File No. S360(Mod.20)-30
Order No. GC26-3603-4

CPS

IBM Systems Reference Library

IBM System/360 Model 20
Card Programming Support
Input/Output Control System

This publication describes the functions, principal features, and use of the Model 20 Card Programming Support, Input/Output Control System (CPS IOCS). The Model 20 IOCS provides tested input/output routines that programmers, by means of macro instructions, can use to control the input and output of data by programs written in the Model 20 Basic Assembler Language.

Included in the publication are sections describing the generation of the IOCS routines, the definition statements that a programmer uses to describe his application to the IOCS, and the macro instructions that the programmer uses in his main source program when he desires an input/output operation to be performed. Also included are sections containing program performance data and a sample program.

The programmer should be familiar with the SRL publication IBM System/360 Model 20 Card Programming Support, Basic Assembler Language, Order No. GC26-3602.

Fifth Edition (April, 1970)

This is a major revision of, and obsoletes, GC26-3603-2, -3, and Technical Newsletters GN33-8502 and GN33-8524. Minimum and maximum machine requirements, as well as Submodel 5 information, have been added. Further changes and additions have been made throughout this publication. 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 version 2, modification 1, of IBM System/360 Model 20, Card Programming Support, Input/Output Control System. The program to which this publication applies falls under Programming Service Classification C. Therefore, no further centralized maintenance of the program or the publication should be expected.

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

Comments concerning the contents of this publication may be addresses to IBM Laboratory, Publications Dept., P.O.Box 24, Uithoorn, Netherlands.

Contents

Card Programming Support, Input/Output Control System

This publication describes the functions, principal features, and use of the Model 20 Card Programming Support Input/Output Control System (CPS IOCS). This IOCS is a set of tested routines and macro instructions which are used to control the input and output of data written in the CPS Basic Assembler language. The CPS IOCS must be used in conjunction with the CPS Basic Assembler language. Therefore, the user should be familiar with the contents of the Systems Reference Library (SRL) publication IBM System/360 Model 20 Card Programming Support, Basic Assembler Language, Order No. GC26-3602.

A major part of most programs written in the Basic Assembler language consists of routines required to read data into the system and to output the results of the processing performed on the input data. A programmer can use the Model 20 IOCS routines in his Basic Assembler source program by means of macro instructions. Programming time is saved because the user does not have to code, to test, or to provide linkages to his own input/output routines. He can concentrate on solving his problem and let IOCS handle the input/output. In addition, the IOCS routines take advantage of the time-sharing capability of the Model 20, thereby optimizing throughput.

USE OF THE IOCS

The user is provided with a complete set of input/output routines to perform all possible Model 20 input/output operations. Not all of these routines need be included in every source program because only certain types of input/output operations may be required for a given application. Therefore, IBM provides, as part of the IOCS, a generator program whose function is to select the routines required by the user and develop them for his application. This procedure minimizes main storage requirements because routines and parts of routines not required are not generated.

The generator program first reads definition statements made by the user describing the input/output operations required by the application. Based on these statements the generator selects the required input/output routines from the IOCS routine library, develops them for the specific application, and punches them into cards in the Basic Assembler source language format. These routines can be assembled in conjunction with a Basic Assembler source program or can be assembled separately.

In the case of separate assembly, the routines are punched into cards in the Basic Assembler relocatable format. They are then loaded together with the Basic Assembler object program by means of the Relocatable Program Loader. One advantage of this procedure is that the Basic Assembler source program can contain a larger number of symbols.

The IOCS routines are inserted into the object program as subroutines. Therefore, there must be a linkage to the correct IOCS routine from each point in the program where an input/output operation is to occur. The user need not provide these linkages. He only writes one instruction (macro instruction) in his source program at the point he desires the input/output operation to occur. When a macro instruction is read during assembly of the source program, the Basic Assembler program automatically inserts the required linkages to and from the IOCS routines.

IOCS ASSEMBLY

The deck of cards containing the IOCS routines in Basic Assembler symbolic language can be assembled either separately or in conjunction (jointly) with the Basic Assembler source program (Figures 1 and 2).

Separate IOCS assembly has two significant advantages:

1. The user needs to assemble the required IOCS portion of his program only once for any given application. This results in significant savings in machine time that would otherwise be required for reassemblies necessitated by changes to the source program.

2. Separate assembly permits the programmer to use nearly the maximum number of symbols in his Basic Assembler source program.

If both the IOCS routines and the main program are assembled separately, they must be assembled in relocatable form. Both must be loaded by the relocatable loader, but the main program need not be relocated. The main program must be loaded first and must not contain the name I001.

If the IOCS routines are assembled separately, these routines use three linkage symbols. Therefore, the programmer can use only eleven linkage symbols (EXTRN and ENTRY statements) in his main program. The IOCS routines require three additional linkage symbols if the IOCS is used in conjunction with the 1419 IOCS and one additional linkage symbol if the IOCS is used in conjunction with the CIOCS (Communication Input/Output Control System).

If the IOCS routines and the main program are assembled jointly, the main source program must not contain any

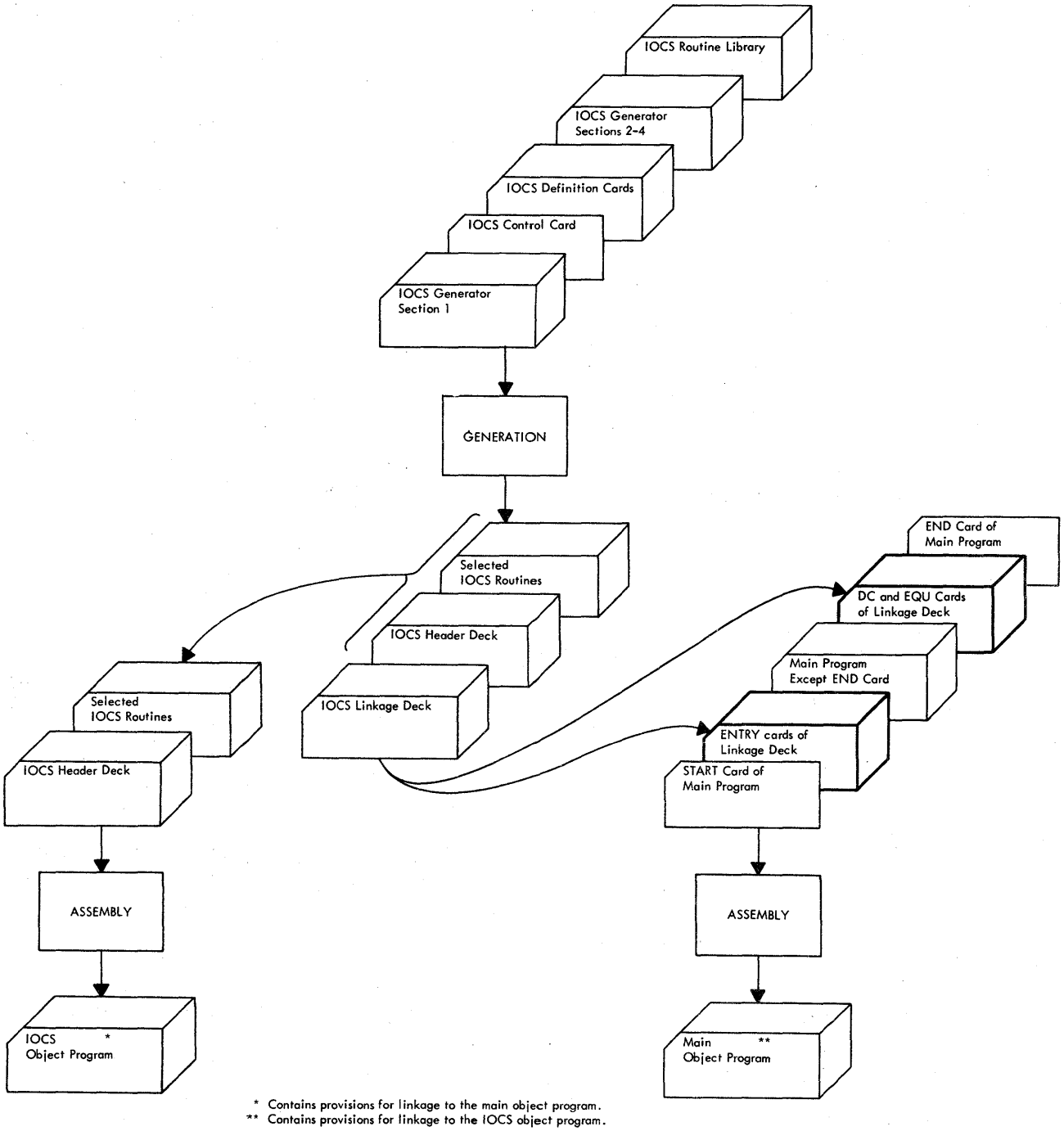

Figure 1. Separate Assembly of IOCS

symbolic names that consist of a letter I followed by three numeric characters (0-9). These symbols may not be used because the IOCS uses them.

The symbolic name assigned to the file to be processed by the IOCS routines must not be used in the name field of any statement in the main program.

To ensure that the base registers used in the main program do not interfere with the base register assignments for the generated IOCS routines, the programmer should write DROP instructions for these base registers at the end of the main program.

The first 156 bytes of main storage (addresses 0-9B) are reserved. They are not available to the user.

Machine Requirements

This section describes the minimum and maximum system configurations for assembling and executing CPS IOCS routines.

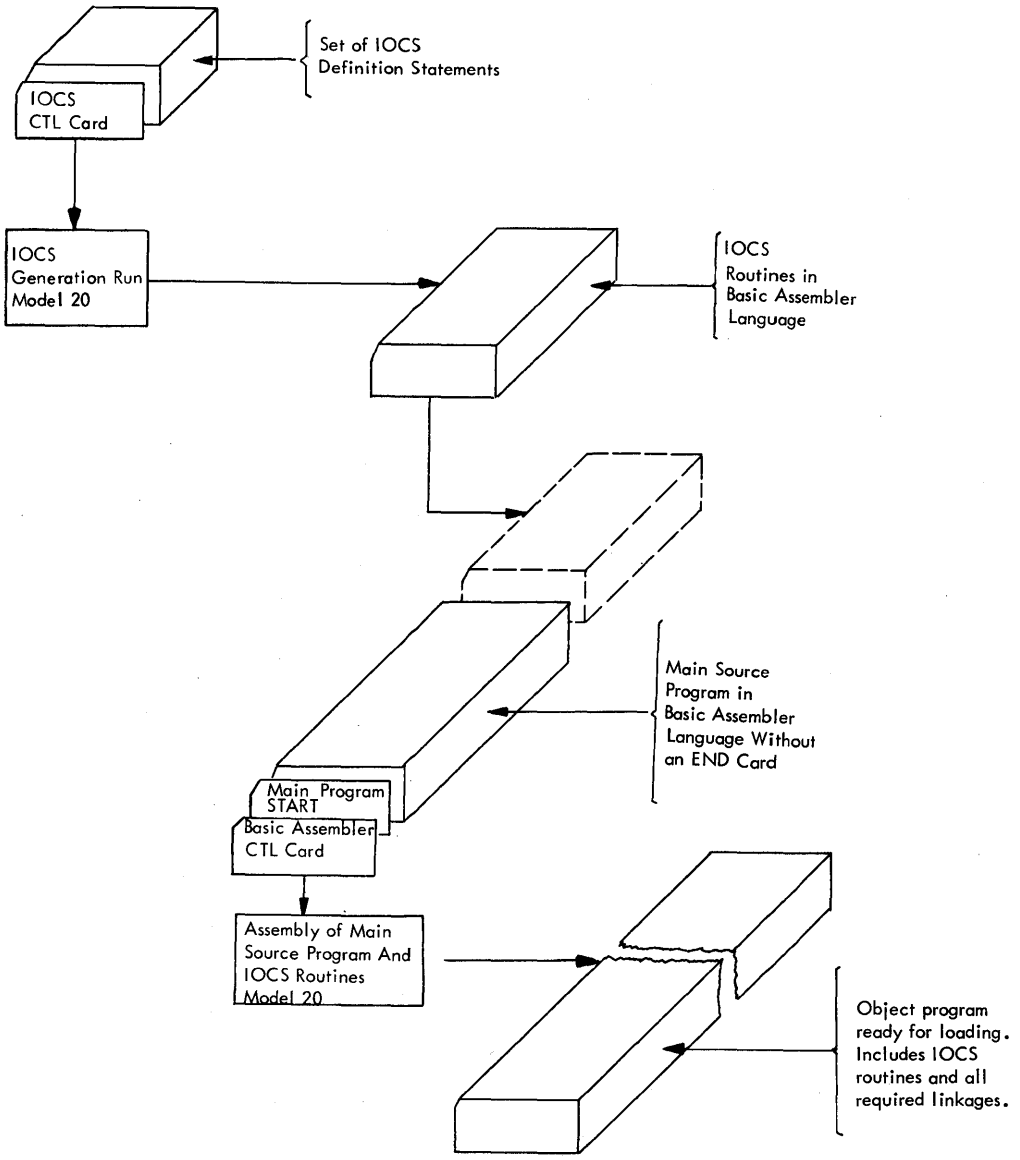

Figure 2. Joint Assembly of IOCS and Main Program

MINIMUM SYSTEM CONFIGURATION

Submodel 2

- an IBM 2020 Central Processing Unit, Model B2 (4096 bytes of main storage);
- one of the following card units:
  IBM 2560 Multi-Function Card Machine, Model A1,
  IBM 2520 Card Read-Punch, Model A1,
  IBM 2501 Card Reader, Model A1 or A2,
  with either an IBM 2520 Card Punch, Model A2 or A3, or an IBM 1442 Card Punch, Model 5;
- one of the following printers:
  IBM 1403 Printer, Model N1, 2, or 7,
  IBM 2203 Printer, Model A1.

Submodel 3

- an IBM 2020 Central Processing Unit, Model B3 (4096 bytes of main storage);
- an IBM 2560 Multi-Function Card Machine, Model A2;
- an IBM 2203 Printer, Model A2.

Submodel 4

- an IBM 2020 Central Processing Unit, Model B4 (4096 bytes of main storage).

Same card unit and printer as for the Submodel 3.

Submodel 5

- an IBM 2020 Central Processing Unit, Model C5 (8192 bytes of main storage).

Same card units and printers as for the Submodel 2.

MAXIMUM SYSTEM CONFIGURATION

Submodel 2

- an IBM 2020 Central Processing Unit, Model D2 (16,384 bytes of main storage);
- 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;

Submodel 3

- an IBM 2020 Central Processing Unit, Model D3 (16,384 bytes of main storage);
- an IBM 2560 Multi-Function Card Machine, Model A2;
- an IBM 2203 Printer, Model A2.

Submodel 4

- an IBM 2020 Central Processing Unit, Model D4 (16,384 bytes of main storage).

Same card unit and printer as for the Submodel 3.

Submodel 5

- an IBM 2020 Central Processing Unit, Model D5 (16,384 bytes of main storage).

Same card units and printers as for the Submodel 2.

DEFINITIONS

This section contains definitions of terms used in this publication.

Record

A record is one unit of information read or punched or printed by one input/output operation. (Model 20 IOCS can process only fixed-length unblocked records.)

File

For purposes of Model 20 IOCS, a file is the total collection of information contained in:

1. All records passed through a given card feed of a punched-card device.

or

2. All records printed as one list during the execution of a given program.

Files can be of two types: simple or combined.

Simple File. A simple file is a file whose records are all either (1) read, or (2) punched, or (3) printed on an output printer during one pass through the system.

Combined File. A combined file is a file consisting of records, some or all of which will be read and/or punched during one pass through the system. (See the section the TYPEFLE Entry.)

Overlap Mode

Overlap Mode is a mode of operation during which execution of input/output operations and processing are performed simultaneously.

Definition Statements

The programmer must use definition statements to describe to the IOCS generator 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 unit used for each file, to define the input/output areas required, etc.

The definition statements are written in symbolic form on the IBM System/360 Assembler Short Coding Form (Form X28-6506) shown in Figure 3. The statements are punched into cards and become the input to the IOCS generator. The generator reads the statements and, based on the information in them, selects the routines required for the user's particular application.

The user must write one DTFSR definition statement for each file to be used by the program. The group of DTFSR statements for the files to be processed by a program must be followed by a DTFEN terminating statement (see DTFEN Terminating Statement).

DTFSR DEFINITION STATEMENT

A DTFSR statement consists of one header entry and a number of detail entries. The number of detail entries depends on the application. The detail entries may be written in any order within a DTFSR statement.

With the exception of the last detail entry card in each DTFSR statement, each header entry card and each detail entry card must have a continuation character (other than blank) punched in column 72. The terminating statement card must not have a continuation character in column 72.

HEADER ENTRY

The header entry consists of an entry in the name field and an entry (DTFSR) in the operation field (columns 32-36). Both of these entries must be provided. The header entry must be the first entry of each definition statement. The name entered in the name field of the header entry must not be assigned to any statement in the main program. This applies to both joint and separate assembly.

The name of the file is specified in the name field beginning in column 25. It may consist of up to four characters. The first character must be alphabetic, the remaining three characters may be alphabetic or numeric. Special characters or embedded blanks must not be used. This name is used in macro instructions to refer to this file.

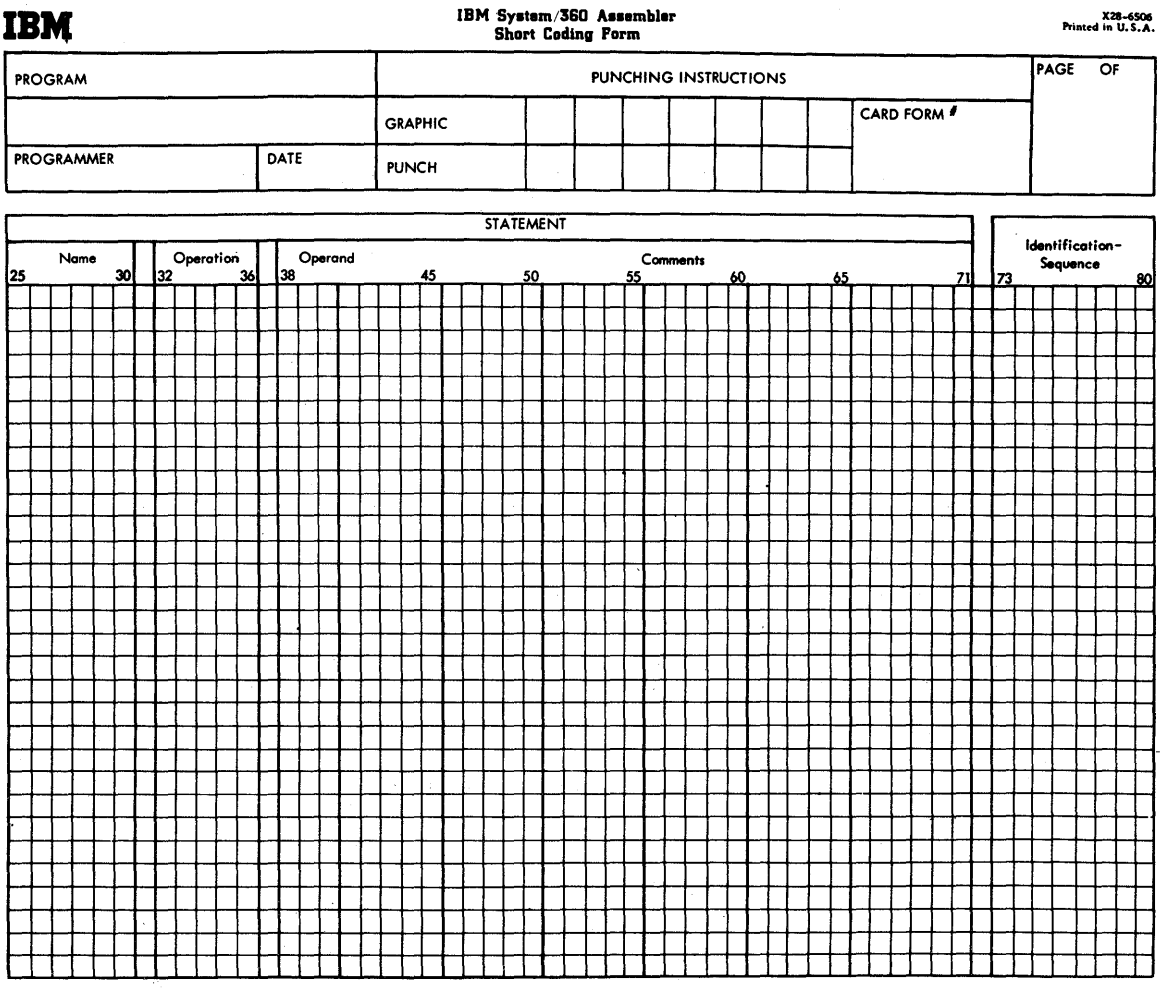

Figure 3. Basic Assembler Short Coding Form

The entry in the operation field must be DTFSR for all files. Figure 4 shows the header entry for a file named INPT.

Name	Operation	Operand
INPT	DTFSR	

Figure 4. DTFSR Header Entry

DETAIL ENTRIES

The detail entries are used to define such information as the device to be used, the mode of processing, etc. Each detail entry is composed of a keyword immediately followed by an equal (=) sign which, in turn, is followed by one specification. A comma must immediately follow the specification of each detail entry except the last entry written for each DTFSR statement.

All detail entries must be written beginning in column 38 on the Short Coding form. Columns 25 through 37 on this form remain blank. A given detail entry may be used only once in each DTFSR statement. Entries that do not apply to a given application must be omitted.

The detail entries can be divided into five categories:

1. Entries applicable to most files.
2. Additional entries for simple files.

3. Additional entries for combined files.

4. Additional entries for card printing.

5. Additional entries that can be used to specify certain checking functions.

DETAIL ENTRIES FOR MOST FILES

The detail entries applicable to most files are:

DEVICE
TYPEFLE
WORKA
PRINTOV
OVERLAP
CONTROL
BINARY
EOFADDR

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

The PRINTOV, OVERLAP, CONTROL, BINARY, and EOFADDR entries must be provided only for certain 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 main source program.

The OVERLAP entry must be provided for all files to be processed in the non-overlap mode.

The CONTROL entry must be provided for a file only if a CNTRL macro instruction referring to that file is used in the main source 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 simple input and combined files.

The DEVICE Entry

This entry identifies the device (or device part) by means of which the file is to be read and/or punched, or printed.

The entry consists of the keyword DEVICE and a specification that identifies the device used by the file. The following specifications are provided:

READ01 file is read by an IBM 2501 Card Reader

PUNCH20 file is punched by an IBM 2520 Card Punch

PUNCH42 file is punched by an IBM 1442 Card Punch, Model 5

PRINTER file is printed by an IBM 2203 Printer, with the standard carriage, or by an IBM 1403 Printer. (Refer to Note below.)

PRINTLF file is printed on the lower carriage of an IBM 2203 with the dual feed carriage. (Refer to Note below.)

PRINTUF file is printed on the upper carriage of an IBM 2203 with the dual feed carriage. (Refer to Note below.)

MFCM1 file is read and/or punched from the primary feed of the IBM 2560 Multi-Function Card Machine.

MFCM2 file is read and/or punched from the secondary feed of the IBM 2560 Multi-Function Card Machine.

CRP20 file is read and/or punched by the IBM 2520 Card Read Punch.

Note: If both feeds of a 2203 Printer 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 in the print file DTFSR statement.

Figure 5 shows an example of the DEVICE detail entry identifying a file read by the 2501 Card Reader.

Name	Operation	Operand
		DEVICE=READ01,

Figure 5. DEVICE Detail Entry

The TYPEFLE Entry

This entry indicates whether the file is an input, an output, or a combined file.

The keyword of this entry is TYPEFLE. The allowable specifications are:

INPUT for a simple input file
OUTPUT for a simple output file
CMBND for a combined file.

Figure 6 shows an example of the TYPEFLE detail entry.

Name	Operation	Operand
		TYPEFLE=INPUT,

• Figure 6. TYPEFLE Detail Entry

The WORKA Entry

This entry indicates that a work area is specified as the second operand of each PUT, GET, or CRDPR macro instruction referring to the file which is always the case in Model 20 IOCS.

The keyword of this entry is WORKA. The specification is YES (Figure 7).

Name	Operation	Operand
		WORKA=YES,

Figure 7. WORKA Detail Entry

The name of the work area used by any macro instruction is specified in that particular instruction and not in the WORKA entry for the file.

Work Area Considerations

The following considerations apply to the use of input/output areas as work areas for files being processed in the overlap and the nonoverlap modes.

OVERLAP MODE. Input and/or output areas of files being processed in the overlap mode may not be used as work areas. During processing, a given record is processed in the work area while other records are simultaneously being read into an input area or being punched or printed from an output area.

NONOVERLAP MODE. For combined files, only the punch areas may also be used as work areas.

For simple files, either the input or the output area may be used as a work area.

Card print areas may never be used as work areas.

The PRINTOV Entry

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

The keyword of this entry is PRINTOV. The specification is YES (Figure 8).

Name	Operation	Operand
		PRINTOV=YES,

Figure 8. PRINTOV Detail Entry

The OVERLAP Entry

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

The keyword of this entry is OVERLAP. The specification is NO.

Printer files are always processed in the overlap mode. Therefore, an OVERLAP=NO entry is not permitted for these files.

Figure 9 shows an OVERLAP entry.

Name	Operation	Operand
		OVERLAP=NO,

Figure 9. OVERLAP Detail Entry

The CONTROL Entry

This entry must be provided if a CNTRL macro instruction referring to this file is used in the main source program.

The keyword of this entry is CONTROL. The specification is YES (Figure 10).

Name	Operation	Operand
		CONTROL=YES,

Figure 10. CONTROL Detail Entry

The BINARY Entry

This entry indicates that the cards of an input file are to be read in the column binary mode by a Model 20 equipped with the Read Column Binary Special Feature. The entry may be provided for both simple and combined input files.

The keyword of this entry is BINARY. The specifications are:

YES for simple files
INPUT for combined files.

Figure 11 shows an example of a BINARY detail entry.

Name	Operation	Operand
		BINARY=INPUT,

Figure 11. BINARY Detail Entry

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 for this file must be specified to contain a number of bytes that is equal to twice the number of card columns to be read.

When the entry BINARY is used, the following detail entries must not be used:

SEQNCE
PFORMTn
RFORMTn

The EOFADDR Entry

This entry is used to specify the name of the user's end-of-file routine in the main program. The entry is mandatory for input and combined files.

The keyword of this entry is EOFADDR. The specification is the name of the user's end-of-file routine. An example of this type of entry is shown in Figure 12.

Name	Operation	Operand
		EOFADDR=END,

Figure 12. EOFADDR Detail Entry

All card files, on which simple input or combined input/output operations will be performed, must be terminated by four end-of-file cards containing the entry /* in the first two columns. A branch occurs to the user's end-of-file routine after the first end-of-file card of the corresponding file has been read.

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

The IOAREA1 Entry

This entry defines the name of the input/output area to be used by a simple file.

The keyword of this entry is IOAREA1. The specification is the name of the input or output area used by the file. This name must be the symbol used by the programmer in defining the area in his main program.

Figure 13 shows an example of an IOAREA1 entry.

Name	Operation	Operand
		IOAREA1=INP1,

Figure 13. IOAREA1 Detail Entry

Printer Files

The IOAREA1 entry must not be provided for a file printed by the standard carriage because IOCS uses the print buffer in main storage (first 144 positions). This area 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 carriage preceding the area for the upper-feed carriage (Figure 14).

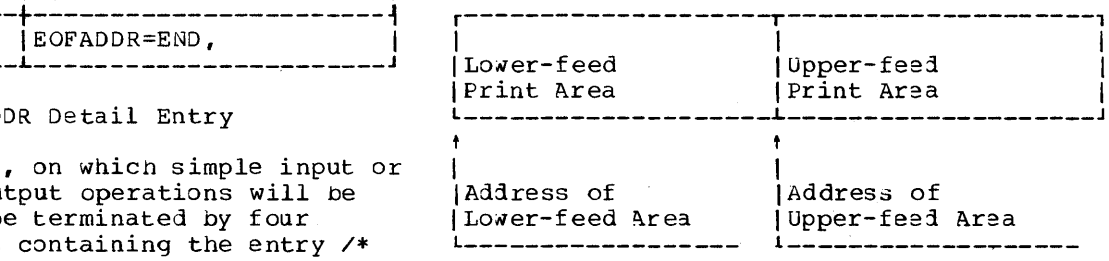

Figure 14. Print-Area Format for Dual Feed Carriage

The IOAREA2 Entry

This entry can be used to define the name of a second input area when the IBM 2501 Card Reader is used in the overlap mode. This 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 if, for example, 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 input or output device or when the 2501, Model A2, is used in nonoverlap mode.

The keyword of this entry is IOAREA2. The specification is the name of the second read area as defined in the main program. This area must be the same length as the first read area whose name is defined in the IOAREA1 entry.

Figure 15 shows an example of an IOAREA2 entry in conjunction with an IOAREA1 entry.

```
|Name  |Operation |Operand                |
|------|----------|-----------------------|
|      |          |IOAREA1=ARA1,          |
|      |          |IOAREA2=ARA2,          |
```

Figure 15. IOAREA1-IOAREA2 Detail Entry Combination

The_BLKSIZE_Entry

This entry specifies the length of the input or output area(s) required by the simple file. The length specified in this entry applies to the area(s) reserved in the main program and referred to in the corresponding IOAREA1 entry or IOAREA1-IOAREA2 combination. In the case of a printer file -- even if an IOAREA1 entry is not provided -- the BLKSIZE entry must be given.

The keyword of this entry is BLKSIZE. The specification is the length of the input/output area in bytes. In the case of an IOAREA1-IOAREA2 combination, the specified length is the length of the individual fields.

Figure 16 shows a set of area definition entries and a BLKSIZE entry for a card file requiring two read areas of 65 characters each.

Maximum record lengths acceptable to the IOCS are as follows:

For cards: 80 bytes (160 bytes for column binary mode).
For printers: 120, 132, or 144 characters, depending on the number of print positions available. If a 2203 printer with a dual carriage feature is used, the total length of areas specified for both feeds must be equal to or less than 144 bytes.

The minimum record lengths acceptable to the IOCS is two bytes (four bytes for column binary mode).

```
|Name  |Operation |Operand                |
|------|----------|-----------------------|
|      |          |IOAREA1=INP1,          |
|      |          |IOAREA2=INP2,          |
|      |          |BLKSIZE=65,            |
```

Figure 16. IOAREA1-IOAREA2 Detail Entries with BLKSIZE Entry

ADDITIONAL DETAIL ENTRIES FOR COMBINED FILES

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

INAREA
INBLKSZ
OUAREA
OUBLKSZ

The_INAREA_Entry

This entry specifies the name of the input area to be used by the combined file.

The keyword of this entry is INAREA. The specification is the name of the input area used by the file. This name must be the symbol used by the programmer in defining the area in his main program (Figure 17).

```
|Name  |Operation |Operand                |
|------|----------|-----------------------|
|      |          |INAREA=INPC,           |
```

Figure 17. INAREA Detail Entry

The_INBLKSZ_Entry

This entry specifies the length of the input area required by the combined file. The length specified in this entry is that of the area reserved in the main program and referred to in the INAREA entry.

The keyword of this entry is INBLKSZ. The specification is the length of the input area in bytes.

Figure 18 shows an INAREA entry and its associated INBLKSZ entry specifying an input area named INPC that is 65 characters in length.

```
┌──────┬─────────┬──────────────────────┐
|Name  |Operation|Operand               |
├──────┼─────────┼──────────────────────┤
|      |         |INAREA=INPC,          |
|      |         |INBLKSZ=65,           |
└──────┴─────────┴──────────────────────┘
```

Figure 18. INAREA Detail Entry with INBLKSZ Entry

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

The OUAREA Entry

This entry specifies the name of the output area to be used by the combined file.

The keyword of the entry is OUAREA. The specification is the name of the output area used by the file. This name must be the symbol used by the programmer in defining the area in his main program (Figure 19).

```
┌──────┬─────────┬──────────────────────┐
|Name  |Operation|Operand               |
├──────┼─────────┼──────────────────────┤
|      |         |OUAREA=OUPC,          |
└──────┴─────────┴──────────────────────┘
```

Figure 19. OUAREA Detail Entry

The OUBLKSZ Entry

This entry specifies the length of the output area required by the combined file. The length specified in this entry is that of the area reserved in the main program and referred to in the OUAREA entry.

The keyword of this entry is OUBLKSZ. The specification is the length of the output area in bytes.

Figure 20 shows an OUAREA entry and its associated OUBLKSZ entry specifying an output area named OUPC that is 65 characters in length.

```
┌──────┬─────────┬──────────────────────┐
|Name  |Operation|Operand               |
├──────┼─────────┼──────────────────────┤
|      |         |OUAREA=OUPC,          |
|      |         |OUBLKSZ=65,           |
└──────┴─────────┴──────────────────────┘
```

Figure 20. OUAREA Detail Entry with OUBLKSZ Entry

The maximum record length permitted is 80 bytes. The minimum record length permitted is one byte.

ADDITIONAL DETAIL ENTRIES FOR CARD PRINTING

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

CRDPRA
CRDPRLn

The entries described in this section can apply only to a simple or combined file of the 2560 MFCM.

The programmer must make the card print detail entries only in one DTFSR statement of any one program. Thus, even if he intends to perform card printing on cards fed by both the primary and the secondary feed of the MFCM, he must make the required detail entries in only one of his DTFSR statements.

It is immaterial in which DTFSR statement the entries appear, since card printing is a function of the CRDPR macro instruction (which does not refer to a file). For details, see the section describing that macro instruction.

The CRDPRA Entry

This 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 keyword of this entry is CRDPRA. The specification is the name of the area.

Figure 21 shows a CRDPRA detail entry.

```
┌──────┬─────────┬──────────────────────┐
|Name  |Operation|Operand               |
├──────┼─────────┼──────────────────────┤
|      |         |CRDPRA=CPAR,          |
└──────┴─────────┴──────────────────────┘
```

Figure 21. CRDPRA Detail Entry

Card Print Areas

The areas containing the data to be printed by the MFCM print heads must be defined as a contiguous area in main storage. The whole card print area may be located anywhere in main storage, but the areas for the individual print heads used must be defined within the whole area in ascending order of the print head numbers (Figure 18).

As shown in Figure 22, the beginning of successive individual print areas must be

separated by exactly 64 bytes of main storage. The length of each print area is defined by the corresponding CRDPRLn entry (see description of that entry below).

During each card print operation, each print head prints the number of positions specified for the longest print area. Therefore, if CRDPRLn entries specify card print areas of 20, 40, and 50 bytes, 50 bytes of information will be printed from each area.

The programmer may use any unused portion of the individual print areas. Thus, if the maximum specified card print area length is 50 bytes, the last 14 bytes in each area may be used for processing.

In each of the three defined print areas shown in Figure 22, an area corresponding to the maximum CRDPRLn entry (length A) will be printed. Only the shaded areas may be used by the programmer for purposes other than printing.

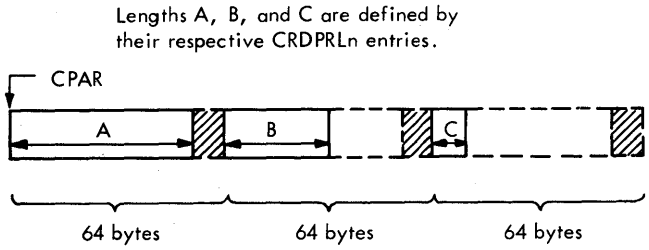

Figure 22. MFCM Card Print Areas

The CRDPRLn Entry

This entry defines a print head number and the length of the area in main storage where the data to be printed by this head is stored. A CRDPRLn entry must be included for each print head to be used.

The keyword of this entry is CRDPRLn, where n is the number of the print head used. The specification is the length (in bytes) of the print area containing the data to be printed by this print head.

Figure 23 shows the CRDPRA and CRDPRLn entries referring to the print areas shown in Figure 22. The entries in Figure 23 assume that area A is 50 bytes, area B is 40 bytes, and area C is 20 bytes in length.

```
|Name  |Operation |Operand                  |
|------|----------|-------------------------|
|      |          |CRDPRA=CPAR,             |
|      |          |CRDPRL1=50,              |
|      |          |CRDPRL2=40,              |
|      |          |CRDPRL5=20,              |
```

Figure 23. CRDPRA Detail Entry with CRDPRLn Entries

Refer to the example in Figures 22 and 23. 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 must contain blanks if no printing is 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.

ADDITIONAL DETAIL ENTRIES FOR CHECKING FUNCTIONS

The following additional detail entries are available for card processing to enable the user to specify certain checking functions:

SEQNCE
SEQXIT
RFORMTn
RFXIT
PFORMTn
PFXIT

The SEQNCE Entry

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 a sequence error is found, the program branches to a user-written routine. (The branch address is specified by the SEQXIT detail entry.) Only one SEQNCE entry is permitted for each file.

The sequence check is made by means of a logical compare operation. If the input data is to be read in the column binary mode, a SEQNCE entry may not be made for this file.

The keyword of this entry is SEQNCE. The specification is:

xxyy

where xx and yy are the numbers of the first and last card columns, respectively, containing the card field to be checked. For card columns 1-9, the leading zero must be punched. The card field to be checked must not be longer than 16 columns.

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 was read by the 2560 MFCM or the 2520 Card Read Punch, it is positioned at the pre-punch station. The next record will be read by the next GET or EOM (Enter Overlap Mode) macro instruction. (Refer to the appropriate sections under The IOCS Macro Instructions for descriptions of GET and EOM.) This record will be compared with the record that preceded the card that led to the error condition.

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-specified SEQXIT routine causes the IOCS to change the processing mode (from overlap to nonoverlap) for the GET macro instruction that detected the error.

This change of 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.

Figure 24 shows an example of a SEQNCE entry. This entry specifies that the contents of card columns 9-15 are to be used for sequence checking.

Name	Operation	Operand
		SEQNCE=0915,

Figure 24. SEQNCE Detail Entry

The SEQXIT Entry

This entry must be used in conjunction with the SEQNCE detail entry to furnish the IOCS with the name of the user's routine to which control is to be transferred if a sequence error occurs.

The keyword of this entry is SEQXIT. The specification is the name of the user's routine. Figure 25 shows an example of a SEQXIT entry.

Name	Operation	Operand
		SEQXIT=NAME,

Figure 25. SEQXIT Detail Entry

The RFORMTn Entry

This entry enables the programmer to check that a specified input card field or fields contain(s) numeric characters or all blanks. If the field is found not to contain numeric characters or blanks, the program branches to a user-written routine (if the branch address is specified by the RFXIT detail entry) or halts. If the input data is to be read in the column binary mode, an RFORMTn entry may 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 provides the programmer with the possibility of writing a maximum of ten different RFORMTn entries and thus having a maximum of ten fields checked. The specification for this entry is:

xxyyz

where xx and yy are the numbers of the first and last card columns, respectively, containing the field to be checked (for column 1-9, the leading zero must be punched), and z is 0 (check for blanks) or 1 (check for numeric characters).

Card fields that are to be checked for numeric contents must not be longer than 16 columns.

When a field is tested for all blanks, control is transferred to a user-written

routine when the test fails (i.e., the field is not blank).

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

bbb...nnnnn

where b = blank
n = numeric character.

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

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-specified RFXIT routine causes the IOCS to change the processing mode (from overlap to nonoverlap) for the GET macro instruction that detected the error.

This change of 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.

Figure 26 shows an example of an RFORMTn entry specifying that columns 73-80 of eachcard in the file are to be checked for the presence of blanks.

Name	Operation	Operand
		RFORM10=73800,

Figure 26. RFORMTn Detail Entry

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.

The programmer may use a maximum of ten RFORMTn entries to perform checks on different card fields. However, he may use only one RFXIT entry for each file.

The RFXIT Entry

This entry is used in conjunction with the RFORMTn detail entry to specify the name of the user's routine to which control is to be transferred if the test made on the field specified by means of an RFORMTn entry is negative (i.e., if a field is found to contain characters other than those specified).

If this entry is omitted and the test is negative, the machine halts. This enables the operator to replace the card that led to the error condition.

Figure 27 shows an example of an RFXIT entry.

Name	Operation	Operand
		RFXIT=FERR,

Figure 27. RFXIT Detail Entry

The PFORMTn Entry

The PFORMTn entry enables the programmer to check cards of a combined file that are not read into the work area by GET macro instructions to ensure that a specified card field (or fields) to be punched contains blanks. 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 (if the branch address is furnished by the PFXIT detail entry) or a system halt occurs.

If a PUT macro instruction is given that refers to a combined file and the program proceeds to the PFORMTn error routine (user's exit), a subsequent GET macro instruction will place the contents of the card that caused the PFORMT error into the work area. If this GET macro instruction is in nonoverlap mode, it is possible to punch this card by means of an additional PUT macro instruction.

The keyword of this entry is PFORMTn, where n is any number from 0 to 9. The n position provides the programmer with the possibility of writing a maximum of ten different PFORMTn entries and thus having a maximum of ten fields checked. The specification for this entry is:

xxyy

where xx and yy are the numbers of the first and last card columns, respectively, containing the field to be checked (for columns 1-9, the leading zero must be punched). The input area must be large enough to permit IOCS to read the information in these columns into storage.

Figure 28 shows an example of a PFORMTn entry specifying that a card field

comprising the columns 1-12 is to be checked for blanks prior to punching.

Name	Operation	Operand
		PFORMT1=0112,

Figure 28. PFORMTn Detail Entry

The programmer may use a maximum of ten PFORMTn entries to perform checks on different card fields. But he may use only one PFXIT entry for each file.

The PFXIT Entry

This entry is used in conjunction with the PFORMTn entry to specify the name of the user's routine to which control is to be transferred if the test made on the field specified by means of the PFORMTn detail entry indicates an error condition.

Figure 29 shows an example of a PFXIT entry.

Name	Operation	Operand
		PFXIT=NAME,

Figure 29. PFXIT Detail Entry

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

If this entry is omitted and the test shows an error condition, the machine halts before punching is initiated. This enables the operator to replace the card that led to the error condition. This card is positioned at the pre-punch station.

DTFEN TERMINATING STATEMENT

This terminating statement is used to indicate the end of the entire set of definition statements for a given application. Only one terminating statement must be provided. It must immediately follow the last DTFSR statement.

The terminating statement consists of the characters DTFEN in the operation field. Figure 30 shows an example of a DTFEN statement.

Name	Operation	Operand
	DTFEN	

Figure 30. DTFEN Terminating Statement

Operation	Operand: Keyword	Operand: Allowable Specifications	Applies to: 2560	2520 Read Punch	2520 Punch	1442 Model 5 Punch	2501	2203	1403	Remarks
DTFSR			x	x	x	x	x	x	x	
	DEVICE	MFCM1	x							
		MFCM2	x							
		CRP20		x						
		PUNCH20			x					
		PUNCH42				x				
		READ01					x			
		PRINTER						x	x	
		PRINTLF						x		for 2203 with dual-feed-carriage
		PRINTUF						x		
	TYPEFLE	INPUT	x	x			x			
		OUTPUT	x	x	x	x		x	x	
		CMBND	x	x						

Figure 31. Definition Statement Summary, Part 1 of 3

DEFINITION STATEMENT SUMMARY

Figure 31 is a summary of the various entries available for the Model 20 IOCS definition statements. The chart shows the allowable entries that can be made in the various fields of the Basic Assembler coding form. In addition, it shows for which input/output unit(s) a specific entry may be required. For example, if a file is being read and/or punched by the 2560 MFCM, the X's in the 2560 column indicate to the programmer which entries he may have to provide.

Operation	Operand: Keyword	Operand: Allowable Specifications	Applies to: 2560	2520 Read Punch	2520 Punch	1442 Model 5 Punch	2501	2203	1403	Remarks
	PRINTOV	YES						x	x	required if PRTOV is macro given for the file
	WORKA	YES	x	x	x	x	x	x	X	mandatory for all files
	OVERLAP	NO	x	x	x	x	x			if omitted, file processed in overlap mode
	CONTROL	YES	x	x	x			x	x	required if CNTRL macro given for file
	EOFADDR	name of user end-of-file routine	x*	x*			x			*mandatory for input and/or combined files
	IOAREA1	name of the user-defined area	x	x	x	x	x	x*		*entry required for 2203 only when dual-feed-carriage is used
	IOAREA2	name of the user-defined area					x			can be used if 2501, Model A2, is used in overlap mode
	BLKSIZE	length of simple file input/output area in bytes	x	x	x	x	x	x	x	specifies length of area specified by IOREA1 - IOAREA2 entries
	BINARY	YES	x*	x*			x			*mandatory for simple files
		INPUT	x	x						only for 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
	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 31. Definition Statement Summary, Part 2 of 3

Operation	Operand: Keyword	Operand: Allowable Specifications	Applies to: 2560	2520 Read Punch	2520 Punch	1442 Model 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 print head number
	SEQNCE	xxyy	x	x			x			indicates sequence check of input cards desired from cols xx to yy
	SEQXIT	name of user-routine used when SEQNCE test fails	x	x			x			must be specified when SEQNCE is specified
	RFORMTn	xxyyz	x	x			x			indicates that a check for numerics or blanks is desired from cols xx to yy in input cards
	RFXIT	name of user-routine used when RFORMTn test fails	x	x			x			
	PFORMTn	xxyy	x	x						indicates that a check for blanks is to be made of field from cols xx to yy prior to punching**
	PFXIT	name of user-routine used when PFORMTn test fails	x	x						

** Applies to combined files only.

Figure 31. Definition Statement Summary, Part 3 of 3

THE IOCS MACRO INSTRUCTIONS

This section describes the format, function, and use of the IOCS macro instructions that are used by the programmer to control input/output operations. These symbolic instructions are used in the main program to cause the Basic Assembler to insert the object-program linkages to the IOCS routines previously defined by means of the IOCS definition statement.

The macro instructions enable the user to design his program free from most detailed considerations concerning input and output.

The filling (i.e., reading into) and emptying (i.e., punching out and printing out) of the input and output areas will be handled automatically by the IOCS. All files will be handled in accordance with the information given the IOCS in the definition statements. Source programs using the IOCS must not contain any Basic Assembler input/output instructions (i.e., XIO, TIOB, CIO and SPSW).

Relative addressing of a work area is the only relative addressing permitted in the operand of a macro instruction.

Like the IOCS definition statements, the macro instructions are written on the System/360 Assembler Short Coding Form. IOCS macro instructions must be written according to the rules specified for the Basic Assembler language with the exception that they may have up to four operands. The mnemonic of the macro instruction is written in the operation field and up to four operands are written in the operand field. Each macro instruction may have a name, which is written in the name field and may consist of up to four characters.

The IOCS macro instructions are:

GET
PUT
OPEN
CLOSE
PRTOV (Check Printer Overflow)
CNTRL (Control)
LOM (Leave Overlap Mode)
EOM (Enter Overlap Mode)
CRDPR (Card Print)
WAITC (Wait Card)

Note: The Basic Assembler program does not perform a specific test of IOCS macro instructions. Therefore, a number of errors in macro instructions are not detected.

The GET Macro Instruction

The GET macro instruction makes the next record of the specified file available in the user-defined work area and, if necessary, transfers control to the appropriate user's routine specified by the SEQXIT, RFXIT, or EOFADDR detail entries.

Each GET macro instruction may have a name in the name field and must contain:

1. GET in the operation field.
2. A first operand, followed by a comma, specifying the file from which the record is to be read.
3. A second operand specifying the work area by a name or a relocatable expression.

   Note: A work area need not be associated with any particular file. Each macro instruction specifies the work area to be used. Each work area used by a file must be at least as long as the input/output area for that file.

For simple files working in the non-overlap mode, the programmer may use the read or punch area as a work area. For combined files working in the nonoverlap mode, the programmer may use only the punch area as a work area.

Figure 32 shows a GET macro instruction that moves the next record from a file called FILE into a work area defined by the user's DS statement called WORK.

Name	Operation	Operand
NAME	GET	FILE,WORK

Figure 32. GET Macro Instruction

PROCESSING IN NONOVERLAP MODE. When a file is processed in the nonoverlap mode, a GET that refers to this file:

1. Initiates a read operation for the next (or first) card of the file. (The data contained in the card is read into the input area).
2. Moves the data read from the input area into the work area after the read operation is complete.
3. Transfers control immediately to either a user's routine (SEQXIT, RFXIT, or EOFADDR) or to the main program.

The read operation for the following card of this file is not initiated until another GET for the same file is executed.

PROCESSING IN OVERLAP MODE. When a file is processed in overlap mode, the OPEN macro instruction for this file initiates a read operation. This causes the contents of the first card of the file to be read into the input area. Therefore, the first and any subsequent GET that refers to this file causes:

1. The record that is contained in the input area to be moved into the work area.
2. A read operation to be initiated for the card following the card whose contents have just been moved into the work area.
3. Immediate transfer of control to either a user's routine (SEQXIT, RFXIT, or EOFADDR) or to the main program.

An overlapping effect is achieved because a read operation for the following card is initiated immediately after the desired record has been moved from the input area into the work area. Processing is performed while the following card is moved through the read station and the contents of this card are read into the input area.

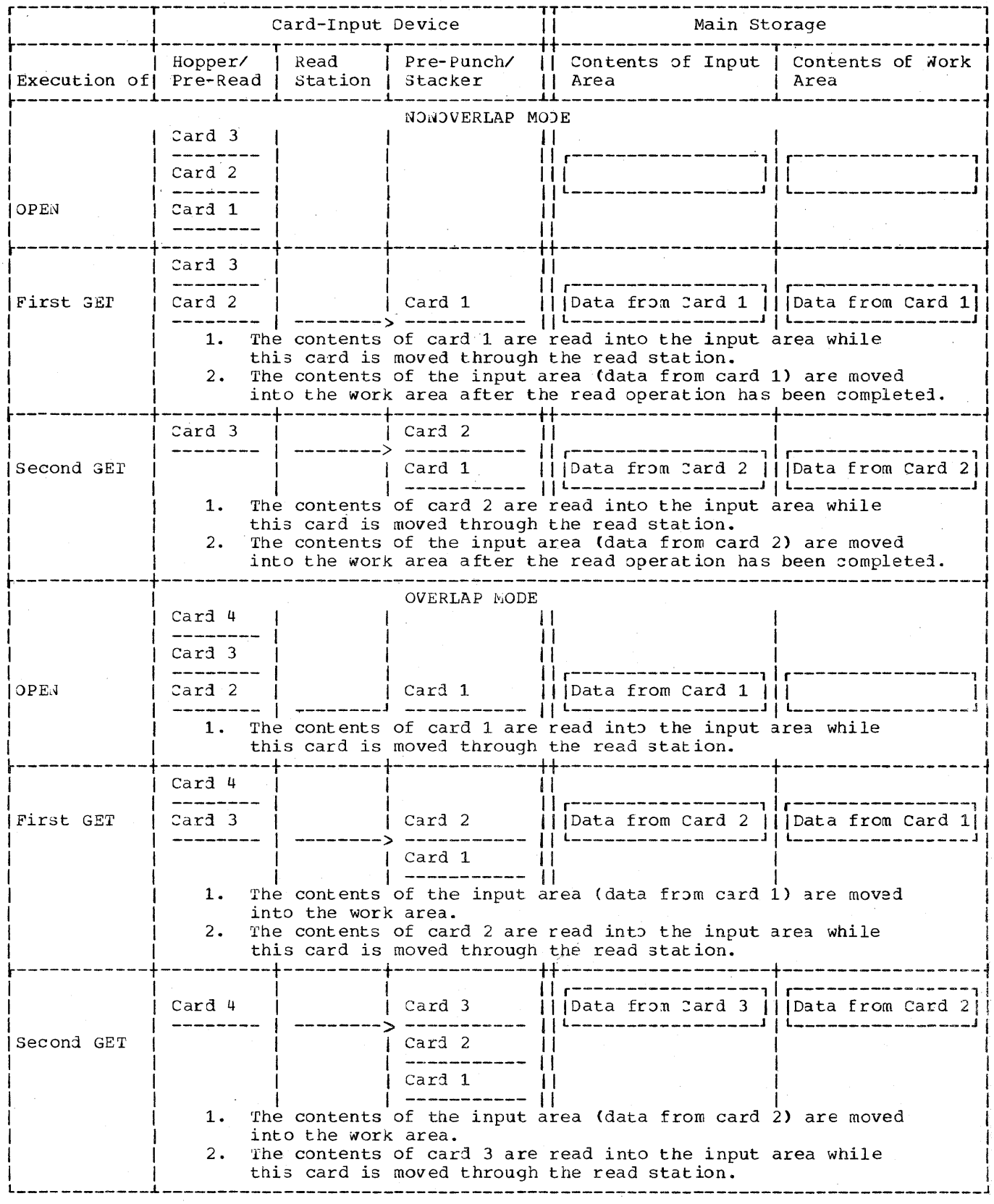

Figure 33. Card Movement and Data Flow when a GET Macro Instruction is Executed

Figure 33 illustrates card movement and data flow from the card input device to main storage for the processing of a file in both the nonoverlap and the overlap modes. It also illustrates the position of a card in the card input device at the time the GET macro instruction for this card is executed.

For an explanation of the relationship between the GET macro instruction and the CRDPR macro instruction, refer to the description of the CRDPR macro instruction.

The PUT Macro Instruction

The PUT macro instruction makes a record from the work area available for punching or printing and, if necessary, transfers control to the user's routine specified by the PFXIT detail entry.

Each PUT macro instruction may have a name in the name field and must contain:

1. PUT in the operation field.
2. A first operand, followed by a comma, specifying the file to which the record is to be made available.
3. A second operand specifying the work area by a name or a relocatable expression (when a simple file is processed in the nonoverlap mode, this can be the name of the punch area).

   Note: A work area need not be associated with any particular file. Each macro specifies the work area. Each work area used by a file must be at least as long as the input/output area for that file.

When processing is being performed in the nonoverlap mode (card punching only), the PUT macro instruction:

1. Moves a record from the work area to the output area.
2. Initiates the punch operation (and the next read operation in the case of a combined file).
3. Transfers control to the main program when the punch operation is complete.

When processing is being performed in the overlap mode, the PUT macro instruction:

1. Moves a record from the work area to the output area.
2. Initiates the punch or print operation (and the next read operation in the case of a combined file).
3. Immediately transfers control to the main program.

Whenever an output data record is transferred to an output device by a PUT macro instruction, the data remains in the work area until it is either cleared or replaced by other data. The IOCS does not clear the work area. Therefore, if the user plans to build another record having data that does not use every position of the work area, he must clear this area before he builds the record. If this is not done, the new record and all of the following records may contain interspersed characters from the preceding record.

Figure 34 shows a PUT macro instruction that makes a record available to a file called DETL from a work area whose beginning address is MSTR+150.

Name	Operation	Operand
NAME	PUT	DETL,MSTR+150

Figure 34. PUT Macro Instruction

For an explanation of the relationship between the PUT macro instruction and the CRDPR macro instruction, refer to the description of the CRDPR macro instruction.

PROGRAMMING CONSIDERATIONS -- COMBINED FILES. If a combined file is being processed by the following sequence of instructions:

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

the following rules apply:

Nonoverlap. The PUT F1 macro instruction causes punching into the card made available by the GET F1 macro instruction.

Overlap. The PUT F1 Macro instruction causes punching into the card that follows the one made available by the GET F1 macro instruction, because the card made available by the GET F1 macro instruction is already past the punch station when the PUT F1 macro instruction is given.

The OPEN Macro Instruction

This macro instruction ensures that all the information required to handle a file has been provided.

For an input file to be processed in overlap mode, the OPEN macro instruction causes the first card to be read. Its contents is then available to be moved from the input area into the work area when the first GET for the file is encountered. For an input file to be processed in non-overlap mode, the function of the OPEN macro instruction depends on the type of the file:

1. For a simple file, the OPEN macro instruction makes the file available for processing.

2. For a combined file, the OPEN macro instruction causes the first card to be read while this card is moved to the pre-punch station.

The OPEN macro instruction must be issued before any other macro instruction regarding the same file is given. The programmer must write a separate OPEN macro instruction for each file.

Each OPEN macro instruction may have a name in the name field and must contain:

1. OPEN in the operation field, and

2. one and only one operand indicating the file to which this OPEN macro instruction applies.

Figure 35 shows an OPEN macro instruction labeled NAME for a file called PAY.

Name	Operation	Operand
NAME	OPEN	PAY

Figure 35. OPEN Macro Instruction

The CLOSE Macro Instruction

This macro instruction ensures proper handling of the file after all records have been processed.

Specifically, the CLOSE macro instruction ensures:

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

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

A CLOSE macro instruction must be given for each file after the processing of all records of the file has been completed. For input files, the CLOSE macro instruction is normally given in the user's end-of-file routine. A file may not be reopened by an OPEN macro instruction after it has been closed.

Each CLOSE macro instruction may have a name in the name field and must contain:

1. CLOSE in the operation field.

2. One and only one operand indicating the file to which this CLOSE macro instruction applies.

Figure 36 shows a CLOSE macro instruction for a file called DETL.

Name	Operation	Operand
NAME	CLOSE	DETL

Figure 36. CLOSE Macro Instruction

The PRTOV Macro Instruction

The PRTOV macro instruction can be used by the programmer to check for printer-overflow conditions.

The PRTOV macro instruction consists of PRTOV in the operation field, followed by two or three operands of the form

```
SYMB,m
```

or

```
SYMB,m,ROUT
```

where SYMB is the symbolic name of the printer file,

m is 9 or 12 to specify which carriage-tape channel indicator is to be checked, and

ROUT is the symbolic name of the user routine to be executed when an overflow occurs.

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

1. Before execution of the last (preceding) PUT macro instruction referring to a printer with the standard carriage.

2. Before the execution of the last PUT macro instruction referring to a printer with the dual feed carriage when only the lower feed is used.

3. Before the execution of the next-to-last PUT macro instruction referring to a printer with the dual feed carriage when both feeds are used.

However, if a skip has been performed or more than one line has been spaced after the last PUT macro instruction (or after the next-to-last PUT macro instruction if the associated DTFSR statements contain PRINTUF and PRINTLF) any punch that may be sensed in channel 9 or channel 12 is lost and cannot be determined by a PRTOV macro instruction.

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

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

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

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

Figure 37 shows a PRTOV macro instruction referring to a file named PRNT and specifying that the program branches to the user routine named OVFL when a channel-9 punch is sensed in the carriage tape.

Name	Operation	Operand
NAME	PRTOV	PRNT,9,OVFL

Figure 37. PRTOV Macro Instruction

The CNTRL Macro Instruction

The CNTRL macro instruction can be used by the programmer to cause printer spacing, printer skipping, and card stacking to be performed in other than the normal manner. There are four different formats available for the CNTRL macro instruction: one for printer spacing, one for printer skipping, and two for card stacking. Each of the four types is described separately in the following sections.

CNTRL Macro Instruction for Printer Spacing

This macro instruction can be used to specify immediate spacing and/or delayed spacing (space after print) of the printer carriage.

The CNTRL macro instruction for spacing consists of CNTRL in the operation field followed by an operand of the form:

SYMB,SP,m,n

where SYMB is the name of the printer file

SP specifies spacing

m is the number of lines the form is to be spaced immediately (m = 0, 1, 2, or 3), and

n is the number of lines the form is to be spaced after printing (n = 0, 1, 2, or 3).

The programmer may omit either m or n. A name may be assigned to each CNTRL macro instruction.

If a CNTRL macro instruction specifying delayed spacing is not given before the next PUT for the printer file, the printer carriage advances one space after the print operation is completed. When two CNTRL macro instructions specifying delayed spacing are given before the next PUT for the printer file, the spacing specified in the second CNTRL macro instruction is effective (i.e., the second CNTRL macro instruction overrides the first). If delayed spacing and skipping are both specified before a PUT for the printer file, only the last specified operation will be performed.

Because of timing considerations, delayed spacing should be used whenever possible in order to increase system throughput.

Figure 38 shows a CNTRL macro instruction referring to a printer file named LIST and specifying that the form is to be spaced three lines immediately and one line after printing.

Name	Operation	Operand
NAME	CNTRL	LIST,SP,3,1

Figure 38. CNTRL Macro Instruction for Printer Spacing (Immediate and Delayed)

Figure 39 shows a CNTRL macro instruction referring to a printer file named LIST and specifying that the form is to be spaced one line immediately.

Name	Operation	Operand
NAME	CNTRL	LIST,SP,1

Figure 39. CNTRL Macro Instruction for Printer Spacing (Immediate)

Similarly, Figure 40 shows a CNTRL macro instruction specifying that 2 lines are to be spaced after printing only. Note that the omission of the operand m is indicated by a comma.

Name	Operation	Operand
NAME	CNTRL	LIST,SP,,2

Figure 40. CNTRL Macro Instruction for Printer Spacing (Delayed)

CNTRL Macro Instruction for Printer Skipping

This macro instruction can be used to specify the channel of the carriage control tape to which the carriage tape is to be skipped immediately and/or after the printing of a line.

The CNTRL macro instruction for printer skipping consists of CNTRL in the operation field followed by an operand of the form:

SYMB,SK,m,n

where SYMB is the name of the printer file,

SK specifies skipping,

m is the number of the tape channel to which the carriage is to be skipped immediately (m = 1, 2,...,12), and

n is the number of the tape channel to which the carriage is to be skipped after printing (n = 1, 2,...,12).

The programmer may omit either m or n. A name may be assigned to each CNTRL macro instruction.

When two CNTRL macro instructions specifying delayed skipping 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 macro instruction overrides the first).

Because of timing considerations, delayed skipping should be used whenever possible in order to increase throughput.

Figure 41 shows a CNTRL macro instruction referring to a printer file called REPT and specifying that control is to be transferred immediately to channel 11 of the printer tape and to channel 12 after printing.

Name	Operation	Operand
NAME	CNTRL	REPT,SK,11,12

Figure 41. CNTRL Macro Instruction for Printer Skipping (Immediate and Delayed)

Figure 42 shows an example of a CNTRL macro instruction referring to a file named UPDT and specifying that carriage control is to be transferred to channel 4 immediately.

Name	Operation	Operand
NAME	CNTRL	UPDT,SK,4

Figure 42. CNTRL Macro Instruction for Printer Skipping (Immediate)

Figure 43 shows a CNTRL macro instruction referring to a file called NEW and specifying that carriage control is to be transferred to channel 7 after printing. Note that the omission of the operand m is indicated by a comma.

Name	Operation	Operand
NAME	CNTRL	NEW,SK,,7

Figure 43. CNTRL Macro Instruction for Printer Skipping (Delayed)

CNTRL Macro Instruction for Stacking

This macro instruction applies only to multistacker devices. It specifies the

stacker into which cards of the file are to be selected.

Two formats of this macro instruction are available: one specifies the name of the file whose cards are to be stacked; the other does not specify a file name. These two formats are described as Format I and Format II, respectively.

FORMAT I. This format of the CNTRL macro instruction specifies the file whose cards are to be stacked.

This macro instruction consists of CNTRL in the operation field followed by an operand of the form:

SYMB,SS,n

where SYMB is the name of the file whose cards are to be stacked

SS identifies this as a stack macro instruction, and

n is the number of the stacker into which the cards are to be selected.

When two Format-I CNTRL macro instructions for stacking pertaining to the same file are given before a GET or PUT macro instruction for that file, the stacker selection specified in the second CNTRL macro instruction is effective (i.e., the second CNTRL macro instruction overrides the first).

Figure 44 shows a CNTRL macro instruction specifying that a card of a file named DAY is to be stacked in stacker 2 of the device handling this file.

Name	Operation	Operand
NAME	CNTRL	DAY,SS,2

Figure 44. CNTRL Macro Instruction for Stacking

The following explains the relationship between a given Format-I CNTRL macro instruction and the GET, PUT, or ECM macro instruction referring to the card to be selected.

OVERLAP MODE. If the file is being processed in the overlap mode, the Format-I CNTRL macro instruction must be the last macro instruction preceding the GET or PUT that refers to the card to be selected.

In the example below, the CNTRL macro instruction shown selects the card whose contents are transferred to or from the work area by the GET (or PUT) macro instruction.

```
------------------
------------------
CNTRL AAAA,SS,n
------------------   no GET or PUT macro
------------------   instruction referring
------------------   to file AAAA
GET(or PUT)AAAA,xxxx
------------------
------------------
```

NONOVERLAP MODE. The CNTRL macro instruction must be issued after the GET macro instruction or before the PUT macro instruction that moves the card to be selected.

In the coding sequence below, the CNTRL macro instruction shown selects the card read by the GET macro instruction.

```
------------------
------------------
GET AAAA,xxxx
------------------   no PUT, GET, or ECM
------------------   macro instruction re-
------------------   ferring to file AAAA
CNTRL AAAA,SS,n
------------------
------------------
```

In the coding sequence below, the CNTRL macro instruction shown selects the card moved by the PUT macro instruction.

```
------------------
------------------
CNTRL AAAA,SS,n
------------------   no PUT, GET, or ECM
------------------   macro instruction re-
------------------   ferring to file AAAA
PUT AAAA,xxxx
------------------
------------------
```

Format II

This format of the CNTRL macro instruction can be used only for the stacking of cards from files read and/or punched by the 2560 MFCM. Format II of the CNTRL macro instruction specifies the desired stacker but not the file whose cards are to be selected.

This format of the CNTRL macro instruction consists of CNTRL in the operation field followed by an operand of the form:

,SS,n

where SS indicates that the CNTRL macro instruction is to be used for stacking, and

n is the number of the stacker into which the cards are to be selected.

A name may be assigned to each CNTRL macro instruction of this format.

When two Format-II CNTRL macro instructions for stacking are given before the stacker selection is performed, the stacker selection specified in the second CNTRL macro instruction is effective (i.e., the second CNTRL macro instruction overrides the first).

Figure 45 shows a CNTRL macro instruction specifying that cards are to be stacked into stacker 3 of the 2560 MFCM.

Name	Operation	Operand
NAME	CNTRL	,SS,3

Figure 45. CNTRL Macro Instruction for Stacking (2560 MFCM)

The relationship between the Format-II CNTRL macro instruction and the GET, PUT, or EOM macro instruction referring to the card to be stacked is the same as that described for the CRDPR macro instruction.

The LOM Macro Instruction

The LOM (Leave Overlap Mode) macro instruction applies to combined files specified to be processed in the overlap mode. The processing of the file begins in nonoverlap mode after the next GET macro instruction for the specified file is executed.

This makes it possible to read and punch into the same card of a combined file that is being processed in the overlap mode.

Each LOM macro instruction may have a name and must contain:

1. LOM in the operation field.
2. An operand specifying the name of the file to which the macro instruction applies.

Figure 46 shows an example of an LOM macro instruction.

Name	Operation	Operand
NAME	LOM	TRAN

Figure 46. LOM Macro Instruction

If an LOM macro instruction is given for a particular file, all subsequent GET macro instructions for that file are performed in nonoverlap mode until an EOM (Enter-Overlap-Mode) macro instruction is given. (Refer to the section The EOM Macro Instruction for further information concerning the use of the LOM with EOM.)

The EOM Macro Instruction

The EOM (Enter-Overlap-Mode) macro instruction applies only to combined files for which a previous LOM macro instruction has been given. This macro instruction (1) causes the next card to be read into the read area, and (2) causes subsequent GET macro instructions addressing the same file to be performed in the overlap mode. The processing of the file in overlap mode begins immediately after the EOM is given.

Each EOM macro instruction may have a name and must contain:

1. EOM in the operation field, and
2. an operand specifying the name of the file to which the macro instruction applies.

Figure 47 shows an example of an EOM macro instruction.

Name	Operation	Operand
NAME	EOM	UPDT

Figure 47. EOM Macro Instruction

Programming Considerations - LOM and EOM Macro Instructions

A card that belongs to 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 operation (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 nonoverlap mode.
2. Do not provide an OVERLAP=NO detail entry for the file. Then 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 nonoverlap mode.

3. Do not provide an OVERLAP=NO detail entry for the file. Then, 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 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 nonoverlap mode and thus can be subsequently punched. However, when punching is not to be performed, the program immediately begins operation in the overlap mode. This third method requires some additional main storage positions for the extra LOM and EOM macro instructions, but it results in a program that runs at nearly the same speed as a program operating entirely in the overlap mode. This programming method is illustrated in the sample program coding shown in Figure 63 (Appendix D).

The CRDPR Macro Instruction

This macro instruction can be used only if the user has a 2560 Multi-Function Card Machine equipped with the Card Print Feature.

The CRDPR macro instruction moves information to be printed on a card from the work area to the card print area specified as the third operand of this macro instruction. For each line to be printed, the programmer must write one CRDPR macro instruction. If two CRDPR macro instructions are given for the same line, only the last one will be executed.

Because the CRDPR macro 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 (Figure 48). The second operand is the name of the work area, and the third operand is the name of the card print area. A name may be assigned to a CRDPR macro instruction.

Execution of a CRDPR macro instruction does not immediately result in printing on the card. Printing occurs when the card on which printing will take place is moved into and through the print station by the execution of a following GET, PUT, or EOM macro instruction. At that time, all specified print lines for a particular card are printed simultaneously.

All lines are printed each time a card print operation is performed. It is not possible to print only with print head 1 during one print operation and then print with print heads 1 and 2 during another print operation. Therefore it is possible not to print any data with one of the print heads by simply not entering any data into its respective print area or, if processing was performed in the area, by clearing the area before printing takes place.

Figure 48 shows an example of a CRDPR macro instruction. This macro instruction moves a line of information from the work area named WORK into the print area named TOTL.

Name	Operation	Operand
NAME	CRDPR	,WORK,TOTL

Figure 48. CRDPR Macro Instruction

As stated previously, the CRDPR macro instruction does not address a particular file. Therefore the card on which printing will take place must be moved into and through the print station by a GET, PUT, or EOM macro instruction. The programmer can determine, from the following three explanations, the sequence of GET, PUT, or EOM instructions required for his application.

1. Processing in the Overlap Mode

If the card on which printing will take place is punched by a PUT macro instruction, the CRDPR macro instruction must be given before any subsequent PUT, GET, or EOM macro instructions addressing an MFCM file. See the following coding sequence.

```
------------------
------------------
------------------
PUT(or GET) F1,xxxx
------------------   no PUTs, GETs or
------------------   EOMs referring to
------------------   MFCM files.
CRDPR ,xxxx,cardprint
------------------
------------------
```

2. Processing in the Nonoverlap Mode - Format A

If the card on which printing will take place is punched by a PUT macro instruction, then the CRDPR macro instruction must be given prior to any other GET, PUT, or EOM macro instruction that addresses an MFCM file. See the following coding sequence.

```
------------------
------------------
------------------
PUT F1,xxxx
------------------   no PUTs, GETs, or
------------------   EOMs referring to
------------------   MFCM files.
CRDPR ,xxxx,cardprint
------------------
------------------
```

There is one exception to the above case.

A GET F1 macro instruction may be inserted between the PUT F1 and the CRDPR macro instructions. See the following coding sequence.

```
------------------
------------------
------------------
PUT F1,xxxx
------------------
------------------   no PUTs, GETs, or
------------------   EOMs referring to
------------------   MFCM files.
------------------
GET F1,xxxx
------------------
------------------   no PUTs, GETs, or
------------------   EOMs referring to
------------------   MFCM files.
CRDPR ,xxxx,cardprint
------------------
------------------
```

3. Processing in the Nonoverlap Mode - Format B.

If the card on which printing will take place is read by a GET macro instruction referring to a file named F1, then another GET F1, EOM F1, or PUT F1 macro instruction must be given before the CRDPR macro instruction. See the following coding sequence.

```
------------------
------------------
GET F1,xxxx
------------------
------------------   any combination of
------------------   macro instructions
------------------   referring to File 2.
GET(or PUT or EOM)
    F1,xxxx
------------------
------------------   no PUTs, GETs, and
------------------   EOMs referring to
------------------   MFCM files.
CRDPR ,xxxx,cardprint
------------------
------------------
```

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 contents of the card that is made available by the first GET (refer to the above example) is to be printed on the same card, the work area specified in the second GET macro instruction must not be the same as the one specified in the first GET macro instruction. Specifying the same work area in both GET macro instructions causes the contents of the card that is read by the second GET to be card-printed on the card that is read by the first GET.

WAITC Macro Instruction

The WAITC macro instruction causes the problem program to wait for the completion of all pending 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 the program.

Figure 49 shows an example of a WAITC macro instruction. WAITC macro instructions may or may not have a name. Since this macro instruction neither refers to a particular file nor requests a particular control function, an operand is not required.

Name	Operation	Operand
[NAME]	WAITC	

Figure 49. Example of a WAITC Macro Instruction

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

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 in nonoverlap mode and a file in the other feed of the IBM 2560 is to be processed in overlap mode.

Except for the condition 2, above, a WAITC macro instruction need not be issued to permit the replacement of an error card if the cards of the file are to be read in nonoverlap 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 operations 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 nonprocess 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 50, which is a summary of the halt and restart information.

DUMMY GET MACRO INSTRUCTIONS: To ensure proper program functions on restart, i.e., resume processing with the record from the corrected card, the programmer must issue either one or two dummy GET macro instructions as shown in Figure 50.

For the explanation below, processing in the overlap mode is assumed, unless it is stated that the information applies to files that are processed in nonoverlap 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 moves the record from the card following the error card into the work area. To make sure that the contents of the corrected error card has been moved into the work area before normal processing is resumed, the first GET macro instruction encountered after restart must be a dummy GET, i.e., no processing must be performed on the record moved into the work area by means of this GET macro instruction. If an IBM 2501 is used to read the cards of a 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 cards of the other (nonerror) file are read in nonoverlap mode and (2) no GET has yet been given for the 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 51.

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.

Figure 50 is provided to facilitate the programming of restart routines and to furnish card-handling information that is not convered in the Model 20 IOCS Operating Procedures. The programmer must inform the operator of the number of cards to be returned to and placed in front of the remaining cards of the input deck. Any runout cards that are not to be returned to the input deck must be placed into the proper stacker manually.

I/O Device	Mode of Operation	WAITC required	Number of Dummy GETs	Number of Cards to be returned	
				Error Feed	Non-error Feed
2501	Nonoverlap	No	0	2	
	Overlap with one I/O area	Yes	1	3	
	Overlap with two I/O areas	Yes	2	4	
2560 Feed 1	Nonoverlap	No*	0	3	3
	Overlap	Yes**	1	4	3
2560 Feed 2	Nonoverlap	No*	0	2	2
	Overlap	Yes**	1	3	2
2520	Nonoverlap	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 50. Programming with the WAITC Macro Instruction -- Halt and Restart Information

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 described in the SRL publication IBM System/360 Model 20, Card Programming Support Input/Output Control System Operating Procedures, Order No. GC26-3803. 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 which do not have to be returned into the respective hopper.

The coding example in Figure 51 illustrates 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 from hopper 1 of 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 nonoverlap 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 51. These instructions are identified by sequence numbers in parentheses in the rightmost column of Figure 51. The sequence numbers are used as a reference in the explanations below.

If a card of file AAA does not contain a 1-punch in column 1, the branch to NERR (7) is not performed and the program executes the WAITC macro instruction (8) that precedes an HPR instruction (9). On restart, the program executes either one or two dummy GET macro instructions. Only one dummy GET macro instruction for file AAA (10) is executed if no GET macro instruction has yet been executed for file BBB. In this case, the branch instruction named SW (11) is executed and the second dummy GET macro instruction (12) is bypassed. Control is returned to the problem program by a branch to REPT 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 because it 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 was 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.

```
┌--------┬------┬--------------------┬-----┐
|        |Opera-|                    |Instr|
|Name    |tion  |Operand             |Sqnce|
├--------┼------┼--------------------┼-----┤
|        | .    |                    |     |
|        | .    |                    |     |
|        | .    |                    |     |
|        |GET   |BBB,WRK2            | (1) |
|        |MVI   |SW+1,X'00'          | (2) |
|        | .    |                    |     |
|        | .    |                    |     |
|        | .    |                    |     |
|        |CNTRL |BBB,SS,4            | (3) |
|        | .    |                    |     |
|        | .    |                    |     |
|        | .    |                    |     |
|REPT    |CNTRL |AAA,SS,2            | (4) |
|        |GET   |AAA,WRK1            | (5) |
|        |CLI   |WRK1,C'1'           | (6) |
|        |BC    |8,NERR              | (7) |
|        |WAITC |                    | (8) |
|        |HPR   |X'FFF',0            | (9) |
|        |GET   |AAA,WRK1            |(10) |
|SW      |BC    |15,BPSS             |(11) |
|        |GET   |BBB,WRK1            |(12) |
|        |CNTRL |BBB,SS,4            |(13) |
|BPSS    |BC    |15,REPT             |(14) |
|NERR    | .    |                    |     |
|        | .    |                    |     |
|        | .    |                    |     |
└--------┴------┴--------------------┴-----┘
```

Figure 51. Coding Example -- Programming with the WAITC Macro Instruction

If the cards of a combined file are also to be card-printed and this file is to be processed in nonoverlap 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 52.

The coding example in Figure 52 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.
3. All cards without a 1-punch in column 1 are error cards and must be replaced by the operator.

```
┌────────┬──────┬──────────────────┬─────┐
│        │Opera-│                  │Instr│
│Name    │tion  │Operand           │Sqnce│
├────────┼──────┼──────────────────┼─────┤
│        │  .   │                  │     │
│        │  .   │                  │     │
│        │  .   │                  │     │
│REPT    │GET   │CMBF,WRKC         │ (1) │
│        │CLI   │WRKC,C'1'         │ (2) │
│        │BC    │8,NERR            │ (3) │
│        │GET   │CMBF,WRKC         │ (4) │
│        │WAITC │                  │ (5) │
│        │HPR   │X'FFF',0          │ (6) │
│        │BC    │15,REPT           │ (7) │
│NERR    │  .   │                  │     │
│        │  .   │                  │     │
│        │  .   │                  │     │
│        │PUT   │CMBF,WRKC         │ (8) │
│        │CRDPR │                  │ (9) │
│        │BC    │15,REPT           │(10) │
│        │  .   │                  │     │
│        │  .   │                  │     │
└────────┴──────┴──────────────────┴─────┘
```

Figure 52. Coding Example -- Programming with the WAITC Macro Instruction Involving Card Printing

The sequence numbers shown in the rightmost column of Figure 52 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, apply also to stacker-select CNTRL macro instructions without having a file name as the first operand.

Appendix A. Approximate Main Storage Requirements of the IOCS Routines

The basic main storage requirement for all programs using the IOCS is 270 bytes. Additional main storage requirements depend on the IOCS features chosen and the input/output devices used. These requirements are listed in Figures 53 through 55. The values shown in these charts can be used to calculate the approximate main storage requirements of the IOCS object program routines.

Device	Model of Operation	Main Storage Requirement
Basic Routines		270
Printer	Standard Carriage	320
	2203 with Dual-Feed Carriage	680
2501 Card Reader	Models A1 and A2 nonoverlap	110
	Model A1 or A3 overlap	160
	Model A2 overlap	220
2520 Card Punch	nonoverlap	100
	overlap	150
2520 Card Read Punch	overlap combined	750
2560 MFCM	If only one file is used (combined without card print)	950
	If two files are used (combined with card print)	1470
1442, Model 5, or 8 Card Punch	nonoverlap	100
	overlap	150

Figure 53. Approximate IOCS Main Storage Requirements (in bytes) of the Routines for the Different Input/Output Devices

Program Feature	Main Storage Requirements: Basic	For Each File: Exit Entry	For Each File: No Exit Entry	For Each Field
RFORMTn detail entry	140*	14	MFCM, 2520: 36 2501: 24	4
PFORMTn detail entry	90*	18	MFCM, 2520: 40	
SEQNCE detail entry	-	24 bytes plus length of sequence field		-

*140 bytes are required for the joint use of RFORMTn and PFORMTn detail entries.

Figure 54. Approximate Main Storage Requirements (in bytes) of Additional IOCS Features

Macro Instruction	Main Storage Requirement
GET	6
PUT	6
OPEN	6
CLOSE	4
CRDPR	8
CNTRL	6
LOM	4
EOM	4
PRTOV	8
WAITC	4

Figure 55. Main Storage Requirements (in bytes) of Each Macro Instruction

Appendix B. Approximate Average Times Required for Generation and Execution of the IOCS Routines

Figures 56 and 57 show the approximate average times (in milliseconds) required for the execution of the various IOCS features and macro instructions. Figure 54 shows the approximate times required for generation of the IOCS symbolic programs. The generation time depends on (1) the device used to punch the IOCS symbolic deck, and (2) the machine configuration defined by the definition statements.

Device	File Type	Mode of Operation	GET Time Required	PUT Time Required
1403 Printer	simple	standard carriage	---	12
2203 Printer Model A1	simple	standard carriage	---	16
		dual-feed carriage	---	15**
2203 Printer Model A2	simple	standard carriage	---	24
		dual-feed carriage	---	22.5
2501 Card Reader, Models A1 and A2	simple	nonoverlap	9 plus read time	---
2501 Card Reader, Model A1	simple	overlap	12	---
2501 Card Reader, Model A2	simple	overlap	10	---
2520 Card Punch	simple	nonoverlap	---	10 plus punch time
		overlap	---	11
1442 Card Punch, Model 5	simple	nonoverlap	---	8 plus punch time
		overlap	---	9
2520 Card Read Punch	simple	nonoverlap	12 plus read time	14 plus punch feed time
		overlap	13	15
	combined	nonoverlap	15 plus read time	20 plus punch time*†
		overlap	18	24*
2560 Multi-Function Card Machine Model A1	simple	nonoverlap	18 plus read time	18 plus punch feed time.
		overlap	20	20
	combined	nonoverlap	19 plus read time	28 plus punch time plus read time*†
		overlap	20	28*

Figure 56. Approximate Average Times Required by the GET and PUT Macro Instructions, Part 1 of 2

Device	File Type	Mode of Operation	GET Time Required	PUT Time Required
2560 Multi-Function Card Machine Model A2	simple	nonoverlap	27 plus read time	27 plus punch time
		overlap	30	30
	combined	nonoverlap	28.5 plus read time	42 plus punch time
		overlap	30	42

*PUT macros for combined files contain a punch and a read command.
**Value assuming alternate lower and upper carriage print operations.
†If a GET follows a PUT for a combined file in nonoverlap mode, the GET and the PUT instructions require 28 msec plus punch time for 2520 and 35 msec plus punch and read time for 2560.

Figure 56. Approximate Average Times Required by the GET and PUT Macro Instructions, Part 2 of 2

Program Feature		msec per 10-Character Field to be Checked: Subm. 2	Subm. 3 or 4	Subm. 5	
SEQNCE detail entry		1.50	2.20	1.35	
RFORMTn detail entry	numeric	5.00	7.50	4.50	minimum
		13.00	19.50	11.70	maximum
	blank	4.00	6.00	3.60	
PFORMTn detail entry		4.00	6.00	3.60	

Figure 57. Approximate Average Times Required by the IOCS Features

Unit(s) Used for Generation	Time Required for Generation
2501 Card Reader Model A1, and 2520 Card Punch Model A2	4 to 6 minutes
2560 MFCM Model A1	6 to 12 minutes
2560 MFCM Model A2	9 to 18 minutes

Figure 58. Approximate Time Required for Generation of Symbolic IOCS Routines

Appendix C. Programming of User Routines

A user routine may be required in the main source program if certain checking functions (SEQNCE, RFORMT, or PFORMT) are desired. When the branch to a user routine occurs, the IOCS will automatically store the main program reentry address in general register 14. The routine must provide the linkage back to the main program.

If the user routine contains any macro instruction (all macro instructions are permitted), the contents of register 14 should be saved before this macro instruction is executed. If this is not done, during execution of the macro instruction in the user routine the reentry address is lost.

If a PUT macro instruction is given that refers to a combined file and the program proceeds to the PFORMTn error routine (user's exit), a subsequent GET macro instruction will place the contents of the card containing the PFORMT error into the work area. If this GET macro instruction is in nonoverlap mode, it is possible to punch this card by means of an additional PUT macro instruction.

USE OF BASE REGISTERS

Base register 15 is reserved for IOCS interrupt and macro routines. It must not be used by the programmer, not even if he saves its contents between two macro instructions.

Base register 14 is also used by the IOCS for all macro instructions. The programmer may use register 14 but does not have to save and restore its contents. However, the contents of the register will be changed during the execution of each macro instruction.

When an IOCS-controlled branch to a user-written routine (PFXIT, RFXIT, or SQXIT) occurs and the user desires to issue an IOCS macro instruction in his routine, he must save the contents of register 14 before the IOCS macro instruction is executed. He must restore the contents of register 14 to their original value before he returns control to the IOCS.

LANGUAGE COMPATIBILITY

Model 20 IOCS is closely patterned after the Basic Programming Support and Basic Operating System 8K Input/Output Control Systems. Since the Model 20 IOCS is designed to support input/output devices that are unique to the Model 20 and 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 models of System/360 should therefore be aware that programs using the Model 20 IOCS require modification prior to generation by the other systems.

PROGRAMMING ERRORS

Programming errors can only be detected during phases 1 and 2 of a generation, that is, when the contents of the CTL card and the definition cards are read, loaded into main storage, and checked.

A programming error prevents the IOCS to generate the specified routines and to punch or print a diagnostic message. If a printer is not used during the generation run, the diagnostic messages -- if any -- are punched into columns 1 through 10 of the incorrect definition cards, or into blank cards if the message refers to more than one definition card.

Programming errors always cause a machine halt at the end of phase 3 of a generation, that is, when the checking of the definition statements has been completed. No programming-error halts can occur during the execution of phases 1 and 2 of a generation run.

CAUTION: If the programmer uses a macro instruction for which no IOCS routine has been generated due to a missing DTFSR detail entry, no diagnostic message is produced during the generation run. Example: The programmer has not included a CONTROL=YES detail entry in the DTFSR statement for a file and the program contains a CNTRL macro instruction referring to this file.

Diagnostic Messages

The diagnostic messages that the IOCS produces upon detection of programming errors provide the programmer with an aid in identifying the errors.

Five types of programming errors are identified during phase 1 of a generation. Another seven types are detected in phase 2 and identified during phase 3 of a generation. Each of these errors causes a diagnostic message to be printed or punched. The occurrence of one or more of these errors causes a machine halt upon completion of phase 3.

Appendix A contains a summary of the diagnostic messages produced during the generation of IOCS routines. The subsequent text is a more detailed description of these diagnostic messages and their meaning.

Messages Produced During Phase 1

The following is a detailed description of the diagnostic messages produced during phase 1 and refers to errors detected during phase 1 of a generation.

Message	Meaning
COMMA	A detail card either has a continuation punch in column 72 and no comma following the specification, or no continuation punch in column 72 and a comma following the specification.
KEYWORD	The keyword in a detail card is not aligned in column 38, or is not followed by an equal sign (=), or is not one of the permitted keywords.
NUMBER	A numeric specification contains nonnumeric character(s), or exceeds the permissible limit.
PARAMETER	A keyword that requires a certain specification, or one of a number of prescribed specifications, is followed by an invalid specification.
SYMBOL	A name (or symbol) does not conform to the appropriate rules.

Messages Produced During Phase 3

During phase 3 of a generation run, IOCS may produce the following diagnostic messages that refer to errors detected during phase 2.

Message	Meaning
CARDBLKSZ	The block-size specification of a card file exceeds 80, or is greater than 160 in case of read or punch binary.
CRDPRREPTD	Card printing is specified for both feeds of the 2560 MFCM.
INCNSISTNT	Two or more detail entries of a file contradict each other, or at least one entry of a file is defined more than once.
INCOMPLETE	A mandatory detail entry is missing.
PRINTBLKSZ	The sum of the block-size specifications of two print files for a printer with a dual-feed carriage exceeds 144; or the block-size specification of a file for a printer with a single-feed carriage exceeds 144.
PRINTFILE	One of the feeds of a printer with a dual-feed carriage has been specified in the device-type entry of a file, while the other feed has not been assigned a file.
REPEATED	The device-type specification of a file contradicts the device-type specification of another file.

RELATIVE ADDRESSING

The programmer may desire to use relative addressing in routines of his program that include IOCS macro instructions. In this case, he must take into consideration the number of bytes required by the IOCS-generated instructions. Figure 55 (Appendix A) shows the number of bytes that the generated instructions require for each of the macro instructions available to users of the IOCS.

Appendix D. Sample Program

This section contains a sample program that illustrates the use of the IOCS definition statements and macro instructions.

The sample program is designed to perform an invoice billing application. The input to the program consists of:

1. An invoice number and date card containing the starting invoice number and today's date.
2. Customer address cards.
3. Order cards containing the order number, date of order, and customer number.
4. Detail item cards specifying the individual items ordered and customer number.

From these input cards the program produces:

1. An invoice for each customer.
2. An invoice summary card for each customer.
3. After all input cards have been processed, a new invoice-number and date card.

In addition, the gross and net amounts for a particular item ordered are punched into that item's detail item card.

The sample program is written for the following machine configuration:

4,096 positions of main storage
IBM 2560 Multi-Function Card Machine
IBM 2501 Card Reader
One Printer.

Input Cards

The input card formats are shown in Figures 59 through 62. Each type of input card has an identification code punched in column 1 for card-type identification purposes. The card types and their respective codes are:

Code	Card Types
0	invoice number and date card
1	customer address card
2	order card
3	detail item card

Invoice Format

The format of the invoices that will be printed by the program is shown in Figure 63. Only the first invoice page for each customer will contain the customer name and address, customer number, invoice number, and today's date. Each successive page, except the last, will contain only order number, order date, and item entries. The invoice gross amount, the percent of discount, the net amount, and the mode of payment will be printed on the last invoice page for each customer.

Invoice Summary Card

An invoice summary card is punched and interpreted for each invoice printed. The format of this card type is shown in Figure 64. A 4 will be punched in column 1 of each invoice summary card for card type identification purposes.

New Invoice Number and Date Card

After the last invoice has been prepared, the program punches a new invoice number and date card containing a 0 in column 1 and the beginning invoice number for the next program run in columns 2-8. Columns 11-16 (date field) are blank.

Program Data Flow

Figure 65 shows the flow of data for the sample program.

The sample program was written with the following assumptions:

1. The invoice number and date card will be the first card read from hopper 1 of the MFCM.
2. The customer address file contains an address card for each customer. The file is assumed to be in ascending sequence according to customer number. A sequence check will be performed on this file.
3. Each order card contains an order number and date and is followed by at least one detail item card. The customer number in each detail card is assumed to equal the number in the preceding order card. This file is assumed to be in sequence according to customer number and will not be sequence checked.

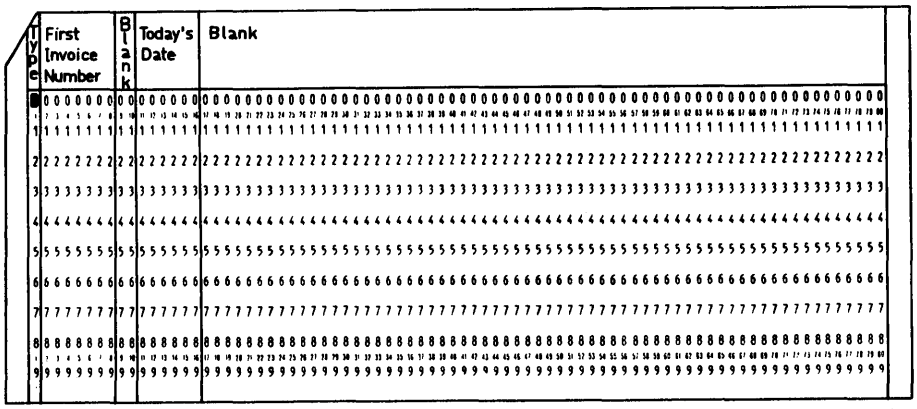

Figure 59. Invoice Summary and Date Card Format

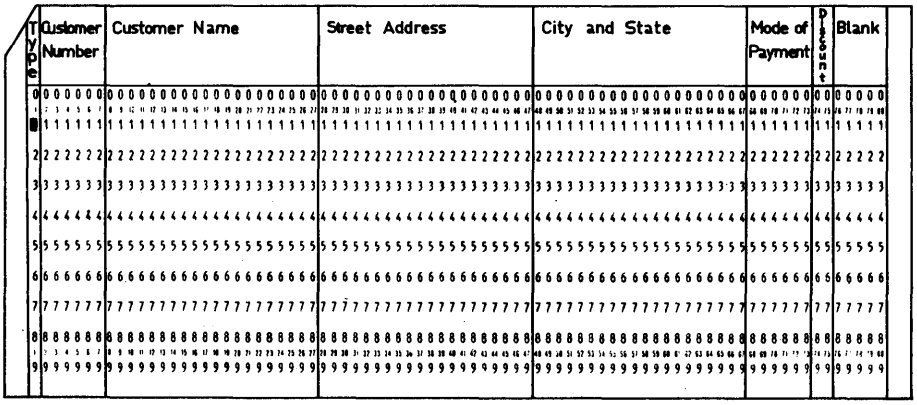

Figure 60. Customer Address Card Format

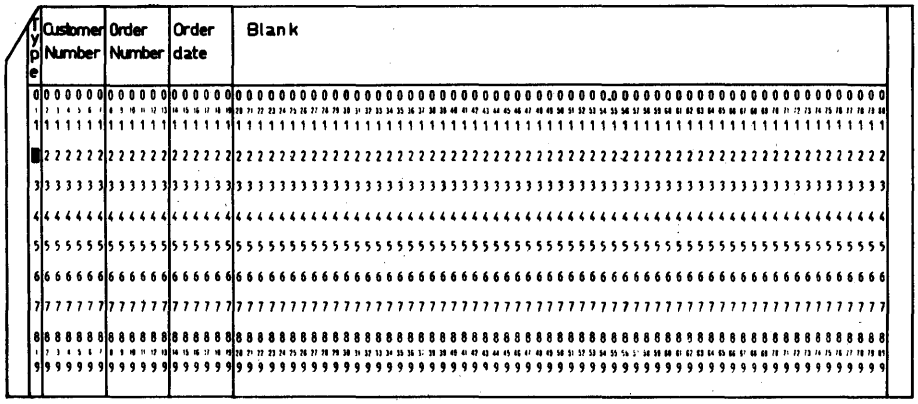

Figure 61. Order Card Format

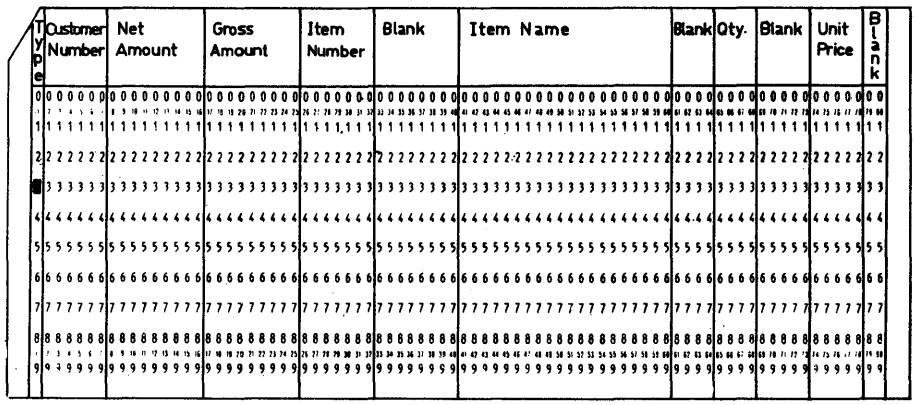

Figure 62. Detail Item Card Format

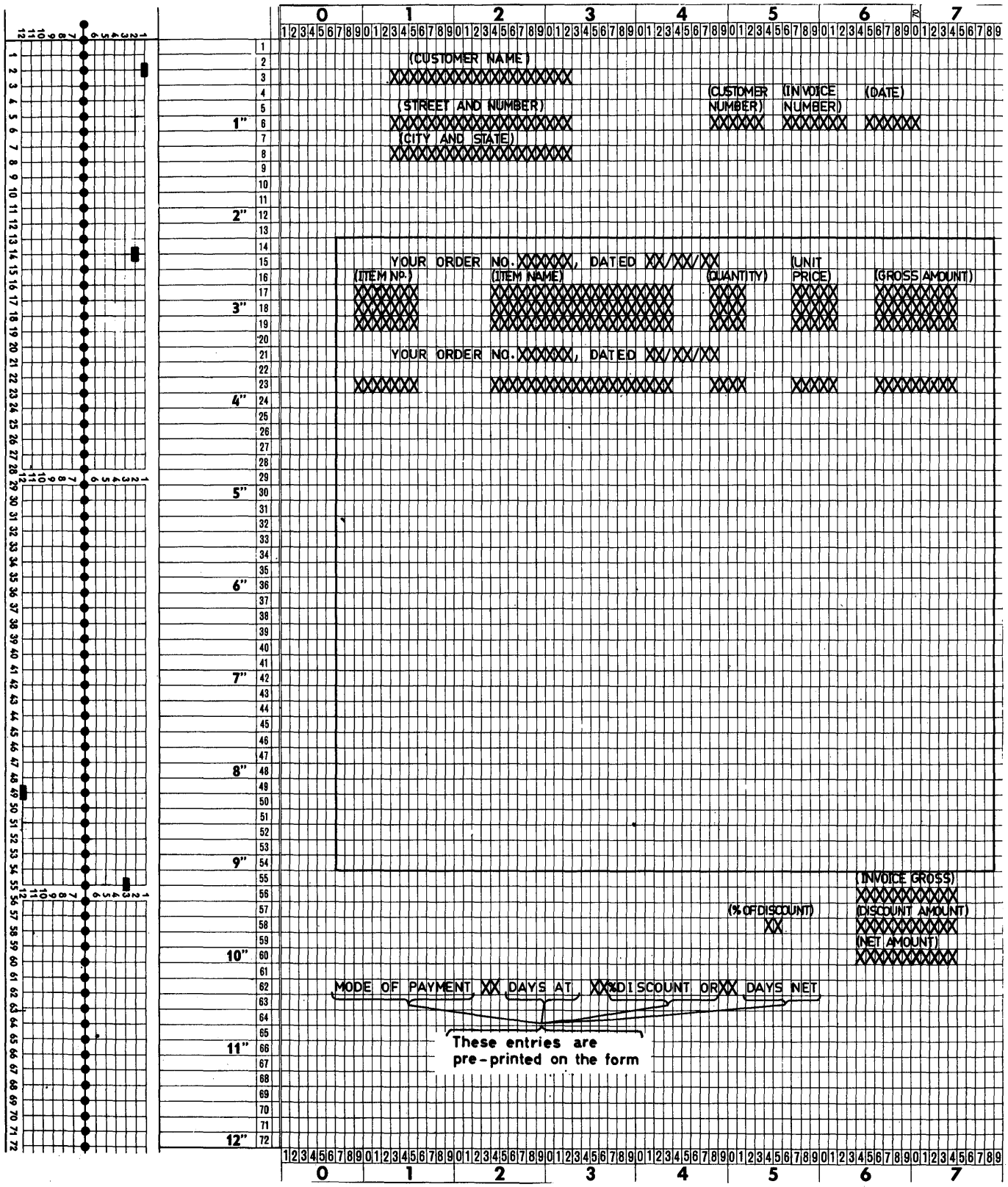

Figure 63. Invoice Format

4. Hopper 1 of the MFCM contains an invoice number and date card, order cards, and detail item cards only.

5. Hopper 2 of the MFCM contains only blank cards.

Definition Statements

Figure 66 shows the definition statements required to generate the IOCS routines for this program. All input/output devices will operate in the overlap mode.

The file named ORDR is a combined file because the detail item cards must be read and punched. Note the RFORMT0, and RFXII entries. The RFORMT0 entry specifies a check of columns 8-25 of each card to be punched to ensure that they are blank. The RFXII specifies that the user routine named PCHF is to be executed when columns 8-25 of a card to be punched are not blank. The PCHF routine selects the detail item card with punching in columns 8-25 and all following cards for the current customer into Stacker 3.

The file named ISUM is composed of the blank cards that will become the invoice summary cards. Note the use of the CRDPRLn entries to specify the print heads.

The file named ADDR contains the customer address cards.

The invoice file which will be printed on the printer is named PRIN.

Program Flow Chart and Coding

Figure 67 shows a flow chart for the main program and Figure 68 shows the main program coding in Model 20 Basic Assembler language.

The program was written assuming joint assembly. Each of the macro instructions shown in the program coding specifies the name of the work area to be used during the input/output operation. Some of the work areas are used for more than one file. For example, at one point in the program, a portion of the area named AR4 is used to contain data to be punched in a detail-item card and at a later point the same portion of AR4 is used to contain the data to be printed on the invoice.

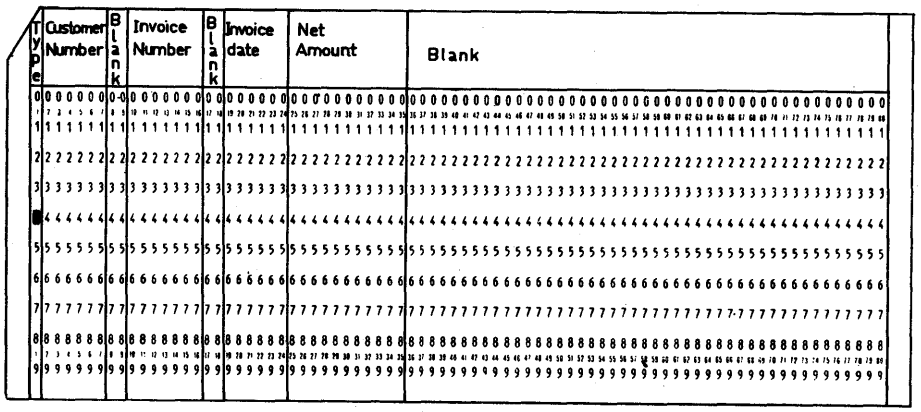

Figure 64. Invoice Summary Card Format

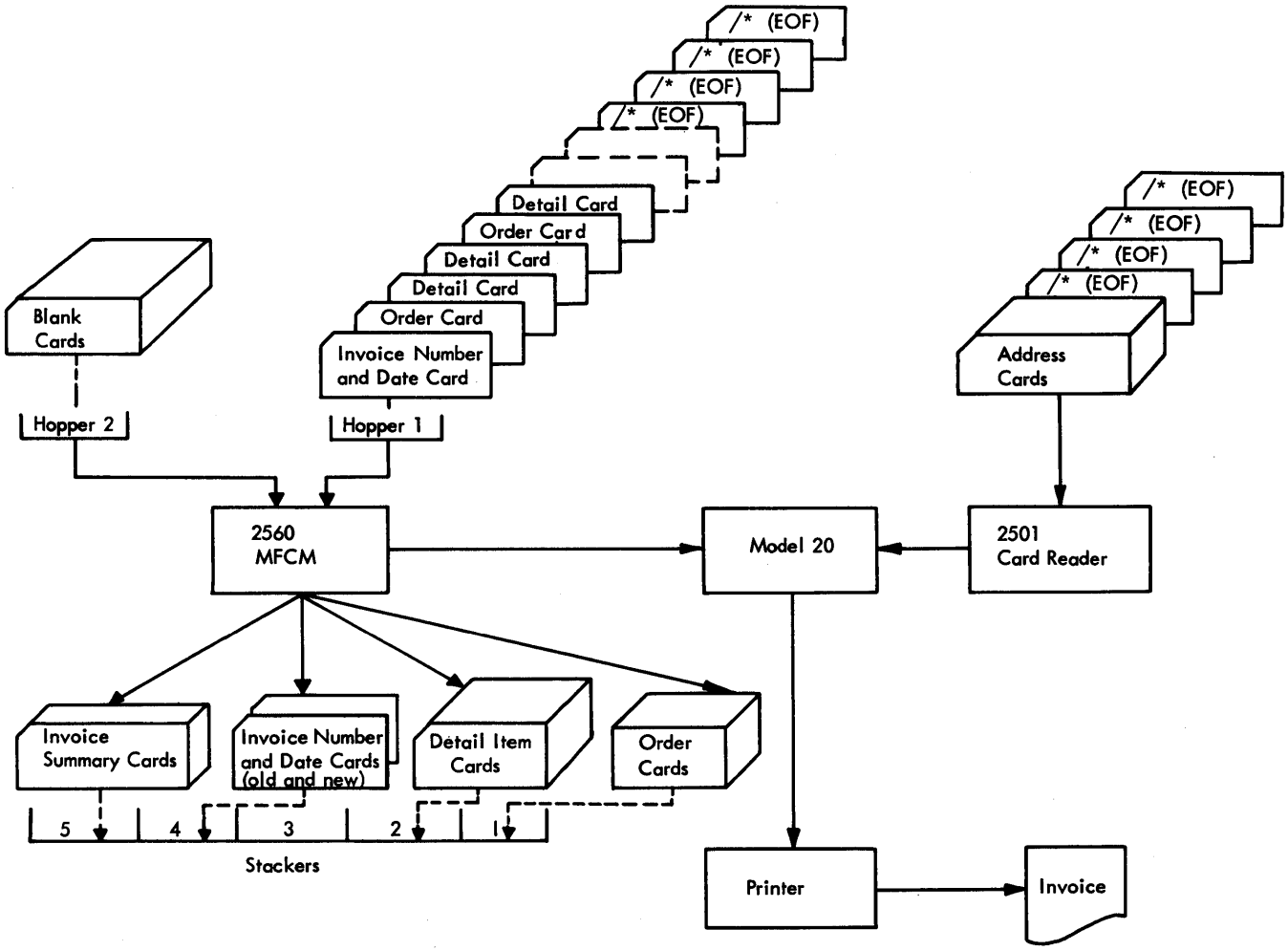

Figure 65. Program Data Flow

IBM System/360 Assembler
Short Coding Form

X28-6506
Printed in U.S.A.

PROGRAM: INVOICING
DEFINITION STATEMENTS
PROGRAMMER: B. J. LUND
DATE:
PUNCHING INSTRUCTIONS
GRAPHIC:
PUNCH:
CARD FORM #:
PAGE 1 OF 2

Name (25–30)	Operation (32–36)	Operand (38–71)	Comments	Identification-Sequence (73–80)
ORDR	DTFSR	DEVICE=MFCM1,	C	
		TYPEFLE=CMBND,	C	
		IMAREA=AR5,	C	
		IMBLKSZ=78,	C	
		OUAREA=AR6,	C	
		OUBLKSZ=25,	C	
		WORKA=YES,	C	
		RFORMT@=@@25@@,	C	
		RFXIT=PCHF,	C	
		CONTROL=YES,	C	
		EOFADDR=EOF1		
ISUM	DTFSR		C	
		DEVICE=MFCM2,	C	
		TYPEFLE=OUTPUT,	C	
		IOAREA1=AR7,	C	
		BLKSIZE=35,	C	
		WORKA=YES,	C	
		CRDPRA=CPL1,	C	
		CRDPRL1=7,	C	
		CRDPRL2=6,	C	
		CRDPRL4=34,	C	
		CONTROL=YES		

Figure 66. Sample Program Definition Statements, Part 1 of 2

IBM System/360 Assembler
Short Coding Form

X28-6506
Printed in U.S.A.

PROGRAM: INVOICING
DEFINITION STATEMENTS
PROGRAMMER: B. J. LUND
DATE:
PUNCHING INSTRUCTIONS
GRAPHIC:
PUNCH:
CARD FORM #:
PAGE 2 OF 2

Name (25–30)	Operation (32–36)	Operand (38–71)	Comments	Identification-Sequence (73–80)
ADDR	DTFSR	DEVICE=READ@1,	C	
		TYPEFLE=INPUT,	C	
		IOAREA1=AR8,	C	
		BLKSIZE=75,	C	
		WORKA=YES,	C	
		SEQNCE=@2@7,	C	
		SEQXIT=SEQR,	C	
		EOFADDR=EOFR		
PRIN	DTFSR		C	
		DEVICE=PRINTER,	C	
		PRINTOV=YES,	C	
		TYPEFLE=OUTPUT,	C	
		BLKSIZE=74,	C	
		CONTROL=YES,	C	
		WORKA=YES		
	DTFEN			

Figure 66. Sample Program Definition Statements, Part 2 of 2

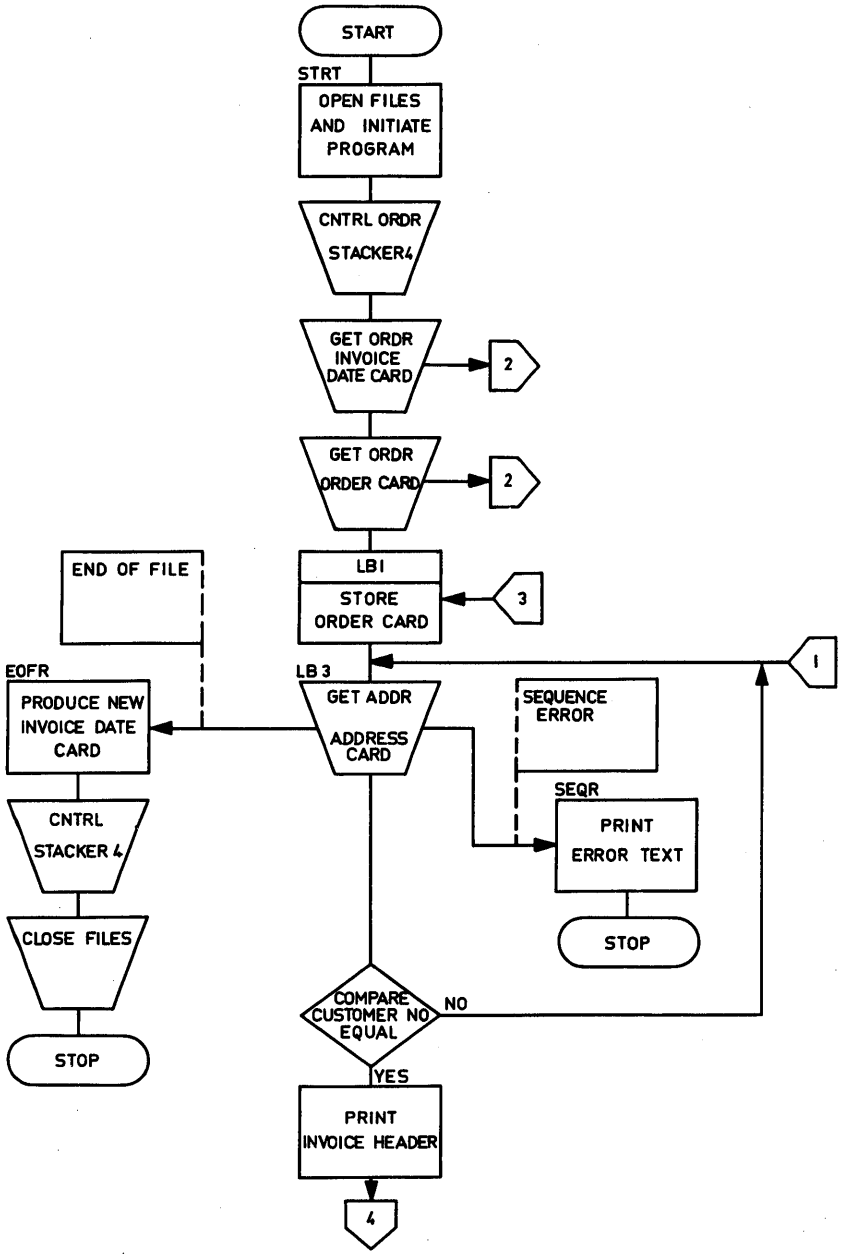

Figure 67. Sample Program Flow Chart, Part 1 of 2

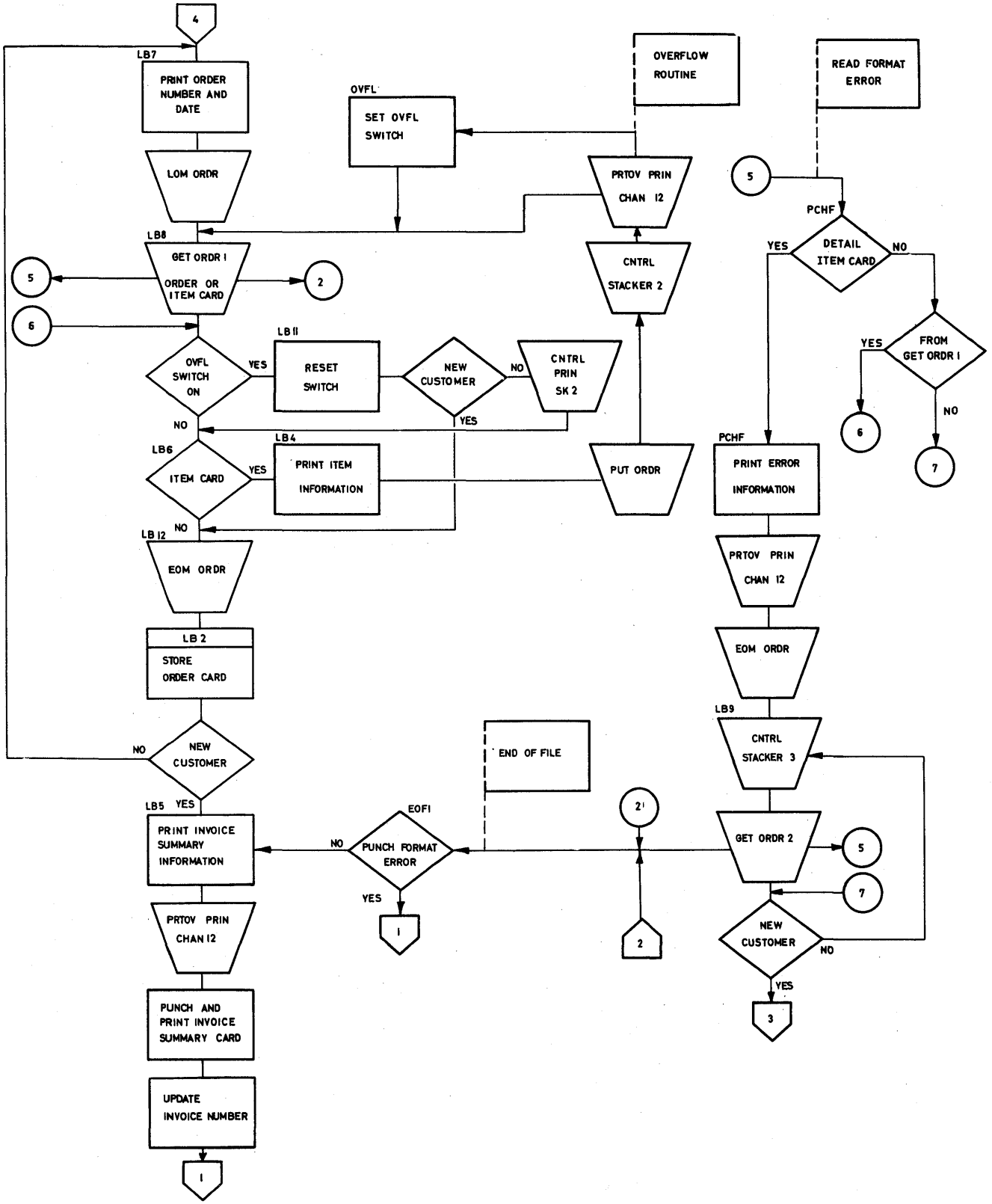

Figure 67. Sample Program Flow Chart, Part 2 of 2

IBM System/360 Assembler
Short Coding Form

X28-6506
Printed in U.S.A.

PROGRAM: INVOICING | PUNCHING INSTRUCTIONS | PAGE 1 OF 8

PROGRAMMER: B. J. LUND | DATE | GRAPHIC | PUNCH | CARD FORM #

Name	Operation	Operand	Comments
	START	256	
	USING	*,0	
*	FOLLOWING AREAS UTILIZE LOADER AREA		
ARS	DS	CL78	IOAREA1 ORDR RD
ARG	DS	CL25	IOAREA1 ORDR PCH
AR7	DS	CL35	IOAREA1 ISUM
AR8	DS	CL75	IOAREA1 ADDR
*	INITIALIZATION ROUTINE		
STRT	OPEN	ORDR	OPEN THE FILES
	OPEN	ISUM	
	OPEN	ADDR	
	OPEN	PRIN	
	CNTRL	PRIN,SK,1	
	CNTRL	ORDR,SS,4	
	GET	ORDR,AR1	INVOICE/DATE CRD
	MVI	AR1,X'40'	
	PACK	NO12,NO2	
	GET	ORDR,AR3	1ST ORDER CARD
	BAS	9,LB1	STORE ORDER CARD
*	LOCATE CUSTOMER ADDRESS CARD		
LB3	GET	ADDR,AR2	ADDRESS CARD
	CLC	NO1,AR2+1	COMPARE CUST NO
	BC	X'7',LB3	NOT EQUAL
	MVC	NO9(8),AR2+67	

Figure 68. Sample Program Coding, Part 1 of 8

IBM System/360 Assembler
Short Coding Form

X28-6506
Printed in U.S.A.

PROGRAM: INVOICING | PUNCHING INSTRUCTIONS | PAGE 2 OF 8

PROGRAMMER: B. J. LUND | DATE | GRAPHIC | PUNCH | CARD FORM #

Name	Operation	Operand	Comments
*	PRINT INVOICE HEADER INFORMATION		
	MVC	AR4+12(20),AR2+7	PRINT NAME
	CNTRL	PRIN,SP,,3	
	PUT	PRIN,AR4	
	MVC	AR4+12(20),AR2+27	PRINT STREET
	MVC	AR4+47(23),NO1	
	CNTRL	PRIN,SP,,2	
	PUT	PRIN,AR4	
	MVC	AR4+47(23),AR4+46	MOVE BLANK
	MVC	AR4+12(20),AR2+47	PRINT CITY+STATE
	CNTRL	PRIN,SK,,2	
	PUT	PRIN,AR4	
	CNTRL	PRIN,SP,,2	
*	PRINT ORDER CARD CONTENTS		
LB7	MVC	AR4+12(36),MO14	
	PUT	PRIN,AR4	
*	READ A CARD FROM ORDR FILE		
	LOCK	ORDR	
LB8	GET	ORDR,AR3	ORDER OR ITEM CD
	BC	X'0',LB11	OVERFLOW SWITCH
LB6	CLI	AR3,X'F3'	
	BC	X'8',LB4	ITEM CARD
LB12	EOM	ORDR	ORDER CARD
	BAS	9,LB2	STORE ORDER CARD

Figure 68. Sample Program Coding, Part 2 of 8

IBM System/360 Assembler
Short Coding Form

X28-6506
Printed in U.S.A.

PROGRAM: INVOICING | PUNCHING INSTRUCTIONS | PAGE 3 OF 8

PROGRAMMER: B.J. LUND | DATE | GRAPHIC | PUNCH | CARD FORM #

Name	Operation	Operand	Comments	Identification-Sequence
*		PREPARATION FOR PRINTING ORDER CARD		
	CLC	NO1,AR3+1	NEW CUSTOMER	
	BC	X'7',LBS	YES	
	CNTRL	PRIN,SP,1,2		
	BC	X'F',LB7		
*		PRINT INVOICE SUMMARY INFORMATION		
LBS	CNTRL	PRIN,SK,3		
	MVC	AR4+62(12),MSK2	GROSS AMOUNT	
	ED	AR4+62(12),NO5		
	CNTRL	PRIN,SP,,2		
	PUT	PRIN,AR4		
	MVC	AR4+53(2),NO10	DISCOUNT	
	PACK	NO8,NO10		
	MP	NO8,NO5(6)		
	OI	NO8+6,X'0F'		
	MAC	AR4+62(12),MSK2		
	ED	AR4+62(12),NO8+1		
	CNTRL	PRIN,SP,,2		
	PUT	PRIN,AR4		
	MVC	AR4+53(2),AR4+52		
	SP	NO5(6),NO8+1(6)	NET AMOUNT	
	MVC	AR4+62(12),MSK2		
	ED	AR4+62(12),NO5		
	SP	NO5(6),NO5(6)		
	CNTRL	PRIN,SP,,2		
	PUT	PRIN,AR4		
	MVC	AR2+24(11),AR4+63	FOR INV CARD	
	MVC	AR4+63(11),AR4+62		
	MVC	AR4+22(12),NO9	MODE OF PAYMENT	

Figure 68. Sample Program Coding, Part 3 of 8

IBM System/360 Assembler
Short Coding Form

X28-6506
Printed in U.S.A.

PROGRAM: INVOICING | PUNCHING INSTRUCTIONS | PAGE 4 OF 8

PROGRAMMER: B.J. LUND | DATE | GRAPHIC | PUNCH | CARD FORM #

Name	Operation	Operand	Comments	Identification-Sequence
	MVC	AR4+34(2),NO9+2		
	MVC	AR4+48(2),NO9+4		
	PUT	PRIN,AR4		
	PRTOV	PRIN,12		
	MVC	AR4+34(16),AR4+33		
*		PRODUCE INVOICE SUMMARY CARD		
	MVC	AR2(24),NO	PUNCH	
	PUT	ISUM,AR2		
	CRDPR	,AR12,CPL1	INVOICE NUMBER	
	CRDPR	,AR13,CPL2	INVOICE DATE	
	MVC	AR2+9(15),AR2+8		
	CRDPR	,AR11,CPL4	AMOUNT	
	MVC	NO1,AR3+1	STORE CUST NO	
*		INCREASE INVOICE NUMBER		
	AP	NO12(4),NO19(1)		
	MVC	AR1(8),MSK1		
	ED	AR1(8),NO12		
	BC	X'F',LB3		
*		STORE CONTENTS OF ORDER CARD		
LB1	MVC	NO1,AR3+1	CUSTOMER NO	
LB2	MVC	NO15,AR3+7	ORDER NO	
	MVC	NO16,AR3+13	MONTH	
	MVC	NO17,AR3+15	DAY	
	MVC	NO18,AR3+17	YEAR	
	MVC	AR3+17(2),AR3+19		
	OCR	X'F',9		

Figure 68. Sample Program Coding, Part 4 of 8

IBM System/360 Assembler
Short Coding Form
X28-6506 Printed in U.S.A.

PROGRAM: INVOICING | PUNCHING INSTRUCTIONS | PAGE 5 OF 8
GRAPHIC | CARD FORM #
PROGRAMMER: B. J. LUND | DATE | PUNCH

Name	Operation	Operand	Comments
*		PROCESSING OF DETAIL ITEM	CARD
LB4	PACK	MO7,AR4+47(4)	GROSS AMOUNT
	PACK	MO6,AR4+56(5)	
	MP	MO7,MO6	
	AP	MO5(6),MO7	
	MVC	AR4+64(10),MSK3	PRINT ITEM CARD
	ED	AR4+64(10),MO7+1	
	PUT	PRIN,AR4	
	PACK	MO8,MO10	NET AMOUNT
	MP	MO8,MO7	
	OI	MO8+6,X'0F'	
	SP	MO7,MO8+1(6)	
	MVC	AR4+55(10),MSK3	PUNCH ITEM CARD
	ED	AR4+55(10),MO7+1	
	MVC	AR4+0(43),AR4+7	
	PUT	OREOR,AREA10	
	CNTRL	,SS,2	
	MVC	AR4+56(18),AR4+55	
	PRTOV	PRIN,12,OVFL	
	BC	X'F',LB8	
*	OVERFLOW CONDITION		
LB11	MVI	LB8+7,X'00'	RESET SWITCH
	CLC	NO1,AR3+1	NEW CUSTOMER
	BC	X'7',LB12	YES
	CNTRL	PRIN,SK,2	
	BC	X'F',LB7	
*	USER ROUTINES		

Figure 68. Sample Program Coding, Part 5 of 8

IBM System/360 Assembler
Short Coding Form
X28-6506 Printed in U.S.A.

PROGRAM: INVOICING | PUNCHING INSTRUCTIONS | PAGE 6 OF 8
GRAPHIC | CARD FORM #
PROGRAMMER: B. J. LUND | DATE | PUNCH

Name	Operation	Operand	Comments
*	READ FORMAT ERROR ROUTINE		
PCHF	CLI	AR3,X'F3'	
	BCR	7,14	
	CNTRL	PRIN,SK,3	
	MVI	EOF1+1,X'00'	SET RFORM INDIC
	MVI	AR4,X'40'	
	MVC	AR4+1(49),AR4	
	MVC	AR4+50(24),NO13	PRINT ERROR TEXT
	PUT	PRIN,AR4	
	PRTOV	PRIN,12	
	MVC	AR4+50(24),AR4+49	
	EOM	ORDR	GET NEW CUSTOMER
LB9	CNTRL	,SS,3	SELECT STACKER 3
	GET	ORDR,AR3	
	CLC	NO1,AR3+1	NEW CUSTOMER
	BC	X'8',LB9	NO
	MVI	EOF1+1,X'F0'	RESET RFORM ERR
	BC	X'F',LB3-4	
*	ORDER FILE EOF ROUTINE		
EOF1	BC	X'F',LB5	NO RFORM ERROR
	BC	X'F',LB3	
*	SEQUENCE CHECK ROUTINE		
SEQR	MVC	AR4+46(28),NO4	PRINT ERROR TEXT
	MVC	AR4+22(2),AR4+21	
	CNTRL	PRIN,SK,3	
	PUT	PRIN,AR4	
	PPR	100,0	
	BC	X'F',*-4	

Figure 68. Sample Program Coding, Part 6 of 8

IBM System 360 Assembler
Short Coding Form

X28-6506
Printed in U.S.A.

PROGRAM: INVOICING | PUNCHING INSTRUCTIONS | PAGE 7 OF 8

GRAPHIC | CARD FORM #

PROGRAMMER: B. J. LUND | DATE | PUNCH

Name	Operation	Operand	Comments
*	ADDRESS	FILE EOF ROUTINE	
EOFR	MVC	AR4+35(7),NO2	A NEW INVOICE NO
	MVI	AR4+34,X'F0'	CARD INDICATOR
	PUT	ISUM,AR9	
	CNTRL	,SS,4	
	CLOSE	ORDR	CLOSE THE FILES
	CLOSE	ISUM	
	CLOSE	ADDR	
	CLOSE	PRIN	
	HPR	100,0	
	BC	X'F',*-4	
*	CHANNEL 12 CONDITION ROUTINE		
OVFL	MVI	LBB+7,X'F0'	SET OVERFLOW SW
	BCR	X'F',14	
*	DEFINITION OF AREAS AND CONSTANTS		
NO	DC	X'F4'	INDICATOR INV
NO1	DS	CL6	CUSTOMER NO
	DC	X'40'	
AR1	DS	CL1	WORKAREA
NO2	DS	CL7	INVOICE NO
	DS	CL2	
NO3	DS	CL6	DATE OF INVOICE
AR2	DS	CL58	WORKAREA
AR3	DS	CL17	WORKAREA
AR4	DS	CL74	WORKAREA
AR9	EQU	AR4+34	
AR10	EQU	AR4+49	
AR11	EQU	AR2+1	

Figure 68. Sample Program Coding, Part 7 of 8

IBM System 360 Assembler
Short Coding Form

X28-6506
Printed in U.S.A.

PROGRAM: INVOICING | PUNCHING INSTRUCTIONS | PAGE 8 OF 8

GRAPHIC | CARD FORM #

PROGRAMMER: B. J. LUND | DATE | PUNCH

Name	Operation	Operand	Comments
AR12	EQU	AR2+9	
AR13	EQU	AR2+18	
CPL1	DS	CL7	CRD PRT AREA L1
	ORG	*+29	
NO4	DC	C'ADDRESS CARD OUT'	
	DC	C' OF SEQUENCE'	
CPL2	DS	CL6	CRD PRT AREA L2
	ORG	*+30	
NO5	DC	XL6'F'	GROSS AMOUNT
NO7	DS	CL6	MULTIPLICATION
NO8	DS	CL8	MULTIPLICATION
MSK1	DC	X'402020202020212020'	
CPL4	DS	CL34	CRD PRT AREA L4
NO6	DS	CL3	UNIT PRICE
NO9	DS	CL6	MODE OF PAYMENT
NO10	DS	CL2	DISCOUNT
NO11	DC	X'00'	INDICATOR
NO12	DS	CL4	INVOICE NO
NO13	DC	C'PUNCH ERROR IN '	
	DC	C'ITEM CARD'	
NO14	DC	C'YOUR ORDER NO.'	
NO15	DS	CL6	ORDER NO
	DC	C', DATED '	
NO16	DS	CL2	MONTH
	DC	C'/'	
NO17	DS	CL2	DAY
	DC	C'/'	
NO18	DS	CL2	YEAR
NO19	DC	X'1F'	PACKED DECIMAL 1
MSK2	DC	X'4020202020202020212020202020'	
MSK3	DC	X'4020202020202120202020202020'	

Figure 68. Sample Program Coding, Part 8 of 8

Appendix E. Summary of Diagnostic Messages

Message	Meaning
CARDBLKSZ	BLKSIZE specification of card file exceeds 80 (or 160 if read or punch binary).
COMMA	comma after detail entry but no continuation punch in col. 72, or no comma but punch in col. 72.
CRDPRREPTD	card printing specified for both feeds of 2560 MFCM.
INCNSISTNT	two or more detail entries of a file contradict each other, or at least one entry of a file is defined more than once.
INCOMPLETE	mandatory detail entry missing.
KEYWORD	keyword not aligned in column 38, or not followed by equal sign, or invalid.
NUMBER	numeric specification invalid.
PARAMETER	specification invalid.
PRINTBLKSZ	dual-feed carriage -- sum of BLKSIZE specifications exceeds 144; single-feed carriage -- BLKSIZE specification exceeds 144.
PRINTFILE	one feed of printer with dual-feed carriage has not been assigned a file.
REPEATED	device-type specifications of two files contradict each other.
SYMBOL	symbolic name invalid.

Index

Syst[illegible]0 Model 20 CPS IOCS, S360(Mod. 20)-30, Printed in U.S.A. GC26-3603-4

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]