape mark. 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 the user's routine specified in the LABADDR=name detail entry in the DTFMT statement for the file. This occurs after the standard label has been written. After building each label, the programmer must return control to the IOCS by use of the LBRET macro instruction. After all trailer labels have been written, the IOCS writes two tape marks, executes the specified rewind function, and deactivates the file.

Two tape marks are written at the end of a tape output file to indicate that no further data follows. If NCRWD has been specified for the file, the IOCS causes the tape to be backspaced by one record. As a result, the second tape mark is overwritten if another output file is written on the same tape.

Reopening Closed Files

If a CLOSE macro instruction has been issued for a card or printer file, this file cannot be reopened by a subsequent OPEN macro instruction.

If further processing of a closed tape file is desired, the file can be reopened. If this is done, the user must be aware that the previous CLOSE for the file has caused the tape to be positioned in accordance with the REWIND detail entry in the DTFMT statement for the file. Therefore,

to resume the processing of tape records at the point where the file was closed, REWIND=NORWD should be specified in the DTFMT statement.

The first record read from the reopened tape 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, the user must identify the first record read as a data record or a file label.

When a multi-volume file is reopened and the DTFMT entry ALTTAPE is included in the definition statement for the file, 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.

FEOV Macro Instruction

The format of this macro instruction (Force End-Of-Volume) is:

Name	Operation	Operand
name	FEOV	filename

The name of the file to which this macro instruction refers (assigned to it by the entry in the name field on the DTFMT header entry line) is the only operand required in this macro instruction.

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

If this macro instruction is issued for an input tape, the IOCS immediately causes execution of the rewind option selected by the user, provides for a reel change in accordance with the ALTTAPE detail entry in the DTFMT statement for the file, and processes the header label (or labels) of the next volume as required.

If this macro instruction is issued for an output tape, the IOCS causes the last block of records to be written, if necessary, followed by a tape mark. 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. provides for a reel change in accordance with the ALTTAPE detail entry in the DTFMT statement for the file; and

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

An example for the use of the FEOV macro instruction is given below.

If FILABL=NSTD or FILABL=NO has been specified for a multi-volume input file, the IOCS has no means to detect an end-of-volume condition. When a tape mark is detected, the IOCS transfers control to the user's EOFADDR routine in which the user must determine the end of a volume. After the last block of a volume has been read, the user must issue an FEOV macro instruction to have the IOCS perform the end-of-volume functions in accordance with his detail entries.

Operation	Operand	Remarks
CLOSE	file1,file2,file3,...,file16	Up to 16 files may be closed with one CLOSE macro instruction
CNTRL	filename,Operation,n filename,Operation,n,m	Control
	Possible operation and n(m)-operands:	
	BSF (backspace to tapemark), no n(m)-operand	Applies to tape files only
	BSR (backspace to inter-block gap) no n(m)-operand	Applies to tape files only
	ERG (erase gap), no n(m)-operand	Applies to tape files only
	FSF (forward space to tape mark), no n(m)-operand	Applies to tape files only
	FSR (forward space to inter-block gap), no n(m)-operand	Applies to tape files only
	REW (rewind tape), no n(m)-operand	Applies to tape files only
	RUN (rewind and unload), no n(m)-operand	Applies to tape files only
	SK (skip) n: 1,2,...,12	n causes immediate skip to specified tape channel
	m: 1,2,...,12	m causes skip to specified tape channel after printing
	SP (form spacing) n: 0,1,2, or 3	n causes immediate form spacing by specified number of lines
	m: 0,1,2, or 3	m causes form spacing by specified number of lines after printing
	SS(stacker select), n-operand required	Applies to multi-stacker card input/output devices only. The n-operand is the number of the stacker into which cards are to be selected.

Figure 13. Summary of Imperative Macro Instructions, Part 1 of 2

Operation	Operand	Remarks
CRDPR	,workname,cardprintarea	Card Print Absence of file-name operand is indicated by a comma. Requires CRDPRA and CRDPRLn DTFSR detail entries.
EOM	filename	Enter Overlap Mode. Applies to combined files for which a previous LOM macro instruction has been given.
FEOV	filename	Force End of Volume. Applies to multi-volume tape files only.
GET	filename,workname	Second operand must be omitted when no work area has been specified.
LBRET	n (n = 1 or 2)	Label Return. Required for return to IOCS from the LABADDR routine.
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.
PRTOV	filename,n,address	Print Overflow. n is either 9 or 12 and denotes the channel indicator to be tested. The third operand specifies a branch address if a branch is desired on an overflow condition. A skip to channel 1 is performed if the third operand has been omitted.
PUT	filename,workname	Second operand must be omitted when no work area has been specified.
RELSE	filename	Release. Applies to blocked record tape input files.
TRUNC	filename	Truncate. Applies to blocked record tape output files.
WAITC		Required if (1) card files are processed in overlap mode and (2) a program using the IOCS is executed in several phases.

Figure 13. Summary of Imperative Macro Instructions, Part 2 of 2

If labeled tape files are to be processed, the IOCS requires two types of control statements which are used by the Job Control program when the program is executed. These two types of control statements are (1) the Tape Volume Statement and (2) the Tape Label Statement. The format of each of these two statements is described below.

The volume and label statements provide the IOCS with the necessary label information to check labels for an input file or to create labels for an output file. For each labeled tape file, one volume statement and one label statement are required.

For a given file, the volume statement must always precede the label statement that describes the file on the volume.

Format of Volume Statement

Name	Operation	Operands
//	VOL	SYSnnn,XXXXXXX

The specified symbolic address of the tape drive used is the location of the first (or only) volume of the file. It is six characters in length. The name assigned to the file by the DTFMT statement must be used in the file name field of the volume statement.

Format of Tape Label Statement

Name	Operation	Operands
//	TPLAB	'XXX.....XXX'

The following information is required in the label information field:

	No. of Char's
File Identification	17
File Serial Number	6
Volume Sequence Number	4
File Sequence Number	4
Generation Number (if used)	4
Version Number (if used)	2
Creation Date (bYYDDD)	6
Expiration Date (bYYDDD)	6

The label information must be written as one character string enclosed in apostrophes, i.e., an apostrophe before the first character and after the last character in the label information field. The precise format of tape labels is described in the SRL publication IBM System/360 Model 20, Tape Programming System, Control and Service Programs, Form C24-9000.

BLOCKING OF RECORDS

Blocking of records is not possible for card and printer files. Blocking of tape file records offers the advantage that no unused gaps exist between individual records written on tape. A gap is provided between the last record of a block and the first record of the following block. This decreases the number of necessary tape starts and stops and permits the user to write more data on a tape.

Blocking of records is accomplished by means of appropriate detail entries in the DTFMT statement for the particular file.

If the records to be blocked are of fixed length, the user must include the following detail entries:

1. RECFORM=FIXBLK to indicate the record format.
2. BLKSIZE=n to indicate the length of a block.
3. RECSIZE=n to indicate the length of the logical records.

Note: If the DTFMT statement for the file does not include the WORKA=YES detail entry, an IOREG=(n) detail entry is required in addition to the three entries mentioned above.

If the records to be blocked are of variable length, the following detail entries are required:

1. RECFORM=VARBLK and
2. BLKSIZE=n.

Note: If the DTFMT statement for the file does not include the WORKA=YES detail entry, the detail entries IOREG=(n) and VARBLD=(n) are required in addition to the two entries mentioned above.

If the user builds his records in the output area the value in the IOREG register is used to address the record space available in the output area. A PUT macro instruction is issued each time a record has been built. The IOCS then checks if the output area is filled. If the output area is not filled, the IOCS increases the value in the IOREG register by the number of bytes contained in the record. Therefore, the IOREG register points to the address of the next available record space where the user can build his next record.

If the output area is filled, the IOCS causes all records currently in the output area to be written on tape as one block of records. In addition, the value in the IOREG register is reset to the starting address of the output area. Note that the OPEN routine places the starting address of the output area into the IOREG register.

If the user builds his records in a work area a PUT macro instruction causes the record in the work area to be moved into the output area. In addition, the IOCS checks to determine if the output area is filled. If the output area is not filled, the IOCS returns control to the main program and the next PUT macro instruction causes the record now in the work area to be moved into the next available record space in the output area. When the output area is filled, the IOCS causes all records in the output area, including the one that was moved into the output area by the PUT macro instruction, to be written on tape as one block of records.

DEBLOCKING OF RECORDS

Records are deblocked by including detail entries in the DTFMT statement for the file as described below.

If the records to be deblocked are of fixed length, the detail entries required are:

1. RECFORM=FIXBLK to indicate the record format.
2. BLKSIZE=n to indicate the length of a block.
3. RECSIZE=n to indicate the length of the logical records.

Note: If the DTFMT statement for the file does not include the WORKA=YES detail entry, an IOREG=(n) detail entry is required in addition to the three entries mentioned above.

If the records to be deblocked are of variable length, the detail entries required are:

1. RECFORM=VARBLK and
2. BLKSIZE=n.

Note: If the DTFMT statement for the file does not include the WORKA=YES detail entry, an IOREG=(n) detail entry is required in addition to the two entries mentioned above.

After a GET has caused the entire block of records to be read into the input area, the first logical record of the block is immediately available for processing (either in the input area or in the work area). The next GET for this file will make the next logical record available for processing. This function is repeated until the last logical record of the block has been made available for processing.

INPUT/OUTPUT-WORK AREA COMBINATIONS

The user must define, in his main source program, an area into which input data can be read and from which output data can be written by the IOCS routines. The name of this area must be used as the specification in the IOAREA detail entry (or entries) for the file. When a work area is not specified, all records are processed in the input/output area(s).

For a particular file, the user can specify one of the following input/output-work area combinations:

1. One input/output area.
2. One input/output area and one work area.
3. Two input/output areas (for tape files only).
4. Two input/output areas and one work area (for files associated with the IBM 2501 Card Reader, or for tape files).

The size of an input/output area must be equal to the length of the longest block to be processed. The size of a work area should be equal to the length of an individual record. However, if variable-length unblocked records or records of undefined format are specified, the programmer must take into consideration the number of bytes specified in the BLKSIZE detail entry when defining his work area(s). (Refer to the description of the BLKSIZE detail entry in the section DTFMT Detail Entries.)

One Input/Output Area

Specifying an input/output area without a work area is permitted for tape files only.

When a GET or a PUT macro instruction is given for a file while another input/output operation is executed, the IOCS enters a waiting loop to wait for the completion of this input/output operation, except when this input/output operation is a tape rewind operation on another tape drive or a print operation on the 1403 printer.

One Input/Output Area and One Work Area

If the processing of records is done in a work area rather than in the input/output area, the programmer must include a WORKA=YES entry in the DTFSR (DTFMT) statement for the file. (For card and printer files, the use of a work area is mandatory.) Also, he must define a work area in his program and assign a name to it. That name is then specified as the second operand whenever he issues a GET or a PUT macro instruction for the file.

Note: Input/Output areas for tape files must not be used as work areas.

When a GET macro instruction is given for a file that uses a work area a record is moved from the input area into the work area. When a PUT macro instruction is given, a record is moved from the work area into the output area.

For card files, the combined use of a work area with one I/O area permits the IOCS to overlap an input/output operation with processing and/or with another input/output operation. The same applies for tape files if a Submodel 5 is used and the RWC feature is specified. This increases system throughput, i.e., it increases the amount of records processed within a given time.

For tape files, the use of a work area provides the advantage of increased system throughput only if the program utilizes card and print devices (excluding the IBM 1403). If a GET is issued for a tape file that uses a work area specified, the IOCS performs as follows:

1. The next record in the input/output area is made available for processing by moving it into the work area.
2. If the record moved into the work area is the last one available from the input/output area, the GET macro instruction causes, at the same time, an actual device request to be issued that will be executed as soon as CPU time is available. Until this request can be executed processing is continued. If CPU time does not become available for execution of the device request before the next GET macro instruction requires the first record of the next block, the program enters a waiting loop.

If a work area is not used, the next block of records is requested by the GET macro instruction that requires the first

Record Format (Blocked or Unblocked)	Number of I/C Areas	Separate Work Area	Amount of Effective Overlap
Unblocked	1	no	No overlap
		yes	Overlap processing of each record (Record move required)
	2	no	Overlap processing of each record (No record move required)
		yes	Overlap processing of each record (No advantage to specify a work area)
Blocked	1	no	No overlap
		yes	Overlap processing of first or last record in each block
	2	no	Overlap processing of full block
		yes	Overlap processing of full block (No advantage to specify a work area)

Note: Overlap given is the maximum achievable.

Figure 14. Summary of Achievable Overlap of Processing and Input/Output when a Submodel 5 is used and the Read/Compute, Write/Compute Overlap Feature is employed.

record of the new block. If the CPU is busy servicing another input/output device that does not permit concurrent tape input/output, processing is not possible because the record to be processed is not yet available. The program then enters a waiting loop until the current input/output operation is completed.

To summarize, the use of a work area causes the device request for a new record block to be given already at the time the last record of the preceding block is made available for processing. If no work area is specified, the device request for a new block is not issued until the first record of the new block is required.

If a PUT macro instruction is given while the CPU is busy servicing another input/output device, the record in the work area is moved into the output area. If this was the last record to be moved into the output area, this PUT macro instruction will also cause a device request to be issued that will be executed immediately after completion of the current input/output operation. Until the request can be executed, the CPU can perform processing on the first record of the next block. If a work area is not specified, the program must enter a waiting loop until the current input/output operation is completed. During this time, no processing is possible and processing time is lost.

When a work area is used, the user can do all his processing in a predetermined area. Also, an IOREG register is not required. These are further advantages offered by the use of a work area.

Two Input/Output Areas

The use of two input/output areas without a work area is only permitted for tape files. The user can only take advantage of specifying a second I/O area when he uses a Submodel 5 and the program makes use of the read/compute, write/compute overlap feature.

a. Input.
The combined use of two input/output areas allows the overlap of the processing of one block with the reading of the next one. When the last logical record of a block in one area has been processed, the reading of the next block from tape into the same area is overlapped with the processing of the block in the other area.

b. Output.
The combined use of two input/output areas allows the overlap of the writing of one block with the building of the next one. When one output area is full, the block is written onto tape. Meanwhile,

Record Format	Number of Input/ output Areas	Work Area Specified	IOREG Required?	VARBLD Required?	RECSIZE=(n) Required?
Fixed Blocked	1 or 2	No	Yes	No	No
Fixed Blocked	1 or 2	Yes	No	No	No
Fixed Unblocked	1	No	No	No	No
Fixed Unblocked	1 or 2	Yes	No	No	No
*Fixed Unblocked	2	Yes	No	No	No
Fixed Unblocked	2	No	Yes	No	No
Variable Blocked	1 or 2	No	Yes	Yes**	No
Variable Blocked	1 or 2	Yes	No	No	No
Variable Unblocked	1	No	No+	No	No
Variable Unblocked	1 or 2	Yes	No	No	No
Variable Unblocked	2	No	Yes	No	No
Undefined	1	No	No+	No	Yes
Undefined	1 or 2	Yes	No	No	Yes
Undefined	2	No	Yes	No	Yes

*Only for card-input files to be read by the IBM 2501 in overlap mode.
**Output files only.
+Required if read backward is specified.

Figure 15. Register Requirements

another block is built in the other area. Thus, a maximum degree of overlap is obtained.

Two Input/Output Areas and One Work Area

The use of two input/output areas together with a work area is permitted only for card input files that are to be read by the IBM 2501 Card Reader in overlap mode, or for tape files.

1. For the IBM 2501 Card Reader: This area combination permits the IOCS to maintain the maximum card reading speed of the IBM 2501.

2. For tape files:
   The user can only take advantage of specifying a second I/O area when he uses a Submodel 5 and the program makes use of the read/compute, write/compute overlap feature (see also Figure 14). The information contained in the previous section, Two Input/Output Areas (subsections a and b), should also be considered when using two input/output areas and a work area.

Register Requirements

The record format in conjunction with the input/output-work area combination used determines whether none, one, or two registers (IOREG, VARBLD and/or RECSIZE) must be specified. Figure 15 indicates when it is necessary to specify a register.

DTF BLOCKS

For each tape file to be processed in a program that uses the IOCS, a table is built in main storage during the generation phase of the assembly. This table is referred to as the DTF block. It contains such information for the file as the specified rewind function the type of labels used, the accumulated block count, etc. Figure 16 shows the layout of and the type of information contained in the DTF block.

BYTE	BIT	CONTENTS
0-1		Address of DTF Block -- in binary notation.
2		OPEN Rewind Option -- command byte of CCW used when the file is opened: REW or NOP.
3		CLOSE Rewind Option -- command byte of CCW used when the file is closed: REW, RUN, or NOP.
4-5		Residual Count -- in binary notation.
6-7		Unit Status Bytes -- reserved for IOCS internal use; the user is not permitted to change the bit configuration of these two bytes.
6	0	Attention
	1	Status Modifier
	2	Control Unit End
	3	Busy
	4	Channel End
	5	Device End
	6	Unit Check
	7	Unit Exception
7	0	Incorrect Record Length
	1	Wait Bit
	2	In Error Recovery
	3	Wait for CPU Availability
	4	Reserved
	5	Device End Significant
	6	Accept Input/Output Error
	7	Reserved
8-9		Logical Unit Displacement -- the displacement of the logical unit block (symbolic device address) with reference to the begin address of the logical unit table in binary notation. The contents of these two bytes may be one of the following (shown in hexadecimal notation) 04 for SYSIPT 06 for SYSOPT 0C through 2A for SYS000 through SYS015, respectively. (In increments of two per symbolic address) The user is not permitted to change the bit configuration of these two bytes.
10-11		Address of CCW.
12	0	Not used.
	1	Type of File -- 1 = Input; 0 = Output
	2	FEOV Switch -- 1 = Yes; 0 = No
	3	EOF/EOV Indicator -- 1 = EOF; 0 = EOV
	4	OPEN Indicator -- 1 = Open 0 = Closed
	5	Alternate Drive Switch -- 1 = Yes; 0 = No
	6	Not used
	7	Not used

Figure 16. Layout of and Type of Information Contained in the DTF Block, Part 1 of 2

BYTE	BIT	CONTENTS
13	0	Standard Labels Indicator -- 1 = Yes; 0 = No
	1	Auxiliary Labels Indicator -- 1 = NSTD; 0 = No Labels
	2	Rewind Unload Switch -- 1 = Yes; 0 = No
	3	Rewind Option Indicator -- 1 = NRWD; 0 = REW
	4	Read Option Indicator -- 1 = Back; 0 = Forward
	5	User LABADDR routine -- 1 = Yes; 0 = No
	6	Tape Mark Option -- 1 = No; 0 = Yes
	7	Not used.
14-15		EOF Address -- address of user-routine in binary notation.
16-17		Alternate Unit Address Displacement -- the displacement of the logical unit block (symbolic device address) for the specified alternate tape drive. The displacement is given with reference to the begin address of the logical unit table.
18-23		CCW for Set Tape Mode Operation.
24-29		CCW for Tape Input/Output Operation.
*30-33		Accumulated Block Count -- in packed decimal format.
*34-35		LABADDR Address -- address of user-routine in binary notation.
*36-42		File Name -- the symbol specified in the name field of the DTFMT statement for the file.
*43-91		Standard Label -- the contents of fields 3 through 10 of the standard file label.

*This type of information is included only in the DTF blocks of files for which standard labels have been specified.

• Figure 16. Layout of and Type of Information Contained in the DTF Block, Part 2 of 2

There are two methods by which the user can obtain access to individual bytes of the DTF block for a file:

1. The first byte (byte 0) of the DTF block is addressable by using the file name as a symbolic address. For example, to refer to the CCW for input/output operations the user must specify the file name+24 in his instruction.

2. If the instruction that is to refer to a location within the DTF block is not preceded by an IOCS macro instruction that refers to another file, register 15 contains the address of the first byte (byte 0) of the DTF block. In this case, the programmer may use register 15 as base register and write an appropriate displacement to obtain access to the desired byte in the DTF block.

Access to the information contained in the DTF block for a file permits the user to test and/or modify the contents of the DTF block. He may, for example, test the accumulated block count on a tape read or write error and correct it after the error recovery.

DEVICE ERROR RECOVERY

Punched-Card Equipment Errors

When errors (read checks, feed checks, etc.) occur on punched-card equipment, the IOCS discontinues the execution of the program to allow the operator to take corrective action. An error indication is displayed on the CPU console to identify the type of error and to indicate the required restart procedure.

Tape Error Routines

The IOCS routines provide that on tape read errors the tape is backspaced and re-read 100 times before the block is considered an error block. The IOCS routines further provide that any error which can not be corrected is indicated to either the main program or the operator. Refer to the

description of the ERROPT detail entry. Indication to the operator is by means of a display on the CPU console. This display indicates the type of error and the device address.

If a tape write error occurs, the IOCS causes the tape to be back-spaced by one block, the error block to be erased, and the output block to be written on a new section of tape. The IOCS makes a total of 10 attempts to rewrite a block as described above. If the tenth attempt does not result in a correctly written output block, a programmed halt occurs.

REGISTER USAGE

The programmer may freely use any or all of the registers 10 through 13.

Registers 8, 9, 14, and 15 are not readily available to the programmer for reasons explained below.

Registers 8 and 9 are used by the IOCS to communicate with the LABADDR routine (see LABADDR=Name). Before branching to that routine, the IOCS saves the values that are contained in these registers. The two registers are restored to their original values if the programmer uses the LBRET macro instruction to return to the main program.

Registers 14 and 15 are used by the IOCS imperative macro instructions (GET, PUT, etc). If the programmer uses one or both of these two registers in his program, he must make sure their contents are no longer required before he issues an IOCS imperative macro instruction. He must save these registers if he requires their contents later in his program.

When an IOCS-controlled branch to a user-routine (ERROPT, LABADDR, PFXIT, RFXIT, SQXIT, or WLERR) occurs, the contents of registers 14 and 15 must not be destroyed. If the user desires to issue an IOCS macro instruction in his routine, he must save the contents of the two registers before this macro instruction is executed. He must restore the contents of the two registers to their original values before he returns control to the IOCS.

Transition Considerations. If the user anticipates transition to a higher System/360 model, he must be aware that the Basic Programming Support or the Basic Operating System Supervisor do not permit the programmer to use registers 12 and 13.

In the LABADDR user routine, the Model 20 IOCS uses registers 8 and 9 as communication registers (see the description of the LABADDR=name detail entry). The Basic Programming System and the Basic Operating System IOCS use registers 0 and 1 for this purpose in the LABADDR and other routines such as ERROPT and WLRERR.

IOCS ASSEMBLY

Both the IOCS portion and the user written program instructions are assembled in one and the same run. Figure 17 shows the arrangement of the input cards for a source program using the IOCS.

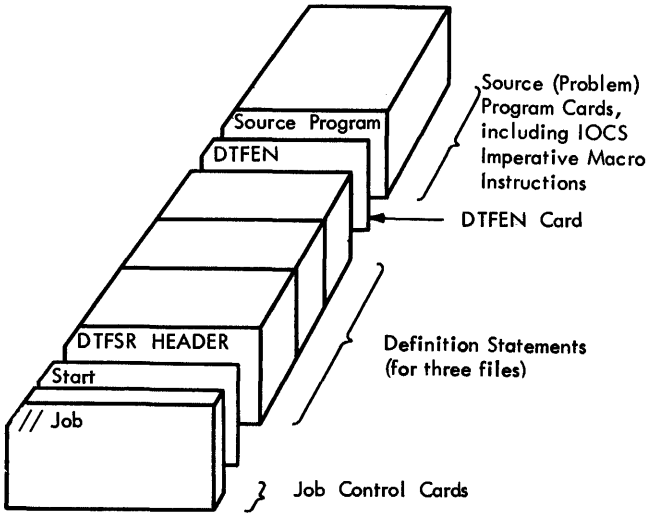


Figure 17. Arrangement of Source Program Cards Using the IOCS

Diagnostics for Source Programs Using the IOCS

During the assembly of a program using the IOCS, the IOCS portion of the program is subject to extensive checking for format errors. This checking is performed in two steps as follows:

1. during each macro phase of the assembly -- i.e., during the generation of a routine that is to replace a macro instruction in the problem program and/or the generation of DTF routines; and

2. during the assembly of the problem program.

The checks performed during step 2 are those that are normally performed by the Assembler.

```
OBJECT CODE  ADDR1 ADDR2 STMNT   SOURCE STATEMENT
                         0362 FILE     DTFMT DEVADDR=SIS001,
                         0363                EOFADDR=NAME,
                         0364                FILABL=NO,
                         0365                IOAREA1=NAME,
                         0366                IOREG=(10),
                         0367                LABADDR=NO,
                         0368                RECFORM=FIXBLK,
                         0369                TPMARK=NO,
                         0370                TYPEFLE=INPUT,
                         0371                VARBLD=SYMBOL,
                         0372                WORKA=YES
                         0373 MNOTE       TAPEMARK OPTION SPEC'D FOR INPUT FILE
    ERR
                         0374 MNOTE       VARBLD SPEC'D BUT NOT REQUIRED
    ERR
                         0375 MNOTE       NO RECSIZE SPEC'D
    ERR
                         0376 MNOTE       IOREG SPEC'D WITH WORKA
    ERR
                         0377 MNOTE       IMPROPER BLKSIZE
    ERR
                         0388 MNOTE       INVALID DEVADDR SPEC
    ERR
                         0379 MNOTE       LABADDR SPEC' D FOR A NO LABEL FILE
    ERR
                         0380 MNOTE       GENERATION TERMINATED
    ERR
```

Note: Continuation punches (column 72) are not shown.

Figure 18. Example of MNOTE Diagnostic Messages

Checking During the Macro Phase. The IBM-supplied IOCS macro definitions include conditional-assembly instructions that cause the user-specified operands and DTFMT (DTFSR) detail entries to be checked. When an error is detected, an appropriate MNOTE diagnostic message is printed in the program listing. Figure 18 shows an example of MNOTE diagnostic messages that were printed as a result of incorrect and/or inconsistent detail entries to a DTFMT statement. The MNOTE diagnostic messages describe the type of error(s) detected during the macro phase of the assembly.

When an error is detected, diagnosing continues until it is completed for all of the macro instruction operands and detail entries involved. However, the generation of source language coding for the macro instruction (definition statement) that contains the error is suppressed.

Some errors cause the printing of more than one MNOTE diagnostic message. For example, assume that FILABL=STAND has been specified for a file with standard labels. Further assume that a label exit routine has been specified for this file. As a result of the erroneous FILABL detail entry, the following two MNOTE diagnostic messages are printed.

```
MNOTE     IMPROPER FILABL SPEC'D

MNOTE     SPEC'D WITH STAND
          LABEL FILE
```

The second of these two MNOTE diagnostic messages is printed because a LABADDR detail entry is only permitted if FILABL=STD has been specified for the file.

Checking During the Assembly of the Problem Program. The source-language instructions generated from IOCS macro instructions and definition statements are assembled together with the instructions in the user's problem program. Both the generated instructions and the user-coded instructions are tested by the Assembler to detect such coding errors as undefined symbols, invalid length values, etc.

Three examples of error indications by the Assembler are given below. These examples explain some of the error conditions that are detected by the Assembler during the assembly of the problem program.

1. Assume that IOAREA1=OUTPUT has been specified in the definition statement for a file and that, in his program, the programmer has erroneously omitted a definition of the symbol OUTPUT. In this case, all references to OUTPUT in IOCS macro instructions and/or instructions in the problem program are identified as undefined-symbol errors.

2. An undefined-symbol error occurs if the program calls for an IOCS function that is not available for the referenced file. For example, a RELSE macro instruction referring to a file named PAYROLL is issued in a problem program and this file has been defined as an input file that consists of unblocked records. Since the RELSE routine is only available for blocked-record files, the generated reference to the RELSE-routine entry point is identified as an undefined-symbol error. A similar situation occurs if a TRUNC macro instruction is issued for an output file that is to consist of unblocked records.

3. Undefined-symbol errors may occur if the generation of an IOCS routine has been terminated because of an error condition during the macro phase of the assembly. In this case, references are made to non-generated routine entry points, and all of these references are identified as undefined-symbol errors.

RESTRICTIONS

When writing his own routines, the programmer has to observe the following:

1. He must not use any names starting with the letter "I" because all names used by the IOCS start with this letter. This is to eliminate the possibility of multiple-defined names.

2. He must not use file names that are longer than seven characters (refer to Header Entries) because the eighth character position is required by the IOCS.

3. To avoid multi-definition of names, the first seven characters of names used by the programmer should not be identical with the first seven characters of names given to files. The assembler derives the entry points to the IOCS routines by adding a character to the end of the file name.

   Example: If READCRD has been assigned as name to an input file, the programmer should not use names such as READCRDA, READCRDB, etc. in his source program.

4. The name assigned to a file to be processed by the IOCS routines must not be used in the name field of a statement in his program.

5. It is not permitted to give an XIO or an SPSW instruction as this would cause an unexpected interrupt and thus interfere with automatic scheduling of input/output operations by the IOCS.

Use of the FETCH Macro Instruction in Programs Using the IOCS

When a program (or a program phase) that includes input/output routines for new files is loaded, the programmer must ensure that all files used in the preceding program have been closed. If any of the files so closed are required during the newly loaded program, such files must be redefined and subsequently opened in the new program.

Note that the operator is required to run-out the cards in the associated card input/output device if a card file is to be closed for the purpose of loading a new program. Also, a second or subsequent program cannot be loaded from a card input device in which data cards were read during any of the preceding programs.

The Model 20 IOCS is closely patterned after the Basic Programming Support IOCS and the Basic Operating System IOCS. Since the Model 20 IOCS is designed to support card and printer input/output devices that are unique to the Model 20 and to achieve optimum performance of all devices, some macro instructions and DTFSR entries are not identical to those of the other systems. Users who anticipate transition from Model 20 to other System/360 models should therefore be aware that programs using the Model 20 IOCS require some modification prior to generation by the other /360 Systems.

All control cards and labels used and/or required by the Model 20 IOCS as well as card and tape data sets created under control of the Model 20 IOCS are fully upward compatible.

PROGRAMMING EXAMPLE 1

This example (Figure 19) illustrates the files and main-storage area assignments for two tape files and one card file. It is a simplified order and inventory job in which a master tape is updated and written onto a new tape and a card file of detail orders is processed. The following assumptions are made:

- The old master inventory tape contains quantity on hand and unit price in addition to the identifying information.
- The card file reflects quantities ordered. It is to be completed with quantity available for shipment, unit price, and the extension of quantity shipped times the unit price.
- The new master inventory tape reflects the decrease in quantity on hand due to the current orders or an increase when items are returned.

The paragraph numbers of the following text correspond to the coded numbers in Figure 19. The illustration shows this setup:

1. Job Control cards to indicate to the Basic Monitor program the type of job (assembly run) to be executed.

2. Definition statements to define the three files.

   a. Old master tape file. This is an input file to be read forward. It contains standard volume and file labels and additional user 80-character file labels. It is a card-image file with a blocking factor of 5. Register 10 is assigned for locating individual records in the input area.

   b. New master tape file. This is an output file with the same characteristics as the input file. Register 11 is assigned for locating the next available output-record area.

   c. Detail card file. This is a combined file used to update input records from the old master tape file. The cards are in the primary feed of an IBM 2560 MFCM and are read in the read station of that machine.

   d. End of the three file definition macro instructions. The user's source program follows this statement.

3. Sample instructions to open files and locate master records that have current activity. This illustrates the following:

   a. The registers used in the program are defined by means of USING instructions. Register 12 is used as base register and the address of the next instruction is loaded into this register. The use of registers 10 and 11 is explained later in this section.

   b. All files to be processed by the IOCS must be opened.

   c. A GET for an unblocked record to be processed in a work area causes the record to be transferred from the input/output device to main storage. This makes the record available to the problem program.

   d. The first GET for a record in a blocked file causes the physical transfer of the block of data from the input/output device to the input/output area. It also places the address of the first record into the specified ICREG register. Each succeeding GET causes the address of the currently available record to be placed in the IOREG register, and may or may not cause a transfer of data.

   e. The master and detail item numbers are compared with each other to determine if either (1) a master record is missing or a card with a new detail item number has been read, or (2) the master record is to be updated, or (3) the master record is to be written unchanged on the output tape.

      To address any field within the OMSTR input record being processed (AREA1), register 10 has been assigned by the Assembler as base register because this register produces the lowest displacement. Since register 10 has also been specified in the ICREG entry for the OMSTR file, the IOCS automatically loads the register and changes its contents to point to the begin address of the record being processed. When this programming technique is used, the programmer need not specify a base register in

instructions referring to fields of the OMSTR input record currently being processed.

The same programming technique is used to address the NMSTR output records. However, register 11 has been assigned by the Assembler as a base register for the addresses of the fields within the NMSTR record being processed (AREA2).

f. The UPDATE routine is entered whenever it is determined that a record from the OMSTR input file requires updating.

When output files specify an IOREG register, the register is initialized by the OPEN routine. A PUT to an output file for which a work area was not specified merely causes the address of the next available record area to be placed into the specified IOREG register. No data is moved within storage, and a transfer of the data to the output device may or may not occur.

g. The ERROR routine is entered when either a master record is missing or a new detail number has been detected. In this routine, the user may build a record for inclusion in the new master file.

4. User routines for processing additional standard labels.

5. All files that have been opened must be closed. This is normally done in an end-of-file routine.

6. Assembler instructions to define the input and output areas for the three files.

   a. Two input/output areas and one work area are reserved for detail card records. Because these are single unblocked records, the individual fields within the records may be defined along with the allocation of the work area as shown.

   b. An input/output area is reserved for each of the two master files. Note the use of the zero-duplication factor to define fields within logical records of the OMSTR input file.

IBM System/360 Assembler Coding Form

PROGRAM IOCS EXAMPLE PROGRAMMER DATE PUNCHING INSTRUCTIONS GRAPHIC PUNCH PAGE 1 OF 6 CARD ELECTRO NUMBER

STATEMENT

```
(1)  // JOB    ASSEMBLER
     // EXEC
*
     READY    START 2500
*
*                        FILE DEFINITION SECTION
*
*                        MASTER FILE DEFINED AS AN INPUT FILE
*
(2a) OMSTR    DTFMT TYPEFLE=INPUT,READ=FORWARD,FILABL=STD,RECFORM=FIXBLK,     X
                    BLKSIZE=400,RECSIZE=80,IOAREA1=AREA1,IOREG=(10),          X
                    LABADDR=CKOLAB,ERROPT=SKIP,EOFADDR=EOFMST,REWIND=UNLOAD,X
                    DEVADDR=SYS002
*
*                        MASTER FILE DEFINED AS AN OUTPUT FILE
*
(2b) NMSTR    DTFMT TYPEFLE=OUTPUT,FILABL=STD,RECFORM=FIXBLK,BLKSIZE=400,     X
                    RECSIZE=80,IOAREA1=AREA2,IOREG=(11),LABADDR=CKNLAB,       X
                    REWIND=UNLOAD,DEVADDR=SYS003
*
```

Figure 19. Programming Example 1, Part 1 of 6

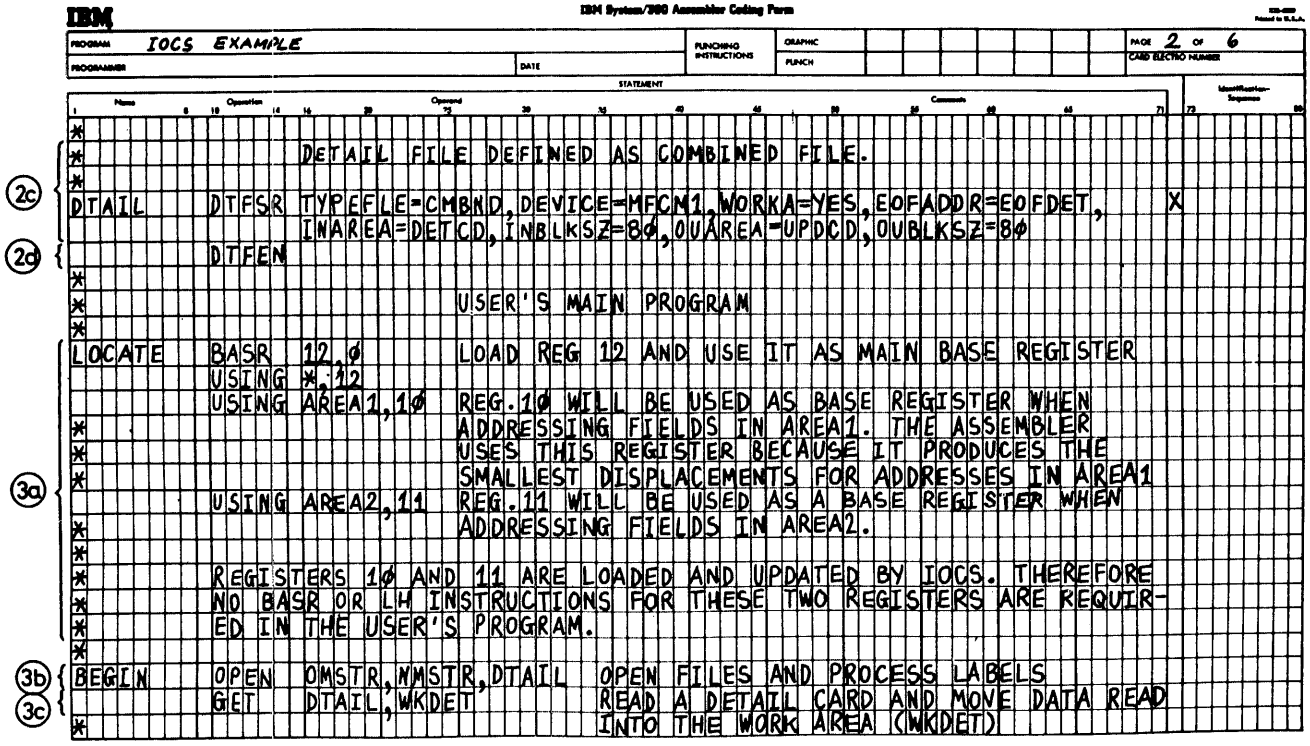


Figure 19. Programming Example 1, Part 2 of 6

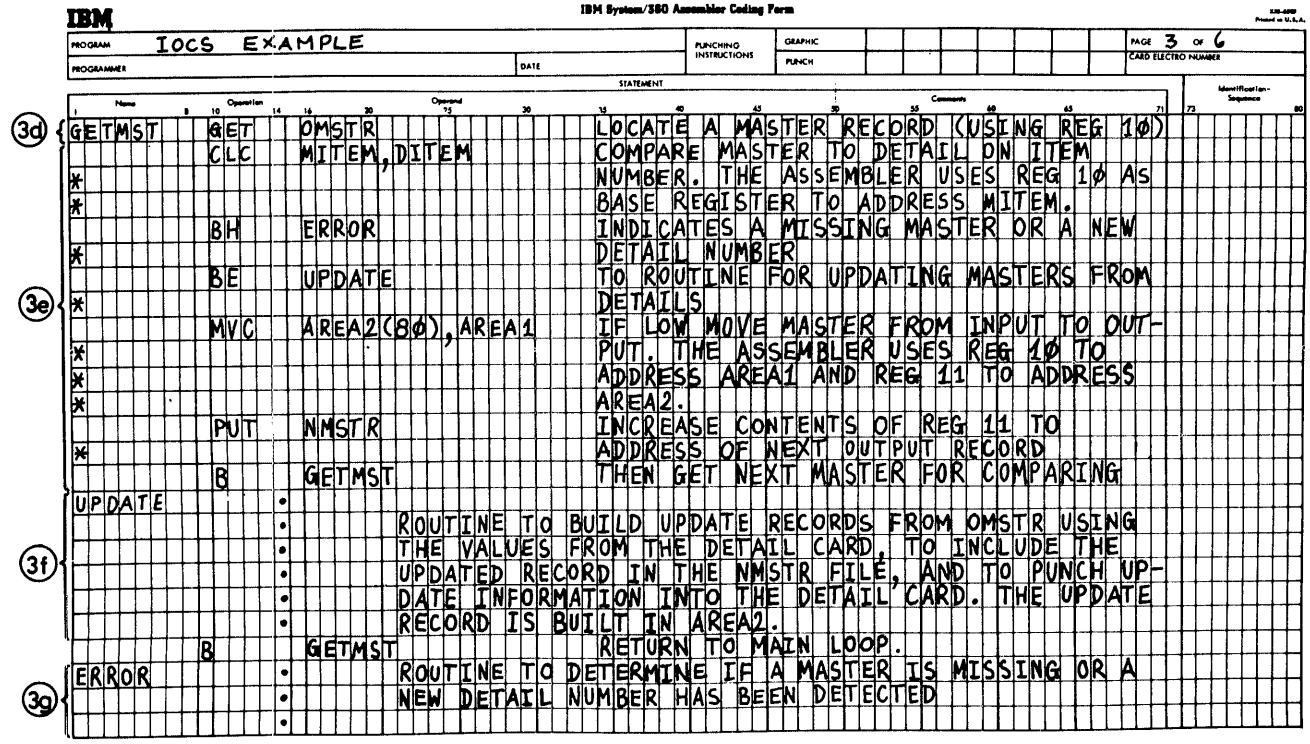


Figure 19. Programming Example 1, Part 3 of 6

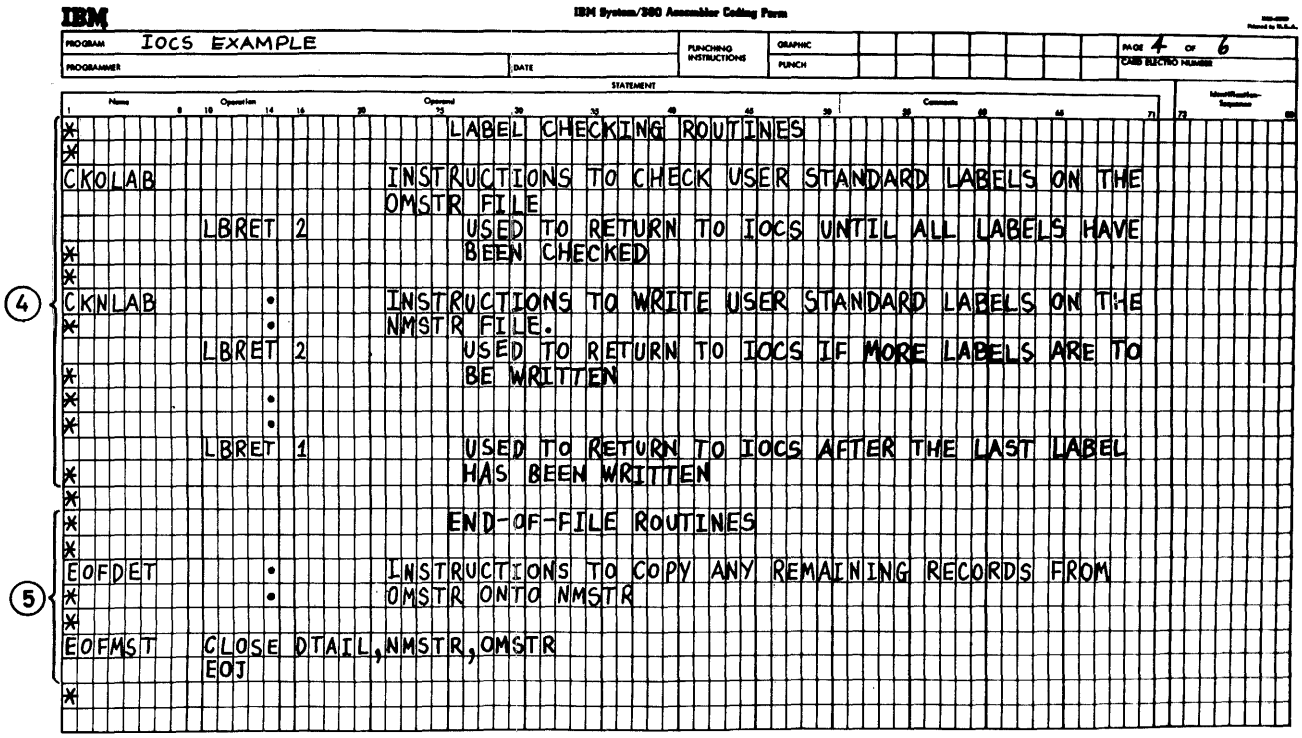


Figure 19. Programming Example 1, Part 4 of 6

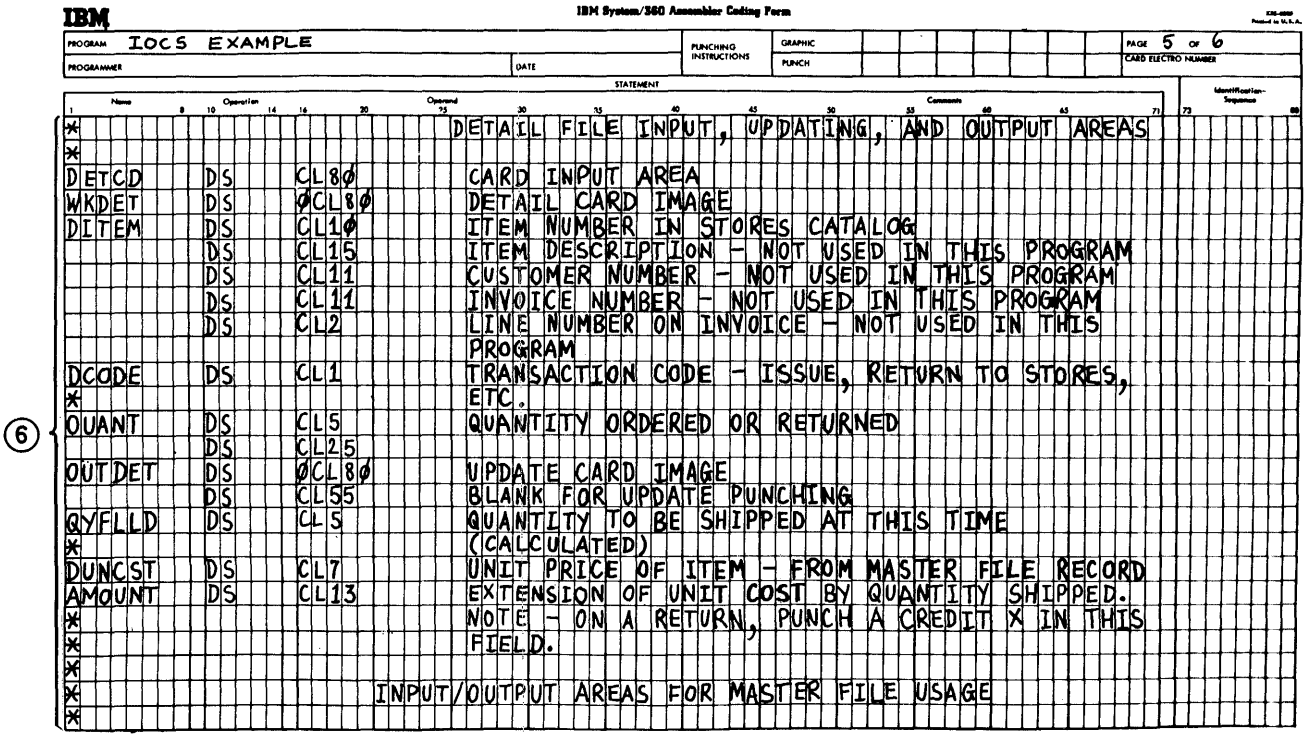


Figure 19. Programming Example 1, Part 5 of 6

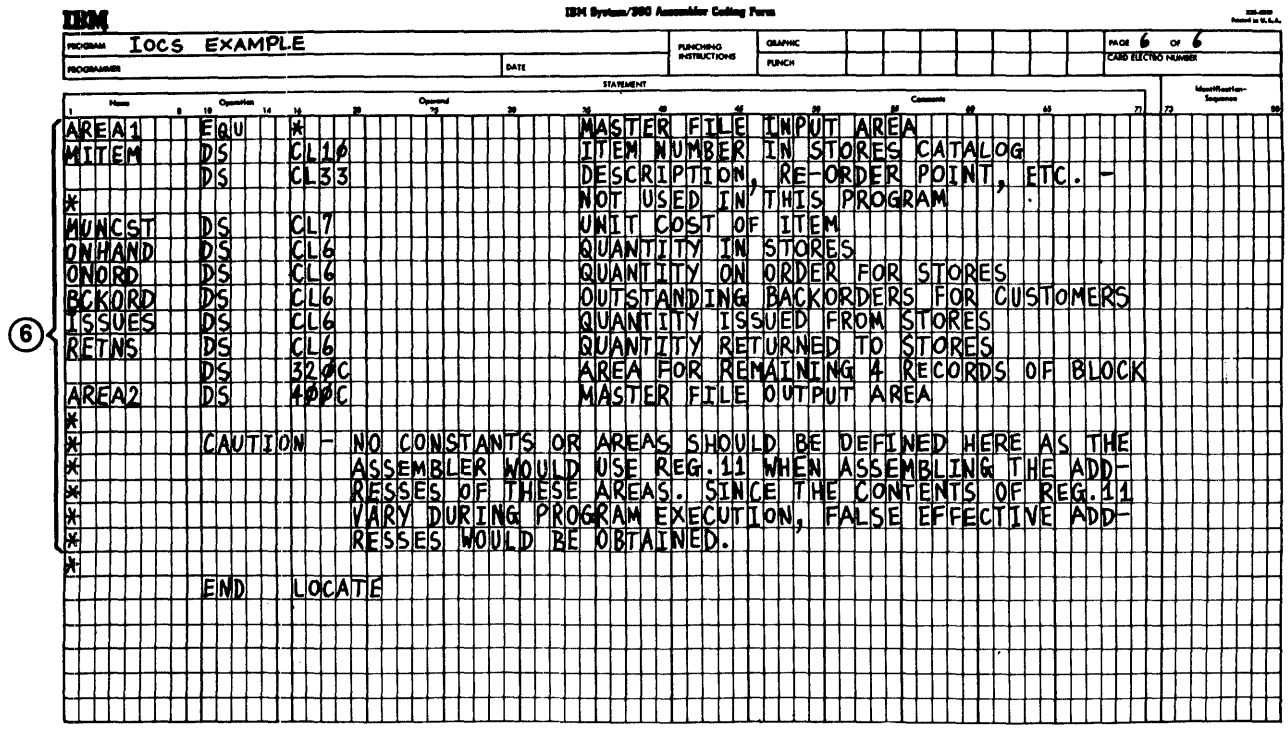


Figure 19. Programming Example 1, Part 6 of 6

The following simple example (Figure 20) illustrates how the read/compute, write/compute overlap feature is used, in connection with two I/O areas and a work area.

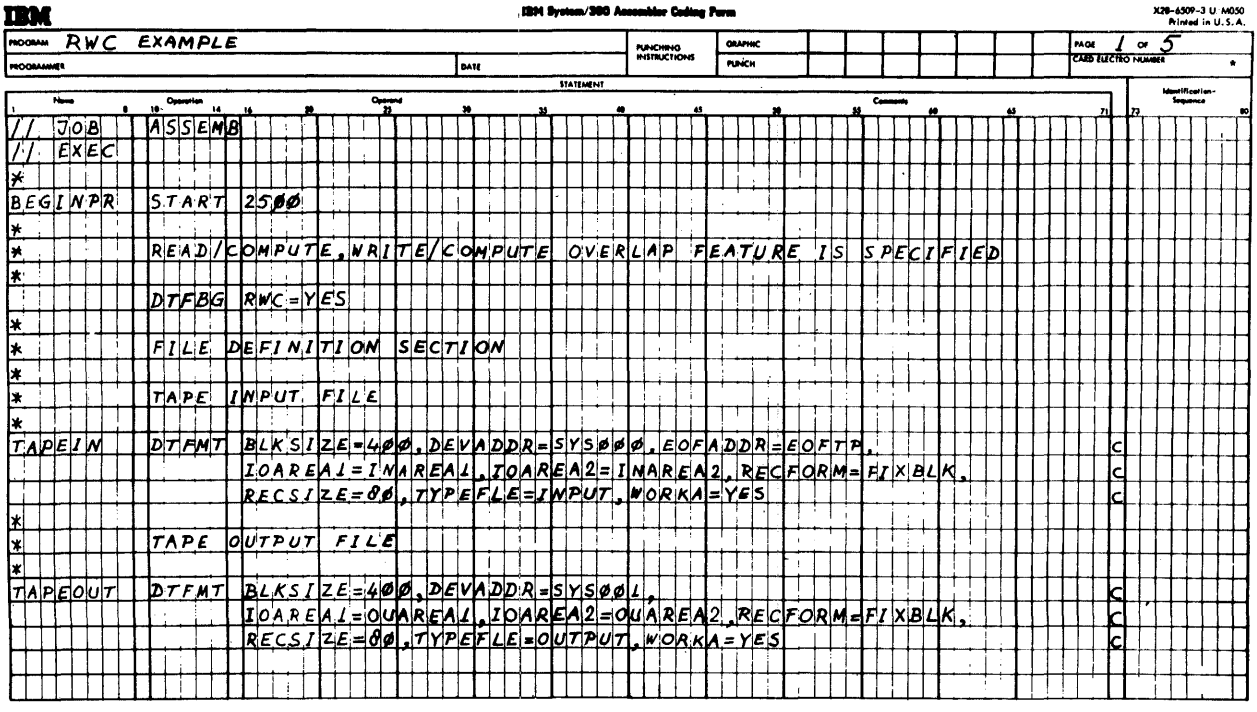
A maximum degree of overlap is achieved as follows:

Input. The combined use of two input areas allows the processing of one block to be overlapped with the reading of the next one. When the last logical record of a block in one area has been processed, the reading of the next block from tape into the same area is overlapped with the processing of the block in the second input area.

Output. The combined use of two output areas allows the writing of one block to be overlapped with the building of the next one. When one output area is full, the block is written onto tape and, at the same time, another block is built in the second output area.

The job being done is basically the same as in the previous example. An old master inventory tape contains quantities on hand together with identifying information. A card input file reflects quantities received or dispatched (only the first 20 columns of each card contain data). The new master output file contains the resulting new quantities. A list is printed of all records on tape; any changed quantities are marked with an asterisk in the last print position.

For simplicity, no error routine is included. The program halts if a card is read for which there is no corresponding tape record, or if the end-of-tape has been reached before the last card has been processed. When the last tape record has been processed, the job is completed.



```
//  JOB   ASSEMB
//  EXEC
*
BEGINPR  START  2500
*
*        READ/COMPUTE,WRITE/COMPUTE OVERLAP FEATURE IS SPECIFIED
*
         DTFBG  RWC=YES
*
*        FILE DEFINITION SECTION
*
*        TAPE INPUT FILE
*
TAPEIN   DTFMT  BLKSIZE=400,DEVADDR=SYS000,EOFADDR=EOFTP,          C
                IOAREA1=INAREA1,IOAREA2=INAREA2,RECFORM=FIXBLK,    C
                RECSIZE=80,TYPEFLE=INPUT,WORKA=YES                 C
*
*        TAPE OUTPUT FILE
*
TAPEOUT  DTFMT  BLKSIZE=400,DEVADDR=SYS001,                        C
                IOAREA1=OUAREA1,IOAREA2=OUAREA2,RECFORM=FIXBLK,    C
                RECSIZE=80,TYPEFLE=OUTPUT,WORKA=YES                C
```

● Figure 20. Programming Example 2, Part 1 of 5

IBM System/360 Assembler Coding Form

PROGRAM: RWC EXAMPLE — PAGE 2 OF 5

```
*
*        CARD INPUT FILE (DATA ARE CONTAINED IN THE FIRST
*                             TWENTY CARD COLUMNS ONLY)
CARDIN   DTFSR BLKSIZE=20,DEVICE=READ01,EOFADDR=EOFCRD,                C
               IOAREA1=CRDAREA,TYPEFLE=INPUT,WORKA=YES                 C
*
*        PRINTER FILE
*
PRINT    DTFSR BLKSIZE=81,CONTROL=YES,DEVICE=PRINTER,                  C
               TYPEFLE=OUTPUT,WORKA=YES                                C
         DTFEN
         USING BEGINPR-2500,0,1,2,3   THIS IS A NON-RELOCATABLE PROGRAM
BEGINEX  OPEN  TAPEIN,TAPEOUT,CARDIN,PRINT    OPEN FILES
         CNTRL PRINT,SK,1             SKIP TO CHANNEL 1
         GET   CARDIN,WORKCRD         READ A CARD
GETTAPE  GET   TAPEIN,WORKTP          MAKE RECORD AVAILABLE IN WORK AREA
COMP     CLC   NOCRD,NOTAPE           COMPARE ITEM NO IN CARD WITH
*                                     ITEM NO IN RECORD
         BE    UPDATE                 RECORD FOUND
         BL    ERROR                  INDICATES A MISSING RECORD ON TAPE
         PUT   TAPEOUT,WORKTP         RECORD NOT YET FOUND, COPY RECORD
*                                     WITH LOWER NUMBER ON OUTPUT TAPE
         PUT   PRINT,WORKTP           PRINT RECORD
```

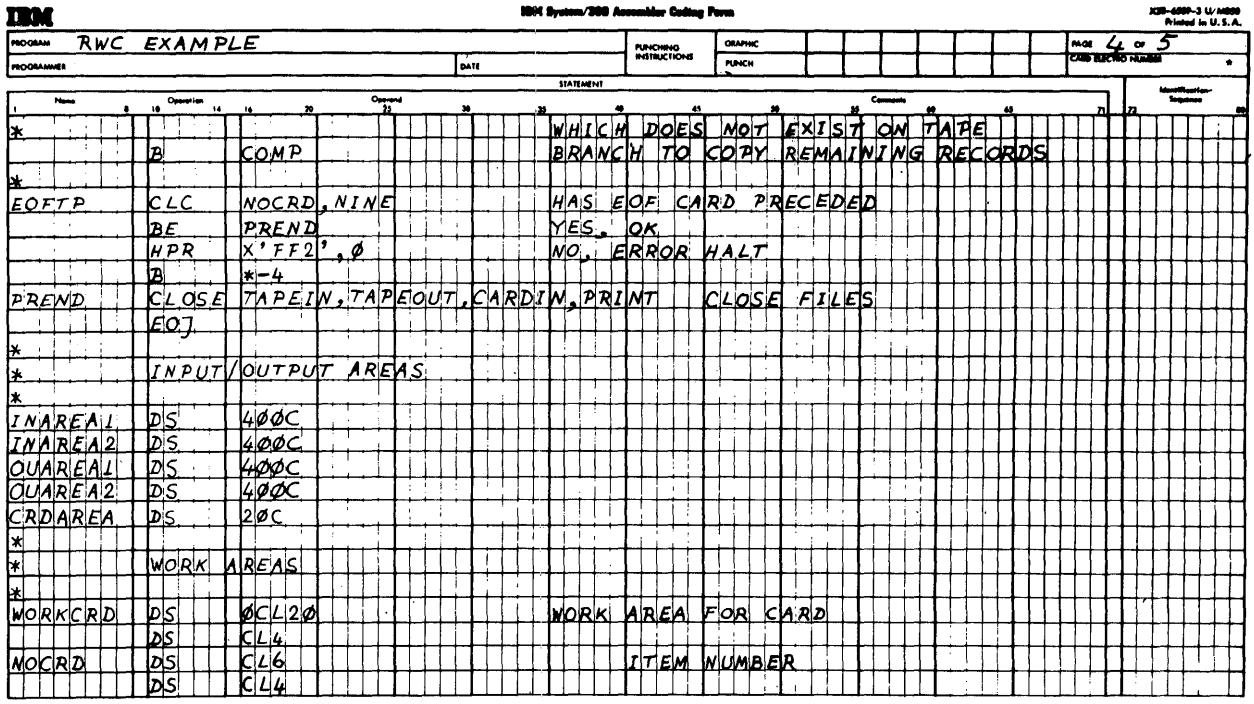
● Figure 20. Programming Example 2, Part 2 of 5

IBM System/360 Assembler Coding Form

PROGRAM: RWC EXAMPLE — PAGE 3 OF 5

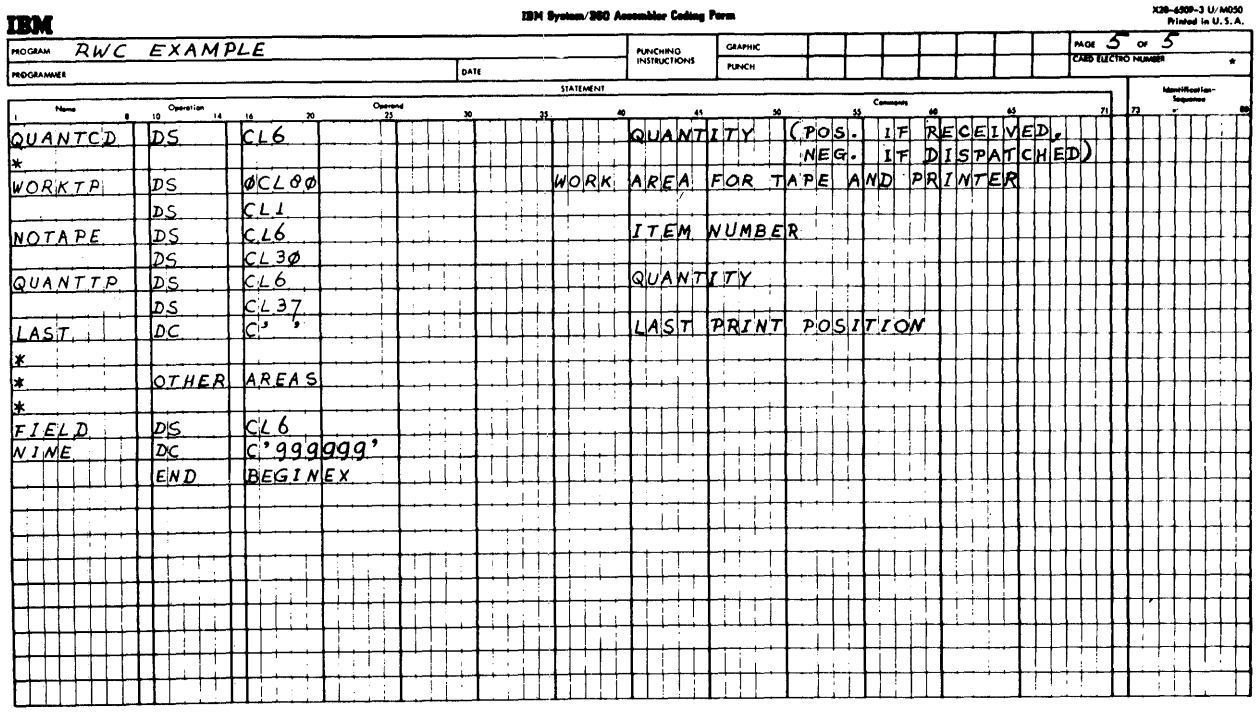
```
         MVI   LAST,C' '              BLANK LAST PRINT POSITION
         B     GETTAPE                BRANCH TO GET NEXT RECORD
*
*        ROUTINE TO UPDATE TAPE RECORD.  THE QUANTITY IN THE TAPE
*        RECORD IS INCREASED BY THE QUANTITY IN THE CARD AND
*        THE PRINT AREA IS MARKED WITH *.
*
UPDATE   PACK  QUANTCD,QUANTCD
         PACK  FIELD,QUANTTP
         AP    FIELD,QUANTCD          ADD QUANTITY (SUM MAY NOT
         UNPK  QUANTTP,FIELD+2(4)                     EXCEED 999999)
         MVI   LAST,C'*'              MOVE * TO LAST PRINT POSITION
         GET   CARDIN,WORKCRD         READ NEXT CARD
         B     COMP                   BRANCH TO COMPARE
*
*        ERROR ROUTINE FOR MISSING RECORD.
*        HERE IT IS ONLY A HALT.
*
ERROR    HPR   X'FF1',0
         B     *-4
*
*        END OF FILE ROUTINES
*
EOFCRD   MVC   NOCRD,NINE             SET CARD NUMBER TO A HIGH VALUE,
```

● Figure 20. Programming Example 2, Part 3 of 5



```
IBM System/360 Assembler Coding Form                                  X28-6509-3 U/M050 Printed in U.S.A.
PROGRAM  RWC EXAMPLE                                  PUNCHING INSTRUCTIONS  GRAPHIC  PUNCH
PROGRAMMER                          DATE                                    PAGE 4 OF 5
                                                                            CARD ELECTRO NUMBER
STATEMENT
Name      Operation   Operand                  Comments                                  Identification-Sequence
*                                              WHICH DOES NOT EXIST ON TAPE
          B           COMP                     BRANCH TO COPY REMAINING RECORDS
*
EOFTP     CLC         NOCRD,NINE               HAS EOF CARD PRECEDED
          BE          PREND                    YES, OK
          HPR         X'FF2',Ø                 NO, ERROR HALT
          B           *-4
PREND     CLOSE       TAPEIN,TAPEOUT,CARDIN,PRINT   CLOSE FILES
          EOJ
*
*         INPUT/OUTPUT AREAS
*
INAREA1   DS          4ØØC
INAREA2   DS          4ØØC
OUAREA1   DS          4ØØC
OUAREA2   DS          4ØØC
CRDAREA   DS          2ØC
*
*         WORK AREAS
*
WORKCRD   DS          ØCL2Ø                    WORK AREA FOR CARD
          DS          CL4
NOCRD     DS          CL6                      ITEM NUMBER
          DS          CL4
```

● Figure 20. Programming Example 2, Part 4 of 5



```
IBM System/360 Assembler Coding Form                                  X28-6509-3 U/M050 Printed in U.S.A.
PROGRAM  RWC EXAMPLE                                  PUNCHING INSTRUCTIONS  GRAPHIC  PUNCH
PROGRAMMER                          DATE                                    PAGE 5 OF 5
                                                                            CARD ELECTRO NUMBER
STATEMENT
Name      Operation   Operand                  Comments                                  Identification-Sequence
QUANTCD   DS          CL6                      QUANTITY (POS. IF RECEIVED,
*                                                         NEG. IF DISPATCHED)
WORKTP    DS          ØCL8Ø                    WORK AREA FOR TAPE AND PRINTER
          DS          CL1
NOTAPE    DS          CL6                      ITEM NUMBER
          DS          CL3Ø
QUANTTP   DS          CL6                      QUANTITY
          DS          CL37
LAST      DC          C' '                     LAST PRINT POSITION
*
*         OTHER AREAS
*
FIELD     DS          CL6
NINE      DC          C'999999'
          END         BEGINEX
```

● Figure 20. Programming Example 2, Part 5 of 5

Address.
1. An identification, as represented by a name, or number, for a register, location in storage, or other data source or destination.
2. Loosely, any part of an instruction which specifies the location of an operand for the instruction.

Allocate. To assign storage locations or areas of storage for specific routines, portions of routines, constants, data, etc.

Alphameric. A generic term for alphabetic letters, numerical digits, and special characters.

Ascending Order. A sequence of records such that the control fields of each successive record collate equal to or higher than those of the preceding record.

Assemble. To prepare an object-language program from a symbolic-language program by substituting machine operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses.

Basic Monitor. The main control program. Available in a card, a tape, and a disk version. Resident in main storage when control required. Loads programs into main storage and causes their execution.

Binary.
1. A characteristic or property involving a selection, choice, or condition in which there are two possibilities.
2. The number representation system with a base of two.

Bit. A binary digit.

Blank Character. Any character or characters used to produce a character space on an output medium.

Block (records).
1. To group records for the purpose of conserving storage space or increasing the efficiency of access or processing.
2. A physical record so constituted, or a portion of a telecommunications message defined to be a unit of data transmission.

Branch.
1. To depart from the normal sequence of executing instructions in a computer.
2. A machine instruction that can cause a departure as in (1). Synonymous with 'transfer'.

Buffer (Program Input/Output). A portion of main storage into which data is read, or from which it is written.

Byte. A sequence of adjacent binary digits operated upon as a unit.

Card Stacker. A mechanism which stacks cards in a pocket after they pass through a machine.

Central Processing Unit. A unit of a computer that includes circuits controlling the interpretation and execution of instructions.

Checkpoint. A point in a program about which sufficient information is stored to permit restarting the problem from that point.

Column Binary. Pertaining to the binary representation of data on punched cards in which adjacent positions in a column correspond to adjacent bits of data.

Command. An instruction in machine language.

Constant. A fixed or invariable value or data item.

Control Program. A set of programs which provide the management functions necessary for continuous operation of a computing system.

Control Statement. Any of the statements in the input to a specific job that define the requirements of the job, its options, or control its actions.

Counter. A device such as a register or storage location used to represent the number of occurrences of an event.

CPU. See Central Processing Unit.

Cycle.
1. An interval of space or time in which one set of events is completed.
2. Any set of operations that is repeated regularly in the same sequence. The operations may be subject to variations on each repetition.

Data. Any representation, such as character quantities, to which meaning might be assigned.

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 Processing. A systematic sequence of operations performed on data.

Data Processing System. A network of machine components capable of accepting information, processing it according to a plan, and producing the desired results.

Data Record. See Logical Record.

Data Set. See Data File.

Decimal.
1. A characteristic or property involving a selection, choice or condition in which there are ten possibilities.
2. The number representation system with a base of ten.

Deck. A collection of punched cards.

Default Value. The operand specification assumed by a program when the value is omitted.

Descending Order. A sequence of records such that the control fields of each successive record collate equal to or lower than those of the preceding record.

Digit.
1. Any of the arabic numerals 1 to 9 and the symbol 0.
2. One of the elements that combine to form numbers in a system other than the decimal system.

EBCDIC. (Extended Binary Coded Decimal Interchange Code) A specific set of eight-bit codes standard throughout System/360.

Error. A general term to indicate that a data value is not correct or that a machine component is malfunctioning.

File. A collection of related records treated as a unit, e.g., in inventory control, one line of an invoice forms an item, a complete invoice forms a record, and the complete set of such records forms a file.

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.

Fixed-Length Record. A record having the same length as all other records with which it is logically or physically associated.

Halfword Boundary. Even-numbered byte position in main storage, coincident with the left byte of a halfword.

Hexadecimal. A number system using the equivalent of the decimal number 16 as a base. The values 0-15 are represented by the digits 0-9 and the alphabetic characters A-F.

Hopper. A device that holds cards and makes them available to a card feed mechanism. Contrast with card stacker.

Indexing. A technique of address modification often implemented by means of index registers.

Index Register. A register whose contents is added to or subtracted from the operand address prior to or during the execution of an instruction.

Initialize. To set certain counters, switches and addresses at specified times in a computer routine.

Input.
1. The data to be processed.
2. The state or sequence of states occurring on a specified input channel.
3. The device or collective set of devices used for bringing data into another device.
4. A channel for impressing a state on a device or logic element.

Input Area. The area of internal storage into which data is transferred from external storage.

Input/Output.
1. Common abbreviation I/O. A general term for the equipment used to communicate with a computer.
2. The data involved in such communication.
3. The media carrying the data for input/output.

Instruction. A statement that specifies an operation and the values or locations of all operands. In this context, the term instruction is preferable to the terms command or order which are sometimes used as synonyms. Command should be reserved for electronic signals. Order should be reserved for sequence, interpolation and related usage.

Instruction Format. The allocation of bits or characters of a machine instruction to specific functions.

Interrupt.
1. A break in the normal flow of a system or routine such that the flow can be resumed from that point at a later time.
2. To cause an interrupt

I/O Area. An area (portion) of main storage into which data is read or from which data is written. I/O means Input/Output.

Job Control Program. A System Control program. Called into main storage between jobs and provides for automatic job-to-job transmission. Processes control statements in the input stream that identify a job or define its requirements and options.

Label. A physical identification record on magnetic tape (or disk).

Linkage. The interconnections between a main routine and a closed routine,i.e., entry and exit for a closed routine from the main routine.

Load. To place data into internal storage.

Location. A position in storage that is usually identified by an address.

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. (Contrast with Physical Record.)

Machine Instruction. An instruction that the particular machine can recognize and execute.

Macro Instruction. A statement that is used in a source program and replaced by a specific sequence of machine instructions in the associated object program.

Macro Library (Tape). An area of the macro library section of the system tape. Has four priority sections, each of which contains the macro definitions required by the macro instructions in user programs.

Magnetic Tape. A tape with a magnetic surface on which data can be stored.

Main Storage. The fastest general purpose storage of a computer. Also, for the Model 20, storage within the CPU that can be addressed both for reading and writing data.

Mnemonic Code. A mnemonic code resembles the original word and is usually easy to remember, e.g., ED for edit and MVC for move characters.

Name. An alphameric character string, normally used to identify a program.

Object Program. A fully assembled program ready to be loaded in the computer.

Operand. That which is operated upon. An operand is usually identified by an address part of an instruction.

Operation.
1. The act specified by a single computer instruction.
2. A program step undertaken or executed by a computer, e.g., addition, multiplication, extraction, comparison, shift, or transfer. The operation is usually specified by the operation part of an instruction.

Operation Code. The code that represents the specific operations of a computer.

Output.
1. Data that has been processed.
2. The state or sequence of states occurring on a specified output channel.
3. The device or collective set of devices used for taking data out of a device.
4. A channel for expressing a state on a device or logic element.

Output Area. The area of internal storage from which data is transferred to external storage.

Overlap. To do something at the same time that something else is being done; for example, to perform input/output operations while instructions are being executed by the central processing unit.

Overlay. To place a phase or subphase into main storage locations occupied by another phase or subphase that has already been processed.

Pack. To combine two or more units of information into a single physical unit to conserve storage.

Padding. A technique used to fill a block of information with dummy records, words or characters.

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. (Contrast with Logical Record.)

Problem Program. A general term for any program that is not a control program.

Program.
1. The plan for the solution of a problem including data gathering, processing and reporting.

2. A group of related routines which solve a given problem.

Process. A systematic sequence of operations to produce a specified result.

Read. To transfer information from an input device to internal or auxiliary storage.

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.

Reader. A device which converts information in one form of storage to information in another form of storage.

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.

Register. A device capable of storing a specified amount of data such as one halfword.

Relocate. In programming, to move a routine from one portion of internal storage to another and to automatically adjust the necessary address references so that the routine, in its new location, can be executed.

Relocation. The modification of address constants required to compensate for a change of origin of a phase or subphase.

Routine. An ordered set of instructions that may have some general or frequent use.

RWC Feature. See Read/Compute, Write/Compute Overlap Feature.

Source Language. A language that is an input to a given translation process.

Source Program. A program written in a source language.

Special Character. In a character set, a character that is neither a numeral nor a letter, e.g., -*$ = and blank.

Statement. In computer programming, a meaningful expression or generalized instruction in a source language.

Storage.
1. Pertaining to a device into which data can be entered and from which it can be retrieved at a later time.
2. Loosely, any device that can store data.

Storage Capacity. The amount of data (in bytes) that can be contained in a storage device.

Store.
1. To enter data into a storage device.
2. To retain data in a storage device.

Subroutine. A routine that can be part of another routine.

Switch.
1. A symbol used to indicate a branching point, or a set of instructions to condition a branch.
2. A physical device which can alter flow.

Symbolic Address. An address expressed in symbols convenient to the programmer.

System.
1. A collection of consecutive operations and procedures required to accomplish a specific objective.
2. An assembly of objects united to form a functional unit.

Tape Mark. 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.

Truncate. To cut off at a specified spot (as contrasted with round or pad).

Unpack. To recover the original data from packed data.

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.

INDEX

IBM System/360 Printed in U.S.A. C24-9003-4

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,
Tape Programming System,
Input/Output Control System

Form C24-9003-4

- 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 U

Fold

Fold

CUT ALONG THIS LINE

IBM System/360 Printed in U.S.A. C24-9003-4

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]