60496200

CONTROL DATA

# DMS-170

# FORM VERSION 1 REFERENCE MANUAL

**CDC® OPERATING SYSTEMS:**

**NOS 2**

**NOS/BE 1**

60496200

CONTROL DATA

DMS-170

# FORM VERSION 1 REFERENCE MANUAL

**CDC® OPERATING SYSTEMS:**

**NOS 2**

**NOS/BE 1**

# REVISION RECORD

| Revision | Description |
|---|---|
| A (11/01/75) | Original printing. |
| B (04/28/78) | Rewritten to reflect FORM Version 1.1 at PSR level 472. Major changes include revised directive syntax, single control statement copies, RT=S requirement when reading or writing IBM files, and requirement of FILE control statement for input and output files whenever block and record type differ from the CRM default. The multiple pass feature is not supported, and FORM is no longer callable from a user program. |
| C (10/31/80) | Rewritten to reflect FORM Version 1.2 at PSR level 528. Major changes include implementation of relational operators, implementation of FORM handling of variable length records, and enhancement of the MAX parameter. Entire manual has been reprinted. |
| D (07/25/83) | Revised to reflect FORM Version 1.2 at PSR level 587. This revision supports NOS Version 2 and includes miscellaneous technical corrections. |
| E (12/16/85) | Revised to reflect FORM Version 1.2 at PSR level 647. This revision deletes all references to operation under NOS 1, incorporates miscellaneous technical changes, and reflects release under NOS 2.4.3. |

REVISION LETTERS I, O, Q, AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
P. O. BOX 3492
SUNNYVALE, CALIFORNIA 94088-3492

or use Comment Sheet in the back of this manual

# LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

| Page | Revision | Page | Revision |
|---|---|---|---|
| Front Cover | - | B-1 | C |
| Title Page | - | B-2 thru B-6 | D |
| ii | E | B-6.1/B-6.2 | D |
| iii/iv | E | B-7 thru B-11 | C |
| v | E | C-1 thru C-5 | C |
| vi | E | D-1 | D |
| vii | D | D-2 | E |
| viii | D | D-3 | D |
| ix | C | E-1 thru E-6 | C |
| 1-1 | B | F-1 | C |
| 1-2 | C | F-2 | C |
| 1-3 | C | F-3 thru F-5 | D |
| 2-1 | C | G-1 | D |
| 2-2 thru 2-6 | D | G-2 thru G-5 | C |
| 2-7 | C | G-6 | D |
| 3-1 | D | H-1 thru H-3 | C |
| 3-2 | D | H-4 | E |
| 3-3 | E | H-5 | C |
| 3-4 | C | I-1 | B |
| 3-5 thru 3-12 | D | J-1 | C |
| 3-12.1/3-12.2 | D | K-1 | D |
| 3-13 thru 3-19 | C | K-2 | E |
| 4-1 | E | K-3 | C |
| 4-2 | E | K-4 | E |
| 4-3 thru 4-5 | D | K-5 | C |
| 4-6 | E | L-1 thru L-4 | C |
| 5-1 | D | M-1 | E |
| 5-2 thru 5-5 | C | M-2 | E |
| 5-6 | D | M-3 | D |
| 5-7 | C | N-1 | D |
| A-1 | D | N-2 | D |
| A-2 | E | Index-1 | E |
| A-3 | D | Index-2 | E |
| A-4 | E | Index-3 | D |
| A-5 | E | Comment Sheet/Mailer | E |
| A-6 | D | Back Cover | - |
| A-7 thru A-20 | D | | |

# PREFACE

The File Organizer and Record Manager (FORM) system is a comprehensive, general purpose utility designed for use with the following operating systems:

- NOS 2 for the CONTROL DATA® CYBER 180 Computer Systems; CYBER 170 Series; CYBER 70 Models 71, 72, 73, 74; and 6000 Series Computer Systems
- NOS/BE 1 for the CDC® CYBER 180 Computer Systems; CYBER 170 Series; CYBER 70 Models 71, 72, 73, 74; and 6000 Series Computer Systems

FORM can be used to manipulate records and reorganize files in formats the same as or different from the originals. By working through the CYBER Record Manager facilities of the NOS 2 or NOS/BE operating system, FORM can be used to create files with sequential, indexed sequential, actual key, or direct access organization. FORM is called through control statements.

The NOS system Information Manual is an online manual that includes brief descriptions of all NOS and NOS product manuals. To access this manual, log in to NOS and enter the command EXPLAIN.

The FORM user is expected to have some familiarity with the listed publications. The publications are listed alphabetically within groupings that indicate relative importance to the reader.

Information necessary for a complete understanding of FORM usage is contained in the publications listed below. The applicable operating systems are also indicated.

The following manuals are of primary interest:

| Publication | Publication Number | NOS 2 | NOS/BE |
|---|---|---|---|
| CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual | 60495700 | X | X |
| CYBER Record Manager Advanced Access Methods Version 2 Reference Manual | 60499300 | X | X |
| INTERCOM Version 5 Reference Manual | 60455010 | | X |
| Network Products Remote Batch Facility Reference Manual | 60499600 | X | |
| NOS Version 2 Reference Set, Volume 3, System Commands | 60459680 | X | |
| NOS Version 2 Reference Set, Volume 4, Program Interface | 60459690 | X | |
| NOS/BE 1 Reference Manual | 60493800 | | X |
| 8-Bit Subroutines Reference Manual | 60495500 | X | X |

The following manual is of secondary interest:

| Publication | Publication Number | NOS 2 | NOS/BE |
|---|---|---|---|
| CYBER Loader Version 1 Reference Manual | 60429800 | X | X |

Sites within the United States can order CDC manuals from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

Other sites can order CDC manuals by contacting the local country sales office.

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

# CONTENTS

## TABLES

# NOTATIONS

Each FORM directive is described in terms of a reference format. The following conventions are used:

| | |
|---|---|
| UPPERCASE | words are keywords and must appear exactly as shown. |
| lowercase | words represent the words or symbols supplied by the user. |
| [ ] Brackets | enclose optional portions of a reference format. All of the format within the brackets can be omitted or included at the user's option. If items are stacked vertically within brackets, only one of the stacked items can be used. |
| { } Braces | enclose two or more vertically stacked items in a reference format when only one of the enclosed items must be used. |
| ... Ellipses | immediately follow a pair of brackets or braces to indicate that the enclosed material can be repeated at the user's option. |
| Δ | indicates a space (blank). |

Punctuation symbols shown within the formats are required unless otherwise noted. Unless otherwise specified all references to numbers are to decimal values.

# INTRODUCTION 1

File Organizer and Record Manager (FORM) is a file management utility callable by control statements. The user specifies a single source of input records, either a tape or disk file or a user-supplied owncode routine from which records are delivered individually. FORM then operates on this input to produce up to twenty restructured output files. Detailed processing is controlled by FORM directives through which the user specifies the functions to be performed. The FORM functions can be used to:

- Perform conversions between IBM sequential 8-bit tape files and CDC internal format, maintaining 8-bit significance where necessary. Once an IBM file has been converted to CDC format, any of the functions valid for CDC files can be performed.
- Convert files from one organization type to another; for example, from sequential to indexed sequential.
- Perform simple file copying.
- Copy selected records from an input file to an output file with specified data conversions taking place.
- Reformat records in terms of data fields, including: field reordering, data conversion, literal insertion, zero or blank suppression, data packing and expanding, and data truncation. These operations can occur in various combinations.
- Sequence number a file.
- Reformat a file for printing.

The user also can specify error processing and collating sequence options as well as owncode subroutines to be executed at various points during execution of FORM.

All input/ouput is handled by CDC® CYBER Record Manager (CRM). The user must supply proper CRM descriptions of files being used, as outlined in section 5. The FORM-CRM functional configuration is shown in figure 1-1.

## DIRECTIVE OVERVIEW

The user communicates with FORM through directives, each of which performs a specific function. A directive is specified by means of a 3-letter mnemonic. The eight directive mnemonics and their associated functions are listed in table 1-1.

The directives are passed to FORM either as part of the job deck (file INPUT) or as a separate file of card images specified on the FORM control statement.

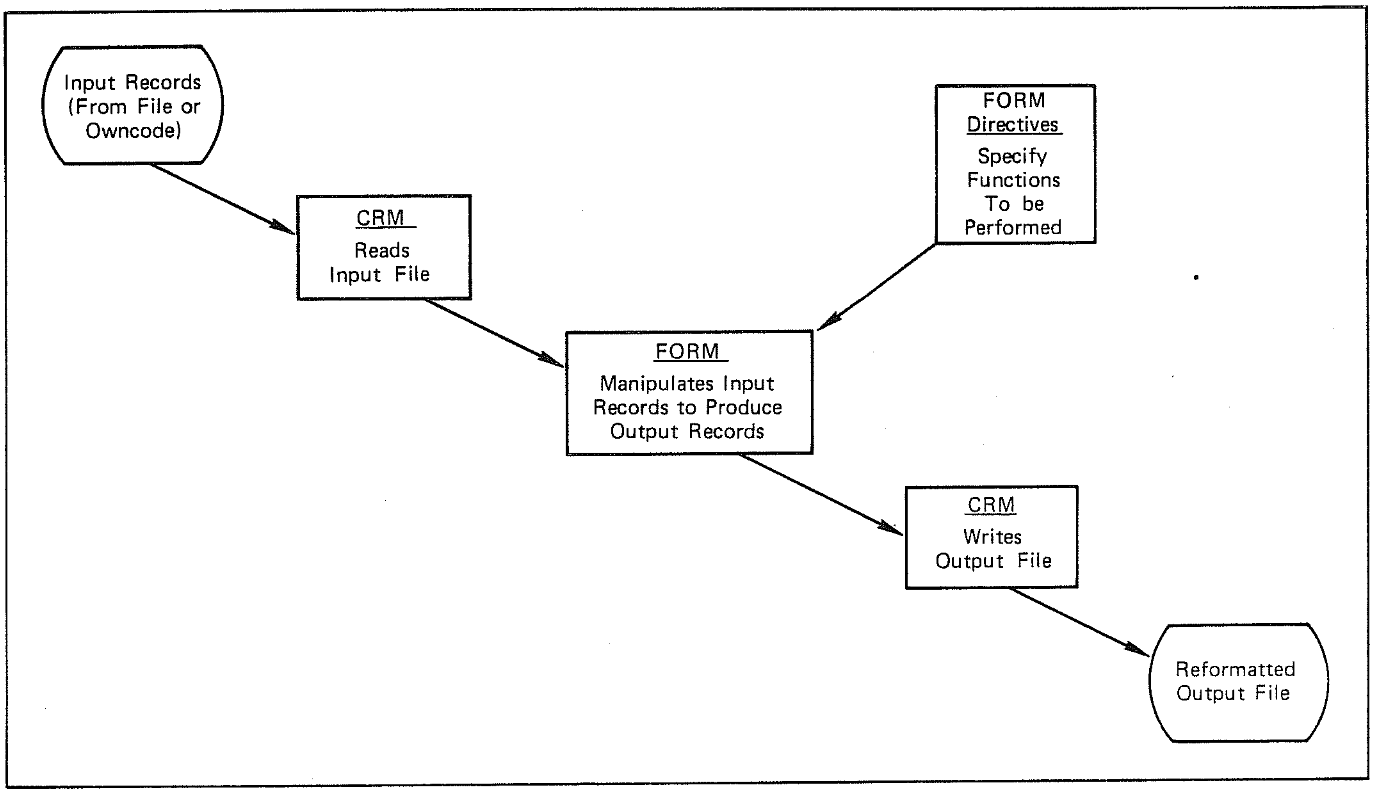

Input Records (From File or Owncode)
CRM Reads Input File
FORM Directives Specify Functions To be Performed
FORM Manipulates Input Records to Produce Output Records
CRM Writes Output File
Reformatted Output File

Figure 1-1. FORM-CRM Functional Configuration

TABLE 1-1. FORM DIRECTIVES

| Directive Function | Directive Mnemonic | Description |
|---|---|---|
| Input | INP | Declares an existing file to be used as a source of input records. |
| Output | OUT | Declares a single output file to be generated by FORM. |
| File Conversion | CON | Performs conversions between IBM and CDC files. |
| Record Qualification | QAL | Selects records from the source file to be sent to an output file. |
| Record Reformatting | REF | Reformats input records for output. |
| Record Sequencing | SEQ | Places a sequence number into any position of an output record. |
| Print Formatting | PAG | Formats a file for printing. |
| Execute | XEQ | Specifies entry point names of owncode routines to perform error processing and to supply input records. |

Typical uses of FORM directives include using INP to specify an input file to be modified; OUT (and PAG if the file is to be printed) to specify an output file; and using QAL, REF, SEQ, and CON to specify how the output file records are to differ from the input file records. As an alternative to the INP and OUT directives, an input and an output file can be specified on the FORM control statement, or input records can be passed to FORM via a user-supplied routine. FORM can process only one input file but can generate up to twenty output files.

## FORM EXECUTION

When FORM is called, it reads all directives, whether they exist as part of the job deck or in card images on another file, and scans for syntax errors. If errors are detected, appropriate diagnostics are written to file OUTPUT. If no syntax errors have occurred, execution of the directives begins when an end-of-record is encountered in the directive stream. FORM does not relinquish control, other than at user-defined owncode exits, until the run terminates.

A FORM run consists of a single call to FORM. During a run, functions are performed according to the directive parameters. When all directives have been executed, the run terminates. FORM executes all directives in a predetermined order, regardless of the order in which they are specified.

An example of a sequence of control statements and directives comprising a FORM run is illustrated in figure 1-2. Control statements are included to attach the input file and to make the output files permanent. In this example, the input and output files are assumed to have other than default record and blocking types; these characteristics are specified to CRM by the FILE statements. The file containing the input records, FILE1, is identified to CRM by the FILE control statement and to FORM by the INP directive. Two output files, FILE2 and FILE3, are declared by the OUT directive. The QAL directive specifies that FILE2 is to contain only those records from FILE1 that satisfy condition-1. The records are to be reformatted according to format-1 of the first REF directive. FILE3 is to contain all of the records of FILE1, reformatted according to format-2 of the second REF directive. The 7/8/9 card (end-of-record) initiates execution of the preceding directives.

```
NOS:

Job statement
USER statement
CHARGE statement
ATTACH(FILE1=PFILE1)              Attach input file
FILE(FILE1, . . . )  )
FILE(FILE2, . . . )  }            Define files to CRM
FILE(FILE3, . . . )  )
DEFINE(FILE2=PFILE2) )            Make output files
DEFINE(FILE3=PFILE3) )            permanent
FORM
7/8/9 in column 1
INP(FILE1)              )
OUT(FILE2)              |
OUT(FILE3)              }         FORM directives
QAL(FILE2,condition-1)  |
REF(FILE2,format-1)     |
REF(FILE3,format-2)     )
7/8/9 in column 1


NOS/BE:

JOB statement
Account statement
ATTACH(FILE1,PFILE1,ID=MYID)      Attach input file
FILE(FILE1, . . . )  )
FILE(FILE2, . . . )  }            Define files to CRM
FILE(FILE3, . . . )  )
REQUEST(FILE2,*PF)   )            Assign output files
REQUEST(FILE3,*PF)   )            to storage device
FORM.
CATALOG(FILE2,PFILE2,ID=MYID) )  Make output files
CATALOG(FILE3,PFILE3,ID=MYID) )  permanent
7/8/9 in column 1
INP(FILE1)              )
OUT(FILE2)              |
OUT(FILE3)              }         FORM directives
QAL(FILE2,condition-1)  |
REF(FILE2,format-1)     |
REF(FILE3,format-2)     )
7/8/9 in column 1
```

Figure 1-2. Example of FORM Job Decks

FORM functions always are executed in the order shown in figure 1-3. User owncode exits are indicated to show the point in processing at which they are called.

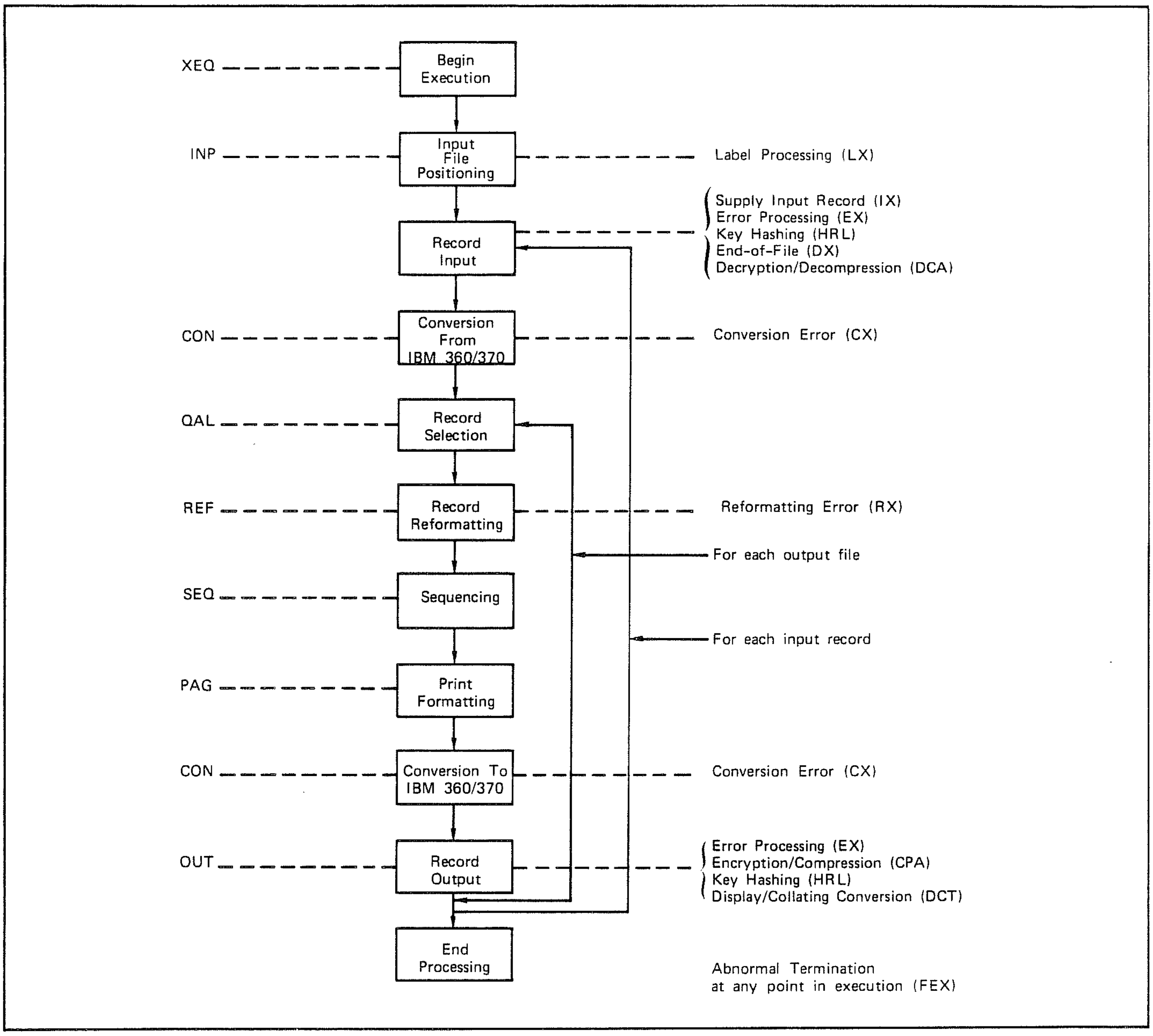


Figure 1-3. FORM Logical Flow

# DIRECTIVE ELEMENTS 2

Elements comprising the FORM directives include characters, literals, keywords, item descriptors, and selector expressions, organized according to the directive format. Equating of logical file names is a shorthand method of duplicating directive specifications.

## DIRECTIVE FORMAT

A FORM directive consists of a directive mnemonic followed by a logical file name and a parameter list. The directive format is shown in figure 2-1. The directive mnemonic and the logical file name must precede the parameter list. Parameters can be specified in any order. The logical file name and the parameter list must be enclosed by parentheses. Parameters must be separated by commas.

fnc(lfn,param-list)

| | |
|---|---|
| fnc | 3-letter mnemonic for the directive. |
| lfn | Logical file name consisting of 1 through 7 letters or digits, beginning with a letter; required in all directives except XEQ. |
| param-list | For the INP, OUT, PAG, SEQ, and XEQ directives, a list of options in the format: keyword-1=opt-1,keyword-2=opt-2, ..... |

Figure 2-1. Directive Format

The XEQ directive is the only directive that does not require a logical file name. If an XEQ directive appears without a parameter list, it must be terminated by a period. Blanks are significant only when used in literals and when preceding and following logical operators; otherwise, blanks are ignored and can be included to improve readability.

FORM directives are coded in the first 72 columns of 80-column cards or as card images. A directive can begin in any column and can span more than one line. The first 72 characters of a line are scanned. Only literals can extend beyond column 72. Literals that extend beyond column 72 are treated as if they continue in column 1 of the next line. All other characters extending beyond column 72 are ignored. Continuation lines are valid and can begin in any column; however continuation and terminating characters must not be placed beyond column 72.

Reformat-items, qualification-items, and conversion-items cannot be split across lines. Literals can be split across lines. Each FORM directive must begin a new line; characters appearing after the right parenthesis terminating the parameter list are ignored by FORM and can be used for comments.

## EQUATING LOGICAL FILE NAMES

Directive specifications for files can be duplicated in a FORM run by equating logical file names. FORM copies the specifications of the directive for the logical file name on the left of the equals sign from the specifications of the identical directive for the logical file name on the right.

The technique of equating logical file names is illustrated in figure 2-2. In the first example, two output files, OUTFIL1 and OUTFIL2, are created, each containing the same records selected by the QAL directives but differing in the formats specified by the REF directives. The equating of logical file names in the QAL directive causes the same record selection criteria to apply to OUTFIL1 and OUTFIL2. The directive containing the specifications must precede the directive which equates the file names.

Example 1.

```
INP(INFILE)
OUT(OUTFIL1)
OUT(OUTFIL2)
QAL(OUTFIL1,condition-1)
QAL(OUTFIL2=OUTFIL1)
REF(OUTFIL1,format-1)
REF(OUTFIL2,format-2)
7/8/9 in column 1
```

Example 2.

```
INP(INFILE)
OUT(OUTFIL3)
OUT(OUTFIL4)
QAL(OUTFIL3,condition-1)
QAL(OUTFIL4,condition-2)
REF(OUTFIL3,format-1)
REF(OUTFIL4=OUTFIL3)
7/8/9 in column 1
```

Figure 2-2. Equating Logical File Names

This technique can be used with any directive in which a logical file name is required. In the second example, the files OUTFIL3 and OUTFIL4 each contain different records as selected by the QAL directives but are reformatted the same way by the REF directives.

## CHARACTER SET

For constructing parameters of a directive, the following characters are recognized by FORM:

- digits 0 through 9
- letters A through Z
- special characters + * / . $ , ( ) = : ; blank

The following characters are recognized as delimiters:

| | |
|---|---|
| $ or * | Delimit literals. |
| ( ) | Clarify selector expressions; enclose directive parameters (reformat strings, conversions strings, and qualification strings). |
| , | Separates parameters within directives. |
| : | Separates selector expressions from reformat items and conversion items. |
| ; | Separates reformat specifications and conversion specifications. |
| = | Acts as a replacement operator. |
| blank | Separates operands and operators within selector expressions. |

One or more blanks adjacent to a valid separator are ignored; blanks can be used for clarity. Elsewhere in this book the symbol Δ indicates a space or a blank.

## KEYWORDS

A keyword is a 2- or 3-letter mnemonic that has a predefined meaning to FORM. Parameters of certain directives are expressed in the following keyword format:

```
keyword=value
```

The order in which keywords are specified within a directive is not fixed. To illustrate the keyword notation, assume that processing of the input file INFILE is to be limited to a maximum of 100 records. The directive is:

```
INP(INFILE,MAX=100)
```

Keywords associated with each directive are discussed in section 3, and a complete list of keywords and directives appears in appendix H.

## LITERALS

In certain directive formats, a literal is required as part of the option specification. A literal is a string of characters that represents an actual value. Two types of literals are recognized by FORM: string literals and numeric literals.

### STRING LITERALS

A string literal is written as a string composed of any symbol or symbols in the character sets in appendix A and enclosed by identical delimiter characters. The characters * or $ are used as delimiters and are not considered part of the literal. Blanks are valid within string literals. When the literal contains either * or $, the other character should be used as the delimiter pair. If the delimiter character must be part of the string, each enclosed occurrence must be doubled. Assume the character string

```
ABC*DEF
```

The character string can be represented by the string literal

```
$ABC*DEF$
```

Alternatively, the character string can be represented by the string literal

```
*ABC**DEF*
```

A literal must not exceed 255 characters, excluding delimiters, when used with the REF or QAL directive and can contain any character in the display code character set. A literal must not exceed 80 characters when used with the CON directive.

Examples:

| Literal | Character String |
|---|---|
| *A,B C* | A,B C |
| $AB$$CD$ | AB$CD |
| *123****456* | 123**456 |
| $$$$ | $ |

A literal symbol within a string must not be doubled unless the same symbol is used as the literal delimiter.

Examples:

| Literal | Character String |
|---|---|
| $AB*CD$ | AB*CD |
| *ABC$**$DEF* | ABC$*$DEF |

### NUMERIC LITERALS

Numeric literals represent actual numeric values. Numeric literals are written in a form similar to the numeric notation of FORTRAN. The permissible forms of numeric literals are the following:

$\pm n \quad \pm n.n \quad \pm n. \quad \pm n.E\pm s \quad \pm n.nE\pm s \quad \pm .nE\pm s$

The numeric value is represented by n, an integer of up to 27 significant digits. The value of the exponent is represented by s. The exponent value for a numeric literal must be an integer and cannot exceed 511. If the sign is omitted, a positive value is assumed.

Examples:

| | |
|---|---|
| 0 | 124.E-4 (represents .0124) |
| -3.5 | -56.2E+3 (represents -56200.) |
| +16. | .314E1 (represents 3.14) |

NOTE

Numeric literals are valid in the CON directive only. When the user wishes to specify a numeric value as an operand in any other directive, the value must be expressed as a string literal; that is, it must be enclosed by a $ or * delimiter pair. FORM performs the necessary conversion from character to numeric format.

When an item descriptor with B format is used, only ones and zeros for the bits can be in the literal. Item descriptors are described later in this section.

## ITEM DESCRIPTORS

Item descriptors are symbolic representations of data items to be manipulated by FORM. Item descriptors are required to describe data items involved in record selection (QAL directive), record reformatting (REF directive), and IBM conversion (CON directive) functions.

Item descriptors describe data fields in terms of starting position within the record, data format, and length. For purposes of data description, each record is considered to be a string of bytes with the byte length determined by the storage device in use: when the field is stored internally, the string contains 6-bit bytes; when stored on IBM tape, the string contains 8-bit bytes. Bits within each byte are numbered 1 through 6 or 1 through 8, from left to right.

The item descriptor formats are as follows:

Tm    iTm

±iTm    i/wTm

±i/wTm

i    is a byte index denoting the initial byte position of the data field; i can have two interpretations:

| | |
|---|---|
| absolute index | i denotes a byte position relative to the beginning of the record and must be written as an unsigned decimal integer. The initial byte is numbered 1. |
| relative index | i denotes a byte position relative to the current byte and must be written as a signed decimal integer. The sign indicates the direction of the move (+ forward, - backward). The current byte is the one next in sequence to the byte previously indicated in a string of descriptors. The initial current byte is the first byte of the record. |

If i is omitted, the current byte is assumed. The byte index, i, is used for all data types. In a CDC record, each byte contains 1 character (6 bits), with 10 characters equivalent to an internal word of 60 bits.

i/w    is a byte and bit index separated by a slash, indicating the starting position for a bit string (data type B) only. In the format i/w, i is a byte index as described previously, and w is an absolute bit position within the byte (the leftmost bit is numbered 1). The value given for w must not exceed the size of the byte in the source medium: 6 bits for CDC records, 8 bits for IBM records.

T    is a one-chararacter mnemonic code indicating the format of the data field.

m    is either a decimal integer or the word ALL.

| | |
|---|---|
| decimal integer | m specifies the length in bytes (bits if a type B item) of the data item; m must be omitted for data types H, W, G, F, L, E, I, U, D. A default of 1 is assumed if m is omitted for data types B, X, P, S, N, Z. |
| ALL | m specifies that the remaining data in the source field (everything to the right of the position specified by i) is to be moved to the destination data field. |

Valid values for T and m are listed in table 2-1. Data type X can be any member of the character set in use. S, N, Z, and P are subsets of X and can contain numeric characters only. These data types, along with type B, describe variable length data fields; m indicates the length of the field. All other data types describe fixed length fields; m cannot be used with these types. Examples of item descriptors are shown in figure 2-3.

TABLE 2-1. VALID T AND m VALUES FOR ITEM DESCRIPTIONS

| T | Description | m† |
|---|---|---|
| | IBM Format†† | |
| B | Bits | Size of field in bits |
| X | 8-bit alphanumeric characters | Size of field in 8-bit characters |
| H | Half-word (16-bit) integer | -- |
| W | Whole-word (32-bit) integer | -- |
| G | Double-word (64-bit) integer | -- |
| F | Floating-point (32-bit) | -- |
| L | Long floating-point (64-bit) | -- |
| E | Extended-precision floating-point (128-bit) | -- |
| P | Packed decimal (IBM COMP-3 COBOL items) | Size of field in 8-bit bytes |
| S | Decimal signed numeric | Size of field in 8-bit bytes |

TABLE 2-1. VALID T AND m VALUES FOR ITEM DESCRIPTIONS (Contd)

| T | Description | m[†] |
|---|---|---|
| CDC Format | | |
| B | Bits | Size of field in bits |
| X | 6-bit alphanumeric characters (Display Code) | Size of field in 6-bit characters |
| A | 12-bit alphanumeric characters (ASCII) | Size of field in 12-bit characters |
| I | Integer (60-bit) | -- |
| U | Unnormalized floating-point (60-bit) | -- |
| E | Normalized floating-point (60-bit) | -- |
| D | Double-precision floating-point (120-bit) | -- |
| S | Numeric, signed overpunch (Display Code) | Size of field in 6-bit characters |
| N | Numeric, unsigned (Display Code) | Size of field in 6-bit characters |
| Z | Numeric, leading zeros suppressed (Display Code) | Size of field in 6-bit characters |

[†]-- indicates that m must be omitted.

[††]IBM 360/370 ASCII and EBCDIC 8-bit sequential tapes can include all these data items.

| | |
|---|---|
| i T m<br>20 X 5 | describes a character field 5 bytes long starting at byte 20 of the record. |
| i T<br>11 E | describes a (60-bit) floating-point word starting at byte 11 of the record. |
| i /w Tm<br>16/4B2 | describes a 2-bit binary field starting at the fourth bit position of byte 16. |
| i T m<br>+5 X 10 | describes a character item 10 bytes (characters) long, starting at the byte position 5 bytes forward from the current byte position. |

Figure 2-3. Examples of Item Descriptors

Additional examples of item descriptors are shown in table 2-2.

## SEARCH DESCRIPTORS

The use of item descriptors presumes that the user knows the exact size and location of all data items within a record. Frequently, however, this is not the case, particularly with regard to variable length records. When variable length fields are referenced, the iTm method of item description must be modified, since exact values of the initial character position and measured length of the item are unknown. A search descriptor can be substituted for an unknown i or m or both.

The search descriptor format is as follows:

oln

o is a byte index denoting the initial byte position (origin) of the data field; o can have the following two interpretations:

absolute index

o denotes a byte position relative to the beginning of the record and must be written as an unsigned decimal integer. The initial byte is numbered 1.

relative index

o denotes a byte position relative to the current byte and must be written as a signed decimal integer. The sign indicates the direction of the move (+ forward, - backward). The current byte is the one next in sequence to the byte previously defined in a string of decriptors. The initial current byte is the first byte of the record. If o is omitted, the current byte is indicated.

l is a literal that delimits the field. The literal cannot exceed 255 characters.

±n is the ordinal (occurrence) of the delimiter l. A plus sign preceding n indicates that the literal is part of the field and is to be moved with it. A minus sign preceding the value n indicates that the literal is not part of the field. Omission of the sign is the same as +n.

TABLE 2-2. EXAMPLES OF SEARCH AND ITEM DESCRIPTORS

| Descriptor Format | Example | Description |
|---|---|---|
| i T m | iTm<br>1XALL | Starting at byte 1, entire data field is moved. |
| | | Starting at byte 3, field is as follows: |
| | iTm<br>3X4 | 4 characters (alphanumeric) |
| | 3B4 | 4-bit string |
| | 3I | 60-bit integer |
| | 3E | 60-bit normalized floating-point |
| | 3D | 120-bit double-precision floating-point |
| | 3U | 60-bit unnormalized floating-point |
| | 3S4 | 4-character signed numeric |
| | 3N4 | 4-character numeric (leading zeros) |
| | 3Z4 | 4-character numeric (leading zeros changed to blanks) |
| i/w T m | i/wTm<br>2/5B4 | Byte 2, bit 5, 4 bits, bits numbered left to right starting with 1. |
| oln T m | o l nTm<br>3*,*2X4 | 4-character field starting with the second occurence of a comma from byte 3; comma is included. |
| | o l nTm<br>21*,*-3X4 | 4-character field following the third comma that appears after byte 21. |
| | o l nTm<br>2$/$-1B6 | 6-bit field starting after the first occurrence of a / after byte 2. |
| T m | Tm<br>X4 | 4 alphanumeric characters starting with the next character. |
| i T oln | iTo l n<br>6X6*$ $*-1 | Alphanumeric field beginning with byte 6 and ending with the byte immediately preceding the literal $$. |
| | iTo l n<br>6X10*$ $*1 | Alphanumeric field beginning with byte 6 and terminating before the first occurrence of the literal $$ occurring after byte 10. $$ is included in the field. |
| oln T oln | o l nT o l n<br>3*$*-1X15*.*1 | Alphanumeric field beginning after the first $ that appears after byte 3 and ending with the first period character (.) appearing after byte 15. The terminating period character (.) is included. |

When FORM encounters a search descriptor, FORM searches for the nth occurrence of the literal l, beginning the search at the byte designated by o. When a search descriptor is substituted for i in iTm, the beginning delimiter of the data field is indicated; when a search descriptor is substituted for m, the terminating delimiter is indicated. Examples of search descriptors are shown in figure 2-4.

Additional examples of search and item descriptors are shown in table 2-2.

## SELECTOR EXPRESSIONS

Selector eoressions are used by the REF, QAL, and CON directives to specify conditions under which records are to be selected for manipulation by these directives.

A selector expression tests one attribute of one data field. If the result of the test is true, the associated ooerations are performed. If the result is false, the associated operations are ignored.

The format for selector expressions is shown in figure 2-5.

| | |
|---|---|
| o I nTm | |
| 16$/$2X6 | Search for the 6-character field beginning with the second / occurring from byte 16 of the record. The / is considered part of the field. |
| o I nTO I n | |
| 20*$*-1X20*$*-2 | Starting at byte 20, search for a character string bounded by the first and second occurrence of a $; the $ is not part of the field. |
| o I nALL | |
| 10$*$1ALL | Search for the first * occurring after byte 10 of the record. The entire data field after the first * is moved. |

Figure 2-4. Examples of Search Descriptors

| | | |
|---|---|---|
| item-descriptor-1 | relation | item-descriptor-2 |
| item-descriptor-1 | relation | {string literal / numeric literal†} |

†Valid in CON directive only.

Figure 2-5. Selector Expression Formats

Blanks are not significant and can be used to improve readability. The relational operators for which data fields in a selector expression can be compared are as follows:

| | |
|---|---|
| LE | less than or equal to |
| LT | less than |
| EQ | equal to |
| NE | not equal to |
| GT | greater than |
| GE | greater than or equal to |

Before the operands in a selector expression can be compared, both operands must be reduced to a common mode. FORM performs the necessary conversions to either a character string or a numeric value according to the procedures outlined in tables 2-3 and 2-4. For example:

| | |
|---|---|
| X3 EQ $123$ | Comparison is in string literal mode. |
| X3 EQ 123 | Comparison is in numeric literal mode (valid in CON directive only). |
| N3 EQ $123$ | Comparison is in numeric mode. |
| E EQ $123.0$ | Comparison is in numeric mode. |

TABLE 2-3. IBM FORMAT COMPARISON MODES FOR SELECTOR EXPRESSIONS

| Item-Descriptor-1 Data Type | Item-Descriptor-2 Data Type | | | |
|---|---|---|---|---|
| | String Literal | Numeric† Literal | B,X | H,W,G,F, L,E,P,S |
| B,X | String mode | Numeric mode | String mode | Numeric mode |
| H,W,G,F, L,E,P,S | Numeric mode | Numeric mode | Numeric mode | Numeric mode |

†Valid in CON directive only.

TABLE 2-4. CDC FORMAT COMPARISON MODES FOR SELECTOR EXPRESSIONS

| Item-Descriptor-1 Data Type | Item-Descriptor-2 Data Type | | | |
|---|---|---|---|---|
| | String Literal | Numeric† Literal | B,X,A | I,E,U,D, S,N,Z |
| B,X,A | String mode | Numeric mode | String mode | Numeric mode |
| I,E,U,D, S,N,Z | Numeric mode | Numeric mode | Numeric mode | Numeric mode |

†Valid in CON directive only.

Numeric quantities are kept to an accuracy of at least 96 bits. Uppercase and lowercase alphabetic characters are mapped to a single uppercase character set. In appendix A, tables A-5 and A-6 provide the translations for display code, EBCDIC, and ASCII characters that occur during mapping.

Shorter strings are treated as if extended on the right with blanks so that both strings are the same length. (Exception: a string derived from a bit string is extended with zeros.)

Some examples of selector expressions are shown in table 2-5.

TABLE 2-5. EXAMPLES OF SELECTOR EXPRESSIONS

| Example | Explanation |
|---|---|
| X3 EQ $ABC$ | A string of 3 characters is compared with the literal ABC. |
| 10X6 NE $ABCDEF$ | The condition is satisfied if the string of characters starting at byte 10 of the record is not equal to the string ABCDEF. |
| L LT *-4.67E+02* | The long floating-point field beginning at the current byte of the record is compared with the numeric value -467.0. |
| 6X1 EQ 10X1 | The condition is satisfied if byte 6 of the record is equal to byte 10. |
| 4I EQ 14I | The condition is satisfied if the integer starting at byte position 4 is equal to the integer starting byte position 14. |
| 6/4B2 EQ $10$ | The condition is satisfied if the 2-bit field starting with bit 4 of byte 6 is equal to the literal value 10. |

# DIRECTIVES 3

Directives allow the user to communicate with FORM. Through the FORM directives, the user can specify input and output file characteristics, record selection criteria, reformatting and conversion to be performed, and printed page options.

## ORDERING OF DIRECTIVES

FORM directives are always executed in the order indicated in figure 1-3 of section 1. The directives can be specified in any order, within the following constraints:

- The INP or OUT directive for a file must precede all other directives for that file in the directive sequence.
- When logical file names are equated, the directive containing the parameter list must precede the directive equating the file names.
- Files are written in the order in which they are specified in the OUT directives.

## INPUT DIRECTIVE (INP)

The INP directive declares a source of input records to be processed by FORM. A single source of input records is required for a FORM run. Input records can be supplied by owncode, by a file specified in the INP parameter of the FORM control statement, or by a file specified in the INP directive. The INP directive must be specified when this directive is a source of input records. If the user specifies an alternate input source (either owncode or the INP parameter of the FORM control statement) the INP directive must be omitted. The format of the INP directive is shown in figure 3-1.

Examples of the INP directive are illustrated in figure 3-2.

If a file contains variable length records, appendix K should be consulted for determination of input record length.

```
INP(lfn,MAX=n,POS=±n,REW=r,HRL=ept,DCA=ept,LX=ept,DX=ept,EX=ept,RFM=r,BLK=b,LRL=n,IRL=len,COD=c,CX=ept)
```

With the exception of lfn, all parmeters are optional and order independent.

| | | |
|---|---|---|
| lfn | Logical file name of file containing input records. | |
| MAX=n | Maximum number of records to be processed. Default is no limit; all records in the file are processed. Value can be from 1 through 16777215 ($2^{24}$ - 1). Both n and n/R refer to record count. | |
| | n | Record count |
| | n/R | Record count |
| | n/P | Partition count† |
| | n/S | Section count†† |
| | 1/P | Processing of exactly one file |
| POS=±n | Number of partitions/sections/records to skip from present position prior to processing. Value can be from ± 1 to ± 2047. Both n and n/P refer to partitions.††† | |
| | n, n/P, n/S, n/R<br>+n, +n/P, +n/S, +n/R | Skips n partitions/sections/records in a forward direction |
| | -n, -n/P, -n/S, -n/R | Skips n partitions/sections/records in a backward direction |
| REW=r | Action to be taken on file at end of run. | |
| | N | No rewind (default) |
| | R | Rewind only |
| | U | Rewind and return (disk files) or unload (tapes) |

Figure 3-1. INP Directive Format (Sheet 1 of 2)

The following parameters specify user-supplied owncode routines that are loaded from the file specified in the OWN parameter of the FORM control statement.

| | |
|---|---|
| HRL=ept | Entry point name of routine to perform key hashing for an input file with direct access organization. |
| DCA=ept | Entry point name or standard routine ordinal ($1 \le n \le 64$) of routine to perform record decompression/decryption for indexed sequential, direct access, and actual key files. |
| LX=ept | Entry point name of routine to perform label processing for tapes. |
| DX=ept | Entry point name of routine to receive control whenever end-of-partition, end-of-section, or end-of-data is encountered on the input file. |
| EX=ept | Entry point name of routine to receive control when an error is detected on the input file. |

The HRL, DCA, LX, DX, and EX routines are called directly by CRM at the appropriate times.

The following parameters are used only when the input file is an IBM format file being converted to CDC format through the CON directive. The presence of any of these parameters causes the input file to be processed as an IBM format file.

RFM=r IBM record/blocking format.

| | |
|---|---|
| F | Variable format (default) |
| V | Variable format |
| U | Unspecified format |
| FB | Fixed blocked format |
| VB | Variable blocked format |
| VSB | Variable spanned blocked format |

IBM record formats are described in appendix D.

| | |
|---|---|
| BLK=b | Block size measured in 8-bit bytes; value can be from 1 through 32767. If the block size is greater than 3840, the LABEL statement for the file must specify an L tape. |
| LRL=n | Maximum record length measured in 8-bit bytes. Value can be from 1 through 32760. |
| IRL=len | Internal length of record after conversion, in 6-bit bytes. Default is LRL. For IBM tapes containing variable length records, the default is $\frac{4*LRL}{3}$. |

COD=c IBM data code.

| | |
|---|---|
| A | ASCII |
| C, E | EBCDIC (default) |

CX=ept Entry point name of routine to be entered whenever a conversion error occurs while processing the file.

RFM, BLK, and LRL are required for processing IBM tapes; COD, IRL, and CX are optional.

† Partition refers to an internally recorded end-of-file mark in the data. (See CYBER Record Manager Basic Access Methods reference manual.)

†† Section refers to an internally recorded end-of-record mark.

††† POS=n is implemented by means of a Record Manager SKIPdF macro, where d is the direction of the skip. This means that POS does not function properly for files with RT=W.

Figure 3-1. INP Directive Format (Sheet 2 of 2)

Example 1.

```
        INP(INFILE)
```

Declare INFILE as the source of input records.

Example 2.

```
        INP(TAPE1,POS=+2,REW=U)
```

Declare TAPE1 as the input source, skip two partitions prior to processing, rewind and return the file at end of run.

Figure 3-2. Examples of INP Directive

## OUTPUT DIRECTIVE (OUT)

The OUT directive declares an output file to be generated by FORM. This directive can be omitted if an output file is declared in the OUT parameter of the FORM control statement. Up to twenty output files can be processed in a single FORM run; each output file requires an OUT directive. Files are written in the order in which they are declared in the OUT directives.

The format of the OUT directive is shown in figure 3-3.

The KEY parameter is used to describe the key field for indexed sequential and direct access output files. This parameter is overridden by the KA parameter of the FILE control statement.

If the input file is sequential, the KEY parameter is required only if the output record key is not contained within the record (nonembedded). The user should specify KEY=+k to designate the field at location k as the record key (k is an item descriptor of the format iTm).

If the input file is indexed sequential, extended indexed sequential actual key, or direct access, the KEY parameter is required only if the output record key is nonembedded and the output key is different from the input key.

- The user should specify +k if the key is to be retained in the output record at location k.
- The user should specify -k if the key at location k of the input record is to be extracted from the record prior to output. The specification of -k decreases the size of each output record.

The user should omit the KEY parameter if the key location does not change or if neither input records nor output records contain keys.

For actual key output files CYBER Record Manager (CRM) returns the key after each record is written. Files are written by FORM in the same order in which they are declared in the OUT directives; therefore an OUT directive for an actual key output file must precede an OUT directive for a file that is to receive the key via a REF directive.

When creating an actual key file, CRM assigns the key to the output record. The key is saved in the item KEYA where it can be referenced by a REF directive for a subsequent output file. The user cannot supply a key for actual key files when using FORM.

Information concerning the use of the OUT directive and variable length records is in appendix K.

Figure 3-4 illustrates some examples of the OUT directive.

## IBM 360/370 CONVERSION DIRECTIVE (CON)

The CON directive is used to perform conversions between IBM 360/370 8-bit sequential 7- or 9-track tapes and CDC internal format files. An IBM file can be manipulated by FORM directives only when a CON directive is also specified to convert the input file data to CDC format.

CON is not always needed for conversions between IBM tapes and CDC files. Section 4, under IBM Tape Files, gives examples where simple conversions are performed without CON or other directives.

IBM tape files can contain the following:

- EBCDIC character string data
- Bit stream data
- Arithmetic data in IBM form
- ASCII character string data

NOTE

ASCII does not refer to NOS ASCII 6/12 data; ASCII refers to ASCII 8/8 data.

The EBCDIC and ASCII codes are listed in appendix A. IBM tape files can be in any one of the record/block format types described in appendix F.

The internal CDC format used in conjunction with the CON directive can contain the following data types:

- 6-bit display code character string data
- Bit stream data
- Arithmetic data in CDC format

The 6-bit data format and CDC format arithmetic data are defined in appendix F.

Both IBM tapes and CDC files must be described to CRM in terms of CRM parameters; CRM descriptions for CDC files are always needed except when the file has W type records in I type blocks. File descriptions are discussed in section 4.

The format of the CON directive is shown in figure 3-5.

Conversion strings are used as input to the CON directive. The conversion string specifies how data items in a record are to be translated. Through the conversion string, the user can specify conversion between any IBM data format and CDC internal data format listed in table 2-1 of section 2.

Except when used in literal string parameters, blanks are ignored and can be used to improve readability.

OUT(lfn,MAX=n,REW=r,KEY=±k,BGD=b,NOSEC,DCT=ept,CPA=ept,HRL=ept,LX=ept,EX=ept,RX=ept,RFM=r,BLK=len,LRL=len,
IRL=len,COD=c,CX=ept)

With the exception of lfn, all parameters are optional and order independent.

| | |
|---|---|
| lfn | Logical file name; can also be written as lfn/R, in which case the file is rewound before use. |
| MAX=n | Maximum number of records to be processed. Default is no limit. Value can be from 1 through 16777215 ($2^{24}$ - 1). Both n and n/R refer to record count. |

| | |
|---|---|
| n | Record count |
| n/R | Record count |
| n/P | Partition count† |
| n/S | Section count†† |
| 1/P | Processing of exactly one file |

REW=r Action to be taken on file at end of run.

| | |
|---|---|
| N | No rewind (default) |
| R | Rewind only |
| U | Rewind and return (disk files) or unload (tapes) |

| | |
|---|---|
| KEY=±k | Location of key for an indexed sequential, actual key, or direct access file being written; k is an item descriptor of the form iTm. |
| BGD=b | Record background of the output record prior to execution of the REF directive. Causes any record fields not described by a REF directive to contain one of the following: |

| | |
|---|---|
| X | Blank (octal 55) |
| Z | Display code zero (octal 33) |
| B | Binary zero |
| C | Same as input record (default) |

NOSEC Causes section boundaries in input file to be ignored. Applicable only to sequential output files created from sequential input files.

The following parameters specify routines that are loaded from the file specified by the OWN parameter of the FORM control statement.

| | |
|---|---|
| DCT=ept | Entry point name of display-collating conversion table for indexed sequential files. |
| CPA=ept | Entry point name or standard ordinal ($1 \leq n \leq 64$) of routine that is to perform record compression/encryption for indexed sequential, direct access, and actual key files. |
| HRL=ept | Entry point name of routine to perform key hashing for a direct access output file. |
| LX=ept | Entry point name of routine to perform label processing for tapes. |
| EX=ept | Entry point name of routine to receive control when an error is detected on the output file. |
| RX=ept | Entry point name of routine to be entered whenever an error is detected while reformatting the record for output. |

The following parameters are used only when the output file is an IBM format file being converted from CDC format. The presence of any of these parameters causes the output file to be processed as an IBM format file.

RFM=r IBM record/blocking format.

| | |
|---|---|
| F | Fixed format |
| V | Variable format |
| U | Unspecified format |
| FB | Fixed blocked format |

Figure 3-3. OUT Directive Format (Sheet 1 of 2)

| | | |
|---|---|---|
| | VB | Variable blocked format |
| | VSB | Variable blocked spanned format |
| BLK=len | Block size measured in 8-bit bytes; value can be from 1 through 32767. If block size is greater than 3840, the LABEL statement for the file must specify an L tape. | |
| LRL=len | Maximum record length measured in 8-bit bytes. Value can be from 1 through 32767. | |
| IRL=lfn | Internal length of record, in 6-bit bytes, before conversion. Default is LRL. | |
| COD=c | IBM data code. | |
| | A | ASCII |
| | C } E } | EBCDIC (default) |
| CX=ept | Entry point name of routine to be entered whenever a conversion error is encountered while processing the file. | |

RFM, BLK, and LRL are required for processing IBM tapes; IRL, COD, and CX are optional.

---

†Partition refers to an internally recorded end-of-file mark in the data. (See CYBER Record Manager Basic Access Methods reference manual.)

††Section refers to an internally recorded end-of-record mark.

Figure 3-3. OUT Directive Format (Sheet 2 of 2)

Example 1.

```
OUT(AFILE,MAX=50)
OUT(PUNCH)
OUT(OUT21,MAX=100,REW=U,EX=ERRMS)
```

Three output files are defined: AFILE, with a limit of 50 records, is to be rewound at end of run. PUNCH is a file with an associated punch disposition. OUT21, with a limit of 100 records, is to be rewound and unloaded at end of run; the routine ERRMS receives control if a parity error is encountered on OUT21.

Example 2.

```
OUT(OUTFILE,KEY=+1X10)
```

The 10-character field starting at the first character position of the output record is designated as the key field.

Example 3.

```
OUT(SISOUT,KEY=-17X6,RX=RFERR)
```

The key from each record of the indexed sequential output file SISOUT is extracted from character positions 17-22 prior to writing the file; the routine RFERR is entered if a reformatting error is detected.

Figure 3-4. Examples of OUT Directive

A conversion string can contain one or more conversion specifications. A conversion specification consists of an optional selector expression followed by one or more conversion items. A conversion item can consist of a simple item conversion or a conversion string. A simple item conversion causes actual data translation. Since a conversion item can consist of a conversion string, nested conversion strings are legal. Nesting can occur to a maximum depth of seven levels. Nested conversion strings must be enclosed in parentheses.

Multiple conversion specifications are separated by semicolons. A selector expression is separated from its conversion items in a conversion specification by a colon. Multiple conversion items are separated by commas. The scope of a selector expression is a single conversion specification and is terminated by a semicolon; when the selector expression is true, the semicolon causes the rest of the conversion string, up to the matching right parenthesis, to be ignored. Selector expressions are defined in section 2.

A conversion item can comprise part or all of a conversion specification or a conversion string. The most basic form is a conversion item that specifies only one conversion; this form is a simple item conversion.

Conversions specified in the CON directive are executed until either end-of-record in the input file or a Q specification in the directive (described later in this section) is encountered.

Information relating to FORM conversion of variable length records can be found in appendix K.

```
CON(lfn,conversion-string)

    lfn    logical file name of the IBM file.

    conversion-string

           conversion-specification [; conversion-specification] . . .

    conversion-specification

           [selector-expression:] conversion-item [, conversion-item] . . .

    conversion-item

                            { simple-item-conversion }
           [repeat-count]   {                        }
                            { (conversion-string)    }

    simple-item-conversion

           Tm1[Tm2]

    repeat-count

           decimal integer indicating the number of times the conversion item is to be executed.
```

Figure 3-5. CON Directive Format

## BYTE LENGTH

Each record in a file is considered to be a string of bytes, with the byte length determined by the source computer. Bytes within IBM-format files contain 8 bits; bytes within CDC-format files contain 6 bits. In 6-bit or 8-bit bytes, bits within each byte are numbered 1 through 6 or 1 through 8, from left to right.

## FIELD ALIGNMENT

When data conversion is initiated, internal pointers are established for the source and destination record areas, each initially pointing to bit 1 byte 1 of its record string. These bits are the initial field positions.

When a source item is converted to a destination item, these pointers are modified as follows:

- Prior to conversion, if the bit pointer for a byte does not equal 1, it is set to 1, and character position is rounded up to the next byte by adding 1 to the byte pointer. If the destination pointer is so affected, skipped bit positions are filled with binary zeros.

  Exception: No rounding takes place for a type B (bit) source or destination item.

- When conversion is complete, the pointers are updated to point to the bit succeeding the last bit read or written; this is the current field position. When conversion terminates mid-word, the remainder of the word is unchanged.

Alignment never is forced to a boundary more significant than a byte position. If word boundary or other alignment is needed, the proper fill items must be supplied explicitly. Data alignment requirements are given in appendix F.

## CONVERSION ITEMS

Conversion items provide directions for translating data items from a source record to a destination record. The format is one of the following:

n simple-item-conversion

n(conversion-string)

### Repeat Count

The optional repeat count, n, is a decimal integer that indicates the number of times the conversion is to be repeated. Specifying a repeat count is equivalent to writing the conversion item n times, separated by commas. If n is 0, 1, or omitted, the simple item-conversion or conversion-string is executed only once.

### Simple Item Conversion

A simple item conversion specifies how the current source record field is to be translated to the current destination record field. Only a simple item conversion specification causes data to be converted. Other parts of the conversion string provide control information that determines the kind of conversions to be performed.

A simple item conversion must be written in one of the following formats:

Tm1Tm2

Tm1

Tm1 and Tm2 must be valid item descriptors; Tm1 describes a source data field and Tm2 describes a destination data field. Execution of a simple item conversion causes data to be moved from the source field to the destination field and translated to the destination field format. CON treats each data field of a record in sequence; data fields must be described in the order in which they appear in the record, so that FORM can adjust for word or byte boundaries. Therefore, the position specification i is not used in the item descriptors. The length specification m must be omitted for certain data types, as indicated in table 2-1 of section 2.

If Tm2 is omitted, CON uses a default value, as specified in table 3-1.

TABLE 3-1. DEFAULT CONVERSIONS IF Tm2 IS NOT SPECIFIED

| Tm1 | Default Tm2 | Tm1 | Default Tm2 |
|---|---|---|---|
| Tape | Internal | Internal | Tape |
| Bm | Bm | Bm | Bm |
| Xm | Xm | Xm | Xm |
| H | I | Am | Xm |
| W | I | I | W |
| G | I | U | F |
| F | E | E | F |
| L | E | D | L |
| E | D | Sm | Sm |
| Pm | Sn (n=2m) | Nm | Sm |
| Sm | Sm | Zm | Xm |
| Note: Refer to appendix G for conversion rules. | | | |

Translation rules for all possible combinations of data types are outlined in appendix G.

Some examples of simple item conversions are shown in table 3-2.

### Conversion Strings Used as Conversion Items

A conversion string can be used wherever a simple item conversion is allowed. This feature allows specification of alternative conversions at interior positions of a record, such as where a fixed initial record segment is followed by a variable format segment, or where certain alternatives can themselves have alternatives.

Conversion strings can occur as items within conversion strings to a maximum depth of seven levels. Examples of nested conversion strings appear later in this section.

## CONVERSION SPECIFICATIONS

A conversion specification consists of an optional selector expression followed by a list of one or more conversion items. A selector expression uses relational operators to specify conditions under which records are to be selected for manipulation by the CON directive. The selector expression must be true for the associated conversion items to be executed; if the selector expression is false, the conversion items are ignored. If the selector expression is missing, the conversion items are executed. The selector expression is separated from its associated list by a colon; conversion items within the list are separated by commas.

The conversion items of a conversion specification are executed in sequence, left to right. The format of a conversion specification is:

```
    selector-expression:  conversion-item

                    , conversion item ...
```

Elements of a conversion specification are illustrated in figure 3-6.

## CONVERSION STRINGS

A conversion string consists of one or more conversion specifications, separated by semicolons. The format of a conversion string is:

```
    conversion-specification

            ;conversion-specification...
```

When a conversion string is encountered during execution, each conversion specification is tested in turn, from left to right, until one is found with a selector expression that is true. That conversion specification is executed and all specifications to its right are ignored. If none are true, no conversion is performed.

### Conversion String Punctuation

A colon separates a selector expression from its associated conversion items in a conversion specification; multiple conversion items are separated by commas. The scope of a selector expression is a single conversion specification that is terminated by a semicolon. When the selector expression is true, the semicolon causes the rest of the parenthesized conversion string, up to the matching right parenthesis, to be ignored. This feature is of special significance when items are nested in parentheses in a conversion string. When a conversion specification is executed, all remaining parts of the conversion string are ignored.

### Nested Conversion Strings

A conversion string can be used wherever a simple item conversion is allowed in a conversion item. Conversion strings can occur within conversion items to a maximum depth of seven levels.

TABLE 3-2. EXAMPLES OF SIMPLE ITEM CONVERSIONS

| Example | Explanation |
|---|---|
| $\underbrace{\text{X5}}_{\text{Tm1}}\underbrace{\text{X5}}_{\text{Tm2}}$ | Translates five 8-bit characters on an ASCII tape to five 6-bit internal display code characters. |
| $\underbrace{\text{X120}}_{\text{Tm1}}\underbrace{\text{X150}}_{\text{Tm2}}$ | Translates a 120-character field to a 150-character field blank-filled on the right. |
| $\underbrace{\text{X4}}_{\text{Tm1}}\underbrace{\text{H}}_{\text{Tm2}}$ | Translates four display code characters to an IBM half-word integer. |
| $\underbrace{\text{G}}_{\text{Tm1}}\underbrace{\text{U}}_{\text{Tm2}}$ | Translates one IBM 64-bit integer to a 60-bit word containing a CDC unnormalized floating-point number. |
| $\underbrace{\text{B60}}_{\text{Tm1}}\underbrace{\text{I}}_{\text{Tm2}}$ | Translates a 60-bit stream on tape to an internal 60-bit integer field. |
| $\underbrace{\text{60}}_{\text{repeat count}}\underbrace{\text{B6}}_{\text{Tm1}}\underbrace{\text{B10}}_{\text{Tm2}}$ | Moves 60 consecutive internal 6-bit fields to consecutive 10-bit fields on tape. Each 10-bit destination field will contain 4 bits of binary zero fill on the right. |
| $\underbrace{\text{4}}_{\text{repeat count}}\underbrace{\text{X10}}_{\text{Tm1}}\underbrace{\text{P6}}_{\text{Tm2}}$ | Translates 4 consecutive 10-character display code fields to 4 consecutive 6-byte packed decimal fields. |
| $\underbrace{\text{6}}_{\text{repeat count}}\underbrace{\text{E}}_{\text{Tm1}}\underbrace{\text{F}}_{\text{Tm2}}$ | Translates 6 CDC floating-point numbers to 6 IBM floating-point numbers. |
| $\underbrace{\text{B60}}_{\text{Tm1}}$ | Moves a 60-bit field to a 60-bit field. |
| $\underbrace{\text{U}}_{\text{Tm1}}$ | Translates a 60-bit internal CDC unnormalized floating-point number to an IBM 32-bit floating-point field. |
| $\underbrace{\text{X5}}_{\text{Tm1}}$ | Translates 5 characters to 5 characters. |
| $\underbrace{\text{40}}_{\text{repeat count}}\underbrace{\text{P5}}_{\text{Tm1}}$ | Translates 40 consecutive IBM packed decimal fields to internal CDC 10-digit signed overpunch numeric 6-bit display code fields. |

$$\underbrace{\underbrace{\text{X80X80}}_{\text{conversion-item-1}}}_{\text{conversion-specification}} \qquad \underbrace{\underbrace{\text{X5X10}}_{\text{conversion-item-1}},\ \underbrace{\text{X10X5}}_{\text{conversion-item-2}}}_{\text{conversion-specification}}$$

$$\underbrace{\underbrace{\text{P5 EQ -456}}_{\text{selector expression}}\ :\ \underbrace{\text{P5S9}}_{\text{conversion-item-1}}}_{\text{conversion-specification}} \qquad \underbrace{\underbrace{\text{P3 GT 0}}_{\text{selector expression}}\ :\ \underbrace{\text{6P3Z10}}_{\text{conversion-item-1}},\ \underbrace{\text{60B8B6}}_{\text{conversion-item-2}}}_{\text{conversion-specification}}$$

Figure 3-6. Elements of a Conversion Specification

This feature permits specification of alternate conversions at various positions within a record, such as where a fixed initial record is followed by a variable format segment, or where certain alternatives can themselves have alternatives.

Figure 3-7 illustrates elements of a conversion string.

### Q Specification

The Q specification terminates all conversion for a record; as a result, the output record contains only converted data. Q can be inserted wherever a conversion item is valid.

The Q specification is useful when performing alternative conversions within a conversion string. Assume the conversion string:

```
10x4 EQ $STOP$:Q; X20X20
```

FORM stops converting if the 4-character field starting at byte 10 of the input record contains the string STOP; otherwise, 20 characters are copied from the input record to the output record.

Table 3-3 contains additional examples of conversion strings.

An example of the CON directive is illustrated in figure 3-8.

## SPECIAL CONVERSION RULES

Rules pertaining to all possible conversions appear in appendix G. Some general principles are described in the following paragraphs.

### Bit to String

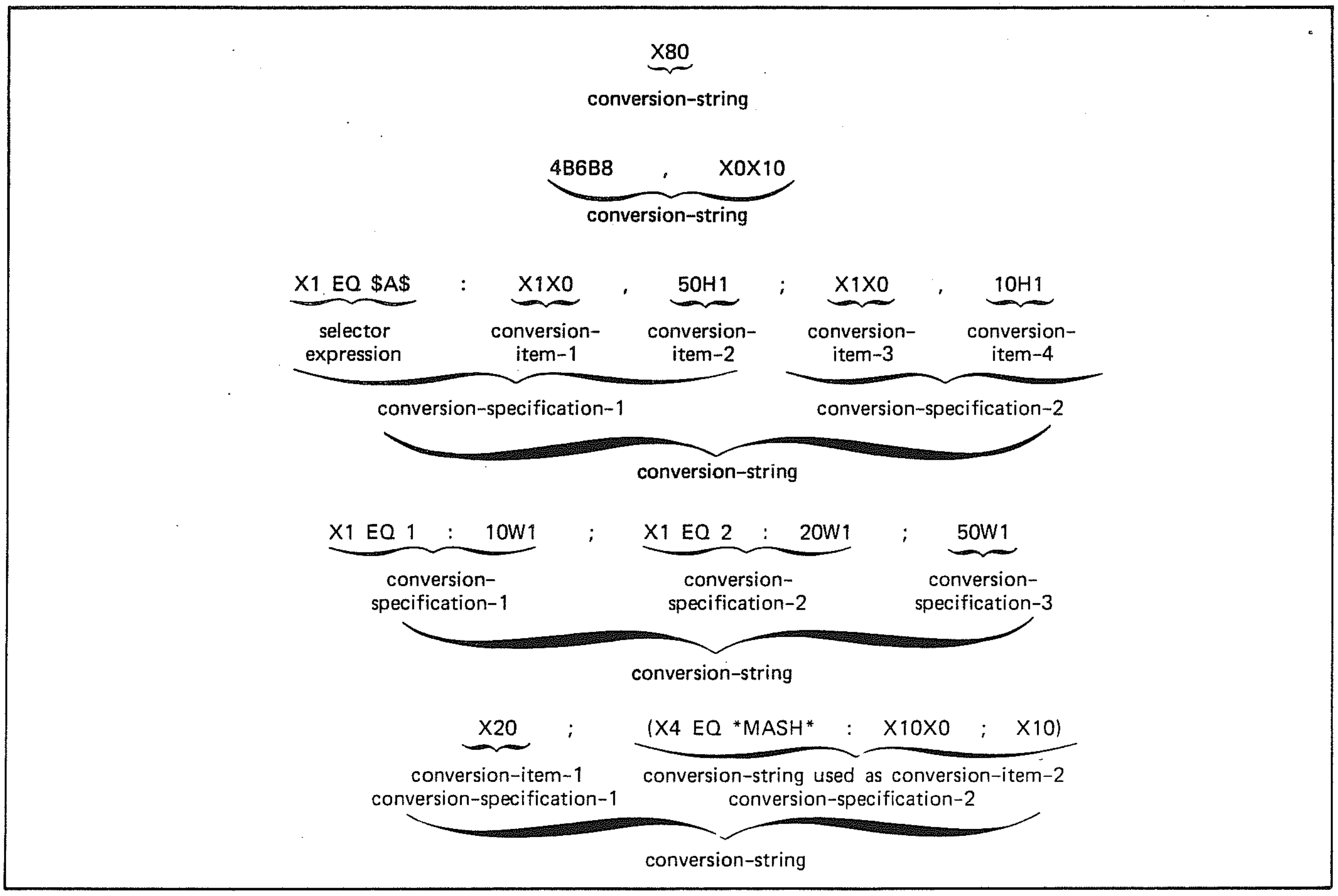
When a bit field is converted to a character string, the number of characters in the character string is the same as the number of bits in the original string. Each bit becomes the character 0 or the character 1. Conversion is from left to right.

### Bit to Numeric

When a bit field is converted to a numeric value, no changes occur in the field; rather, the bit field is interpreted as a positive integer. The binary point is assumed to follow the rightmost bit of the field.

### String to Numeric

Literal strings or X items to be converted to numeric items must conform to the rules described in section 2 under Numeric Literals. An error results if the source string is not in this format. Spaces in the string are ignored.



X80
conversion-string
4B6B8 , X0X10
conversion-string
X1 EQ $A$ : X1X0 , 50H1 ; X1X0 , 10H1
selector expression
conversion-item-1
conversion-item-2
conversion-item-3
conversion-item-4
conversion-specification-1
conversion-specification-2
conversion-string
X1 EQ 1 : 10W1 ; X1 EQ 2 : 20W1 ; 50W1
conversion-specification-1
conversion-specification-2
conversion-specification-3
conversion-string
X20 ; (X4 EQ *MASH* : X10X0 ; X10)
conversion-item-1
conversion-specification-1
conversion-string used as conversion-item-2
conversion-specification-2
conversion-string

Figure 3-7. Elements of a Conversion String

TABLE 3-3. CONVERSION STRING EXAMPLES†

| Example | Explanation |
|---|---|
| X5X6 | Moves the 5 characters ABCDE from tape to the 6-character internal field. The rightmost character is a blank. |
| X5 | Converts five 8-bit EBCDIC characters to five 6-bit display code characters. |
| X1 EQ *A* : X5X6 | Because the first character on the tape is A, the first 5 characters ABCDE are moved from the 8-bit EBCDIC tape record to the 6-bit display code internal format record. The rightmost character of the internal field is a blank. |
| X1 EQ *1* : X5X6 | Since the first character on the tape is not 1, the conversion item is ignored. |
| X1 EQ *1* : X5X6 ; X1 EQ *A* : X3 | Since the first selector expression is false, the X5X6 conversion item is ignored. The second selector expression is true; therefore, the characters ABC are moved. |
| X3,(X1 EQ $D$ : Q ; X3),X5X6 | The first 3 characters ABC are moved. Since the fourth character is D, the Q conversion item terminates conversion string execution at that point. The selector expression refers to the current byte of the source record which contains the character D. If the fourth character was not D, 3 more characters would have been moved. In either case, the next 5 characters are moved to a 6-character field. |
| 3X1 EQ *C* : X5X6 | Since the third character in the source record is C, the first 5 characters ABCDE are moved from tape to the 6-character internal field. |
| +3X1 EQ *D* : X5 | The selector expression is true, as D is the fourth character in the record; therefore, the first five characters ABCDE are moved. |
| X3,(-3X1 EQ *A* : X23X24) | The first 3 characters on tape record ABC are moved before the first character on the record is tested. Since the first character is A, the selector expression is true; and the remaining 23 characters in the alphabetic sequence are moved to a 24-character field. |
| X26,(-26X26 EQ X26 : 2X26 | Moves one entire alphabet group from the tape to the internal record, then compares the next 26 characters with the first 26. Since they match, two more entire alphabet groups are moved to the internal record. |

†These examples assume a 9-track tape containing only 8-bit EBCDIC characters as the input source to be converted to internal CDC records. The tape contains multiple repetitions of the alphabet in uppercase only.

### Character Skipping and Blank/Zero Fill

To specify bit or character skipping, the source field length must be specified as greater than the destination length in the conversion items. For example:

- B10B0 causes 10 bits to be skipped.
- X5X0 causes 5 characters to be skipped.
- X10X5 causes 5 characters to be transferred and the next 5 to be skipped.

To insert blanks or zeros in the destination record, destination field size must be greater than the source field size in the conversion item. For example:

- X0X5 causes 5 EBCDIC spaces to be placed in the destination field.
- B0B60 causes 60 bits of zero to be placed in the destination field.
- X10X100 causes 10 characters to be transferred to the destination field with 90 blanks on the right.

### Floating-Point to Integer

Conversion is possible for the formats listed in table 2-1 of section 2. The restrictions for each conversion are noted in table 2-5 of section 2. For example, conversion can be performed between the IBM floating-point formats of 32, 64, or 128 bits and the CDC floating and double-precision floating formats of 60 or 120 bits. Conversions to single-precision floating-point are rounded to 48-bit precision; conversions to double-precision are rounded to 96-bit precision.

```
INP(AFILE,RFM=F,BLK=3000,LRL=1000,COD=A,CX=CNERR)
CON(AFILE,25B2 EQ $01$:  X24X24,B4,P8S16;25B2 EQ $10$:  3P8I,B4,GI)
```

The IBM file AFILE is converted to CDC format. AFILE has the following characteristics:

Input records are of fixed length
Maximum block length is 3000 8-bit characters
Record length is 1000 8-bit characters
Records contain ASCII coded data

Control transfers to entry point CNERR if a conversion error occurs.

The conversion string performs the following conversions:

25B2 EQ $01$ If the bit string of length 2 beginning at byte 25 has the value 01, only the following conversion items are executed:

| | |
|---|---|
| X24X24 | translate 24 8-bit characters to 24 6-bit display code characters |
| B4 | copy 4-bit field to 4-bit field |
| P8S16 | translate 8-byte packed decimal number into 16 numeric characters (sign is overpunched in low-order digit) |

25B2 EQ $10$ If the 2-bit string beginning at byte 25 is equal to the binary string 10, only the following conversion items are executed:

| | |
|---|---|
| 3P8I | translate 3 packed decimal numbers into 60-bit integer format |
| B4 | copy 4-bit field to 4-bit field |
| GI | translate a full-word fixed-point binary number into a 60-bit binary integer |

If the bit string does not have either of the values 10 or 01, no conversion is performed and the record is not sent to the output file.

Figure 3-8. Example of CON Directive

Conversion from the internal record to an external IBM floating-point format yields a minimum precision of 21 bits for floating-point, 53 bits for long floating-point, and 109 bits for extended-precision floating-point.

### Binary Data

Any data can be considered binary and manipulated on a bit-by-bit basis. Bits can be copied in strings, or they can be copied selectively by skipping bits or replacing bit groups in a string with zeros.

Bit strings can be converted to any other valid format within the limitations expressed in appendix G.

Item descriptors in selector expressions can address any bit in a character or bit string. For example:

- 2/5B1 references the fifth bit of the second byte in the source record.
- 2/5B1 NE $1$: X1X0; X1X1 translates the second character to the destination record only if its fifth bit is 1. If the fifth bit is not 1, the character is skipped. If the fifth bit is 1, the character is converted. When a record is referenced using data type B, references must be based on 6-bit bytes for CDC records and 8-bit bytes for IBM records.

## QUALIFICATION DIRECTIVE (QAL)

The QAL directive specifies the criteria by which records from the input file are selected to be sent to an output file. The selection criteria are expressed in the form of a qualification string that consists of one or more selector expressions joined by logical operators. The format of the QAL directive is shown in figure 3-9.

Numeric literals must be enclosed by $ or * delimiter pairs.

QAL(lfn,qualification-string)

lfn logical file name of the output file to receive records satisfying the criteria specified in the qualification string.

qualification-string

$$[\text{NOT}] \begin{Bmatrix} \text{selector-expression} \\ \text{qualification-string} \end{Bmatrix} \left[ \begin{Bmatrix} \text{AND} \\ \text{OR} \end{Bmatrix} [\text{NOT}] \begin{Bmatrix} \text{selector-expression} \\ \text{qualification-string} \end{Bmatrix} \right] \ldots$$

Figure 3-9. QAL Directive Format

A qualification string can be substituted where a selector expression is valid. Qualification strings can be nested to a maximum depth of seven levels. Blanks are ignored and can be used freely to improve readability.

The QAL function is executed in sequence preceding the REF, CON, PAG, and SEQ functions. Records from the input file satisfying the criteria established by the qualification string are selected for further processing. Subsequent directives apply only to those records that have been selected by QAL; records not meeting the criteria receive no further processing and are not sent to the output file.

Logical operations are executed in the following order: NOT, AND, OR. However, this precedence can be altered by the use of parentheses. Expressions within the innermost level of parentheses are evaluated first.

All CDC internal data types can be processed by the QAL directive. QAL performs the necessary conversion of data types prior to comparison of values. Conversion and comparison are performed according to procedures outlined in table 2-3 of section 2.

Search descriptors in the oln format are permitted in iTm specifications used in qualification strings.

Examples of the QAL directive are shown in figure 3-10.

Example 1.

```
QAL(FILEA,5E EQ $256.1$)
```

Selects and copies to FILEA those records in which the 60-bit floating-point number starting at byte 5 has a value of 256.1.

Example 2.

```
QAL(FILEB,11E GT 32X 10)
```

Selects and copies to FILEB those records in which the 60-bit floating-point number starting at byte 11 is greater in algebraic value than the 10-character string starting at character position 32. Conversion is performed in numeric mode.

Example 3.

```
QAL(FILEA,10X EQ $A$ AND 20X EQ $A$)
```

Selects for output to FILEA records that contain the character A in the tenth and twentieth bytes.

Example 4.

```
QAL(OUTFILE,32X EQ $A$ OR 32X EQ $B$ AND 36X EQ $C$)
```

The AND operation is executed before the OR operation. Hence, this directive selects records for output to OUTFILE if byte 32 contains the character A or if byte 32 contains the character B and byte 36 contains the character C.

Example 5.

```
QAL(MYFILE,(32X EQ $A$ OR 32X EQ $B$) AND 36X EQ $C$)
```

In this example, precedence has been established by the use of parentheses. Records are selected for output to MYFILE if byte 32 contains an A or B character, and byte 36 contains the character C.

Example 6.

```
QAL(FILEA,11X4 NE 20$,$-2X20$,$-3)
```

This example illustrates the use of the oln search descriptor. QAL selects for output to FILEA those input records in which the 4 character alphanumeric field starting at byte 11 is not equal to the variable length character field bounded by the second and third commas occurring after byte 20. The commas are not included in the comparison. Search descriptors are substituted for both i and m in iTm.

Figure 3-10. Examples of QAL Directive

## REFORMATTING DIRECTIVE (REF)

The REF directive controls arrangement and format of data fields in the output records. The REF directive can be used to insert literals at any position within records, to place keys in output records, and to convert data fields from one data type to another. REF supports all CDC internal data types, and can perform conversions between any of those data types. When overlapping fields are specified, the last specification establishes the final format.

A separate REF directive is required for each output file to be reformatted. Only one REF can be used for an output file. The format of the REF directive is shown in figure 3-11.

A reformat string can contain one or more reformat specifications. A reformat specification consists of an optional selector expression followed by a list of one or more reformat items. A reformat item consists of a simple reformat or a reformat string prefixed by an optional repeat count. Simple reformats are composed of item descriptors that describe the input and output data fields. A reformat item can comprise all or part of a reformat string. Since a reformat item can contain a reformat string, nesting is legal and can occur to a maximum depth of seven levels. With the exception of literal strings, blanks within reformat stings are ignored and can be used to improve readability. Appendix K has information about reformatting variable length records.

### REFORMAT ITEMS

Reformat items provide directions for reformatting data items and moving them from an input record to an output record. A reformat item can consist of one of the three following formats:

simple-reformat

n(simple-reformat)

n(reformat-string)

The optional decimal repeat count (n) indicates the number of times the reformat item is to be repeated. Specification of a repeat count is equivalent to writing the item n times, separated by commas.

```
REF(lfn,reformat-string)

    lfn       logical file name of output file.

    reformat-string:

              reformat-specification [;reformat-specification] . . .

    reformat-specification:

              [selector-expression:] reformat-item [, reformat-item] . . .

    reformat-item:

             { simple-reformat                                        }
             { repeat-count     { (simple-reformat) }                 }
             {                  { (reformat-string) }                 }

    simple-reformat:

             { iTm=iTm                 }
             { iTm= { $literal$ }      }
             {      { *literal* }      }
             { iTm=KEY                 }
             { iTm=KEYA                }
             { iTm                     }

    repeat-count:

              decimal integer specifying the number of times the reformat item is to be repeated.
```

Figure 3-11. REF Directive Format

## Simple Reformats

A simple reformat causes a data field in an input record to be reformatted and moved to an output record. A simple reformat is written in one of the following forms:

iTm=iTm

iTm= [ $literal$ or *literal* ]

iTm=KEY

iTm=KEYA

iTm

iTm=olnTm

iTm=iToln

iTm=olnToln

A simple reformat causes the data field described by the specification on the right of the equal sign to be reformatted and moved to the data field described by the specification on the left. The descriptor on the right of the equal sign describes a data field in the input file; the descriptor on the left decribes a data field in the output file. The item descriptor iTm is as defined in section 2, with the exception that the oln search descriptor can be substituted for i and m only for source items; oln is not valid in destination items.

The length of the output field takes precedence over the length of the input field. Literals exceeding the length of the output field are truncated on the right. If the length of the output field exceeds the length of the input field, the string is left-justified and blank filled.

If m is omitted from either the source or destination side, a value of 1 is assumed.

When the data type of the source field differs from that of the destination field, the data is converted to the destination field format. Data can be converted between any two of the CDC internal data types listed in table 2-1 of section 2. Table G-1 of appendix G summarizes the rules for all valid conversions.

When a signed numeric field (data type S) is converted to a character field (data type X), a positive sign is dropped. A negative sign is inserted as the first character of the receiving field, and, if the receiving field cannot accommodate the sign, the number is truncated on the left.

If the input file is one of the CYBER Record Manager Advanced Access Methods (AAM) file organizations (indexed sequential, direct access, or actual key), the source item KEY can be used to insert the key into the output records. KEY cannot be used for ORG=NEW files that contain embedded keys, as CRM does not return the key to FORM.

KEY is an X (character) item with length specified by the KL parameter of the FILE control statement. Assume the reformat item:

1X10=KEY

The key from the input record is inserted starting at the first byte of the output record.

The source item KEYA is an I (integer) item. FORM sets KEYA to the value of the last key returned by CRM for an actual key output file. KEYA can then be used in the REF directive to insert the key in another output file.

When a reformat item is written in the form iTm, specifying a single item descriptor instead of a source and destination item descriptor, the item describes a data field in the source record. Data is moved from the indicated source field to a field of the same format in the output record. This is equivalent to the form iTm=iTm with the destination descriptor having the following values:

| | |
|---|---|
| i | The current byte (bit for data type B) in the destination record. The current byte is defined as the one that follows the last byte referenced in a reformat string. Initially i points to the first byte in the record. |
| T | Same as T for the source field. |
| m | Same as m for the source field. |

This form is especially useful when referencing fields of variable length, because it allows the destination field to assume the length of the source field. Assume the reformat item:

```
1*,*1X1*,*2
```

The character field in the input record bounded by and including the first and second commas is moved to the field starting at the current byte of the output record. The output field length is equal to the input field length.

Some examples of reformat items are shown in table 3-4.

### Reformat Strings Used as Reformat Items

Wherever a simple reformat is allowed, a reformat string can be used. This feature allows specification of alternate reformats within a record, and in situations where alternatives can themselves have alternatives.

Reformat strings can occur as items within reformat strings to a maximum depth of seven levels.

## REFORMAT SPECIFICATIONS

The format of a reformat specification is:

[selector-expression:] reformat-item

[,reformat-item...]

A reformat specification consists of an optional selector expression followed by a list of one or more reformat items. The selector expression is suffixed by a colon. The selector expression must be true for the associated reformat items to be executed. If the selector expression is false, the list of reformat items is ignored. If the selector expression is omitted, all reformat items in the reformat specification are executed. Selector expressions are defined in section 2.

TABLE 3-4. EXAMPLES OF REFORMAT ITEMS

| Example | Explanation |
|---|---|
| 1E = 1S10 | Converts first ten signed numeric characters of the input record to floating-point and places them in the output record starting at the first character position. |
| 21X10 = 11I | Converts the integer at word 2 (character position 11) of the input record to alphanumeric characters and inserts them in the 10-character field of the output record starting at character position 21. |
| 11I = $4.56$ | Truncates the numeric literal 4.56 and converts it to integer 4, then inserts it into the 60-bit output field starting at character position 11. |
| 10(X3) | Copies ten contiguous data fields three characters long from the input record to the output record, starting at the first character position. |
| 2X4 = 20X4, 24X4 | Moves the 4-character data field starting at byte 20 of the input record to the 4-character field starting at byte 2 of the output record. Then moves the 4-character field starting at byte 24 to the field starting at the current byte (byte 6) of the output record. |
| 20X10 = 64X10 | Moves 10 characters starting at byte 64 of the input record to the field starting at byte 20 of the output record. |
| 1E=21E,4(E=E) | Starting at word 3 (byte 21) of the input record, moves 5 contiguous 60-bit floating-point data items to the field starting at byte 1 of the output record. Since E-type data items have a fixed length, the length m is omitted from the iTm specification (see table 2-1 of section 2). |
| 32X2 = $ABC$ | Places the string AB in the field of the output record starting at character position 32. |
| 32X4 = $ABC$ | Places the characters ABC and one trailing blank in the output record starting at byte 32. |
| 20X4 = $    $ | Places 4 blanks in the field of the output record starting at character position 20. |

Numeric values used as operands in selector expressions must be written as string literals enclosed by $ or * delimiter pairs; numeric literals are not valid in reformat specifications.

Elements of a reformat specification are illustrated in figure 3-12.

$$\underbrace{\text{16X4=\$ABCD\$}}_{\text{reformat-specification}}$$

$$\underbrace{\underbrace{\text{16X4=\$ABCD\$,}}_{\text{reformat-item-1}}\quad\underbrace{\text{1N6=91S6}}_{\text{reformat-item-2}}}_{\text{reformat-specification}}$$

$$\underbrace{\underbrace{\text{X1 EQ \$A\$:}}_{\text{selector expression}}\quad\underbrace{\text{E=65E}}_{\text{reformat-item}}}_{\text{reformat-specification}}$$

$$\underbrace{\underbrace{\text{B2 GT 0:}}_{\text{selector expression}}\quad\underbrace{\text{11X3=\$ABC\$,}}_{\text{reformat-item-1}}\quad\underbrace{\text{21X3=\$XYZ\$}}_{\text{reformat-item-2}}}_{\text{reformat-specification}}$$

Figure 3-12. Elements of a Reformat Specification

## REFORMAT STRINGS

The format of a reformat string is:

reformat-specification

[;reformat-specification...]

A reformat string consists of one or more alternative reformat specifications. When a reformat string is encountered during execution, each reformat specification within the string is tested in turn, from left to right, until one is found with a selector expression that is true. That reformat specification is executed, and all specifications to its right are ignored. If none are true, no reformatting is performed.

### Reformat String Punctuation

A colon separates a selector expression from its associated reformat items in a reformat specification; multiple reformat items are separated by commas. The scope of a selector expression is a single reformat specification; the scope is terminated by a semicolon. When the selector expression is true, the semicolon causes the rest of the parenthesized reformat string, up to the matching right parenthesis, to be ignored. This feature is of special significance when items are nested in parentheses in a reformat string. When a reformat specification is executed, all remaining parts of the reformat string are ignored.

### Nested Reformat Strings

A reformat string can be used wherever a simple reformat is allowed in a reformat item. Reformat strings can occur within reformat strings to a maximum depth of seven levels. This nesting feature permits the specification of alternate reformatting at various positions within a record.

### Q Specification

The Q specification causes all reformatting for a record to terminate. The contents of the unreformatted portion of the output record are determined by the BGD parameter of the OUT directive. Q can be inserted wherever a reformat item is valid.

The Q specification is useful when performing alternative reformats. Assume the reformat string:

```
10X4 EQ $STOP$:Q; 1X20=1X20
```

FORM stops reformatting if the 4-character field starting at byte 10 of the input record contains the string STOP; otherwise, 20 characters are copied from the input record to the output record.

Figure 3-13 illustrates elements of a reformat string.

## ADDITIONAL NOTATION TECHNIQUES

The value of i in the iTm specification determines the position within the record of the data field. FORM maintains internal pointers for the source and destination records, each initially pointing to byte 1 bit 1 of the record. After a reformat item is executed, the pointers are set to the sum of i and m, to point to the byte (bit for data type B) following the field just processed. If i is omitted, FORM uses these pointers to find the next field in the record. Assume the reformat string:

```
10X2=25X2, X4=X4, X7=X7
```

From a starting position of 25 in the input record, three fields of sizes 2, 4, and 7 characters are moved to a starting position of 10 in the output record.

The i need not be omitted from both sides of the equal sign. If the byte index is omitted on the left, FORM accesses the next position in the output record. If the byte index is omitted on the right, FORM accesses the next position in the input record. FORM maintains current position pointers for both records. For example:

```
10X2=*AB*,12X3=*GHI*,15X4=*QRST*
```

could be written as:

```
10X2=*AB*,X3=*GHI*,X4=*QRST*
```

Prior to reformatting, the pointers are rounded up to the next byte, except for data type B; no rounding takes place for a type B source or destination item.

Alignment is never forced to a word boundary. If word boundary or other alignment is needed, the user must supply the proper fill items explicitly.

A specification of X=literal can be shortened by omitting X=. The previous string becomes:

```
10X2=*AB*,*GHI*,*QRST*
```

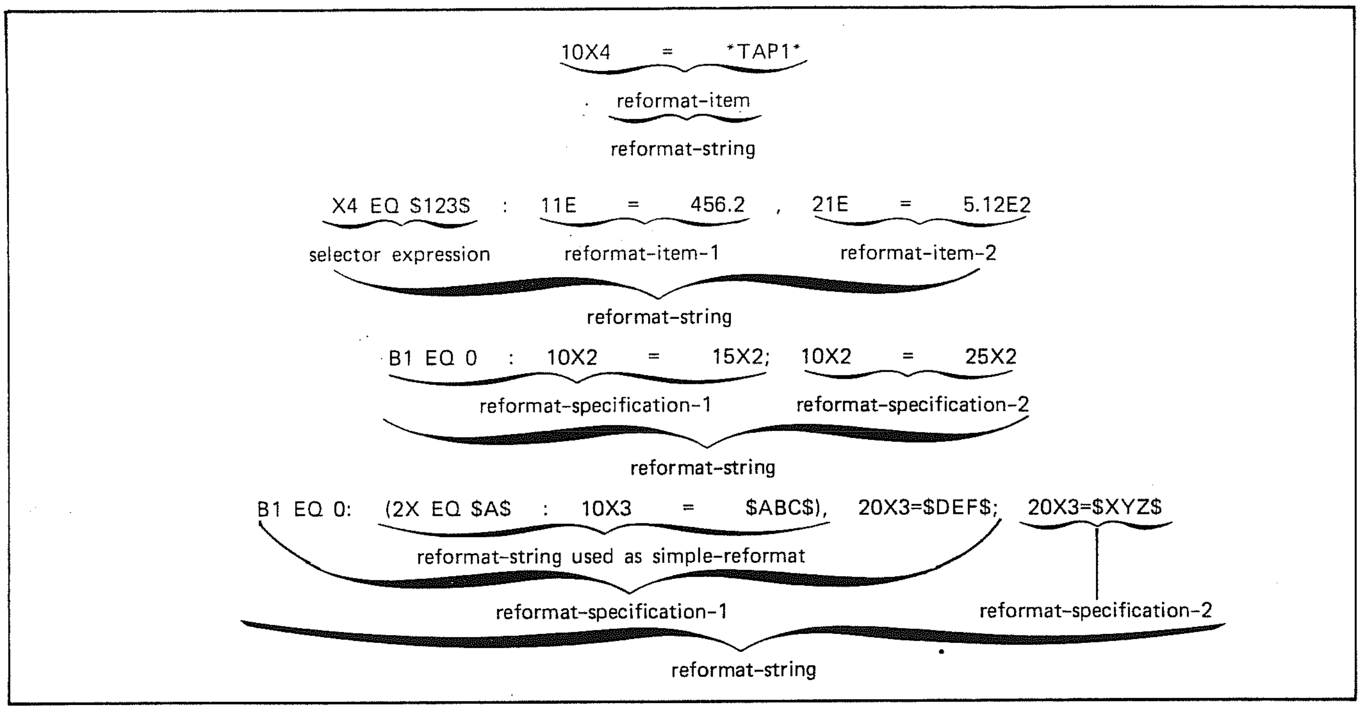


Figure 3-13. Elements of a Reformat String

In summary;

- i can be omitted, causing the next position in the record to be used.
- X = can be omitted when a literal is to be inserted.

Examples of the REF directive are illustrated in figure 3-14.

## SEQUENCING DIRECTIVE (SEQ)

The SEQ directive places sequencing information in each record of an output file. The sequence number replaces the current contents of the designated field in the record. The sequence number can have a maximum value of $2^{48}-1$. A separate SEQ directive must be specified for each output file to be sequenced. The format of the SEQ directive is shown in figure 3-15.

Some examples of the SEQ directive are illustrated in figure 3-16.

## PAGE DIRECTIVE (PAG)

The PAG directive provides page formatting options for an output file to be printed. A separate PAG directive must be included for each file to be printed. Each PAG directive requires an OUT directive declaring the print file.

The PAG directive causes a page number to be printed on line one of all pages, except when formatting is controlled by the user with the FMT=A option. The PAG directive format is shown in figure 3-17.

A print line is 136 characters long (excluding the carriage control character); records are truncated if necessary. An appropriate carriage control character is inserted in the first character position of each record, except when FMT=A is specified. No checks are made on record types or content; the user is responsible for providing printable data. Examples of the PAG directive are illustrated in figure 3-18.

## EXECUTE DIRECTIVE (XEQ)

The XEQ directive specifies the names of the IX and FEX owncode routines to FORM. This is the only function of XEQ; if no IX or FEX routines are to be used, this directive can be omitted. XEQ can appear anywhere in the directive sequence. The format of the XEQ directive is shown in figure 3-19.

If an IX exit is specified, the INP directive must be omitted from the directive sequence.

Examples of the XEQ directive are illustrated in figure 3-20.

Example 1.

```
OUT(OUTFILE,BGD=C)
REF(OUTFILE,1E=$67.5$, $XYZ$)
```

The 60-bit floating-point number 67.5 is placed in the first word of the output record of OUTFILE; the characters XYZ are placed starting at the next consecutive position in bytes 11, 12, and 13. The remainder of the output record is identical to the input record.

Example 2.

```
OUT(MYFILE,BGD=N)
REF(MYFILE,I EQ $0$: 5(I); 5(S10=I))
```

The output record is initially set to zero. If the integer in the first word of the input records is zero, five 60-bit integers are copied to the output record starting at word 1. If the first word is nonzero, the first five consecutive integers of the input record are converted to signed numeric fields 10 characters in length and moved to the output record starting at word 1.

Example 3.

```
REF(OUTFILE,20X=48$,$-2X48$,$3)
```

This example illustrates the use of the oln search descriptor. Input records are reformatted for the variable length character field delimited by commas. The field moved starts immediately after the second comma appearing after character position 48 and continues through the third comma found after position 48. The field is moved to the output record of OUTFILE starting at character position 20. The length m of the destination field is omitted, since it is unknown. The search descriptor is substituted for both i and m in the source field.

Example 4.

```
REF(FILEA, 10X=KEY)
```

The key from the indexed sequential input file is inserted into the output record starting at character position 10, thereby increasing the size of the record.

Figure 3-14. Examples of REF Directive

```
SEQ(lfn,NBR=d,BEG=n,ADD=n)
```

With the exception of lfn, the parameters can be specified in any order.

| | |
|---|---|
| lfn | Output file name. |
| NBR=d | Item descriptor of the form iTm that specifies where in the record the sequencing information is to be written. |
| | i — Beginning character position of sequencing field in the output record; i can be expressed as i/w for data type B. |
| | T — Data type of sequence field; can be any CDC data type.[†] The sequence numbers are translated to the data type represented by T. |
| | m — Length of sequence field; must conform to rules.[†] |
| BEG=n | Initial value (decimal) of sequence number; any value from 0 through $2^{48}$ - 1. Default is 1. |
| ADD=n | Sequence increment (decimal); any value from 1 through $2^{48}$ - 1. Default is 1. |

[†]Refer to table 2-1 of section 2.

Figure 3-15. SEQ Directive Format

Example 1.

```
SEQ(EMPLNR,NBR=1X6,ADD=5)
```

Records of EMPLNR are assigned a 6-digit sequence number, starting in the first character position of each record. The sequence number has an initial value of 1 (default) and an increment of 5.

Example 2.

```
SEQ(FILE1,NBR=19B6,BEG=0)
```

Records of FILE1 contain a 6-bit binary sequence number starting in character position 19. The sequence number has an increment of 1 (default), starting at 0.

Figure 3-16. Examples of SEQ Directive

```
PAG(lfn,FMT=f,PGL=n,TOP=n,TTL=lit)
```

With the exception of lfn, the parameters can be specified in any order.

| | |
|---|---|
| lfn | Logical file name of output file in print format. |
| FMT=f | Line spacing. FORM inserts the necessary carriage control character when f is 1, 2, or D. |
| | 1 Single space (default). |
| | 2 Double space. |
| | D Dump option; output records are single-spaced in 100-character lines (or record length if less than 100 characters). A decimal record number and character count are printed for each record. The character count reported is the record length as returned by CRM. |
| | A Character one of each record is used for carriage control; and page numbering is suppressed. The user must supply the carriage control character for the A option. The operating system reference manuals contain a complete list of these characters. |
| PGL=n | Number of print lines per page including the title line and any blank lines. The maximum value is 511. If omitted, default is 60. Must be omitted when FMT=A is specified. |
| TOP=n | Top margin in lines. Title or first print line occurs immediately after the margin. Maximum value is 511. If omitted, default is 2. Must be omitted when FMT=A is specified. |
| TTL=lit | Character string to be used as page title. Must be expressed as a literal, enclosed by the characters * or $. The title follows the top margin and is followed by a blank line. If omitted, a blank line is generated. If FMT=A is specified, the title appears on the first page only, and is not followed by a blank line. Maximum string length is 115 characters. |

NOTE

For the convenience of FORM 1.0 users, PRT is accepted as PAG. However, the FORM 1.0 and FORM 1.2 directives are not identical in effect, and might produce different output.

Figure 3-17. PAG Directive Format

Example 1.

```
OUT(LIST)
PAG(LIST,FMT=2,PGL=25,TOP=4,
TTL=$ EMPNO NAME SALARY$)
```

The output file LIST is formatted for double spacing; printing begins with the title on the fifth line of each page. 25 lines per page are printed.

Example 2.

```
OUT(OUTFILE)
PAG(OUTFILE,FMT=D)
```

Records of OUTFILE are printed in dump format.

Example 3.

```
OUT(PRINTA)
PAG(PRINTA,FMT=A)
```

File PRINTA is printed with format controlled by the file. The first character position of each record is assumed to contain a carriage control character.

Figure 3-18. Examples of PAG Directive

```
XEQ(IX=ept,FEX=ept,FIN)
```

| | |
|---|---|
| IX=ept | Entry point name of owncode routine to transmit input records to FORM. This option can be used in place of the INP directive; it must be used when an INP directive is not included and the INP parameter is not specified in the FORM control statement. |
| FEX=ept | Entry point name of owncode routine to which control transfers whenever an error occurs during execution which would cause FORM to terminate abnormally. The FORM run terminates immediately after execution of this routine. |
| FIN | Signals the end of run. This option is not required; it is included for FORM 1.0 compatibility. |

The IX and FEX parameters are optional and order independent.

Figure 3-19. XEQ Directive Format

Example 1.

```
XEQ(IX=INREC,FEX=FERR)
```

FORM receives input records from entry point INREC. If an unrecoverable error occurs during execution, control transfers to entry point FERR prior to termination.

Example 2.

```
OUT(OUTPUT)
PAG(OUTPUT,FMT=D)
XEQ(FEX=MYCODE)
```

Records supplied at entry point MYCODE are written to the file OUTPUT in dump format.

Figure 3-20. Examples of XEQ Directive

# EXECUTION 4

A FORM job deck must contain control statements necessary for communicating with the operating system, as well as statements describing input and output files to CYBER Record Manager (CRM). If owncode routines are to be used during execution, the user must make certain information available to FORM.

## CONTROL STATEMENTS

A control statement must be included to make input files available to FORM, call for execution of FORM, and provide for proper disposition of output files. Refer to the appropriate operating system reference manual for detailed information on operating system control statements.

### FORM CONTROL STATEMENT

FORM is called with an operating system control statement. The control statement format is shown in figure 4-1.

```
FORM(I=lfn,OWN=lfn,L=lfn)
```

| | |
|---|---|
| I=lfn | Name of file containing input directives in card image format (CRM record type Z with C blocking). Default is I=INPUT. A specification of I alone is equivalent to I=COMPILE. |
| OWN=lfn | Name of file containing binary decks for owncode routines. This file is loaded by FORM. Default is OWN=LGO. |
| L=lfn | Name of file to receive summary of the FORM run, including diagnostic messages. Default is L=OUTPUT. |

All parameters are optional and can be specified in any order.

Figure 4-1. FORM Control Statement Format

If the I, L, and OWN parameters are omitted, then the following form of the control statement is used:

```
FORM.
```

Alternatively, the following special call can be made if only a simple file copy is desired:

```
FORM(INP=input-file,OUT=output-file)
```

where

| | |
|---|---|
| INP=input-file | Name of input file |
| OUT=output-file | Name of output file |

No I or OWN file can be specified and no directives can be included. A summary report is not generated. This form of the call is equivalent to

```
FORM.
```

with input directives

```
INP(input-file)

OUT(output-file)
```

### IBM TAPE FILES

ASCII or EBCDIC files on either 7-track or 9-track IBM sequential tapes can be processed by the CON directive. The user should refer to the appropriate operating system reference manual for the correct method of requesting 7- and 9-track tapes to be processed by FORM.

The programmer should be familiar with the LABEL statement. Complete decriptions of the 7- and 9-track tape parameters are in the appropriate operating system reference manual.

Input tape files can be copied to disk or other devices that can be substituted for tapes, provided the data format of the copied files is identical to that of IBM tapes when the files are read by CRM.

The CON directive is not always needed for IBM-to-CDC or CDC-to-IBM conversions. The following examples demonstrate simple conversions using FORM without CON or any other directives. The tape used in this example is a single-file tape that contains only alphanumeric data.

If only a read of an IBM tape into a CDC file is required, the following control statements are sufficient:

```
LABEL, IBMTAPE, F=S, CV=EB
FILE, IBMTAPE, BT=K, RT=F, FL=80, RB=20, CM=YES
FILE, CDCDISK, BT=C, RT=Z, MRL=80
FORM, INP=IBMTAPE, OUT=CDCDISK
```

Note that, when this conversion is made, lowercase characters are converted to uppercase.

The following example shows how a CDC file can be converted to a blocked EBCDIC tape.

```
LABEL, CDCFILE, PO=W, NT, D=PE, F=S, CV=EB
FILE, CDCFILE, BT=C, RT=F, FL=136, CM=NO
FILE, IBMTAPE, BT=K, RT=F, FL=182, RB=10, CM=YES
FORM, INP=CDCFILE, OUT=IBMTAPE
```

This approach can be used to create ASCII tapes.

Information concerning use of FORM and variable length records on IBM tapes can be found in appendix K.

### CYBER RECORD MANAGER INTERFACE

CYBER Record Manager (CRM) performs all the input/output functions of FORM. Before CRM can process a file, it must establish a file information table (FIT) to define all record, blocking, and file organization characteristics. All file processing is based on the contents of this table.

The user is responsible for adequately describing the structure of each file to be processed. For any given FIT field, the value can be provided either by a parameter in the FILE control statement or by acceptance of the CRM default. The FILE control statement can be used to change any default without affecting any other values or defining the entire FIT. This statement overrides any default values when the named file is opened. The user must supply all FIT values for which the default is not to be used. Refer to the Advanced Access Methods (AAM) reference manual or the Basic Access Methods (BAM) reference manual for detailed information about file descriptions and CRM.

Appendix K has information about determination of input record length and output record length when using CRM files with variable length records.

FORM overrides some CRM defaults. Table 4-1 summarizes the default FIT field values provided by CRM and FORM. The user can change any of these values through the FILE control statement.

TABLE 4-1. SUMMARY OF FIT FIELD DEFAULT VALUES[†]

| Field and Value | Default Meaning |
|---|---|
| FO=SQ | File organization is sequential. |
| RT=W[††] | Record type is W. |
| BT=I[††] | Block type is I. |
| ERL=0 | Unlimited nonfatal errors are allowed. |
| MRL<br>FL | No release default is allowed for FORM; user must specify a nonzero MRL or FL for each file. |
| MBL=0, 640, 1280, or 5120 | Maximum block size for BT=I is 5120; for BT=C with NOS/BE SI coded, MBL=1280; with all other systems, MBL=5120; for other BT, MBL=0. |
| OF=N | File is not rewound before open. |
| CF=N | File is not rewound after close. |
| VF=U | Tape file is unloaded if volume is closed. |
| CM=NO | No code conversion is performed for tape read or write. |

[†]Refer to CYBER Record Manager Basic Access Methods or Advanced Access Methods reference manuals for more information about CRM defaults.

[††]For files named INPUT, OUTPUT, or PUNCH: RT=Z,BT=C. When RT is set to S, any block type setting on BT is disregarded.

Since CRM cannot block or deblock records in IBM format, it is necessary to include a FILE control statement describing each IBM 8-bit sequential tape file to be processed by FORM in terms of CRM parameters. The parameters required in the FILE control statement for IBM tape files are shown in table 4-2.

TABLE 4-2. PARAMETERS REQUIRED IN THE FILE CONTROL STATEMENT FOR IBM TAPE FILES

| Required Parameter for IBM Tapes | CRM Parameter |
|---|---|
| RT=S | Record type (only when the CON directive is used) |
| MBL=nnnn | Maximum block size |
| MRL=nnnn | Maximum record length |
| CM=NO | Conversion mode |

nnnn=(BLK*8)/6 since BLK refers to 8-bit characters, and CRM expects sizes in terms of 6-bit characters. A fraction must be rounded up to the next highest integer.

For 9-track tape input/output, two additional parameters are necessary for noise record skipping:

| | |
|---|---|
| MNR=24 | Minimum Record Length |
| MNB=24 | Minimum Block Length |

Any block containing fewer than MNR or MNB frames is considered noise and discarded by the system.

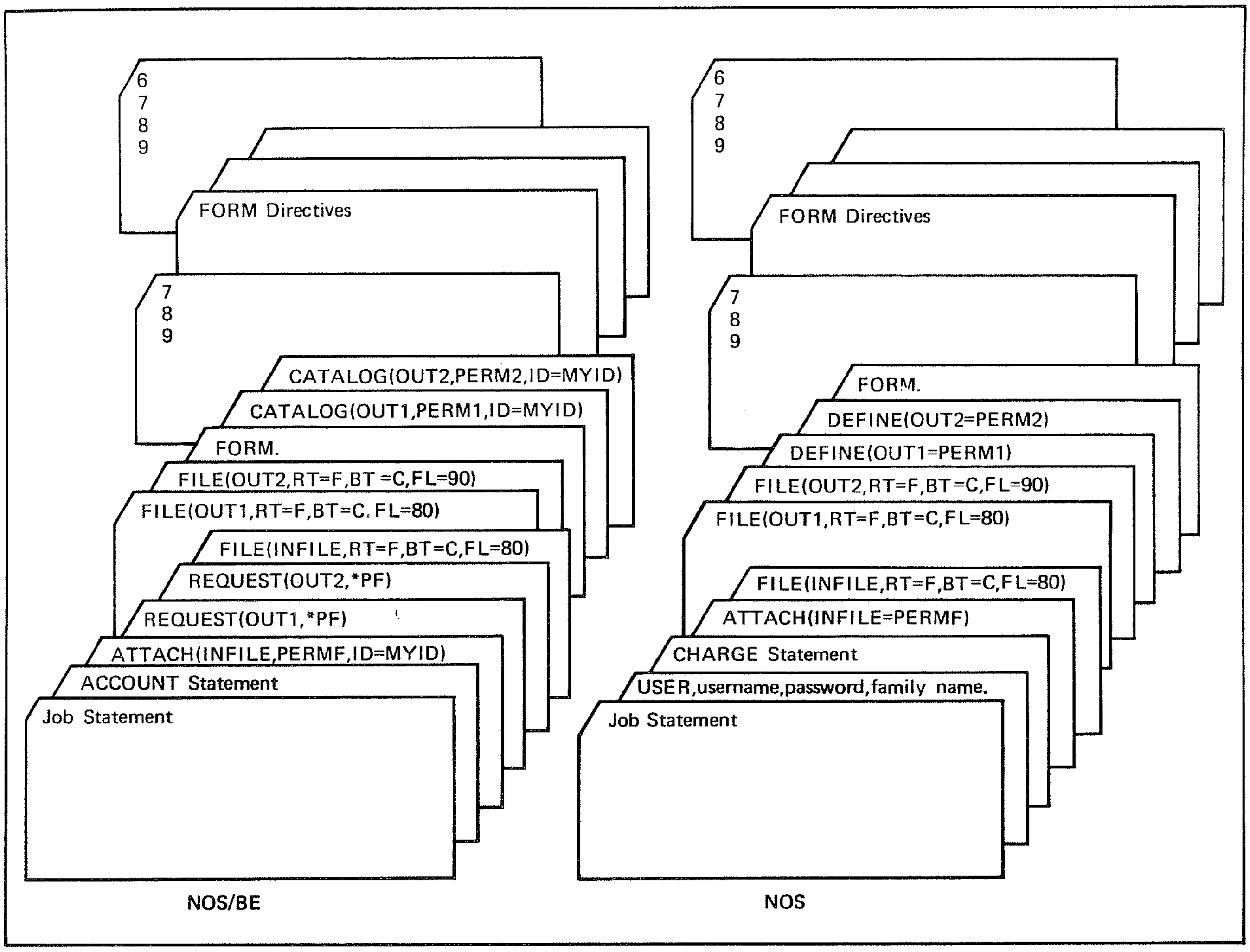
For 7-track tapes the operating system can discard blocks prior to passing them to CRM. A block of less than 7 frames (NOS) or 14 frames (NOS/BE) is considered noise and ignored.

Under NOS, the noise size of a tape file is set to 7 frames (by default) and can be overridden by the NS parameter of the LABEL control statement.

A sample deck structure is shown in figure 4-2. In this example, FORM is to read an input file and create two output files. Control statements are included to attach the input file, request permanent file device residence for the output files (NOS/BE), and make the output files permanent. The FILE statements override the CRM defaults for record type and block type.

## OWNCODE INTERFACE

The FORM user can specify subroutines to be executed at any or all of eleven owncode exits during FORM execution. The exits are listed in table 4-3. The entry points to the user-supplied routines are specified in the directive that corresponds to the appropriate exit. Figure 1-3 of section 1 shows the point in processing at which the exits occur.



6
7
8
9
FORM Directives
7
8
9
CATALOG(OUT2,PERM2,ID=MYID)
CATALOG(OUT1,PERM1,ID=MYID)
FORM.
FILE(OUT2,RT=F,BT =C,FL=90)
FILE(OUT1,RT=F,BT=C,FL=80)
FILE(INFILE,RT=F,BT=C,FL=80)
REQUEST(OUT2,*PF)
REQUEST(OUT1,*PF)
ATTACH(INFILE,PERMF,ID=MYID)
ACCOUNT Statement
Job Statement
NOS/BE
6
7
8
9
FORM Directives
7
8
9
FORM.
DEFINE(OUT2=PERM2)
DEFINE(OUT1=PERM1)
FILE(OUT2,RT=F,BT=C,FL=90)
FILE(OUT1,RT=F,BT=C,FL=80)
FILE(INFILE,RT=F,BT=C,FL=80)
ATTACH(INFILE=PERMF)
CHARGE Statement
USER,username,password,family name.
Job Statement
NOS

Figure 4-2. Sample Deck Structures

Owncode routines to be executed during a FORM run must be stored in relocatable binary form in a single file. The easiest way to do this is to compile the source code into a single binary file. The file name is then specified in the OWN parameter of the FORM control statement. FORM loads the routines from the specified file via the user-call loader. Owncode routines cannot originate from a library. Owncode routines can, however, call library routines.

The IX, CX, and RX owncode routines are coded in function format; the FEX routine is coded in subroutine format. Each routine consists of an initial function or subroutine statement followed by the function or subroutine body. The initial statement contains the name that is used as the entry point name in the FORM directive, and a parameter list through which the user communicates with FORM. With the exception of the IX routine, FORM passes information to the user; the IX routine passes information to FORM.

In most cases, the user must assign a return code to the function name at some point within the function body. When FORM regains control, it tests the return code and takes the appropriate action.

Areas used as function arguments must be of sufficient size, specified according to the rules of the language in which the function is coded. Refer to the appropriate reference manual for additional information on the coding of functions.

FORTRAN Version 5 is used to illustrate the function formats described in the following paragraphs.

## RX EXIT

The routine speicfied by the RX parameter in the OUT directive is called whenever a reformatting error occurs during REF directive processing.

The FORTRAN function format for the RX routine is shown in figure 4-3. The arguments, passed by FORM to the routine, consist of a 60-bit word with the pointer format shown in figure 4-4.

Upon regaining control, FORM tests for the following return code values:

| | |
|---|---|
| 0 | continue execution |
| ≠0 | terminate |

TABLE 4-3. OWNCODE EXITS

| Exit | Explanation |
|---|---|
| LX | Performs tape label processing. |
| DX | Receives control whenever an end-of-partition, end-of-section, or end-of-data is encountered in the input or output file. |
| EX | Transfers control to a user recovery routine when an error (fatal or trivial) occurs. |
| CX | Receives control whenever a conversion error is encountered while reading or writing an IBM file. This exit is used only when a CON directive is active. |
| DCA | Performs record decompression/decryption for AAM input files. This exit can be an owncode or a standard CRM routine. |
| CPA | Performs record compression/encryption for AAM output files. This exit can be an owncode or a standard CRM routine. |
| HRL | Performs key hashing for direct access files. |
| IX | Supplies input records to FORM. This exit is valid only when the INP directive is omitted. |
| FEX | Receives control whenever an error occurs that would cause the FORM run to terminate abnormally. Termination occurs when FORM regains control. |
| RX | Receives control when a reformat error occurs during REF processing. |
| DCT | Makes user-supplied collating sequence conversion table available for indexed sequential files. |

```
INTEGER FUNCTION fn(inptr,outptr,strptr)
  .
  .
  function body
  .
  .
RETURN
END
```

| | |
|---|---|
| fn | function name |
| inptr | pointer to input record area |
| outptr | pointer to output record area |
| strptr | pointer to reformat string |

Figure 4-3. RX Function Format

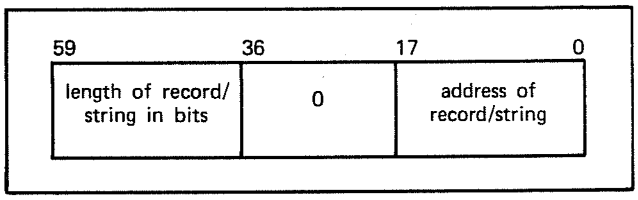


Figure 4-4. Pointer Format for RX Function

A FORTRAN function for the RX exit is illustrated in figure 4-5. Logical operators are used to mask the length and address fields from the pointers passed by FORM. A value of one is assigned to the function name; this value tells FORM to terminate immediately after regaining control.

```
          INTEGER FUNCTION RXFUNC(IPTIN,IPTOUT,IPTSTR)
          DATA MASKA/O"777777"/,MASKB/O"77777777"/
C
C.... FETCH ADDRESS AND LENGTH OF INPUT RECORD
C
          IADDI=IPTIN.AND.MASKA
          LENIN=SHIFT(IPTIN,24).AND.MASKB
C
C.... FETCH ADDRESS AND LENGTH OF OUTPUT RECORD
C
          IADDO=IPTOUT.AND.MASKA
          LENOUT=SHIFT(IPOUT,24).AND.MASKB
                .
                .
                .
C
C....ASSIGN RETURN CODE AND RETURN CONTROL TO FORM
C
          RXFUNC=1
          RETURN
          END
```

Figure 4-5. Example of RX Owncode Routine

## CX EXIT

The function associated with the CX exit has the format shown in figure 4-6. The entry point name is specified by the CX parameter in the INP directive. This exit is taken whenever an error occurs while converting to or from IBM format. FORM tests for the following return code values:

| | |
|---|---|
| >0 | terminate |
| 0 | continue execution |
| -1 | process as end-of-section |
| -2 | process as end-of-partition |
| -3 | process as end-of-information |

```
        INTEGER FUNCTION fn(wsa,rec,constr,ierr,len)
              .
              .
              .
          function body
              .
              .
              .
        RETURN
        END
```

| | |
|---|---|
| fn | Function name. |
| wsa | Working storage area used by FORM. Size is determined by the RFM parameter in the INP directive, as follows:<br>RFM=F,FB,U,V, and reading of VB<br>$\text{space} = 6 + \left\lceil \frac{BLK}{7.5} \right\rceil$ words[†]<br>RFM = VSB, and writing of VB<br>$\text{space} = 6 + \left\lceil \frac{LRL}{7.5} \right\rceil + \left\lceil \frac{BLK}{7.5} \right\rceil$ words |
| rec | IBM record area. |
| constr | Area containing conversion string that was input to CON directive. |
| ierr | Error code from FORM:<br>1 conversion error<br>0 no error<br>-1 end-of-section encountered on input file<br>-2 end-of-partition encountered on input file<br>-3 end-of-information encountered on input file |
| len | IBM input record length, in 8-bit bytes. Must be omitted when conversion is from CDC to IBM. |

[†] A fraction is always rounded up to the next higher integer.

Figure 4-6. CX Function Format

## IX EXIT

The IX routine supplies input records to FORM. The entry point name is specified by the IX parameter in the XEQ directive. A single source of records is required for a FORM run; therefore the IX exit can be used only if the INP directive is omitted for a FORM run. The format of the associated function is shown in figure 4-7.

```
        INTEGER FUNCTION fn(iaddr,len)
              .
              .
              .
          function body
              .
              .
              .
        RETURN
        END
```

| | |
|---|---|
| fn | function name |
| iaddr | address of input record |
| len | length (characters) of input record |

Figure 4-7. IX Function Format

The values for the address and length of the input record must be assigned within the function and passed to FORM. On regaining control, FORM tests for the following return code values:

| | |
|---|---|
| 1 | I/O error |
| 0 | no error, data record supplied |
| -1 | end-of-section |
| -2 | end-of-partition |
| -3 | end-of-information |

An example of a FORTRAN routine, executable under NOS, that reads card images and passes them to FORM, is illustrated in figure 4-8. The FTN5 control statement compiles the FORTRAN program. The resulting binary resides on the LGO file where it is accessed by FORM. The function IXFN reads cards and stores them in the array INREC. A $ in column 1 of the input data indicates end-of-input.

## FEX EXIT

The routine specified by the FEX parameter on the XEQ directive is called whenever an error occurs that would otherwise cause FORM to terminate abnormally. The FEX routine is written in the format of a subroutine. The FORTRAN format is shown in figure 4-9. No information is passed to or received from FORM, and no argument list need be specified. When FORM regains control, the run terminates.

```
JOB statement
USER statement
CHARGE statement
FILE(OUTF,RT=Z,BT=C,MRL=80)
FTN5.
FORM(OWN=LGO)
REWIND,OUTF.
COPYBF,OUTF,OUTPUT.
7/8/9
C
C.... IXFN READS CARD IMAGES AND
C.... PASSES THEM TO FORM.
C
      INTEGER FUNCTION IXFN(IADDR,LEN)
      DIMENSION INREC(8)
C
C.... LOCF FETCHES THE ADDRESS
C.... OF INREC
C
      IADDR=LOCF(INREC)
      LEN=80
      READ 100, INREC
  100 FORMAT(8A10)
      IXFN=0
C
C.... A $ IN COLUMN 1 MEANS
C.... THE END OF DATA
      IF(INREC(1).EQ.'$') IXFN=-3
      RETURN
      END
7/8/9
OUT(OUTF)
PAG(OUTF,TTL=$--OUTF--$)
QAL(OUTF,30X1 EQ $+$)
XEQ(IX=IXFN)
7/8/9
  THIS IS THE FIRST RECORD     +
  THIS IS THE SECOND RECORD
  THIS IS THE THIRD RECORD     +
  THIS IS THE FOURTH RECORD
$
6/7/8/9
```

Figure 4-8. Example of IX Owncode Routine

```
          SUBROUTINE subname
                .
                .
                .
           subroutine
              body
                .
                .
                .
          RETURN
          END
```

Figure 4-9. FEX Subroutine Format

## DCA,CPA,HRL,LX,DX,DCT,EX EXITS

The routines specified by the DCA, CPA, HRL, LX, DX, and EX parameters are called by CRM. DCA, HRL, LX, DX, and EX exits are specified in the INP directive; CPA, HRL, LX, DCT, and EX exits are specified in the OUT directive. Specifying these parameters causes the proper FIT field to be set; if any are not specified, CRM default action is taken. Refer to the AAM and BAM reference manuals for details on the function and usage of these routines.

# SAMPLE PROGRAMS

This section contains two sample programs illustrating the use of FORM.

## CDC FILE REFORMATTING

FORM can be used to select, reformat, and write records from a CDC file to an output file.

The original file contains information about a group of university students. The file contains one record for each student; each record contains the following information:

| | |
|---|---|
| name | (20 characters) |
| number | (6 characters) |
| age | (2 characters) |
| sex | (1 character) |
| department code | (2 characters) |
| year in school | (1 character) |
| level | (2 characters) |
| address | (46 characters) |

The input record format and some sample records are illustrated in figure 5-1.

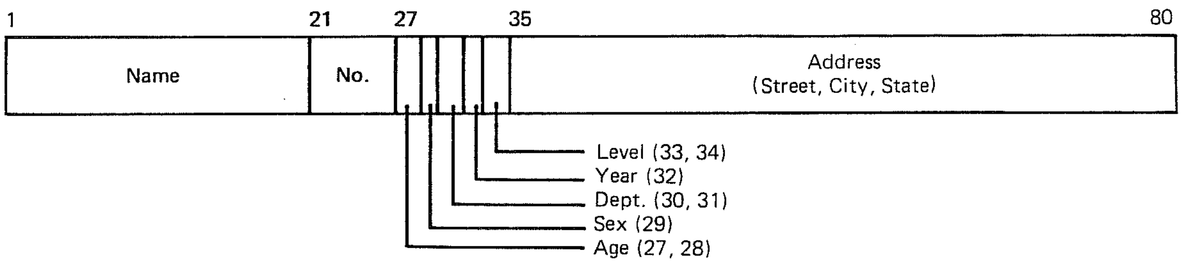
A new file is to be created containing records only for senior (SR) level students. The remaining records are to be reformatted to contain the following information:

| | |
|---|---|
| name | (20 characters) |
| number | (6 characters) |
| year in school | (1 character) |
| department code | (2 characters) |
| blank field | (3 characters) |
| city of residence | (30 characters) |

A test score is to be inserted into the blank field at a later date. In addition, all students in department 2 are being transferred to department 4. Output records are to be 62 characters in length. The output record format is shown in figure 5-2. The record reformatting is shown in figure 5-3.

The deck setups for running under NOS and NOS/BE are shown in figure 5-4. The OUT directive specifies BGD=X so that unreformatted portions of the output record will contain blanks. The QAL directive selects those records from the input file in which the level field contains SR. The REF directive reformats the selected records: the selector expression causes a 2 in a department code field to be changed to a 4; the expression X3=$   $ inserts three blank characters; search descriptors are used to extract the city name from the address field of the input records.

Control statements are included to attach the input file STDNTF, to describe input and output file characteristics to CYBER Record Manager (CRM), and to assign the output file GRADF to a permanent storage device. Numeric parameters refer to character position or count.

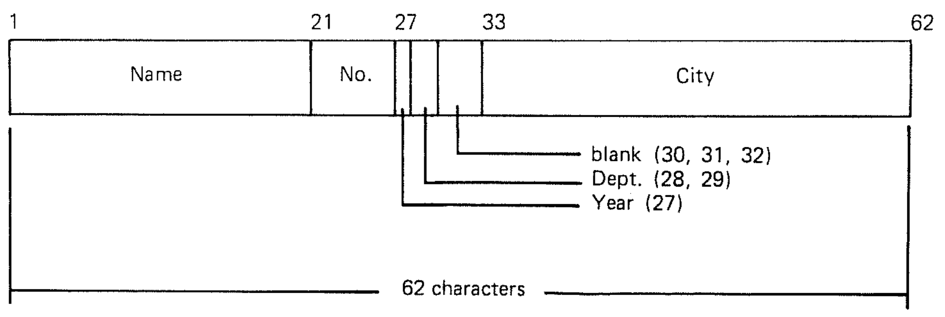


Input Record Format

```
ANDER, A. T.          12345621M023JR123 A ST., POZO, CALIF.
BINDER, R. L.         61234524F045SR456 NORTH ST., BOONEVILLE, CALIF.
KETTLE, V. H.         56123422M034SR33 MAIN ST., SAN JOSE, CALIF.
MCKEE, N. R.          45612325F026SR2216 BUSH ST., BEAR VALLEY, CALIF.
```

Sample Input Records from File STDNTF

Figure 5-1. Input Record Format and Sample Records



Output Record Format

```
BENDER, R. L.        012345504    BOONEVILLE
KETTLE, V. H.        561234403    SAN JOSE
MCKEE, N. R.         456123604    BEAR VALLEY
```

Sample Output Records from File GRADF

Figure 5-2. Output Record Format

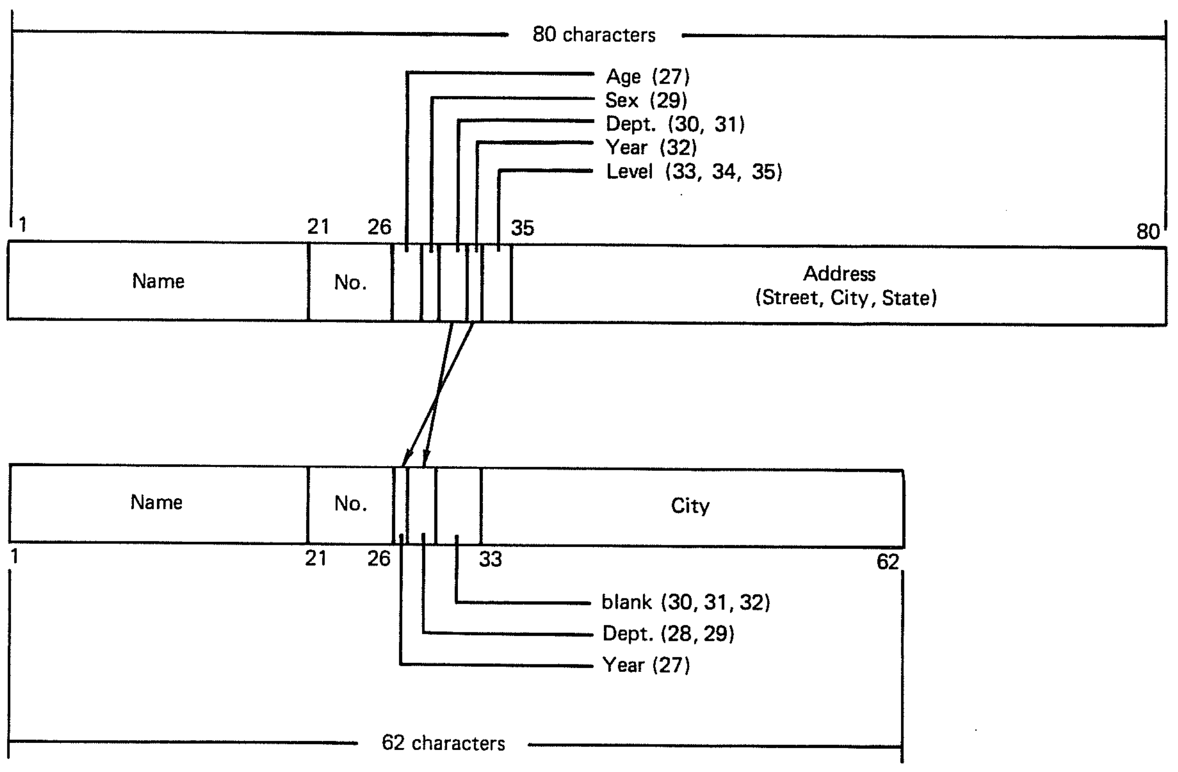


Figure 5-3. Record Reformatting

The parameters in the FILE control statements have the following meaning:

| | |
|---|---|
| RT | Record type (Z-type for cards) |
| BT | Block type (C-type for cards) |
| MRL | Maximum record length |

Refer to the AAM and BAM reference manuals for more information on FILE control statements.

An output listing of the FORM run under NOS is shown in figure 5-5. FORM directives are listed with a summary of action taken by FORM during the run. The reformatted output record is printed. The dayfile includes control statements encountered by the operating system.

```
NOS:

JOB statement
USER statement
CHARGE  statement
GET,STDNTF.
FILE(STDNTF,BT=C,RT=Z,MRL=80)
FILE(GRADF,BT=C,RT=Z,MRL=62)
FORM.
REWIND,GRADF.
COPYBF,GRADF,OUTPUT.
7/8/9 in column 1
INP(STDNTF)
OUT(GRADF,BGD=X)
QAL(GRADF,33X2 EQ $SR$)
REF(GRADF,X20=X20,N6=N6,N1=32N1,
    (30N2 EQ $02$: N2=$04$;N2=30N2),
    X3=$    $,35$,$-1X35$,$-2)
6/7/8/9 in column 1


NOS/BE:

MYJOB statement
ACCOUNT statement
ATTACH(STDNTF,ID=MYID)
FILE(STDNTF,BT=C,RT=Z,MRL=80)
FILE(GRADF,BT=C,RT=Z,MRL=62)
FORM.
REWIND,GRADF.
COPYBF,GRADF,OUTPUT.
7/8/9 in column 1
INP(STDNTF)
OUT(GRADF,BGD=X)
QAL(GRADF,33X2 EQ $SR$)
REF(GRADF,X20=X20,N6=N6,N1=32N1,
    (30N2 EQ $02$: N2=$04$;N2=30N2),
    X3=$    $,35$,$-1X35$,$-2)
6/7/8/9 in column 1
```

Figure 5-4. Job Decks for Reformatting Example

## IBM-CDC CONVERSION

FORM can be used to convert an IBM payroll file to two reformatted CDC files.

The original payroll file is to be converted to a CDC file. The unlabeled CDC tape has the blocking structure and record format defined in FILE statements, and is written in coded mode on a 7-track tape.

IBM records are supplied to FORM by the INP directive. The files CDCPAYR and OUTC are specified in the OUT directive. The CON directive converts the IBM type data file to the CDC type data file, CDCPAYR. The REF directive equates OUTC to CDCPAYR. OUTC is then printed using the FMT parameter of the PAG directive.

The IBM tape maximum block size is 3960 characters, arrived at by multiplying the number of 8-bit bytes in the block by 4/3 to convert to the 6-bit equivalent needed for CRM descriptions.

Figure 5-6 shows the data records definition and structure as created by IBM COBOL. Some sample IBM records are shown in figure 5-7. Figure 5-8 illustrates the corresponding CDC data record structure as it would be output by FORM, and the CDC COBOL definition of this record. Figure 5-9 shows some sample CDC records. Figure 5-10 shows the record reformatted from an IBM input record to a CDC output record. The deck setups for the NOS and NOS/BE operating systems are illustrated in figure 5-11.

```
INP(STDNTF)

OUT(GRADF,BGD=X)

QAL(GRADF,33X2 EQ $SR$)

REF(GRADF,X20=X20,N6=N6,N1=32N1,

    (30N2 EQ $02$: N2=$04$;N2=30N2),

     X3=$    $,35$,$-1X35$,$-2)

    .

SUMMARY

 INPUT FILE - STDNTF      5 RECORDS READ

 OUTPUT FILE - GRADF      5 RECORDS WRITTEN

END OF RUN.

BENDER,R.L.          612345504   BOONEVILLE
KETTLE,V.H.          561134403   SAN JOSE
MCKEE,N.R.           456123604   BEAR VALLEY

 ACMAAMF. 80/01/31.(22) SVL SN614 NOS

07.41.03.BIN014G.
07.41.03.UCCR, 6145,         0.016KCDS.
07.41.03.USER,CDSXXXX,.
07.41.03.CHARGE,591X,693XXXX.
07.41.03.GET,STDNTF.
07.41.03.FILE(STDNTF,BT=C,RT=Z,MRL=80)
07.41.04.FILE(GRADF,BT=C,RT=Z,MRL=62)
07.41.04.FORM.
07.41.07.REWIND,GRADF.
07.41.07.COPYSBF,GRADF,OUTPUT.
07.41.07. EOI ENCOUNTERED.
07.41.07.UEAD,        0.002KUNS.
07.41.07.UEPF,        0.010KUNS.
07.41.07.UEMS,        0.770KUNS.
07.41.07.UECP,        0.537SECS.
07.41.07.AESR,        2.766UNTS.
08.06.37.UCLP, 6141,        0.320KLNS.
```

Figure 5-5. Printout of a FORM Run Under NOS

The parameters in the FILE control statement for the input file IBMPAYR have the following meaning:

| | |
|---|---|
| LT | Label type; UL indicates unlabeled |
| RT | Record type; S indicates system record type |
| MBL | Maximum block length in characters |
| MRL | Maximum record length |
| CM | Conversion mode; NO indicates no conversion |

```
       8    12
       FD   PAYROLL-MASTER
            BLOCK CONTAINS 30 RECORDS
            RECORD CONTAINS 89 CHARACTERS
            LABEL RECORDS OMITTED
            DATA RECORD IS MASTER-RECORD
            RECORDING MODE IS BINARY.
       01   MASTER-RECORD.
            02 SOC-SEC                       PIC 9(9).
            02 NAME                          PIC X(28).
            02 DEPT                          PIC 9(4).
            02 EMPLOYEE-NUMBER               PIC 9(5).
            02 DATE-OF-BIRTH.
                  03 YY                      PIC 99.
                  03 MM                      PIC 99.
                  03 DD                      PIC 99.
            02 SEX                           PIC X.
            02 RATE                          PIC 99V99 COMP-3.
            02 HRS-WORKED                    PIC 99V9 COMP-3.
            02 FICA                          PIC 9(4)V99 COMP-3.
            02 SALARY-LAST-MO                PIC 9(4)V99 COMP-3.
            02 SALARY-TO-DATE                PIC 9(5)V99 COMP-3.
            02 HOURS-VACATION                PIC 9(5).
            02 OVERTIME-HISTORY.
                  03 OVERTIME                PIC 99 OCCURS 7 TIMES.
```

| Social Security Number | Employee Name | Dept. | Emp. No. | Birth Date | SEX | Rate | Hr. Wrk. | FICA | Sal. Last Mo. | Sal. to Date | Hrs. Vac. | Overtime History |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10 | 38 | 42 | 47 | 53 | 54 | 57 | 59 | 63 | 67 | 72 | 76 |

89 Bytes

Figure 5-6. IBM COBOL Record Structure

```
558700700ROSCOE              222212345040447M0525350000156905250020255000400       50
558700660BOSCOE              456751234120743M0450400001237006000029503500070        0
569087651BRISCOE             456745123060844F0500400001100005950028000000010       16
577380902BREE                222234512092551F035025000085000350001500000015       240
```

Figure 5-7. Sample IBM Records

The parameters in the FILE control statement for the output file CDCPAYR have the following meaning:

| | |
|---|---|
| LT | Label type; UL indicates unlabeled |
| BT | Block type; K indicates RB records per block |
| MBL | Maximum block length |
| RB | Records per block in a sequential file K type block |
| RT | Record type; F indicates fixed length |
| FL | Number of characters in the F-type record (CRM expands the actual record with blanks if necessary) |
| CM | Conversion mode; YES indicates the tape file is BCD coded because it is a 7-track tape |

Figure 5-12 shows an output listing of the FORM run. The output records are printed using the FMT parameter of the PAG directive. The dayfile indicates NT57 has been assigned to the tape containing IBMPAYR; MT52 has been assigned to CDCPAYR. Refer to the appropriate reference manual for information on accessing tapes.

```
      8       12
      FD      PAYROLL-FILE
              RECORD CONTAINS 86 CHARACTERS
              BLOCK CONTAINS 30 RECORDS
              LABEL RECORDS OMITTED
              DATA RECORD IS PAY-MASTER.
      01      PAY-MASTER.
              02 EMPLOYEE-NUMBER             PIC 9(5).
              02 EMP-NAME                    PIC X(30).
              02 SOCIAL-SECURITY-NR          PIC 9(9).
              02 DATE-OF-BIRTH.
                    03 YEAR                  PIC 99.
                    03 MONTH                 PIC 99.
                    03 DAY                   PIC 99.
              02 SEX                         PIC X.
              02 DEPT                        PIC 9(4).
              02 RATE                        PIC 99V99.
              02 HRS-WORKED                  PIC 99V9.
              02 FICA                        PIC 9(4)V99.
              02 SALARY-LAST-MO              PIC 9(4)V99.
              02 SALARY-TO-DATE              PIC 9(5)V99.
              02 HOURS-VACATION-ACCRUED      PIC 9(5).
```

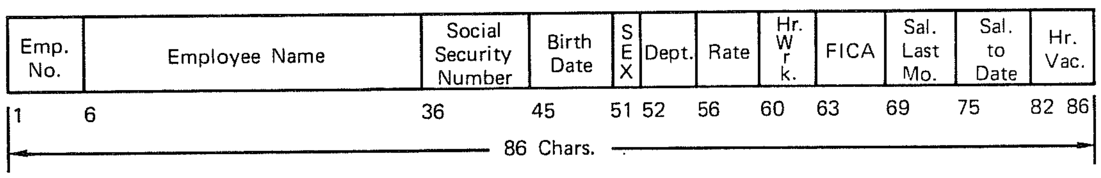


Figure 5-8. CDC COBOL Record Format

```
12345R0SCOE                 558700700040447M2222052535000015690525002C25500040
51234R0SCOE                 558700660120743M4567045C40000123700600002950350007
451239RISCOE                569C87651060844F4567050C4C00011000059500280000001
34512RREE                   577380902092551F2222035C25000085000350001500000001
```

Figure 5-9. Sample CDC Records

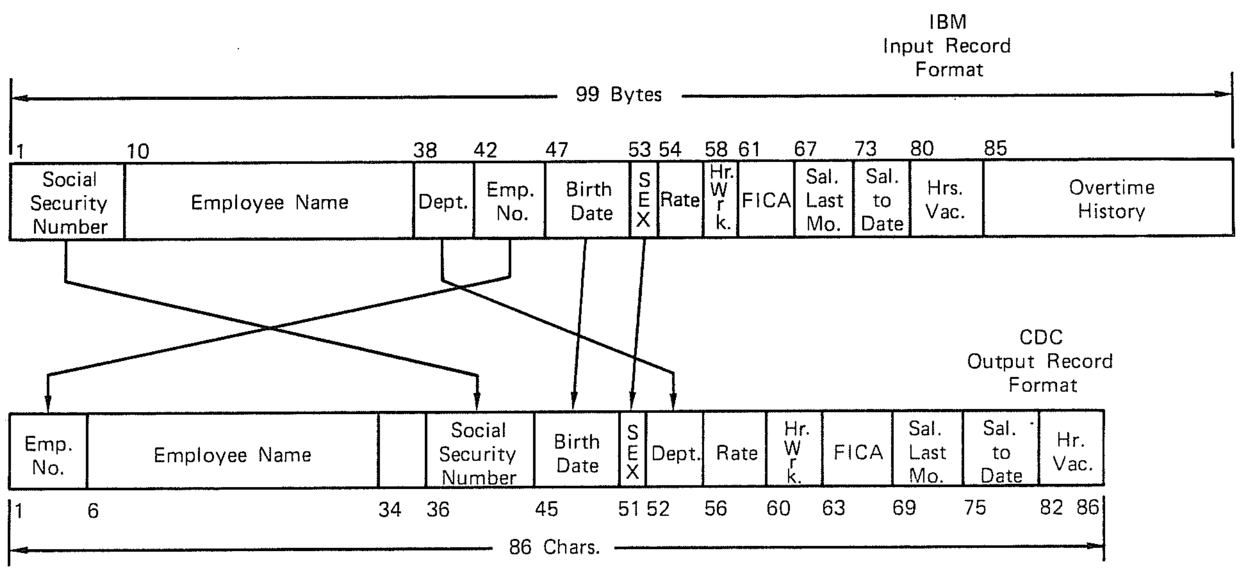


Figure 5-10. Record Reformatting

```
NOS:

JOB  statement
USER  statement
CHARGE  statement
RESOURC(HD=1,HY=1)
LABEL(IBMPAYR,D=HD,F=S,PO=R,VSN=3996)
LABEL(CDCPAYR,MT,F=S,PO=W,VSN=SCRATCH)
FILE(IBMPAYR,LT=UL,RT=S,MBL=3960,MRL=3960,MRL=3960,CM=NO)
FILE(CDCPAYR,LT=UL,BT=K,MBL=2580,RB=30,RT=F,FL=86,CM=YES)
FILE(OUTC,BT=C,RT=Z,MRL=90)
FORM.
REWIND,OUTC.
COPYCF,OUTC,OUTPUT.
RETURN(CDCPAYR,IBMPAYR)
7/8/9  in column 1
INP(IBMPAYR,REW=U,RFM=F,BLK=2970,LRL=90)
OUT(CDCPAYR)
OUT(OUTC)
CON(IBMPAYR,X53,P3X4,P2X3,2P4X6,P4X7,X20)
REF(CDCPAYR,1N5=42N5,X30=10X28,N9=1N9,X6=47X6,X1=53X1,N4=38N4,
    N4=54N4,N3=58N3,N6=61N6,N6=67N6,N7=73N7,N4=80N4)
REF(OUTC=CDCPAYR)
PAG(OUTC,FMT=D)
6/7/8/9  in column 1


NOS/BE:

JOB,NT1,MT1.
ACCOUNT  statement
LABEL(IBMPAYR,NT,HD,S,NORING,VSN=3996)
LABEL(CDCPAYR,MT,S,RING,VSN=SCRATCH)
FILE(IBMPAYR,LT=UL,RT=S,MBL=3960,MRL=3960,CM=NO)
FILE(CDCPAYR,LT=UL,BT=K,MBL=2580,RB=30,RT=F,FL=86,CM=YES)
FILE(OUTC,BT=C,RT=Z,MRL=90)
FORM.
REWIND,OUTC.
COPYCF,OUTC,OUTPUT.
7/8/9  in column 1
INP(IBMPAYR,REW=U,RFM=F,BLK=2970,LRL=90)
OUT(CDCPAYR)
OUT(OUTC)
CON(IBMPAYR,X53,P3X4,P2X3,2P4X6,P4X7,X20)
REF(CDCPAYR,1N5=42N5,X30=10X28,N9=1N9,X6=47X6,X1=53X1,N4=38N4,
    N4=54N4,N3=58N3,N6=61N6,N6=67N6,N7=73N7,N4=80N4)
REF(OUTC=CDCPAYR)
PAG(OUTC,FMT=D)
6/7/8/9  in column 1
```

Figure 5-11. Job Decks for Conversion Example

```
INP(IBMPAYR,REW=U,RFM=F,BLK=2970,LRL=90)

OUT(CDCPAYR)

OUT(OUTC)

CON(IBMPAYR,X53,P3X4,P2X3,2P4X6,P4X7,X20)

REF(CDCPAYR,1N5=42N5,X30=10X28,N9=1N9,X6=47X6,X1=53X1,N4=38N4,

     N4=54N4,N3=58N3,N6=61N6,N6=67N6,N7=73N7,N4=80N4)

REF(OUTC=CDCPAYR)

PAG(OUTC,FMT=D)


SUMMARY

 INPUT FILE - IBMPAYR      4 RECORDS READ

OUTPUT FILE - CDCPAYR      4 RECORDS WRITTEN

OUTPUT FILE - OUTC         4 RECORDS WRITTEN

END OF RUN.
```

```
 RECORD 1               90 CHARS
 12345ROSCOE                            558700700040447M2222052535000015690525002025500040

 RECORD 2               90 CHARS
 51234BOSCOE                            558700660120743M4567045040000123700600002950350007

 RECORD 3               90 CHARS
 45123BRISCOE                           569087651060844F4567050040000110000595002800000001

 RECORD 4               90 CHARS
 34512BREE                              577380902092551F2222035025000085000350001500000001
```

```
 ACMACKV. 80/02/01.(22) SVL SN614 NOS

14.47.59.BIN014G.
14.47.59.UCCR, 6145,      0.029KCDS.
14.47.59.USER,CDSXXXX,.
14.48.00.CHARGE,59XX,693AXXX.
14.48.02.RESOURC(HD=1,HY=1)
14.48.02.LABEL(IBMPAYR,D=HD,F=S,PO=R,VSN=3996)
14.52.07.NT57, ASSIGNED TO IBMPAYR, VSN=3996  .
14.52.07.LABEL(CDCPAYR,MT,F=S,PO=W,VSN=SCRATCH)
15.10.58.MT52, ASSIGNED TO CDCPAYR, VSN=****55.
15.10.58.FILE(IBMPAYR,LT=UL,RT=S,MBL=3960,MRL=3960,
15.10.58.CM=NO)
15.11.00.FILE(CDCPAYR,LT=UL,BT=K,MBL=2580,RB=30,RT=F,
15.11.00.FL=86,CM=YES)
15.11.04.FILE(OUTC,BT=C,RT=Z,MRL=90)
15.11.05.FORM.
15.11.21.REWIND,OUTC.
15.11.21.COPYCF,OUTC,OUTPUT.
15.11.21. EOI ENCOUNTERED.
15.11.21.RETURN(CDCPAYR,IBMPAYR)
15.11.21.UEAD,       0.005KUNS.
15.11.21.UEPF,       0.089KUNS.
15.11.21.UEMT,       0.029KUNS.
15.11.21.UEMS,       3.009KUNS.
15.11.21.UECP,       1.179SECS.
15.11.21.AESR,       6.210UNTS.
15.22.21.UCLP, 6141,      0.320KLNS.
```

Figure 5-12. Printout of a FORM Conversion Example

# CHARACTER DATA INPUT, OUTPUT, AND CENTRAL MEMORY REPRESENTATION A

This appendix describes the code and character sets used by the operating system local batch device driver programs, magnetic tape driver programs, and network terminal communication products. This appendix does not describe how other products associate certain graphic or control characters with specific binary code values for collating or syntax processing purposes. The main text of this manual describes such associations that are relevant to the reader. CDC and ASCII character set collating sequence tables are located in appendix N.

## CHARACTER SETS AND CODE SETS

A character set differs from a code set. A character set is a set of graphic and/or control character symbols. A code set is a numbering system used to represent each character within a character set. Characters exist outside the computer system and communication network; codes are received, stored, retrieved, and transmitted within the computer system and network.

When this manual mentions the ASCII 128-character set or the 7-bit ASCII code set, it is referring to the character set and code set defined as the American National Standard Code for Information Interchange (ASCII, ANSI Standard X3.4-1977). References in this manual to an ASCII character set or an ASCII code set do not necessarily apply to the 128-character, 7-bit ASCII code set.

## GRAPHIC AND CONTROL CHARACTERS

A graphic character can be displayed or printed. Examples of graphic characters are the characters A through Z, a blank, and the digits 0 through 9. A control character is not a graphic character; a control character initiates, modifies, or stops a control operation. An example of a control character is the backspace character, which moves the terminal carriage or cursor back one space. Although a control character is not a graphic character, some terminals use a graphic representation for control characters.

## CODED AND BINARY CHARACTER DATA

Character codes can be interpreted as coded character data or as binary character data. Coded character data is converted by default from one code set representation to another as it enters or leaves the computer system; for example, data received from a terminal or sent to a magnetic tape unit is converted. Binary character data is not converted as it enters or leaves the system. Character codes are not converted when moved within the system; for example, data transferred to or from mass storage is not converted.

The distinction between coded character data and binary character data is important when reading or punching cards and when reading or writing magnetic tape. Only coded character data can be properly reproduced as characters on a line printer. Only binary character data can properly represent characters on a punched card when the data cannot be stored as display code.

The distinction between binary character data and characters represented by binary data (such as peripheral equipment instruction codes) is also important. Only binary noncharacter data can properly reproduce characters on a plotter.

## CHARACTER SET TABLES

The character set tables in this appendix are designed so that the user can find the character represented by a code (such as in a dump) or find the code that represents a character. To find the character represented by a code, the user looks up the code in the column listing the appropriate code set and then finds the character on that line in the column listing the appropriate character set. To find the code that represents a character, the user looks up the character and then finds the code on the same line in the appropriate column.

## NETWORK OPERATING SYSTEMS

NOS and NOS/BE support the following character sets:

- CDC graphic 64-character set
- CDC graphic 63-character set
- ASCII graphic 64-character set
- ASCII graphic 63-character set
- ASCII graphic 95-character set
- ASCII 128-character set

Each site selects either a 64-character set or a 63-character set. The differences between the codes of a 63-character set and the codes of a 64-character set are described under Character Set Anomalies. Any reference in this appendix to a 64-character set implies either a 63- or 64-character set unless otherwise stated.

NOS supports the following code sets to represent its character sets in central memory:

- 6-bit display code
- 12-bit ASCII code
- 6/12-bit display code

NOS/BE supports the following code sets to represent its character sets in central memory:

- 6-bit display code
- 12-bit ASCII code

Under both NOS and NOS/BE, the 6-bit display code is a set of octal codes from 00 to 77, inclusive.

Under both NOS and NOS/BE, the 12-bit ASCII code is the ASCII 7-bit code right-justified in a 12-bit byte. The bits are numbered from the right starting with 0; bits 0 through 6 contain the ASCII code, bits 7 through 10 contain zeros, and bit 11 distinguishes the 12-bit ASCII 0000 code from the 12-bit 0000 end-of-line byte. The octal values for the 12-bit codes are 0001 through 0177 and 4000.

Under NOS, the 6/12-bit display code is a combination of 6-bit codes and 12-bit codes. The octal values for the 6-bit codes are 00 through 77, excluding 74 and 76. (The interpretation of the 00 and 63 codes is described under Character Set Anomalies in this appendix.) The octal 12-bit codes begin with either 74 or 76 and are followed by a 6-bit code. Thus, 74 and 76 are escape codes and are never used as 6-bit codes within the 6/12-bit display code set. The octal values of the 12-bit codes are: 7401, 7402, 7404, 7407, and 7601 through 7677. The other 12-bit codes, 74xx and 7600, are undefined.

## CHARACTER SET ANOMALIES

The operating system input/output software and some products interpret two codes differently when the installation selects a 63-character set rather than a 64-character set. If a site uses a 63-character set: the colon (:) graphic character is always represented by a 6-bit display code value of 63 octal; 6-bit display code 00 is undefined (it has no associated graphic or punched card code); the percent (%) graphic does not exist, and translations produce a space (55 octal).

However, under NOS, if the site uses a 64-character set, output of an octal 7404 6/12-bit display code or a 6-bit display code value of 00 produces a colon. In ASCII mode, a colon can be input only as a 7404 6/12-bit display code. Undefined 6/12-bit display codes in output files produce unpredictable results and should be avoided.

Under NOS/BE, if the site uses a 64-character set, output of a 6-bit display code value of 00 produces the colon (:) graphic character. A colon can be input only as a 00 code.

Under both NOS and NOS/BE, two consecutive 6-bit display code values of 00 can be confused with the 12-bit 0000 end-of-line byte and should be avoided.

Translation of 12-bit ASCII to 6-bit display code causes character set folding from the 128-character ASCII set to the 63- or 64-character ASCII subset, with the special character substitutions shown in figure A-1.

## INTERACTIVE TERMINAL USERS

NOS and NOS/BE systems support display consoles and teletypewriters that use code sets other than 7-bit ASCII codes for communication or use graphics other than those defined in an ASCII character set. Data exchanged with such terminals is translated as described under Terminal Transmission Modes in this appendix. The following description applies only to terminals which use 7-bit ASCII codes and the ASCII character set.

### ASCII Data Exchange Modes

Table A-1 shows the character sets and code sets available to an Interactive Facility (IAF) or INTERCOM user. Table A-2 shows the octal and hexadecimal 7-bit ASCII code for each ASCII character, and can be used to convert codes from octal to hexadecimal. (Under NOS using network product software, certain terminal definition commands require hexadecimal specification of a 7-bit ASCII code.)

#### IAF Usage

IAF supports both normalized mode and transparent mode transmissions through the network. These transmission modes are described under Terminal Transmission Modes in this appendix. For additional information refer to the NOS Version 2 Reference Set, Volume 3 System Commands.

IAF treats normalized mode transmissions as coded character data. IAF converts these transmissions to or from either 6-bit or 6/12-bit display code.

IAF treats transparent mode transmissions as binary character data. Transparent mode input or output uses 12-bit bytes, with bit 11 always set to 1; for ASCII terminals, transparent mode input and output occurs in the 12-bit ASCII code shown in table A-1, but the leftmost digit is 4 instead of 0.

When the NORMAL command is in effect, IAF assumes the ASCII graphic 64-character set is used and translates all input and output to or from display code. When the ASCII command is in effect, IAF assumes the ASCII 128-character set is used and translates all input and output to or from 6/12-bit display code.

| | | 63- or 64-Character Subset | | |
|---|---|---|---|---|
| 12-Bit ASCII (Octal) | | 6-Bit Display Code (Octal) | | 12-Bit ASCII (Octal) |
| 0140 (`) | | 74 (@) | | 0100 (@) |
| 0173 ({) | | 61 ([) | | 0133 ([) |
| 0174 (\|) | Input → | 75 (\) | Output → | 0134 (\) |
| 0175 (}) | | 62 (]) | | 0135 (]) |
| 0176 (~) | | 76 (^) | | 0136 (^) |

Figure A-1. ASCII Character Folding

The IAF user can convert a 6/12-bit display code file to a 12-bit ASCII code file using the NOS FCOPY control statement. The resulting 12-bit ASCII file can be routed to a line printer but the file cannot be output through IAF.

INTERCOM Usage

INTERCOM supports only normalized mode transmissions through the network. This transmission mode is described under Terminal Transmission Modes in this appendix. Refer to the INTERCOM Version 5 Reference Manual for additional information.

INTERCOM treats normalized mode transmissions as coded character data; INTERCOM converts these transmissions to or from 6-bit display code unless the ASCII-128 or ASCII-256 option is used. All communication between INTERCOM and ASCII terminals using any parity setting occurs in the 12-bit ASCII code shown in table A-1.

COMPASS and FORTRAN users can send or receive a 12-bit code byte if their terminal supports the code set contained in the byte. In ASCII-128 mode, INTERCOM assumes the ASCII 128-character set is used and does not translate to or from display code. In ASCII-256 mode, INTERCOM assumes the ASCII 128-character set is used and does not translate to or from display code, or set and clear the eighth bit of the 12-bit byte.

BASIC users can send and receive lowercase and uppercase characters or control codes using the 12-bit ASCII code if the terminal supports such characters; BASIC represents coded character data in central memory using 6/12-bit display code under both NOS and NOS/BE.

## Terminal Transmission Modes

Coded character data can be exchanged with an interactive terminal in two transmission modes. These two modes, normalized mode and transparent mode, correspond to the types of character code editing and translation performed by the network software during input and output operations.

Under NOS, the terminal operator can change the input transmission mode by using a terminal definition command (sometimes called a Terminal Interface Program command). The application program providing the terminal facility service can change the input or output transmission mode.

Under NOS/BE, the input and output transmission modes are both fixed as normalized mode, but an application program with a connected input or output file can control character set mapping and parity bit settings.

Normalized Mode Transmissions

Normalized mode is the initial and default mode used for both input and output transmissions. The network software translates normalized mode data to or from the transmission code used by the terminal into or from the 7-bit ASCII code shown in table A-2. (Tables A-1 and A-3 through A-7 are provided for use while coding a program to run under the operating system; they do not describe character code conversions within the network.) Translation of a specific terminal transmission code to or from a specific 7-bit ASCII code depends on the terminal class in which the network software places the terminal.

The following paragraphs summarize the general case for translations. The reader can extend this generalized description by using the other tables to determine character set mapping for functions initiated from a terminal. For example, the description under Terminal Output Character Sets can be used to predict whether a lowercase ASCII character stored in 6/12-bit display code can appear on an EBCDIC or external BCD terminal; if an ASCII character passes through the network represented in 7-bit ASCII code as normalized mode data, it probably can be represented on an EBCDIC terminal, but it is always transformed to an uppercase character on a mode 4A ASCII terminal.

Table A-2 contains the ASCII 128-character set supported by both NOS and NOS/BE network software. The ASCII 96-character subset in the rightmost six columns minus the deletion character (DEL) comprises the graphic 95-character subset; the DEL is not a graphic character, although some terminals graphically represent it. The graphic 64-character subset comprises the middle four columns. Only the characters in this 64-character subset have 6-bit display code equivalents.

Terminals which support an ASCII graphic 64-character subset actually use a subset of up to 96 characters, consisting of the graphic 64-character subset and the control characters of columns 1 and 2; often, the DEL character in column 7 is included. Terminals which support an ASCII graphic 95-character or 96-character subset actually might use all 128 characters.

The hexadecimal value of the 7-bit code for each character in table A-2 consists of the character's column number in the table, followed by its row number. For example, N is in row E of column 4, so its hexadecimal value is 4E. The octal value of the code when it is right-justified in an 8-bit byte appears beneath the character graphic or mnemonic. The binary value of the code consists of the bit values shown, placed in the order given by the subscripts for the letter b; for example, N is 1001110.

Terminal Output Character Subsets -- Although the network supports the ASCII 128-character set, some terminals restrict output to a smaller character set. This restriction is supported by replacing the control characters in columns 0 and 1 of table A-2 and the DEL character in column 7 with blanks to produce the ASCII graphic 95-character subset, and replacing the characters in columns 6 and 7 with the corresponding characters from columns 4 and 5, respectively, to produce the ASCII graphic 64-character subset.

Terminal Input Character Subsets and Supersets -- Although the network supports the ASCII 128-character set, some terminals restrict input to a smaller character set or permit input of a larger character set. A character input from a device using a character set other than ASCII is converted to an equivalent ASCII character; terminal characters without ASCII character equivalents are represented by the ASCII code for a space.

Under NOS, site-written terminal-servicing facility programs can also cause input or output character replacement, conversion, or deletion by exchanging data with the network in 6-bit display code.

Input Restrictions -- Under both NOS and NOS/BE, the network software by default automatically deletes codes associated with terminal communication protocols or terminal hardware functions. These codes usually represent the cancel, backspace, linefeed, carriage return, and deletion characters. If device input control or paper tape support is requested, the device control 3 and device control 1 codes also are deleted. Some of these code deletions can be suppressed under NOS by using editing options (refer to the FA and SE terminal definition parameters in the NOS Version 2 Reference Set, Volume 3 System Commands).

NOS Output Restrictions -- All codes sent by an application program are transmitted to the terminal. However, the 12-bit ASCII code 0037 (octal), the 6/12-bit display code 7677 (octal), and the 7-bit ASCII code 1F (hexadecimal) should be avoided in normalized mode output. The network software interprets the unit separator character represented by these codes as an end-of-line indicator. The processing of application program-supplied unit separators causes incorrect formatting of output and can cause loss of other output characters.

NOS/BE Output Restrictions -- The 7-bit ASCII null code cannot be distinguished from the 12-bit 0000 end-of-line byte and cannot be transmitted.

NOS Input Parity Processing -- The network software does not preserve the parity of the terminal transmission code in the corresponding ASCII code. An ASCII code received by the terminal-servicing facility program always contains zero as its eighth bit.

NOS/BE Input Parity Processing -- The network software does not preserve the parity of the terminal transmission code in the corresponding 12-bit ASCII code unless the ASCII-256 option of INTERCOM is used. When the ASCII-128 option is used, the 12-bit ASCII code received by an application program always contains a zero as its eighth bit.

Output Parity Processing -- Under NOS, the network software provides the parity bit setting appropriate for the terminal being serviced, even when the software is translating from ASCII characters with zero parity bit settings. Under NOS/BE using either display code or the ASCII-128 option, the network software provides the parity bit setting appropriate for the terminal being serviced, even when the software is translating from ASCII character codes with zero parity bit settings.

### Transparent Mode Transmissions

Transparent mode is selected separately for input and output transmissions.

During transparent mode input, the parity bit is stripped from each terminal transmission code (unless the N or I parity option has been selected by a terminal definition command), and the transmission code is placed in an 8-bit byte without translation to 7-bit ASCII code. Line transmission protocol characters are deleted from mode 4 terminal input. When the 8-bit bytes arrive in the host computer, a terminal servicing facility program can right-justify the bytes within a 12-bit byte.

During transparent mode output, processing similar to that performed for input occurs. When the host computer transmits 12-bit bytes, the leftmost 4 bits (bits 11 through 8) are discarded. The code in each 8-bit byte received by the network software is not translated. The parity bit appropriate for the terminal class is altered as indicated by the parity option in effect for the terminal. The codes are then transmitted to the terminal in bytes of a length appropriate for the terminal class. Line transmission protocol characters are inserted into mode 4 terminal output.

## LOCAL BATCH USERS

Table A-3 lists the CDC graphic 64-character set, the ASCII graphic 64-character set, and the ASCII graphic 95-character set available on local batch devices. This table also lists the code sets and card keypunch codes (026 and 029) that represent the characters.

The 64-character sets use 6-bit display code as their code set; the 95-character set uses 12-bit ASCII code. The 95-character set is composed of all the characters in the ASCII 128-character set that can be printed at a line printer (refer to Line Printer Output). Only 12-bit ASCII code files can be printed using the ASCII graphic 95-character set. The octal 12-bit ASCII codes 0040 through 0176 represent the 95-character set. An octal 12-bit ASCII code outside of the range 0040 through 0176 represents an unprintable character.

To print a 6/12-bit display code file, the user must convert the file to 12-bit ASCII code. The FCOPY command can change the 6/12-bit display codes shown in table A-3 to the 12-bit ASCII codes shown.

### Line Printer Output

The printer train used on the line printer to which a file is sent determines which batch character set is printed. The following CDC print trains match the batch character sets in table A-3:

| Character Set | Print Train |
|---|---|
| CDC graphic 64-character set | 596-1 |
| ASCII graphic 64-character set | 596-5 |
| ASCII graphic 95-character set | 596-6 |

The characters of the default 596-1 print train are listed in the table A-3 column labeled CDC Graphic (64-Character Set); the 596-5 print train characters are listed in the table A-3 column labeled ASCII Graphic (64-Character Set); and the 596-6 print train characters are listed in the table A-3 column labeled ASCII Graphic (95-Character Set).

#### Under NOS

If an unprintable character exists in a line, NOS marks the condition by printing the number sign (#) in the first printable column of the line. A space replaces the unprintable character within the line.

When a transmission error occurs during the printing of a line, NOS makes up to five attempts to reprint the line. The CDC graphic print train prints a concatenation symbol (↱) in the first column of the repeated line following a line containing errors. The ASCII print trains print an underline (_) instead of the concatenation symbol.

After the fifth attempt, the setting of sense switch 1 for the batch input and output control point determines further processing. NOS either rewinds the file and returns it to the print queue, or ignores the transmission errors.

Under NOS/BE

If an unprintable character exists in a line, NOS/BE marks the condition by printing a diagnostic message on the next line. A space replaces the unprintable character within the line.

When a transmission error occurs during the printing of a line, NOS/BE stops the printer and alerts the system console operator. The console operator can either ignore the condition and continue printing the file, or the operator can rewind the file and return it to the print queue.

### Punched Card Input and Output

A character represented by multiple punches in a single column has its punch pattern identified by numbers and hyphens. For example, the punches representing an exclamation point are identified as 11-0; this notation means both rows 11 and 0 are punched in the same column.

A multiple punch pattern that represents something other than a character is identified by numbers and slashes. For example, the punches representing the end of an input file are identified as 6/7/8/9; this notation means rows 6 through 9 are punched in the same column.

Under NOS

Coded character data is exchanged with card readers or card punches according to the translations shown in table A-3. As indicated in the table, other card keypunch codes are available for input of the ASCII and CDC characters [ and ]. NOS cannot read or punch the 95-character set as coded character data.

Each site chooses either 026 or 029 as its default keypunch code. NOS begins reading an input deck in the default code (regardless of the character set in use). The user can specify the alternate keypunch code by punching a 26 or 29 in columns 79 and 80 of any job card, 6/7/9 card, or 7/8/9 card. The specified translation continues throughout the job unless the alternate keypunch code translation is specified on a subsequent 6/7/9 or 7/8/9 card.

A 5/7/9 card with a punch in column 1 changes keypunch code translation if the card is read immediately before or after a 7/8/9 card. A space (no punch) in column 2 of such a 5/7/9 card selects 026 translation mode; a 9 punch in column 2 selects 029 translation mode. The specified translation continues until a similar 5/7/9 card or a similar 7/8/9 card is encountered, or the job ends.

The 5/7/9 card also allows literal input when 4/5/6/7/8/9 is punched in column 2. Literal input can be used to read 80-column binary character data within a punched card deck of coded character data.

Literal cards are stored with each column represented in a 12-bit byte (a row 12 punch is represented by a 1 in bit 11, row 11 by a 1 in bit 10, row 0 by a 1 in bit 9, and rows 1 through 9 by 1's in bits 8 through 0 of the byte), using 16 central memory words per card. Literal input cards are read until another 5/7/9 card with 4/5/6/7/8/9 punched in column 2 is read. The next card can specify a new conversion mode.

If the card following the 5/7/9, 6/7/9, or 7/8/9 card has a 7 and a 9 punched in column 1, the section of the job deck following it contains system binary cards (as described in the NOS Version 2 Reference Set, Volume 3 System Commands).

Under NOS/BE

Coded character data is exchanged with card readers or card punches according to the translations shown in table A-3. As indicated in the table, other card keypunch codes are available for input of the CDC characters ∨ and < or the ASCII characters ! and <. NOS/BE cannot read or punch the ASCII 95-character set as coded character data.

Each site chooses either 026 or 029 as its default keypunch code. NOS/BE begins reading an input deck in the default code (regardless of the character set in use). The user can specify the alternate keypunch code by punching a 26 or 29 in columns 79 and 80 of the job statement or in columns 79 and 80 of any 7/8/9 card. The specified translation continues throughout the job unless the alternate keypunch code translation is specified on a subsequent 7/8/9 card.

A card with all 12 rows of column 1 punched and all of one other column punched can be followed by 80-column cards of free-form binary data. These binary data cards are read as described for NOS literal data until another card with 12 punches in column 1 and in one other column occurs, or until the job ends. The next card is interpreted as coded data.

If the card following the job card or 7/8/9 card has a 7 and a 9 punched in column 1, the section of the job deck following it contains standard binary cards (as described in the NOS/BE Version 1 Reference Manual).

## REMOTE BATCH USERS

Remote batch console input and output is restricted to normalized mode transmission. Normalized mode is described under Terminal Transmission Modes in this appendix.

The abilities to select alternate keypunch code translations, to read binary cards, to output plotter files, and to print lowercase characters depend upon the remote terminal equipment.

Remote batch terminal support under NOS is described in the Remote Batch Facility Version 1 Reference Manual. Remote batch terminal support under NOS/BE is described in the INTERCOM Version 5 Reference Manual.

## MAGNETIC TAPE USERS

The character and code sets used for reading and writing magnetic tapes depend on whether coded or binary data is read or written and on whether the tape is 7-track or 9-track.

## Coded Data Exchanges

Coded character data to be copied from mass storage to magnetic tape is assumed to be stored in a 63- or 64-character 6-bit display code. The operating system magnetic tape driver program converts the data to 6-bit external BCD code when writing a coded 7-track tape and to 7-bit ASCII or 8-bit EBCDIC code (as specified on the tape assignment statement) when writing a coded 9-track tape.

Coded character data copied to mass storage from magnetic tape is stored in a 63- or 64-character 6-bit display code. The operating system magnetic tape driver program converts the data from 6-bit external BCD code when reading a coded 7-track tape and from 7-bit ASCII or 8-bit EBCDIC code (as specified on the tape assignment statement) when reading a coded 9-track tape.

To read and write lowercase character 7-bit ASCII or 8-bit EBCDIC codes or to read and write control codes, the user must assign a 7-track or 9-track tape in binary mode.

### Seven-Track Tape Input and Output

Table A-4 shows the code and character set conversions between 6-bit external BCD and 6-bit display code for 7-track tapes. Because only 63 characters can be represented in 7-track even parity, one of the 64 display codes is lost in conversion to and from external BCD code.

Figure A-2 shows the differences in 7-track tape conversion that depend on whether the system uses the 63-character or 64-character set. The ASCII character for the specified character code is shown in parentheses. The output arrows show how the display code changes when it is written on tape in external BCD. The input arrows show how the external BCD code changes when the tape is read and converted to display code.

63-Character Set

| Display Code | | External BCD | | Display Code |
|---|---|---|---|---|
| 00 | | 16(%) | | 00 |
| 33(0) | Output → | 12(0) | Input → | 33(0) |
| 63(:) | | 12(0) | | 33(0) |

64-Character Set

| Display Code | | External BCD | | Display Code |
|---|---|---|---|---|
| 00(:) | | 12(0) | | 33(0) |
| 33(0) | Output → | 12(0) | Input → | 33(0) |
| 63(%) | | 16(%) | | 63(%) |

Figure A-2. Magnetic Tape Code Conversions

### Nine-Track Tape Input and Output

Table A-5 lists the conversions between the 7-bit ASCII code used on the tape and the 6-bit display code used within the system. Table A-6 lists the conversions between the 8-bit EBCDIC code used on the tape and the 6-bit display code used within the system.

When an ASCII or EBCDIC code representing a lowercase character is read from a 9-track magnetic tape, it is converted to its uppercase character 6-bit display code equivalent. Any EBCDIC code not listed in table A-6 is converted to display code 55 (octal) and becomes a space. Any code between 80 (hexadecimal) and FF (hexadecimal) read from an ASCII tape is converted to display code 00.

## Binary Character Data Exchanges

Binary character data exchanged between central memory files and magnetic tape is transferred as a string of bytes without conversion of the byte contents. The grouping of the bytes and the number of bits in each byte depend on whether 7-track or 9-track tape is being used.

### Seven-Track Tape Input and Output

Each binary data character code written to or read from 7-track magnetic tape is assumed to be stored in a 6-bit byte, such as the system uses for 63- or 64-character 6-bit display code. Seven-bit ASCII and 8-bit EBCDIC codes can only be read from or written to 7-track magnetic tape as binary character data if each code is stored within a 12-bit byte as if it were two character codes.

### Nine-Track Tape Input and Output

Each binary data character code exchanged between central memory files and 9-track magnetic tape is assumed to be stored in an 8-bit or 12-bit byte. During such binary data transfers, the 6/12-bit display codes and 12-bit ASCII codes shown in table A-1, the 7-bit ASCII codes shown in table A-2, or or the 8-bit hexadecimal EBCDIC codes shown in table A-7 can be read or written. The 7-bit ASCII codes and 8-bit EBCDIC codes can be exchanged either in an unformatted form or right-justified within a zero-filled 12-bit byte of memory.

When 9-track tape is written, every pair of 12-bit memory bytes becomes three 8-bit tape bytes; when 9-track tape is read, every three 8-bit tape bytes become a pair of 12-bit memory bytes. Because of the 12-bit byte pairs, codes not packed in 12-bit bytes are exchanged in their unpacked form, while codes packed in 12-bit bytes are exchanged in packed form.

When an odd number of central memory words is read or written, the lower four bits of the last 8-bit byte (bits 0 through 3 of the last word) are not used. For example, three central memory words are written on tape as 22 8-bit bytes (7.5 pairs of 12-bit bytes) and the remaining four bits are ignored.

## CODE CONVERSION AIDS

Table A-7 contains the octal values of each 8-bit EBCDIC code right-justified in a 12-bit byte with zero fill. This 12-bit EBCDIC code can be written or read using the FORM and the 8-Bit Subroutines utilities.

## SCOPE 2 OPERATING SYSTEM

Interactive terminal and remote batch operations through HELLO7 under SCOPE 2 affect character data in the manner described for INTERCOM under NOS/BE. Magnetic tape and line printer operations affect character data in the manner described for NOS/BE. Refer to the SCOPE Version 2 Reference Manual for details.

Coded character data is exchanged with local batch card readers or card punches according to the translations shown in table A-3. As indicated in the table, other card keypunch codes are available for input of the CDC characters ∨ and < , or the ASCII characters ! and < . SCOPE 2 cannot read or punch the 95-character set as coded character data.

When SCOPE 2 reads an input deck, no keypunch code default exists (regardless of the character set in use). The 026 or 029 keypunch code translations are specified by a 26 or 29 punched in columns 79 and 80 of the job statement or in columns 79 and 80 of any 7/8/9 card. The specified translation continues throughout the job unless the alternate keypunch code translation is specified on a subsequent 7/8/9 card.

A job card or 7/8/9 card with blanks in columns 79 and 80 indicates that the following section of the job deck is either coded or binary data, as indicated by the subsequent card.

If the subsequent card has all of column 1 punched (that is, 12 punches in column 1) and all of one other column punched, it can be followed by 80-column cards of free-form binary data. These binary data cards are read in the manner described for NOS literal data cards until another card with 12 punches in column 1 and in one other column occurs, or until the job ends. The next card is interpreted as coded data.

If the card following the keypunch-code changing card has a 7 and a 9 punched in column 1, the section of the job deck following it contains SCOPE 2 binary cards (as described in the SCOPE Version 2 Reference Manual).

If the card following the keypunch-code changing card is anything other than a free-form binary or SCOPE 2 binary indicator card, the section of the deck contains cards punched in the code last selected.

TABLE A-1. INTERACTIVE TERMINAL CHARACTER SETS

| Character Sets | | Code Sets | | |
|---|---|---|---|---|
| ASCII Graphic (64-Character Set) | ASCII Character (128-Character Set) | Octal 6-Bit Display Code | Octal 6/12-Bit Display Code† | Octal 12-Bit ASCII Code |
| : colon†† | | 00†† | | |
| A | A | 01 | 01 | 0101 |
| B | B | 02 | 02 | 0102 |
| C | C | 03 | 03 | 0103 |
| D | D | 04 | 04 | 0104 |
| E | E | 05 | 05 | 0105 |
| F | F | 06 | 06 | 0106 |
| G | G | 07 | 07 | 0107 |
| H | H | 10 | 10 | 0110 |
| I | I | 11 | 11 | 0111 |
| J | J | 12 | 12 | 0112 |
| K | K | 13 | 13 | 0113 |
| L | L | 14 | 14 | 0114 |
| M | M | 15 | 15 | 0115 |
| N | N | 16 | 16 | 0116 |
| O | O | 17 | 17 | 0117 |
| P | P | 20 | 20 | 0120 |
| Q | Q | 21 | 21 | 0121 |
| R | R | 22 | 22 | 0122 |
| S | S | 23 | 23 | 0123 |
| T | T | 24 | 24 | 0124 |
| U | U | 25 | 25 | 0125 |
| V | V | 26 | 26 | 0126 |
| W | W | 27 | 27 | 0127 |
| X | X | 30 | 30 | 0130 |
| Y | Y | 31 | 31 | 0131 |
| Z | Z | 32 | 32 | 0132 |
| 0 | 0 | 33 | 33 | 0060 |
| 1 | 1 | 34 | 34 | 0061 |
| 2 | 2 | 35 | 35 | 0062 |
| 3 | 3 | 36 | 36 | 0063 |
| 4 | 4 | 37 | 37 | 0064 |
| 5 | 5 | 40 | 40 | 0065 |
| 6 | 6 | 41 | 41 | 0066 |
| 7 | 7 | 42 | 42 | 0067 |
| 8 | 8 | 43 | 43 | 0070 |
| 9 | 9 | 44 | 44 | 0071 |
| + plus | + plus | 45 | 45 | 0053 |
| - hyphen (minus) | - hyphen (minus) | 46 | 46 | 0055 |
| * asterisk | * asterisk | 47 | 47 | 0052 |
| / slant | / slant | 50 | 50 | 0057 |
| ( opening parenthesis | ( opening parenthesis | 51 | 51 | 0050 |
| ) closing parenthesis | ) closing parenthesis | 52 | 52 | 0051 |
| $ dollar sign | $ dollar sign | 53 | 53 | 0044 |
| = equals | = equals | 54 | 54 | 0075 |
| space | space | 55 | 55 | 0040 |
| , comma | , comma | 56 | 56 | 0054 |
| . period | . period | 57 | 57 | 0056 |
| # number sign | # number sign | 60 | 60 | 0043 |
| [ opening bracket | [ opening bracket | 61 | 61 | 0133 |
| ] closing bracket | ] closing bracket | 62 | 62 | 0135 |
| % percent sign†† | % percent sign†† | 63†† | 63†† | 0045 |
| " quotation mark | " quotation mark | 64 | 64 | 0042 |
| _ underline | _ underline | 65 | 65 | 0137 |
| ! exclamation point | ! exclamation point | 66 | 66 | 0041 |
| & ampersand | & ampersand | 67 | 67 | 0046 |
| ' apostrophe | ' apostrophe | 70 | 70 | 0047 |
| ? question mark | ? question mark | 71 | 71 | 0077 |

TABLE A-1. INTERACTIVE TERMINAL CHARACTER SETS (Contd)

| Character Sets | | Code Sets | | |
|---|---|---|---|---|
| ASCII Graphic (64-Character Set) | ASCII Character (128-Character Set) | Octal 6-Bit Display Code | Octal 6/12-Bit Display Code† | Octal 12-Bit ASCII Code |
| < less than | < less than | 72 | 72 | 0074 |
| > greater than | > greater than | 73 | 73 | 0076 |
| @ commmercial at | @ commercial at | 74†† | 7401†† | 0100 |
| \ reverse slant | \ reverse slant | 75 | 75 | 0134 |
| ^ circumflex | | 76 | | |
| ; semicolon | ; semicolon | 77 | 77 | 0073 |
| | ^ circumflex | 76†† | 7402 | 0136 |
| | : colon†† | 74†† | 7404†† | 0072 |
| | ` grave accent | | 7407 | 0140 |
| | a | | 7601 | 0141 |
| | b | | 7602 | 0142 |
| | c | | 7603 | 0143 |
| | d | | 7604 | 0144 |
| | e | | 7605 | 0145 |
| | f | | 7606 | 0146 |
| | g | | 7607 | 0147 |
| | h | | 7610 | 0150 |
| | i | | 7611 | 0151 |
| | j | | 7612 | 0152 |
| | k | | 7613 | 0153 |
| | l | | 7614 | 0154 |
| | m | | 7615 | 0155 |
| | n | | 7616 | 0156 |
| | o | | 7617 | 0157 |
| | p | | 7620 | 0160 |
| | q | | 7621 | 0161 |
| | r | | 7622 | 0162 |
| | s | | 7623 | 0163 |
| | t | | 7624 | 0164 |
| | u | | 7625 | 0165 |
| | v | | 7626 | 0166 |
| | w | | 7627 | 0167 |
| | x | | 7630 | 0170 |
| | y | | 7631 | 0171 |
| | z | | 7632 | 0172 |
| | { opening brace | 61†† | 7633 | 0173 |
| | \| vertical line | 75†† | 7634 | 0174 |
| | } closing brace | 62†† | 7635 | 0175 |
| | ~ tilde | 76†† | 7636 | 0176 |
| | NUL | | 7640 | 4000 |
| | SOH | | 7641 | 0001 |
| | STX | | 7642 | 0002 |
| | ETX | | 7643 | 0003 |
| | EOT | | 7644 | 0004 |
| | ENQ | | 7645 | 0005 |
| | ACK | | 7646 | 0006 |
| | BEL | | 7647 | 0007 |
| | BS | | 7650 | 0010 |
| | HT | | 7651 | 0011 |
| | LF | | 7652 | 0012 |
| | VT | | 7653 | 0013 |
| | FF | | 7654 | 0014 |
| | CR | | 7655 | 0015 |
| | SO | | 7656 | 0016 |
| | SI | | 7657 | 0017 |
| | DEL | | 7637 | 0177 |
| | DLE | | 7660 | 0020 |

TABLE A-1. INTERACTIVE TERMINAL CHARACTER SETS (Contd)

| Character Sets | | Code Sets | | |
|---|---|---|---|---|
| ASCII Graphic (64-Character Set) | ASCII Character (128-Character Set) | Octal 6-Bit Display Code | Octal 6/12-Bit Display Code† | Octal 12-Bit ASCII Code |
| | DC1 | | 7661 | 0021 |
| | DC2 | | 7662 | 0022 |
| | DC3 | | 7663 | 0023 |
| | DC4 | | 7664 | 0024 |
| | NAK | | 7665 | 0025 |
| | SYN | | 7666 | 0026 |
| | ETB | | 7667 | 0027 |
| | CAN | | 7670 | 0030 |
| | EM | | 7671 | 0031 |
| | SUB | | 7672 | 0032 |
| | ESC | | 7673 | 0033 |
| | FS | | 7674 | 0034 |
| | GS | | 7675 | 0035 |
| | RS | | 7676 | 0036 |
| | US | | 7677 | 0037 |

†Available only on NOS.

††Character or code interpretation depends on context. Refer to Character Set Anomalies in the text.

## TABLE A-2. 7-BIT ASCII CODE AND CHARACTER SETS

|← 128-Character Set (columns 0–7) →|
|← 96-Character Subset (columns 2–7) →|
|← Graphic 64-Character Subset (columns 2–5) →|

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $b_7$ → | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $b_6$ → | | | | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $b_5$ → | | | | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Bits $b_4$ ↓ | $b_3$ ↓ | $b_2$ ↓ | $b_1$ ↓ | Row ↓ Column → | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 0 | 0 | 0 | NUL 000 | DLE 020 | SP 040 | 0 060 | @ 100 | P 120 | ` 140 | p 160 |
| 0 | 0 | 0 | 1 | 1 | SOH 001 | DC1 021 | ! 041 | 1 061 | A 101 | Q 121 | a 141 | q 161 |
| 0 | 0 | 1 | 0 | 2 | STX 002 | DC2 022 | " 042 | 2 062 | B 102 | R 122 | b 142 | r 162 |
| 0 | 0 | 1 | 1 | 3 | ETX 003 | DC3 023 | # 043 | 3 063 | C 103 | S 123 | c 143 | s 163 |
| 0 | 1 | 0 | 0 | 4 | EOT 004 | DC4 024 | $ 044 | 4 064 | D 104 | T 124 | d 144 | t 164 |
| 0 | 1 | 0 | 1 | 5 | ENQ 005 | NAK 025 | % 045 | 5 065 | E 105 | U 125 | e 145 | u 165 |
| 0 | 1 | 1 | 0 | 6 | ACK 006 | SYN 026 | & 046 | 6 066 | F 106 | V 126 | f 146 | v 166 |
| 0 | 1 | 1 | 1 | 7 | BEL 007 | ETB 027 | ' 047 | 7 067 | G 107 | W 127 | g 147 | w 167 |
| 1 | 0 | 0 | 0 | 8 | BS 010 | CAN 030 | ( 050 | 8 070 | H 110 | X 130 | h 150 | x 170 |
| 1 | 0 | 0 | 1 | 9 | HT 011 | EM 031 | ) 051 | 9 071 | I 111 | Y 131 | i 151 | y 171 |
| 1 | 0 | 1 | 0 | A | LF 012 | SUB 032 | * 052 | : 072 | J 112 | Z 132 | j 152 | z 172 |
| 1 | 0 | 1 | 1 | B | VT 013 | ESC 033 | + 053 | ; 073 | K 113 | [ 133 | k 153 | { 173 |
| 1 | 1 | 0 | 0 | C | FF 014 | FS 034 | , 054 | < 074 | L 114 | \ 134 | l 154 | \| 174 |
| 1 | 1 | 0 | 1 | D | CR 015 | GS 035 | - 055 | = 075 | M 115 | ] 135 | m 155 | } 175 |
| 1 | 1 | 1 | 0 | E | SO 016 | RS 036 | . 056 | > 076 | N 116 | ^ 136 | n 156 | ~ 176 |
| 1 | 1 | 1 | 1 | F | SI 017 | US 037 | / 057 | ? 077 | O 117 | _ 137 | o 157 | DEL† 177 |

†The graphic 95-character subset does not include DEL; refer to Terminal Transmission Modes in the text.

LEGEND:

Numbers under characters are the octal values for the 7-bit character codes used within the network.

TABLE A-3. LOCAL BATCH DEVICE CHARACTER SETS

| Character Sets | | | Code Sets | | | Card Keypunch Code | |
|---|---|---|---|---|---|---|---|
| CDC Graphic (64-Character Set) | ASCII Graphic (64-Character Set) | ASCII Graphic (95-Character Set) | Octal 6-Bit Display Code | Octal 6/12-Bit Display Code† | Octal 12-Bit ASCII Code | 026 | 029 |
| : colon†† | : colon†† | | 00†† | | | 8-2 | 8-2 |
| A | A | A | 01 | 01 | 0101 | 12-1 | 12-1 |
| B | B | B | 02 | 02 | 0102 | 12-2 | 12-2 |
| C | C | C | 03 | 03 | 0103 | 12-3 | 12-3 |
| D | D | D | 04 | 04 | 0104 | 12-4 | 12-4 |
| E | E | E | 05 | 05 | 0105 | 12-5 | 12-5 |
| F | F | F | 06 | 06 | 0106 | 12-6 | 12-6 |
| G | G | G | 07 | 07 | 0107 | 12-7 | 12-7 |
| H | H | H | 10 | 10 | 0110 | 12-8 | 12-8 |
| I | I | I | 11 | 11 | 0111 | 12-9 | 12-9 |
| J | J | J | 12 | 12 | 0112 | 11-1 | 11-1 |
| K | K | K | 13 | 13 | 0113 | 11-2 | 11-2 |
| L | L | L | 14 | 14 | 0114 | 11-3 | 11-3 |
| M | M | M | 15 | 15 | 0115 | 11-4 | 11-4 |
| N | N | N | 16 | 16 | 0116 | 11-5 | 11-5 |
| O | O | O | 17 | 17 | 0117 | 11-6 | 11-6 |
| P | P | P | 20 | 20 | 0120 | 11-7 | 11-7 |
| Q | Q | Q | 21 | 21 | 0121 | 11-8 | 11-8 |
| R | R | R | 22 | 22 | 0122 | 11-9 | 11-9 |
| S | S | S | 23 | 23 | 0123 | 0-2 | 0-2 |
| T | T | T | 24 | 24 | 0124 | 0-3 | 0-3 |
| U | U | U | 25 | 25 | 0125 | 0-4 | 0-4 |
| V | V | V | 26 | 26 | 0126 | 0-5 | 0-5 |
| W | W | W | 27 | 27 | 0127 | 0-6 | 0-6 |
| X | X | X | 30 | 30 | 0130 | 0-7 | 0-7 |
| Y | Y | Y | 31 | 31 | 0131 | 0-8 | 0-8 |
| Z | Z | Z | 32 | 32 | 0132 | 0-9 | 0-9 |
| 0 | 0 | 0 | 33 | 33 | 0060 | 0 | 0 |
| 1 | 1 | 1 | 34 | 34 | 0061 | 1 | 1 |
| 2 | 2 | 2 | 35 | 35 | 0062 | 2 | 2 |
| 3 | 3 | 3 | 36 | 36 | 0063 | 3 | 3 |
| 4 | 4 | 4 | 37 | 37 | 0064 | 4 | 4 |
| 5 | 5 | 5 | 40 | 40 | 0065 | 5 | 5 |
| 6 | 6 | 6 | 41 | 41 | 0066 | 6 | 6 |
| 7 | 7 | 7 | 42 | 42 | 0067 | 7 | 7 |
| 8 | 8 | 8 | 43 | 43 | 0070 | 8 | 8 |
| 9 | 9 | 9 | 44 | 44 | 0071 | 9 | 9 |
| + plus | + plus | + plus | 45 | 45 | 0053 | 12 | 12-8-6 |
| - hyphen (minus) | - hyphen (minus) | - hyphen (minus) | 46 | 46 | 0055 | 11 | 11 |
| * asterisk | * asterisk | * asterisk | 47 | 47 | 0052 | 11-8-4 | 11-8-4 |
| / slant | / slant | / slant | 50 | 50 | 0057 | 0-1 | 0-1 |
| ( open. paren. | ( open. paren. | ( open. paren. | 51 | 51 | 0050 | 0-8-4 | 12-8-5 |
| ) clos. paren. | ) clos. paren. | ) clos. paren. | 52 | 52 | 0051 | 12-8-4 | 11-8-5 |
| $ dollar sign | $ dollar sign | $ dollar sign | 53 | 53 | 0044 | 11-8-3 | 11-8-3 |
| = equals | = equals | = equals | 54 | 54 | 0075 | 8-3 | 8-6 |
| space | space | space | 55 | 55 | 0040 | no punch | no punch |
| , comma | , comma | , comma | 56 | 56 | 0054 | 0-8-3 | 0-8-3 |
| . period | . period | . period | 57 | 57 | 0056 | 12-8-3 | 12-8-3 |
| ≡ equivalence | # number sign | # number sign | 60 | 60 | 0043 | 0-8-6 | 8-3 |
| [ open. bracket | [ open. bracket | [ open. bracket | 61 | 61 | 0133 | 8-7 | 12-8-2 or 12-0††† |
| ] clos. bracket | ] clos. bracket | ] clos. bracket | 62 | 62 | 0135 | 0-8-2 | 11-8-2 or 11-0††† |
| % percent sign†† | % percent sign†† | % percent sign†† | 63†† | 63†† | 0045 | 8-6 | 0-8-4 |

TABLE A-3. LOCAL BATCH DEVICE CHARACTER SETS (Contd)

| Character Sets | | | Code Sets | | | Card Keypunch Code | |
|---|---|---|---|---|---|---|---|
| CDC Graphic (64-Character Set) | ASCII Graphic (64-Character Set) | ASCII Graphic (95-Character Set) | Octal 6-Bit Display Code | Octal 6/12-Bit Display Code† | Octal 12-Bit ASCII Code | 026 | 029 |
| ≠ not equals | " quotation mark | " quotation mark | 64 | 64 | 0042 | 8-4 | 8-7 |
| ↱ concatenation. | _ underline | _ underline | 65 | 65 | 0137 | 0-8-5 | 0-8-5 |
| ∨ logical OR | ! exclamation pt. | ! exclamation pt. | 66 | 66 | 0041 | 11-0 | 12-8-7 |
| | | | | | | or | or |
| | | | | | | 11-8-2§ | 11-0§ |
| ∧ logical AND | & ampersand | & ampersand | 67 | 67 | 0046 | 0-8-7 | 12 |
| ↑ superscript | ′ apostrophe | ′ apostrophe | 70 | 70 | 0047 | 11-8-5 | 8-5 |
| ↓ subscript | ? question mark | ? question mark | 71 | 71 | 0077 | 11-8-6 | 0-8-7 |
| < less than | < less than | < less than | 72 | 72 | 0074 | 12-0 | 12-8-4 |
| | | | | | | or | or |
| | | | | | | 12-8-2§ | 12-0§ |
| > greater than | > greater than | > greater than | 73 | 73 | 0076 | 11-8-7 | 0-8-6 |
| ≤ less/equal | @ commercial at | @ commercial at | 74†† | 7401†† | 0100 | 8-5 | 8-4 |
| ≥ greater/equal | \ reverse slant | \ reverse slant | 75 | 75 | 0134 | 12-8-5 | 0-8-2 |
| ¬ logical NOT | ^ circumflex | | 76 | | | 12-8-6 | 11-8-7 |
| ; semicolon | ; semicolon | ; semicolon | 77 | 77 | 0073 | 12-8-7 | 11-8-6 |
| | | ^ circumflex | 76†† | 7402 | 0136 | | |
| | | : colon†† | | 7404†† | 0072 | | |
| | | ` grave accent | 74†† | 7407 | 0140 | | |
| | | a | | 7601 | 0141 | | |
| | | b | | 7602 | 0142 | | |
| | | c | | 7603 | 0143 | | |
| | | d | | 7604 | 0144 | | |
| | | e | | 7605 | 0145 | | |
| | | f | | 7606 | 0146 | | |
| | | g | | 7607 | 0147 | | |
| | | h | | 7610 | 0150 | | |
| | | i | | 7611 | 0151 | | |
| | | j | | 7612 | 0152 | | |
| | | k | | 7613 | 0153 | | |
| | | l | | 7614 | 0154 | | |
| | | m | | 7615 | 0155 | | |
| | | n | | 7616 | 0156 | | |
| | | o | | 7617 | 0157 | | |
| | | p | | 7620 | 0160 | | |
| | | q | | 7621 | 0161 | | |
| | | r | | 7622 | 0162 | | |
| | | s | | 7623 | 0163 | | |
| | | t | | 7624 | 0164 | | |
| | | u | | 7625 | 0165 | | |
| | | v | | 7626 | 0166 | | |
| | | w | | 7627 | 0167 | | |
| | | x | | 7630 | 0170 | | |
| | | y | | 7631 | 0171 | | |
| | | z | | 7632 | 0172 | | |
| | | { open. brace | 61†† | 7633 | 0173 | | |
| | | \| vertical line | 75†† | 7634 | 0174 | | |
| | | } clos. brace | 62†† | 7635 | 0175 | | |
| | | ~ tilde | 76†† | 7636 | 0176 | | |

†Available only on NOS.

††Character or code interpretation depends on context. Refer to Character Set Anomalies in the text.

†††Available for input only, on NOS.

§Available for input only, on NOS/BE or SCOPE 2.

TABLE A-4. 7-TRACK CODED TAPE CONVERSIONS

| External BCD | ASCII Character | Octal 6-Bit Display Code | External BCD | ASCII Character | Octal 6-Bit Display Code |
|---|---|---|---|---|---|
| 01 | 1 | 34 | 40 | - hyphen (minus) | 46 |
| 02 | 2 | 35 | 41 | J | 12 |
| 03 | 3 | 36 | 42 | K | 13 |
| 04 | 4 | 37 | 43 | L | 14 |
| 05 | 5 | 40 | 44 | M | 15 |
| 06 | 6 | 41 | 45 | N | 16 |
| 07 | 7 | 42 | 46 | O | 17 |
| 10 | 8 | 43 | 47 | P | 20 |
| 11 | 9 | 44 | 50 | Q | 21 |
| 12† | 0 | 33 | 51 | R | 22 |
| 13 | = equals | 54 | 52 | ! exclamation point | 66 |
| 14 | " quotation mark | 64 | 53 | $ dollar sign | 53 |
| 15 | @ commercial at | 74 | 54 | * asterisk | 47 |
| 16† | % percent sign | 63 | 55 | ′ apostrophe | 70 |
| 17 | [ opening bracket | 61 | 56 | ? question mark | 71 |
| 20 | space | 55 | 57 | > greater than | 73 |
| 21 | / slant | 50 | 60 | + plus | 45 |
| 22 | S | 23 | 61 | A | 01 |
| 23 | T | 24 | 62 | B | 02 |
| 24 | U | 25 | 63 | C | 03 |
| 25 | V | 26 | 64 | D | 04 |
| 26 | W | 27 | 65 | E | 05 |
| 27 | X | 30 | 66 | F | 06 |
| 30 | Y | 31 | 67 | G | 07 |
| 31 | Z | 32 | 70 | H | 10 |
| 32 | ] closing bracket | 62 | 71 | I | 11 |
| 33 | , comma | 56 | 72 | < less than | 72 |
| 34 | ( opening parenthesis | 51 | 73 | . period | 57 |
| 35 | _ underline | 65 | 74 | ) closing parenthesis | 52 |
| 36 | # number sign | 60 | 75 | \ reverse slant | 75 |
| 37 | & ampersand | 67 | 76 | ^ caret | 76 |
|  |  |  | 77 | ; semicolon | 77 |

†As the text explains, conversion of these codes depends on whether the tape is read or written.

TABLE A-5. ASCII 9-TRACK CODED TAPE CONVERSION

| ASCII | | | | 6-Bit Display Code††† | |
|---|---|---|---|---|---|
| Code Conversion† | | Character and Code Conversion†† | | | |
| Code (Hex) | Character | Code (Hex) | Character | ASCII Character | Code (Octal) |
| 20 | space | 00 | NUL | space | 55 |
| 21 | ! exclamation point | 7D | } closing brace | ! exclamation point | 66 |
| 22 | " quotation mark | 02 | STX | " quotation mark | 64 |
| 23 | # number sign | 03 | ETX | # number sign | 60 |
| 24 | $ dollar sign | 04 | EOT | $ dollar sign | 53 |
| 25 | % percent sign§ | 05 | ENQ | % percent sign§ | 63§ |
| 26 | & ampersand | 06 | ACK | & ampersand | 67 |
| 27 | ′ apostrophe | 07 | BEL | ′ apostrophe | 70 |
| 28 | ( opening parenthesis | 08 | BS | ( opening parenthesis | 51 |
| 29 | ) closing parenthesis | 09 | HT | ) closing parenthesis | 52 |
| 2A | * asterisk | 0A | LF | * asterisk | 47 |
| 2B | + plus | 0B | VT | + plus | 45 |
| 2C | , comma | 0C | FF | , comma | 56 |
| 2D | - hyphen (minus) | 0D | CR | - hyphen (minus) | 46 |
| 2E | . period | 0E | SO | . period | 57 |
| 2F | / slant | 0F | SI | / slant | 50 |
| 30 | 0 | 10 | DLE | 0 | 33 |
| 31 | 1 | 11 | DC1 | 1 | 34 |
| 32 | 2 | 12 | DC2 | 2 | 35 |
| 33 | 3 | 13 | DC3 | 3 | 36 |
| 34 | 4 | 14 | DC4 | 4 | 37 |
| 35 | 5 | 15 | NAK | 5 | 40 |
| 36 | 6 | 16 | SYN | 6 | 41 |
| 37 | 7 | 17 | ETB | 7 | 42 |
| 38 | 8 | 18 | CAN | 8 | 43 |
| 39 | 9 | 19 | EM | 9 | 44 |
| 3A | : colon§ | 1A | SUB | : colon§ | 00§ |
| 3B | ; semicolon | 1B | ESC | ; semicolon | 77 |
| 3C | < less than | 7B | { opening brace | < less than | 72 |
| 3D | = equals | 1D | GS | = equals | 54 |
| 3E | > greater than | 1E | RS | > greater than | 73 |
| 3F | ? question mark | 1F | US | ? question mark | 71 |
| 40 | @ commercial at | 60 | ` grave accent | @ commercial at | 74 |
| 41 | A | 61 | a | A | 01 |
| 42 | B | 62 | b | B | 02 |
| 43 | C | 63 | c | C | 03 |
| 44 | D | 64 | d | D | 04 |
| 45 | E | 65 | e | E | 05 |
| 46 | F | 66 | f | F | 06 |

TABLE A-5. ASCII 9-TRACK CODED TAPE CONVERSION (Contd)

| ASCII | | | | 6-Bit Display Code††† | |
|---|---|---|---|---|---|
| Code Conversion† | | Character and Code Conversion†† | | | |
| Code (Hex) | Character | Code (Hex) | Character | ASCII Character | Code (Octal) |
| 47 | G | 67 | g | G | 07 |
| 48 | H | 68 | h | H | 10 |
| 49 | I | 69 | i | I | 11 |
| 4A | J | 6A | j | J | 12 |
| 4B | K | 6B | k | K | 13 |
| 4C | L | 6C | l | L | 14 |
| 4D | M | 6D | m | M | 15 |
| 4E | N | 6E | n | N | 16 |
| 4F | O | 6F | o | O | 17 |
| 50 | P | 70 | p | P | 20 |
| 51 | Q | 71 | q | Q | 21 |
| 52 | R | 72 | r | R | 22 |
| 53 | S | 73 | s | S | 23 |
| 54 | T | 74 | t | T | 24 |
| 55 | U | 75 | u | U | 25 |
| 56 | V | 76 | v | V | 26 |
| 57 | W | 77 | w | W | 27 |
| 58 | X | 78 | x | X | 30 |
| 59 | Y | 79 | y | Y | 31 |
| 5A | Z | 7A | z | Z | 32 |
| 5B | [ opening bracket | 1C | FS | [ opening bracket | 61 |
| 5C | \ reverse slant | 7C | \| vertical line | \ reverse slant | 75 |
| 5D | ] closing bracket | 01 | SOH | ] closing bracket | 62 |
| 5E | ^ caret | 7E | ~ tilde | ^ caret | 76 |
| 5F | _ underline | 7F | DEL | _ underline | 65 |

†When these characters are copied from or to a tape, the characters remain the same and the code changes from or to ASCII to or from display code.

††These characters do not exist in display code. When the characters are copied from a tape, each ASCII character is changed to an alternate display code character. The corresponding codes are also changed. Example: When the system copies a lowercase a, 61 (hexadecimal), from tape, it writes an uppercase A, 01 (octal).

†††A display code space always translates to an ASCII space.

§Character or code interpretation depends on context. Refer to Character Set Anomalies in the text.

TABLE A-6. EBCDIC 9-TRACK CODED TAPE CONVERSION

| EBCDIC | | | | 6-Bit Display Code††† | |
|---|---|---|---|---|---|
| Code Conversion† | | Character and Code Conversion†† | | | |
| Code (Hex) | Character | Code (Hex) | Character | ASCII Character | Code (Octal) |
| 40 | space | 00 | NUL | space | 55 |
| 4A | ¢ cent sign | 1C | IFS | [ opening bracket | 61 |
| 4B | . period | 0E | SO | . period | 57 |
| 4C | < less than | C0 | { opening brace | < less than | 72 |
| 4D | ( opening parenthesis | 16 | BS | ( opening parenthesis | 51 |
| 4E | + plus | 0B | VT | + plus | 45 |
| 4F | \| vertical line | D0 | } closing brace | ! exclamation point | 66 |
| 50 | & ampersand | 2E | ACK | & ampersand | 67 |
| 5A | ! exclamation point | 01 | SOH | ] closing bracket | 62 |
| 5B | $ dollar sign | 37 | EOT | $ dollar sign | 53 |
| 5C | * asterisk | 25 | LF | * asterisk | 47 |
| 5D | ) closing parenthesis | 05 | HT | ) closing parenthesis | 52 |
| 5E | ; semicolon | 27 | ESC | ; semicolon | 77 |
| 5F | ¬ logical NOT | A1 | ~ tilde | ^ caret | 76 |
| 60 | - hyphen (minus) | 0D | CR | - hyphen (minus) | 46 |
| 61 | / slant | 0F | SI | / slant | 50 |
| 6B | , comma | 0C | FF | , comma | 56 |
| 6C | % percent sign§ | 2D | ENQ | % percent sign§ | 63§ |
| 6D | _ underline | 07 | DEL | _ underline | 65 |
| 6E | > greater than | 1E | IRS | > greater than | 73 |
| 6F | ? question mark | 1F | IUS | ? question mark | 71 |
| 7A | : colon§ | 3F | SUB | : colon§ | 00§ |
| 7B | # number sign | 03 | ETX | # number sign | 60 |
| 7C | @ commercial at | 79 | \ reverse slant | @ commercial at | 74 |
| 7D | ' apostrophe | 2F | BEL | ' apostrophe | 70 |
| 7E | = equals | 1D | IGS | = equals | 54 |
| 7F | " quotation mark | 02 | STX | " quotation mark | 64 |
| C1 | A | 81 | a | A | 01 |
| C2 | B | 82 | b | B | 02 |
| C3 | C | 83 | c | C | 03 |
| C4 | D | 84 | d | D | 04 |
| C5 | E | 85 | e | E | 05 |
| C6 | F | 86 | f | F | 06 |
| C7 | G | 87 | g | G | 07 |
| C8 | H | 88 | h | H | 10 |
| C9 | I | 89 | i | I | 11 |
| D1 | J | 91 | j | J | 12 |
| D2 | K | 92 | k | K | 13 |
| D3 | L | 93 | l | L | 14 |

TABLE A-6. EBCDIC 9-TRACK CODED TAPE CONVERSION (Contd)

| EBCDIC | | | | 6-Bit Display Code††† | |
|---|---|---|---|---|---|
| Code Conversion† | | Character and Code Conversion†† | | | |
| Code (Hex) | Character | Code (Hex) | Character | ASCII Character | Code (Octal) |
| D4 | M | 94 | m | M | 15 |
| D5 | N | 95 | n | N | 16 |
| D6 | O | 96 | o | O | 17 |
| D7 | P | 97 | p | P | 20 |
| D8 | Q | 98 | q | Q | 21 |
| D9 | R | 99 | r | R | 22 |
| E0 | \ reverse slant | 6A | \| vertical line | \ reverse slant | 75 |
| E2 | S | A2 | s | S | 23 |
| E3 | T | A3 | t | T | 24 |
| E4 | U | A4 | u | U | 25 |
| E5 | V | A5 | v | V | 26 |
| E6 | W | A6 | w | W | 27 |
| E7 | X | A7 | x | X | 30 |
| E8 | Y | A8 | y | Y | 31 |
| E9 | Z | A9 | z | Z | 32 |
| F0 | 0 | 10 | DLE | 0 | 33 |
| F1 | 1 | 11 | DC1 | 1 | 34 |
| F2 | 2 | 12 | DC2 | 2 | 35 |
| F3 | 3 | 13 | TM | 3 | 36 |
| F4 | 4 | 3C | DC4 | 4 | 37 |
| F5 | 5 | 3D | NAK | 5 | 40 |
| F6 | 6 | 32 | SYN | 6 | 41 |
| F7 | 7 | 26 | ETB | 7 | 42 |
| F8 | 8 | 18 | CAN | 8 | 43 |
| F9 | 9 | 19 | EM | 9 | 44 |

†When these characters are copied from or to a tape, the characters remain the same (except EBCDIC codes 4A (hexadecimal), 4F (hexadecimal), 5A (hexadecimal), and 5F (hexadecimal)) and the code changes from or to EBCDIC to or from display code.

††These characters do not exist in display code. When the characters are copied from a tape, each EBCDIC character is changed to an alternate display code character. The corresponding codes are also changed. Example: When the system copies a lowercase a, 81 (hexadecimal), from tape, it writes an uppercase A, 01 (octal).

†††A display code space always translates to an EBCDIC space.

§Character or code interpretation depends on context. Refer to Character Set Anomalies in the text.

TABLE A-7. FULL EBCDIC CHARACTER SET

| Hexa-decimal EBCDIC Code | Octal 12-Bit EBCDIC Code | EBCDIC Graphic or Control Character† | Hexa-decimal EBCDIC Code | Octal 12-Bit EBCDIC Code | EBCDIC Graphic or Control Character† | Hexa-decimal EBCDIC Code | Octal 12-Bit EBCDIC Code | EBCDIC Graphic or Control Character† |
|---|---|---|---|---|---|---|---|---|
| 00 | 0000 | NUL | 4A | 0112 | ¢ cent sign | A7 | 0247 | x |
| 01 | 0001 | SOH | 4B | 0113 | . period | A8 | 0250 | y |
| 02 | 0002 | STX | 4C | 0114 | < less than | A9 | 0251 | z |
| 03 | 0003 | ETX | 4D | 0115 | ( open. paren. | AA | 0252 | undefined |
| 04 | 0004 | PF | 4E | 0116 | + plus | thru | thru | |
| 05 | 0005 | HT | 4F | 0117 | ¦ logical OR | BF | 0277 | undefined |
| 06 | 0006 | LC | 50 | 0120 | & ampersand | C0 | 0300 | { open. brace |
| 07 | 0007 | DEL | 51 | 0121 | undefined | C1 | 0301 | A |
| 08 | 0010 | undefined | thru | thru | | C2 | 0302 | B |
| 09 | 0011 | undefined | 59 | 0131 | undefined | C3 | 0303 | C |
| 0A | 0012 | SMM | 5A | 0132 | ! exclam. point | C4 | 0304 | D |
| 0B | 0013 | VT | 5B | 0133 | $ dollar sign | C5 | 0305 | E |
| 0C | 0014 | FF | 5C | 0134 | * asterisk | C6 | 0306 | F |
| 0D | 0015 | CR | 5D | 0135 | ) clos. paren. | C7 | 0307 | G |
| 0E | 0016 | SO | 5E | 0136 | ; semicolon | C8 | 0310 | H |
| 0F | 0017 | SI | 5F | 0137 | ¬ logical NOT | C9 | 0311 | I |
| 10 | 0020 | DLE | 60 | 0140 | - minus | CA | 0312 | undefined |
| 11 | 0021 | DC1 | 61 | 0141 | / slant | CB | 0313 | undefined |
| 12 | 0022 | DC2 | 62 | 0142 | undefined | CC | 0314 | ⑀ |
| 13 | 0023 | TM | thru | thru | | CD | 0315 | undefined |
| 14 | 0024 | RES | 69 | 0151 | undefined | CE | 0316 | ⑂ |
| 15 | 0025 | NL | 6A | 0152 | ¦ vertical line | CF | 0317 | undefined |
| 16 | 0026 | BS | 6B | 0153 | , comma | D0 | 0320 | } clos. brace |
| 17 | 0027 | IL | 6C | 0154 | % percent sign | D1 | 0321 | J |
| 18 | 0030 | CAN | 6D | 0155 | _ underline | D2 | 0322 | K |
| 19 | 0031 | EM | 6E | 0156 | > greater than | D3 | 0323 | L |
| 1A | 0032 | CC | 6F | 0157 | ? question mark | D4 | 0324 | M |
| 1B | 0033 | CU1 | 70 | 0160 | undefined | D5 | 0325 | N |
| 1C | 0034 | IFS | thru | thru | | D6 | 0326 | O |
| 1D | 0035 | IGS | 78 | 0170 | undefined | D7 | 0327 | P |
| 1E | 0036 | IRS | 79 | 0171 | ` grave accent | D8 | 0330 | Q |
| 1F | 0037 | IUS | 7A | 0172 | : colon | D9 | 0331 | R |
| 20 | 0040 | DS | 7B | 0173 | # number sign | DA | 0332 | undefined |
| 21 | 0041 | SOS | 7C | 0174 | @ commercial at | thru | thru | |
| 22 | 0042 | FS | 7D | 0175 | ' apostrophe | DF | 0337 | undefined |
| 23 | 0043 | undefined | 7E | 0176 | = equals | E0 | 0340 | \ reverse slant |
| 24 | 0044 | BYP | 7F | 0177 | " quotation mark | E1 | 0341 | undefined |
| 25 | 0045 | LF | 80 | 0200 | undefined | E2 | 0342 | S |
| 26 | 0046 | ETBB | 81 | 0201 | a | E3 | 0343 | T |
| 27 | 0047 | ESCE | 82 | 0202 | b | E4 | 0344 | U |

TABLE A-7. FULL EBCDIC CHARACTER SET (Contd)

| Hexa-decimal EBCDIC Code | Octal 12-Bit EBCDIC Code | EBCDIC Graphic or Control Character† | Hexa-decimal EBCDIC Code | Octal 12-Bit EBCDIC Code | EBCDIC Graphic or Control Character† | Hexa-decimal EBCDIC Code | Octal 12-Bit EBCDIC Code | EBCDIC Graphic or Control Character† |
|---|---|---|---|---|---|---|---|---|
| 28 | 0050 | undefined | 83 | 0203 | c | E5 | 0345 | V |
| 29 | 0051 | undefined | 84 | 0204 | d | E6 | 0346 | W |
| 2A | 0052 | SM | 85 | 0205 | e | E7 | 0347 | X |
| 2B | 0053 | CU2 | 86 | 0206 | f | E8 | 0350 | Y |
| 2C | 0054 | undefined | 87 | 0207 | g | E9 | 0351 | Z |
| 2D | 0055 | ENQ | 88 | 0210 | h | EA | 0352 | undefined |
| 2E | 0056 | ACK | 89 | 0211 | i | EB | 0353 | undefined |
| 2F | 0057 | BEL | 8A | 0212 | undefined | EC | 0354 | ⑁ |
| 30 | 0060 | undefined | thru | thru |  | ED | 0355 | undefined |
| 31 | 0061 | undefined | 90 | 0220 | undefined | thru | thru |  |
| 32 | 0062 | SYN | 91 | 0221 | j | EF | 0357 | undefined |
| 33 | 0063 | undefined | 92 | 0222 | k | F0 | 0360 | 0 |
| 34 | 0064 | PN | 93 | 0223 | l | F1 | 0361 | 1 |
| 35 | 0065 | RS | 94 | 0224 | m | F2 | 0362 | 2 |
| 36 | 0066 | UC | 95 | 0225 | n | F3 | 0363 | 3 |
| 37 | 0067 | EOT | 96 | 0226 | o | F4 | 0364 | 4 |
| 38 | 0070 | undefined | 97 | 0227 | p | F5 | 0365 | 5 |
| 39 | 0071 | undefined | 98 | 0230 | q | F6 | 0366 | 6 |
| 3A | 0072 | undefined | 99 | 0231 | r | F7 | 0367 | 7 |
| 3B | 0073 | CU3 | 9A | 0232 | undefined | F8 | 0370 | 8 |
| 3C | 0074 | DC4 | thru | thru |  | F9 | 0372 | 9 |
| 3D | 0075 | NAK | A0 | 0240 | undefined | FA | 0372 | \| vertical line |
| 3E | 0076 | undefined | A1 | 0241 | ~ tilde | FB | 0373 | undefined |
| 3F | 0077 | SUB | A2 | 0242 | s | thru | thru |  |
| 40 | 0100 | space | A3 | 0243 | t | FF | 0377 | undefined |
| 41 | 0101 | undefined | A4 | 0244 | u |  |  |  |
| thru | thru |  | A5 | 0245 | v |  |  |  |
| 49 | 0111 | undefined | A6 | 0246 | w |  |  |  |

†Graphic characters shown are those used on the IBM System/370 standard (PN) print train. Other devices support subsets or variations of this character graphic set.

# DIAGNOSTICS B

FORM issues diagnostic messages during both the scanning and execution of the FORM directives. The 8-bit subroutines, which perform many of the FORM functions, also produce execution diagnostics.

This appendix lists the diagnostics issued both by FORM and by the 8-bit subroutines.

## SCAN DIAGNOSTICS

A diagnostic issued by FORM during the directive scan appears in the program listing immediately after the directive containing the error. Either an arrow or an apostrophe (on an ASCII-64 printer) indicates the approximate location of the error. When an error is detected, scanning continues to the end of the directive sequence, but execution does not occur; the run stops when the scan is complete.

A typical set of error messages is illustrated in figure B-1. The FORM scan diagnostic messages are listed in table B-1.

```
INP(STCNTF,EX= )
              ↑
ENTRY NAME EXPECTED
              ↑
MISSING TERMINATOR

OUT(OUT1,MAX=100,MAX=1000)
                          ↑
MAX SPECIFIED TWICE

OUT(OUT2,BGD=A)
              ↑
INVALID BGD OPTION
```

Figure B-1. Examples of Scan Diagnostics

## EXECUTION DIAGNOSTICS

During directive execution, diagnostics can be produced both by FORM and by the 8-bit subroutines.

### FORM EXECUTION DIAGNOSTICS

Diagnostic messages issued by FORM during execution appear in the program listing immediately after the directive list. Wherever possible, the parameter list of the executing directive is printed and the character being processed at the time of the error is indicated. When an error occurs, FORM finishes processing the record, if possible, and then stops.

An example of a set of execution diagnostics, including both FORM and 8-bit subroutine messages is illustrated in figure B-2. The execution diagnostic messages are listed in table B-2.

### 8-BIT SUBROUTINE DIAGNOSTICS

The 8-bit subroutines produce messages during execution of the FORM directives. These messages are intermixed with the FORM messages and often provide more detailed information. The format of the message is as follows:

ERROR DETECTED BY xname - cause - message

| | |
|---|---|
| xname | Name of subroutine that detected the error |
| cause | Probable cause of the error |
| message | Diagnostic message as listed in table B-2 |

The 8-bit subroutine diagnostic messages are listed in table B-3.

```
 ERROR DETECTED BY XREAD  -   CONVERT- BAD SYNTAX IN Z,S,N OR P FIELD
      CALLED FROM FM$RUN    AT LINE      397
      CALLED FROM FORM      AT LINE      397
↑
UNRECOVERABLE INPUT FILE ERROR
```

Figure B-2. Examples of Execution Diagnostics

TABLE B-1. FORM SCAN DIAGNOSTICS

| Message | Significance | Action |
|---|---|---|
| ADD SPECIFIED TWICE | The ADD parameter can appear only once in a SEQ directive. | Correct the error and rerun. |
| BEG SPECIFIED TWICE | The BEG parameter can appear only once in a SEQ directive. | Correct the error and rerun. |
| BGD SPECIFIED TWICE | The BGD parameter can appear only once in an OUT directive. | Correct the error and rerun. |
| BLK NOT SPECIFIED | The BLK parameter must be specified in the INP or OUT directive when a CON directive is active. | Correct the error and rerun. |
| BLK OUT OF RANGE | The maximum value is 32760. | Correct the error and rerun. |
| BLK SPECIFIED TWICE | The BLK parameter can appear only once in an INP or OUT directive. | Correct the error and rerun. |
| CANT DUMP TO IBM FORMAT | The FMT=D parameter is invalid when print file is IBM format. | Dump to the CDC format file. |
| CANT OPEN FILE; POSSIBLE CRM ERROR | A CRM error condition was detected during a CRM file open operation. | Check validity of file control statement parameters and data file format. |
| CANT SPECIFY INP= OR OUT= WITH I= | If INP= or OUT= is used in the FORM control statement, I= cannot be specified. | Specify only one parameter set and rerun. |
| COD SPECIFIED TWICE | The COD parameter can appear only once in an INP or OUT directive. | Correct the error and rerun. |
| CON AND PAG NOT VALID IF FT NOT SQ | The CON and PAG directives are valid only for sequential files. | Check the FO parameter of the FILE statement. |
| CON SPECIFIED TWICE | Only one CON directive can appear in a FORM run. | Correct the error and rerun. |
| CPA SPECIFIED TWICE | The CPA parameter can appear only once in an OUT directive. | Correct the error and rerun. |
| CX SPECIFIED TWICE | The CX parameter can appear only once in an INP or OUT directive. | Correct the error and rerun. |
| DCA SPECIFIED TWICE | The DCA parameter can be specified only once in an INP directive. | Correct the error and rerun. |
| DCT SPECIFIED TWICE | The DCT parameter can appear only once in an OUT directive. | Correct the error and rerun. |
| DUPLICATE FILENAME | Multiple occurrences of a directive must specify unique file names. | Specify unique file names; check for redundant directives; use selector expressions to combine reformatting operations. |
| DX SPECIFIED TWICE | The DX parameter can appear only once in an INP directive. | Correct the error and rerun. |
| ENTRY NAME EXPECTED | An owncode entry point name has been incorrectly specified. | Specify the correct subprogram name. |
| ERROR WRITING TITLE | An I/O error occurred while writing the file specified in a PAG directive. | Follow site-defined procedures for reporting software errors or operational problems. |
| EX SPECIFIED TWICE | The EX parameter can appear only once in an INP and OUT directive. | Correct the error and rerun. |

TABLE B-1. FORM SCAN DIAGNOSTICS (Contd)

| Message | Significance | Action |
|---|---|---|
| FEX SPECIFIED TWICE | The FEX parameter can appear only once in an XEQ directive. | Correct the error and rerun. |
| FILE ALREADY SPECIFIED | The file name has been specified more than once in the FORM control statement. | Correct the error and rerun. |
| FILE NOT DEFINED | The file name has not been specified in an INP or OUT directive. | Specify the correct file name. |
| FMT SPECIFIED TWICE | The FMT parameter can appear only once in a PAG directive. | Correct the error and rerun. |
| HRL SPECIFIED TWICE | The HRL parameter can be specified only once in an INP or OUT directive. | Correct the error and rerun. |
| I= ALREADY SPECIFIED | The I parameter has been specified more than once in the FORM control statement. | Correct the error and rerun. |
| IBM FILE ORGANIZATION INVALID IF FT NOT = SQ | Word addressable or AAM organization types are not permitted with IBM organizations. | Convert file to sequential format. |
| INVALID BGD OPTION | Valid options are X, Z, B, C. | Correct the error and rerun. |
| INVALID COD OPTION | Valid options are A, C, E. | Correct the error and rerun. |
| INVALID FILENAME | A file name must contain 1 through 7 letters or digits, beginning with a letter. | Correct the file name and rerun. |
| INVALID FMT OPTION | Valid options are 1, 2, A, D. | Correct the error and rerun. |
| INVALID REW OPTION | Valid options are N, R, U. | Correct the error and rerun. |
| INVALID RFM OPTION | Valid options are F, V, U, FB, VB, VSB. | Correct the error and rerun. |
| IRL OUT OF RANGE | The maximum value is 2. | Correct the error and rerun. |
| IRL SPECIFIED TWICE | The IRL parameter can appear only once in an INP or OUT directive. | Correct the error and rerun. |
| IX SPECIFIED TWICE | The IX parameter can appear only once in an XEQ directive. | Correct the error and rerun. |
| KEY LOCATION MISSING | The iTm descriptor is missing in KEY=iTm. | Supply the iTm descriptor and rerun. |
| KEY SPECIFIED TWICE | The KEY parameter can appear only once in an OUT directive. | Correct the error and rerun. |
| KEYTYPE MUST BE *I, F, X* | Data types other then I, F, or X are invalid for record keys. | Correct the error and rerun. |
| L= ALREADY SPECIFIED | The L parameter has been specified more than once in the FORM control statement. | Correct the error and rerun. |
| LRL OUT OF RANGE | The maximum value is 32760. | Correct the error and rerun. |
| LRL SPECIFIED TWICE | The LRL parameter can appear only once in an INP or OUT directive. | Eliminate the excess LRL parameter. |
| LX SPECIFIED TWICE | The LX parameter can be specified only once in an INP or OUT directive. | Correct the error and rerun. |
| MAX OUT OF RANGE | The MAX parameter value exceeds 16777215. | Reduce the value of MAX and rerun. |
| MAX SPECIFIED TWICE | The MAX parameter can appear only once in an INP or OUT directive. | Correct the error and rerun. |

TABLE B-1. FORM SCAN DIAGNOSTICS (Contd)

| Message | Significance | Action |
|---|---|---|
| MISSING FILENAME | The file name has been omitted from the parameter list. | Correct the error and rerun. |
| MISSING TERMINATOR | A directive must be terminated by a right parenthesis. A termination or continuation character was placed beyond column 72. Only the first 72 characters are scanned. | Correct the error and rerun. |
| NBR SPECIFIED TWICE | The NBR parameter can appear only once in a SEQ directive. | Correct the error and rerun. |
| NO INPUT FILE SPECIFIED | A FORM run requires a source of input records. | Use the INP directive, the INP parameter in the FORM control statement, or the IX exit. |
| NO OUTPUT FILE SPECIFIED | At least one output file must be declared for a FORM run. | Use the OUT directive or the OUT parameter in the FORM control statement. |
| NO SEQ FORMAT SPECIFIED | The NBR=iTm parameter is missing in the SEQ directive. | Supply NBR=iTm and rerun. |
| NOSEC SPECIFIED TWICE | The NOSEC parameter can appear only once in an OUT directive. | Correct the error and rerun. |
| NUMERIC FIELD EXPECTED | Nonnumeric characters appear in a numeric field. | Correct the error and rerun. |
| ONLY ONE INPUT FILE ALLOWED | Only a single input file is allowed per FORM run. | Correct the error and rerun. |
| OWN= ALREADY SPECIFIED | The OWN parameter appears more than once in the FORM statement. | Correct the error and rerun. |
| PARAMETER MISSING | A required parameter was not found when the directive was scanned. | Correct the directive and rerun. |
| POS OUT OF RANGE | The POS parameter value exceeds 2047. | Reduce the value of the POS parameter and rerun. |
| POS SPECIFIED TWICE | The POS parameter can appear only once in an INP directive. | Correct the error and rerun. |
| QAL SPECIFIED TWICE | Only one QAL directive can be specified for a given file. | Correct the error and rerun. |
| REF SPECIFIED TWICE | Only one REF directive can be specified for a given file. | Correct the error and rerun. |
| REW SPECIFIED TWICE | The REW parameter can appear only once in an INP or OUT directive. | Correct the error and rerun. |
| RFM NOT SPECIFIED | The RFM parameter must be specified when a CON directive is active. | Correct the error and rerun. |
| RFM SPECIFIED TWICE | The RFM parameter can appear only once in an INP or OUT directive. | Correct the error and rerun. |
| RX SPECIFIED TWICE | The RX parameter can appear only once in an OUT directive. | Correct the error and rerun. |
| STRING DELIMITER MISSING | Literal strings must be delimited by enclosing * or $. | Enclose the literal string with * or $. |
| STRING TERMINATOR MISSING | Conversion, qualification, or reformat string must be terminated with a right parenthesis. | Correct the error and rerun. |

TABLE B-1. FORM SCAN DIAGNOSTICS (Contd)

| Message | Significance | Action |
|---|---|---|
| TOO MANY OUTPUT FILES | The limit is 20 output files. | Reduce the number of output files to 20. |
| TOO MANY OWNCODE ENTRIES | The limit is 50 unique entry points. | Reduce the number of entry points and rerun. |
| TOP SPECIFIED TWICE | The TOP parameter can appear only once in a PAG directive. | Correct the error and rerun. |
| TTL SPECIFIED TWICE | The TTL parameter can appear only once in a PAG directive. | Correct the error and rerun. |
| UNRECOGNIZED DIRECTIVE | The directive mnemonic is misspelled. | Correct the spelling and rerun. |
| UNRECOGNIZED OPTION | The indicated keyword is misspelled. | Correct the spelling and rerun. |
| UNRECOGNIZED PARAMETER | A keyword is misspelled or misplaced. | Correct the spelling or the placement. |
| ZERO MRL/FL ILLEGAL | The CRM default for the MRL or FL parameter cannot be used. | Correct the FILE control statement and rerun. |

TABLE B-2. FORM EXECUTION DIAGNOSTICS

| Message | Significance | Action |
|---|---|---|
| AND, OR, NOT MISPLACED | Logical operator in qualification string is not in proper order. | Correct the error and rerun. |
| BIT NUMBER > 6 | The bit position indicator w in i/wTm must not exceed 6 for CDC files or 8 for IBM files. | Correct the error and rerun. |
| BIT POSITION ONLY VALID WITH TYPE -B- | The i/wTm format cannot be used to describe data items other than bit strings (type B). | Use the iTm descriptor format. |
| BIT SPECIFIER NON-NUMERIC | The w must be a digit in an i/wTm descriptor for a bit field. | Correct the error and rerun. |
| CANNOT -QUIT- TEST | The Q specification is invalid in this context. | Remove the Q specification and rerun. |
| CANT LOAD CAPSULE | An unrecoverable loader error has occurred. | Follow site-defined procedures for reporting software errors or operational problems. |
| CANT SEARCH WITH BIT OFFSET | The starting position specification n of the oln search descriptor must be a decimal integer character count. | Correct the error and rerun. |
| CANT UNLOAD CAPSULE | An unrecoverable loader error has occurred. | Follow site-defined procedures for reporting software errors or operational problems. |
| COLON NOT VALID | A colon cannot be used in the indicated format. | Correct the error and rerun. |
| CONVERSION ERROR | The type of data found in the input record does not match the type of data expected. | Correct the error and rerun. |

TABLE B-2. FORM EXECUTION DIAGNOSTICS (Contd)

| Message | Significance | Action |
|---|---|---|
| CONVERSION NOT ALLOWED HERE | A conversion or reformat item was encountered in a QAL directive or in a selector expression. | Correct the error and rerun. |
| DIGIT DOES NOT FOLLOW +/- | A + or - must not precede a nonnumeric character. | Correct the error and rerun. |
| FIELD BEYOND RECORD ON LEFT | A descriptor references a data field that begins before the beginning of the record. | Decrease the magnitude of i or m in -iTm. |
| FIELD BEYOND RECORD ON RIGHT | A descriptor references a data field that extends beyond the end of the record. | Decrease the magnitude of i or m in iTm or +iTm. |
| FIELD NOT IN RECORD ON LEFT | A descriptor references a data field that precedes the first character of the record. | Decrease the magnitude of i in -iTm. |
| FIELD NOT IN RECORD ON RIGHT | A descriptor references a character beyond the end-of-record. | Decrease the magnitude of i in iTm or +iTm. |
| INVALID LOGICAL TERM | The selector expression contains an invalid operand. A logical operand must be a literal or item descriptor. | Change the logical operand to a literal or an item descriptor. |
| INVALID M VALUE FOR DATA TYPE | M value in the iTm item descriptor is invalid. | Correct the error and rerun. |
| INVALID -T- TYPE IN IN ITM | See table 2-1 for valid -T- types. | Correct the error and rerun. |
| K- NOT KEY OR KEYA | The KEY or KEYA parameter is misspelled. | Correct the error and rerun. |
| KEYA NOT DEFINED AT THIS POINT | The OUT directive for an actual key (specifying FO=AK) file does not precede the OUT directive associated with this REF directive. | Specify an OUT directive for the actual key output file before specifying an OUT directive for a file that is to receive the key via a REF directive. |
| LITERAL FOLLOWED BY -T- CODE | A literal must not precede T in an iTm specification. | Change the literal to a byte index. |
| LITERAL REPLACEMENT > 80 CHARACTERS | A literal used in a conversion item must not exceed 80 characters. | Correct the error and rerun. |
| MISSING AND, OR, NOT CONNECTOR | Logical operator not found between two selector-expressions or qualification-strings. | Correct the error and rerun. |
| MISSING -N- SPECIFIER IN OLN | The starting position specification n of the oln search descriptor is missing. | Insert the starting specification n and rerun. |
| MISSING OPERATOR | A + - = or a logical operator was expected at the indicated position. | Correct the error and rerun. |
| MISSING RIGHT PARENTHESIS | Right and left parentheses must balance. | Correct the error and rerun. |
| MISSING RIGHT-SIDE OF SPECIFICATION | The source field item descriptor is missing from the reformat item. | Correct the error and rerun. |
| MISSING SEPARATOR | FORM parameters must be separated by a comma. | Correct the error and rerun. |
| NO CONVERSION SPEC SEEN | The directive format requires a conversion specification at the indicated position. | Correct the error and rerun. |
| PARENTHESIS MISPLACED | Parenthesis is mispositioned in the qualification-string. | Correct the error and rerun. |

TABLE B-2. FORM EXECUTION DIAGNOSTICS (Contd)

| Message | Significance | Action |
|---|---|---|
| POSITION OR REPEAT COUNT NOT VALID WITH Q | A Q specification must not be preceded by a repeat count. | Correct the error and rerun. |
| REPEAT COUNT NOT VALID | The repeat count cannot be specified in the indicated format. | Correct the error and rerun. |
| SEARCH INVALID ON DESTINATION SPEC | An oln search descriptor must not be used in the destination field item descriptor of a reformat item. | Correct the error and rerun. |
| STRING LITERAL NOT ALLOWED LEFT OF RELATION | String literals are allowed only on the right side of relational operators. | Correct the error and rerun. |
| SYNTAX ERROR AT COMMA | A fatal error occurred near the indicated comma. | Correct the error and rerun. |
| TEST CONVERSION ERROR | The data types in the selector expression cannot be reduced to a common mode. | Correct the data type and rerun. |
| TEST NOT ALLOWED HERE | The selector expression is invalid in this context. | Remove selector expression and rerun. |
| TOO MANY PARENTHESIS LEVELS | The limit is seven levels. | Correct the error and rerun. |
| UNRECOGNIZED DATA TYPE | An invalid T code is specified in an iTm descriptor. | Correct the data type and rerun. |
| UNRECOGNIZED OPERATION | An invalid operation is specified or an operator is misspelled. | Specify the correct operation for the applicable directive. |
| UNRECOVERABLE ERROR WRITING TITLE | A parity error has occurred on the print file. | Follow site-defined procedures for reporting software errors or operational problems. |
| UNRECOVERABLE INPUT FILE ERROR | A parity error has occurred on the input file. | Follow site-defined procedures for reporting software errors or operational problems. |
| +ITM AND -ITM ILLEGAL IN SELECTOR | A + or - preceding the iTM descriptor in a QAL directive is illegal. | Correct the error and rerun. |
| +/- INVALID ON REPEAT COUNT | A conversion item or reformat item repeat count cannot be preceded by a + or -. | Correct the error and rerun. |
| +/- INVALID ON SIZE SPECIFIER | The field length specification m of an iTm descriptor must not be preceded by a + or -. | Correct the error and rerun. |

TABLE B-3. 8-BIT SUBROUTINE DIAGNOSTICS

| Message | Significance | Action | Issued By |
|---|---|---|---|
| BAD OFFSET IN PARAMETER DESCRIPTOR | Illegal bit or byte position specified. | Check position specification in iTm or i/wTm. | XREAD, XWRITE |
| BAD SYNTAX IN Z, S, N, OR P FIELD | Data being converted is in illegal format. | Correct the format and rerun. | XREAD, XWRITE |
| BIT SPECIFICATION ILLEGAL FOR NON-BIT FIELD | w in i/w item locator format in a selector expression can be specified only if the T field is B. | Change position specification to i form to indicate character position. | XREAD, XWRITE |
| BLKSIZE EXCEEDS 32760 BYTES | BLK parameter in an INP or OUT directive is too big. | Correct the error and rerun. | XREAD, XWRITE |
| BLKSIZE NOT SPECIFIED | BLK parameter in an INP or OUT directive has been omitted. | Specify BLK parameter. | XFILE |
| BLOCK SHORTER THAN V-HEADER | Input IBM record is in the wrong format. Block descriptor word contents or record descriptor word contents do not agree with actual block/record size. | Check for tape error or incorrect RFM specification. | XREAD |
| CONVERSION STRINGS NESTED TOO DEEPLY | Can be nested only to depth of seven levels. | Reduce number of levels. | XREAD, XWRITE |
| DOUBLY SPECIFIED PARAMETER IN FILE STRING | Duplicate parameter specified in an INP or OUT directive. | Delete or correct parameter. | XFILE |
| EMPTY BLOCK IN VS-RECORD | No data in input block. | Rewrite file. | XREAD |
| FILE NOT DECLARED | File name not specified in INP or OUT directive. | Correct the error and rerun. | XFILE |
| FILE NOT SPECIFIED AS READ MODE | File not declared in INP directive. | Correct the error and rerun. | XREAD |
| FILE NOT SPECIFIED AS WRITE MODE | File not declared in OUT directive. | Correct the error and rerun. | XWRITE |
| FILE PARAMETER IS NOT A FILE NAME | File name is misspelled. | Check file name. | XFILE |
| FILE STRING DOES NOT BEGIN WITH -(- | Parameter list must always begin with a left parenthesis. | Insert left parenthesis. | XFILE |
| FILE STRING DOES NOT TERMINATE WITH -)- | Parameter list must always terminate with a right parenthesis. | Insert right parenthesis. | XFILE |
| FILE TYPE NOT SPECIFIED | Internal error. | Follow site-defined procedures for reporting software errors or operational problems. | XFILE |
| FILE USAGE NOT SPECIFIED | Internal error. | Follow site-defined procedures for reporting software errors or operational problems. | XFILE |
| FIRST CHARACTER OF CONVERSION STRING IS NOT -(- | Conversion strings must always begin with a left parenthesis. | Insert left parenthesis. | XREAD, XWRITE |

TABLE B-3. 8-BIT SUBROUTINE DIAGNOSTICS (Contd)

| Message | Significance | Action | Issued By |
|---|---|---|---|
| FIRST ITEM IN SELECTOR-EXPRESSION NOT RECOGNIZED | The iTm field in the selector expression is specified incorrectly. | Specify the correct iTm field. | XREAD, XWRITE |
| ILLEGAL FIRST ITEM TYPE | T in the Tm1 field is specified incorrectly. | Check the T specification in the Tm1 field. | XREAD, XWRITE |
| ILLEGAL SECOND ITEM TYPE | T in the Tm2 field is specified incorrectly. | Check the T specification in the Tm2 field. | XREAD, XWRITE |
| INCOMPLETE VS-RECORD AT END-OF-DATA | Data is missing from input file. Length specified in the header information of the last record does not agree with actual length. | Rewrite file. | XREAD |
| INCONSISTENT PARAMETERS IN FILE STRING | File description parameters are inconsistent in INP or OUT directive. | Correct the FILE description parameters. | XFILE |
| INDEFINITE SOURCE VALUE NOT REPRESENTABLE | Floating-point source item has indefinite value. | Check source record for bad data. | XWRITE |
| INDEFINITE VALUE FOR INTEGER DESTINATION FIELD | Item to be stored in integer destination field has indefinite value. | Check source record for bad data. | XWRITE |
| INFINITE SOURCE VALUE NOT REPRESENTABLE | Floating-point source item has infinite value. | Check source record for bad data. | XWRITE |
| INFINITE VALUE FOR INTEGER DESTINATION FIELD | Item to be stored in integer destination field has infinite value. | Check source record for bad data. | XWRITE |
| INTEGER VALUE TOO LARGE FOR FIELD | Receiving field does not contain enough characters to represent all digits in the source field. | Increase length specification of receiving field descriptor. | XREAD, XWRITE |
| INVALID DATA TYPE | Legal data types are A for ASCII, C for EBCDIC, and X for Display Code. | Change T in iTm to proper type. | XREAD, XWRITE |
| INVALID PARAMETER VALUE IN FILE STRING | File description parameter in INP or OUT directive is incorrect. | Change the FILE description parameter. | XFILE |
| KEYWORD NOT FOLLOWED BY = IN FILE STRING | An = sign must follow KEYWORD in parameter list. | Insert an = sign after KEYWORD. | XFILE |
| LITERAL STRING IS TOO LONG | Literal strings in selector expression are limited to 80 characters. | Correct literal string. | XREAD, XWRITE |
| LRECL NOT SPECIFIED | Necessary LRL specification in parameter list of an INP or OUT directive was omitted. | Specify LRL parameter. | XFILE |
| LRECL TOO SMALL FOR V-RECORD HEADER | Actual record length does not agree with length specified in record descriptor word (RDW). | Check for incorrect LRL specification. | XREAD |
| LRECL TOO LARGE FOR BLKSIZE | If the blocking type is not spanned LRL must be less than BLK. | Correct LRL or BLK specification. | XFILE |

TABLE B-3. 8-BIT SUBROUTINE DIAGNOSTICS (Contd)

| Message | Significance | Action | Issued By |
|---|---|---|---|
| M SPECIFICATION ILLEGAL FOR DATA TYPE | The m in Tm1 or Tm2 is incorrectly specified. | Specify correct m and rerun. | XREAD, XWRITE |
| MISSING PARAMETER LIST | Directive requires a parameter list. | Specify parameter list. | Any 8-bit subroutine |
| MISSING RELATIONAL OPERATOR IN SELECTOR-EXPRESSION | Relationship must be stated between two items in a selector expression | Supply a relational operator in selector expression. | XREAD, XWRITE |
| MISSING RIGHT PARENTHESIS | Parameter list must always end with a right parenthesis. | Insert right parenthesis. | XFILE, XREAD, XWRITE |
| MISSING RIGHT PARENTHESIS OR SEMICOLON | Missing punctuation in conversion string. | Insert necessary punctuation. | XREAD, XWRITE |
| MISSING RIGHT STRING DELIMITER | Literal string terminator * or $ is missing. | Insert required string delimiter. | XREAD, XWRITE |
| MISSING SEPARATOR AFTER CONVERSION ITEM | Conversion item must be followed by a comma, semicolon, or right parenthesis, depending on circumstances. | Insert necessary separator. | XREAD, XWRITE |
| MISSING SOURCE-1 PARAMETER | Directive parameter list is incomplete. | Correct parameter list. | XCOMP, XMOVE |
| MISSING SOURCE-2 OR DESTINATION PARAMETER | Directive parameter list is incomplete | Correct parameter list. | XCOMP, XMOVE |
| MISSING LEFT STRING DELIMITER | Missing * or $ in a literal string. | Insert required string delimiter. | XREAD, XWRITE |
| MORE DATA AFTER RECORD IN V-UNBLOCKED FILE | Actual record length in input file exceeds that specified in record descriptor word (RDW). | Check for incorrect RFM specification - should be VB. | XREAD |
| MORE DATA AFTER VS-RECORD SEGMENT | Actual segment record length in the input file exceeds that specified in the segment descriptor word (SDW). | A bad file copy could have occurred. Recopy file. | XREAD |
| NO FILE STRING GIVEN | File description parameters are missing in an INP or OUT directive. | Specify file-string. | XFILE |
| NO PARAMETERS | Necessary parameters were not specified. | Specify required parameters. | Any 8-bit subroutine |
| NO PARAMETERS SUPPLIED TO SUBROUTINE | Parameters are missing from directive. | Specify parameters. | Any 8-bit subroutine |
| NO STATUS PARAMETER | Directive parameter list contains an error or a parameter is missing. | Specify correct parameter. | XCOMP |
| NO WORKING STORAGE AREA PROVIDED | Parameter list contains an error. | Specify RFM parameter in the INP directive. | XFILE |

TABLE B-3. 8-BIT SUBROUTINE DIAGNOSTICS (Contd)

| Message | Significance | Action | Issued By |
|---|---|---|---|
| NUMERIC LITERAL EXPONENT .GE. 512 | Exponent value cannot exceed 511. | Decrease exponent value and rerun. | XREAD, XWRITE |
| NUMERIC LITERAL OUT OF RANGE (INFINITE) | Numeric literal has infinite value. | Correct the literal. | XWRITE |
| PARAMETER IS NOT A DATA ITEM | A literal was supplied for a data-item in COBOL. | Replace with a variable or array name. | Any 8-bit sub-routine |
| RECFM NOT SPECIFIED | RFM parameter was omitted from an INP or OUT directive. | Specify RFM parameter. | XFILE |
| RELATIONAL OPERATOR NOT RECOGNIZED | Illegal relationship mnemonic specified in a selector expression. | Specify valid operation and rerun. | XREAD, XWRITE |
| SECOND SELECTOR-EXPRESSION ITEM NOT RECOGNIZED | The second iTm field in a selector expression is in illegal format. | Specify correct iTm field. | XREAD, XWRITE |
| SELECTOR-EXPRESSION NOT TERMINATED BY COLON | The selector expression must be terminated by a colon. | Insert required colon. | XREAD, XWRITE |
| SOURCE CHARACTER NOT 0 OR 1, TO BIT STRING | In a character-to-bit conversion item, the source character must be 0 or 1. | Specify source character of 0 or 1. | XREAD, XWRITE |
| SOURCE EXPONENT TOO LARGE, NOT REPRESENTABLE | The exponent of the floating-point number in the source field is too large, and the conversion cannot be performed. | Correct the error and rerun. | XREAD, XWRITE |
| STRING NOT IN NUMERIC SYNTACTIC FORM | The character string is not in the proper format for conversion to numeric type. | Correct the format and rerun. | XREAD, XWRITE |
| STRING RELATION IS NOT .EQ. OR .NE. | Only the relationships .EQ. or .NE. are legal for string value fields in selector expressions. | Correct the relationship. | XREAD, XWRITE |
| SYNTAX...NO DIGIT AFTER -E- IN NUMERIC LITERAL | Numeric literals used as operands in selector expressions must fit the numeric literal definition. | Place digit after E in numeric literal. | XREAD, XWRITE |
| TEST FIELD EXTENDS PAST END OF RECORD | The item descriptor specified in a selector expression begins within the record but extends beyond its logical length. | Reduce length specification of the item descriptor. | XREAD, XWRITE |
| TEST FIELD NOT IN RECORD, ON LEFT | The locator field specified in a selector expression references a character preceding the first one in the logical record. | Decrease magnitude of i in -iTm. | XREAD, XWRITE |
| TEST FIELD NOT IN RECORD, ON RIGHT | The locator field specified in a selector expression references a character beyond the last one in the logical record. | Reduce value of i in iTm. | XREAD, XWRITE |

TABLE B-3. 8-BIT SUBROUTINE DIAGNOSTICS (Contd)

| Message | Significance | Action | Issued By |
|---|---|---|---|
| TOO MANY DIGITS IN Z, S, N, OR P FIELD -- OVERFLOW | The magnitude of the number to be stored in the field exceeds the number of digits specified. | Increase length specification of the destination field descriptor. | XREAD, XWRITE |
| TOO MANY PARAMETERS | Extraneous parameters appear in the parameter list. | Specify valid parameters. | Any 8-bit subroutine |
| UNRECOGNIZED CODE SET SPECIFIED | Legal code sets are ASCII (A), EBCDIC (C), or Display Code (X). | Specify correct code set. | XMOVE, XCOMP |
| UNRECOGNIZED KEYWORD IN FILE STRING | A parameter in an INP or OUT directive is misspelled. | Correct spelling and rerun. | XFILE |
| UNRECOVERABLE ERROR ON WRITE FILE | Parity error. | Follow site-defined procedures for reporting software errors or operational problems. | XWRITE |
| V-BLOCK HAS SHORT RECORD FRAGMENT | The actual record size is less than that specified in the record descriptor word (RDW). | Probable file error - rewrite file. | XREAD |
| V-RECORD LENGTH EXCEEDS BLOCK SIZE | The actual record size exceeds the specified block size. | Specify larger block size in BLK parameter. | XREAD |
| V-RECORD LENGTH LESS THAN 4 BYTES | Variable records must contain 4 bytes for the record descriptor word (RDW). | Rewrite tape. | XREAD |
| VALUE TOO LARGE FOR FIELD WIDTH | Numeric value contains too many digits and/or symbols for the receiving field. | Increase length specification of the receiving field descriptor. | XREAD, XWRITE |
| VS-RECORD FINAL SEGMENT MISSING | The final segment of a variable spanned logical record is missing from the input file. | Rewrite tape. | XREAD |
| VS-RECORD FOUND IN TYPE V FILE | A segment descriptor word (SDW) was found in a file that was not spanned. | Rewrite tape. | XREAD |
| VS-RECORD INITIAL SEGMENT MISSING | The first segment of a spanned logical record is missing from the input file. | Rewrite tape. | XREAD |
| WORKING STORAGE AREA TOO SMALL | The size of the workspace buffer specified in the XFILE call is too small. | Follow site-defined procedures for reporting software errors or operational problems. | XFILE |
| WSA NOT ALIGNED ON WORD BOUNDARY | The workspace parameter in a COBOL calling sequence to XFILE must have a beginning character position of 0. | Declare working storage area (WSA) as level 01 item or use SYNCHRONIZE clause for proper alignment. | XFILE |

# GLOSSARY

Actual Key File -
A mass storage file in which each record is stored at the location specified by the block and record slot number in the primary key associated with that record.

Advanced Access Methods (AAM) -
A file manager that processes indexed sequential, direct access, and actual key file organizations and supports the Multiple Index Processor. (See CYBER Record Manager.)

ASCII -
The American Standard Code for Information Interchange, used under NOS as the ASCII 128-character set with either 6- or 12-bit characters and under NOS/BE as the ASCII 95-character set with 12-bit characters stored eight bits right-justified in a 12-bit byte.

ASCII Graphic 63-Character Set -
A subset of the ASCII 128-character graphic and control set. The % character and related card code do not exist.

ASCII Graphic 64-Character Set -
A subset of the ASCII 128-character graphic and control set.

ASCII Graphic 95-Character Set -
Consists of all the characters in the ASCII 128-character set that can be printed at a line printer. Only 12-bit ASCII code files can be printed using the ASCII graphic 95-character set. The 95-character set is represented by the 12-bit ASCII codes $0040_8$ through $0176_8$.

ASCII Graphic 128-Character Set -
Consists of all letters (uppercase and lowercase), digits, special symbols, and device control characters.

ASCII 8-Bit Code -
Eight bits stored in an 8-bit byte. IBM data can be ASCII 8/8.

ASCII 12-Bit Code -
The ASCII 7-bit code ( as defined by ANSI Standard x3.4-1977) right-justified in a 12-bit byte. Assuming that the bits are numbered from the right, bits 0 through 6 contain the ASCII code, bits 7 through 10 contain zeros, and bit 11 distinguishes the 12-bit ASCII $0000_8$ code from the end-of-line byte. The 12-bit codes are $0001_8$ through $0177_8$ and $4000_8$.

Basic Access Methods (BAM) -
A file manager that processes sequential and word addressable file organizations. (See CYBER Record Manager.)

Beginning-of-Information (BOI) -
The start of the first user record in a file. System information, such as tape labels of sequential files or indexes of indexed sequential files, does not affect the beginning-of-information.

Block -
The term block has several meanings depending on context. On tape, a block is information between interrecord gaps on tape. CYBER Record Manager defines several blocks depending on organization, as shown in table C-1.

TABLE C-1. BLOCK TYPES

| Organization | Blocks |
|---|---|
| Indexed sequential | Data block; index block |
| Direct access | Home block; overflow block |
| Actual key | Data block |
| Sequential | Block type I, C, K, E |

Byte -
A group of bits. Unless prefixed (for example, a 6-bit byte), the term implies 8-bit groups. When used for encoding character data, a byte represents a single character.

CON Directive -
The FORM directive that performs conversions between CDC and IBM files.

Conversion Item -
A conversion string component that causes a data item in a CDC or IBM input record to be converted to IBM or CDC format and transmitted to an output record. Also refers to a nested conversion string.

Conversion Specification -
A component of a conversion string. A conversion specification consists of an optional selector expression and a list of associated conversion items.

Conversion String -
A string of one or more conversion specfications; used as input to the CON directive.

CYBER Record Manager -
A generic term relating to the common products AAM and BAM that run under the NOS and NOS/BE operating systems and that allow a variety of record types, blocking types, and file organizations to be created and accessed.

Direct Access File -
In the context of AAM, a direct access file is one of three file organizations implemented according to AAM initial direct or extended direct files. A direct access file is characterized by the system hashing of the unique key within each file record to distribute records randomly in blocks called home blocks of the file.

In the context of NOS, a permanent file is a direct access file that is accessed and modified directly.

Directive -
A statement consisting of a 3-letter mnemonic and a parameter list through which the user specifies processing to be performed by FORM. The directive mnemonics are: INP, OUT, CON, QAL, REF, SEQ, PAG, and XEQ.

End-of-Information (EOI) -
CRM defines end-of-information in terms of the file organization and file residence, as shown in table C-2.

TABLE C-2. END-OF-INFORMATION BOUNDARIES

| File Organization | File Residence | Physical Position |
|---|---|---|
| Sequential | Mass storage | After the last user record. |
| | Labeled tape in SI, I, S, or L format | After the last user record and before any file trailer labels. |
| | Unlabeled tape in SI or I format | After the last user record and before any file trailer labels. |
| | Unlabeled tape in S or L format | Undefined. |
| Word Addressable | Mass storage | After the last word allocated to the file, which might be beyond the last user record. |
| Indexed Sequential, Actual Key | Mass storage | After the record with the highest key value. |
| Direct Access | Mass storage | After the last record in the most recently created overflow block or home block with the highest relative address. |

End-of-Record (EOR) -
The end of a logical record of data.

Entry Point -
A location within a program to which control can be transferred from another program. Each entry point has a unique name.

Extended -
A term used in conjunction with indexed sequential, direct access, and actual key files to denote a specific type of internal processing by AAM. Contrast with Initial.

Field -
A portion of a word or record; a subdivision of information within a record; also, a generic entry in a file information table identified by a mnemonic.

File -
A logically related set of information; the largest collection of information that can be addressed by a file name. It starts at beginning-of-information and ends at end-of-information. Every file in use by a job must have a logical file name.

FILE Control Statement -
A control statement that supplies file information table values after a source language program is compiled or assembled but before the program is executed. In applications such as those with a control statement call to the FORM utility, FILE control statements must be used. Basic file characteristics such as organization, record type, and description can be specified in the FILE control statement.

File Information Table (FIT) -
A table through which a program communicates with CRM. For direct processing through CRM, a user must initiate establishment of this table. All file processing executes on the basis of information in this table. The user can set FIT fields directly, or indirectly by using parameters in a file access call that sets the fields indirectly.

File Organizer and Record Manager (FORM) -
A file management utility callable by control statements.

Hashing -
The method of using primary keys to search for relative home block addresses of records in a file with direct access storage structure.

Home Block -
A block in a file with direct access storage structure whose relative address is computed by hashing keys. A home block contains synonym records whose keys hash to that of the relative address. If all the synonym records cannot be accommodated in the home block, an overflow block can be created by the system. A user creating a direct access file must define the number of home blocks with the HMB parameter in the FILE control statement.

Indexed Sequential File -
A file organization in which AAM maintains files in sorted order by use of a user-defined primary key, which need not be within the record. Keys can be integer, floating-point (initial indexed sequential files only) or symbolic; access is random or sequential. Files contain index blocks and data blocks.

Initial -
A term used in conjunction with indexed sequential, direct access, and actual key files to denote a specific type of internal processing by AAM. Contrast with Extended.

INP Directive -
The FORM directive through which the user specifies the input file name and certain input file characteristics. One input file can be declared for a FORM run.

Item Descriptor -
A FORM element that describes a data field in an input or output record in terms of starting position, data type, and field length.

Key -
A group of contiguous characters or numbers the user defines to identify a record.

Keyword -
A word that has a special meaning to FORM when used in a specific context.

Logical File Name (LFN) -
The name given to a file being used by a job. The name must be unique for the job, and must consist of one to seven letters or digits, the first of which must be a letter.

Longest Logical Record (LRL) -
A parameter specified in the INP and OUT directives. LRL describes length, in number of 8-bit bytes, of the longest logical record expected.

OUT Directive -
The FORM directive through which the user specifies the name of the output file to be generated by FORM and certain output file characteristics. Up to 20 output files can be generated in a FORM run.

Owncode -
A routine written by the user to process certain conditions. Control passes automatically to user owncode routines defined for:

| | |
|---|---|
| LX | Tape label processing |
| DX | End-of-partition, end-of-section, end-of-data |
| EX | Error condition |
| CX | IBM conversion error |
| DCA | Record decompression/decryption |
| CPA | Record compression/encryption |
| HRL | Key hashing |
| IX | Transmitting input records |
| FEX | Fatal error condition |

PAG Directive -
The FORM directive through which the user specifies page formatting options for a file to be printed.

Partition -
CRM defines a partition as a division within a file with sequential organization. Generally, a partition contains several records or sections. Implementation of a partition boundary is affected by file structure and residence, as shown in table C-3.

Q Specification -
A character input to the REF and CON directives that terminates reformatting or conversion.

TABLE C-3. PARTITION BOUNDARIES

| Device | Record Type (RT) | Block Type (BT) | Physical Boundary |
|---|---|---|---|
| PRU device | W | I | A short PRU of level 0 containing a one-word deleted record pointing back to the last I block boundary, followed by a control word with a flag indicating a partition boundary. |
| | W | C | A short PRU of level 0 containing a control word with a flag indicating a partition boundary. |
| | D,F,R, T,U,Z | C | A short PRU of level 0 followed by a zero-length PRU of level $17_8$. |
| | S | - | A zero-length PRU of level number $17_8$. |
| S or L format tape | W | I | A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a deleted one-word record pointing back to the last I block boundary, followed by a control word with a flag indicating a partition boundary. |
| | W | C | A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a control word with a flag indicating a partition boundary. |
| | D,F,T, R,U,Z | C,K,E | A tapemark. |
| | S | - | A tapemark. |
| Any other tape format | - | - | Undefined. |

QAL Directive -
The FORM directive through which the user specifies criteria by which records are selected from an input file for inclusion in an output file.

Qualification String -
A string of selector expressions joined by logical operators and used as input to the QAL directive.

Random File -
In the context of CRM, a file with word addressable, indexed sequential, direct access, or actual key organization in which individual records can be accessed by the values of their keys.

Record -
The largest collection of information passed between CRM and a user program in a single read or write operation. The user defines the structure and characteristics of records within a file by declaring a record format. The beginning and ending points of a record are implicit in each format.

Record Type -
CRM defines eight record types established by the RT field in the file information table.

REF Directive -
The FORM directive through which the user specifies record reformatting options.

Reformat Item -
A reformat string component that causes a data field from an input record to be reformatted and transmitted to an output record; can also refer to a nested reformat string.

Reformat Specification -
A component of a conversion string consisting of an optional selector expression and a list of one or more associated reformat items.

Reformat String -
A string of one or more reformat specifications; used as input to the REF directive.

Search Descriptor -
A FORM element used in conjunction with item descriptors to describe data fields of unknown length or starting position.

Section -
CRM defines a section as a division within a file with sequential organization. Generally, a section contains more than one record and is a division within a partition of a file. A section terminates with a physical representation of a section boundary, as shown in table C-4.

The NOS and NOS/BE operating systems equate a section with a system-logical-record of level 0 through $16_8$.

TABLE C-4. SECTION BOUNDARIES

| Device | Record Type (RT) | Block Type (BT) | Physical Representation |
|---|---|---|---|
| PRU device | W | I | A deleted one-word record pointing back to the last I block boundary followed by a control word with flags indicating a section boundary. At least the control word is in a short PRU of level 0. |
| | W | C | A control word with flags indicating a section boundary. The control word is in a short PRU of level 0. |
| | D,F,R, T,U,Z | C | A short PRU with a level less than $17_8$. |
| | S | - | Undefined. |
| S or L format tape | W | I | A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a deleted one-word record pointing back to the last I block boundary, followed by a control word with flags indicating a section boundary. |
| | W | C | A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a control word with flags indicating a section boundary. |
| | D,F,R, T,U,Z | C,K,E | Undefined. |
| | S | - | Undefined. |
| Any other tape format | - | - | Undefined. |

Selector Expression -
An expression used in the QAL, CON, and REF directives to perform a relational test on a data field; if the test is satisfied, the associated operations are performed; if not, the associated operations are ignored.

SEQ Directive -
The FORM directive through which the user specifies sequence numbering to be performed on output records.

Sequential File -
A file with records stored in the physical order in which they were written. No logical order exists other than the relative physical record position.

Simple Item Conversion -
A conversion string component that performs data conversions to or from IBM format. (See Conversion Item.)

Simple Reformat -
A reformat string component that performs data reformatting. (See Reformat Item.)

Word Addressable File -
A mass storage file containing continuous data or space for data. Words within a word addressable file are numbered from 1 to n, each word containing 10 characters. Retrieving or writing of data at any given word within the file is specified by the word number, called the word address.

Working Storage Area (WSA) -
An area within the user's field length intended for receipt of data from a file or transmission of data to a file.

XEQ Directive -
The FORM directive that specifies the names of the IX and FEX owncode routines to FORM.

# IBM TAPE FILE RECORD AND BLOCKING FORMATS

Files can be written and read in IBM format by FORM. Parameters are specified in the INP and OUT directives to describe the record and blocking format to be used. These parameters, described below, correspond to the data control block (DCB) subparameters RECFM, BLKSIZE, and LRECL on a data definition (DD) statement in IBM job control language (JCL).

## BLK PARAMETER

The BLK parameter specifies length, in number of 8-bit bytes, of the longest tape block expected, regardless of format. For variable (V) format records, the specification must include space for a 4-byte block header. The value for BLK must not exceed 32760.

## LRL PARAMETER

The LRL parameter specifies length, in number of 8-bit bytes, of the longest logical record (LRL) expected. For V format records, the specification must include space for a 4-byte record header. The value for LRL must not exceed 32756 for V format and 32760 for fixed (F) and unspecified (U) formats. LRL must not exceed BLK for all formats except variable spanned blocked (VBS).

## RFM PARAMETER

The RFM parameter describes the physical arrangement for records in blocks and in files. The record formats are fixed (F), fixed blocked (FB), variable (V), variable blocked (VB), variable spanned blocked (VSB), and unspecified (U).

### FIXED RECORD FORMATS

The two type of fixed format records are F and FB.

#### F RECORD FORMAT

For F record formats, every record is exactly LRL bytes long. Each block contains exactly one record. F format is shown in figure D-1.

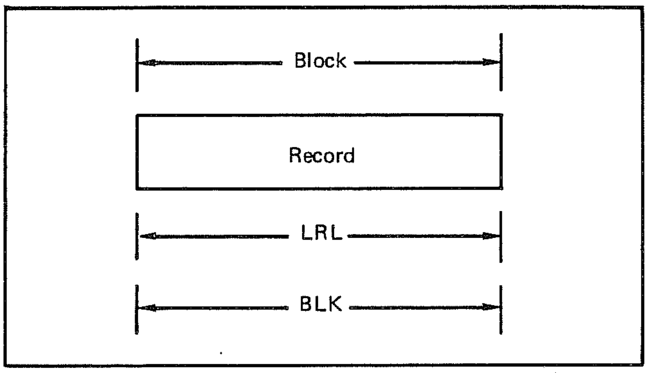


Figure D-1. F Format Record

#### FB RECORD FORMAT

For FB record formats, every record is exactly LRL bytes long. Each block contains an integral number of records. Total block length must not exceed BLK. FB format is shown in figure D-2.

### VARIABLE RECORD FORMATS

The variable formats are V, VB, and VSB.

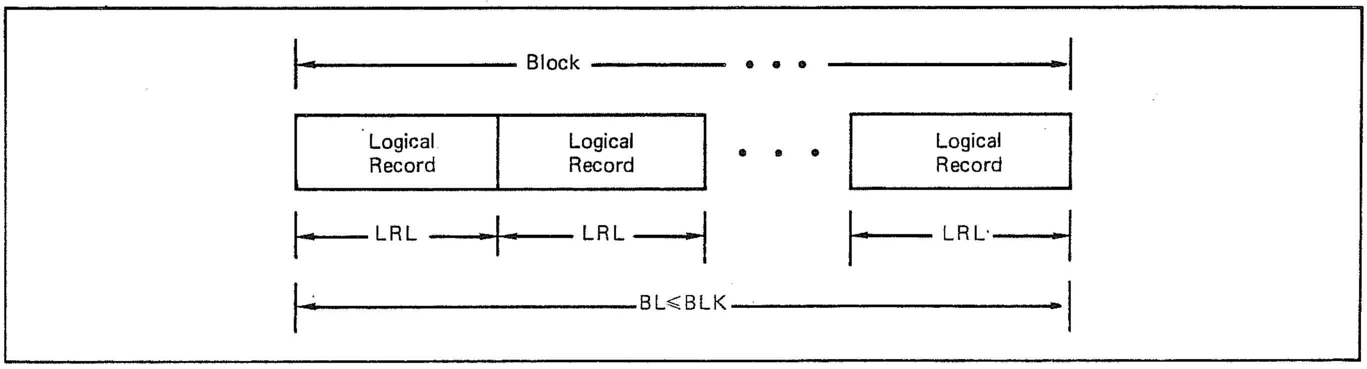


Figure D-2. FB Format Block

## V RECORD FORMAT

The maximum record length for V record formats is LRL-4 bytes. Each record is prefixed by a 4-byte record descriptor word (RDW) as shown in figure D-3. Each block contains exactly one record and is prefixed by a block descriptor word (BDW) as shown in figure D-4.

## VB RECORD FORMAT

The maximum record length for VB record formats is LRL-4 bytes. This format is identical to V, except that several records can occupy the block. The block length must not exceed BLK. VB format is shown in figure D-5.

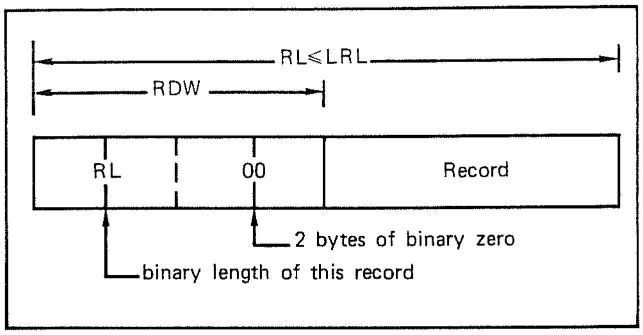


Figure D-3. V Format Record

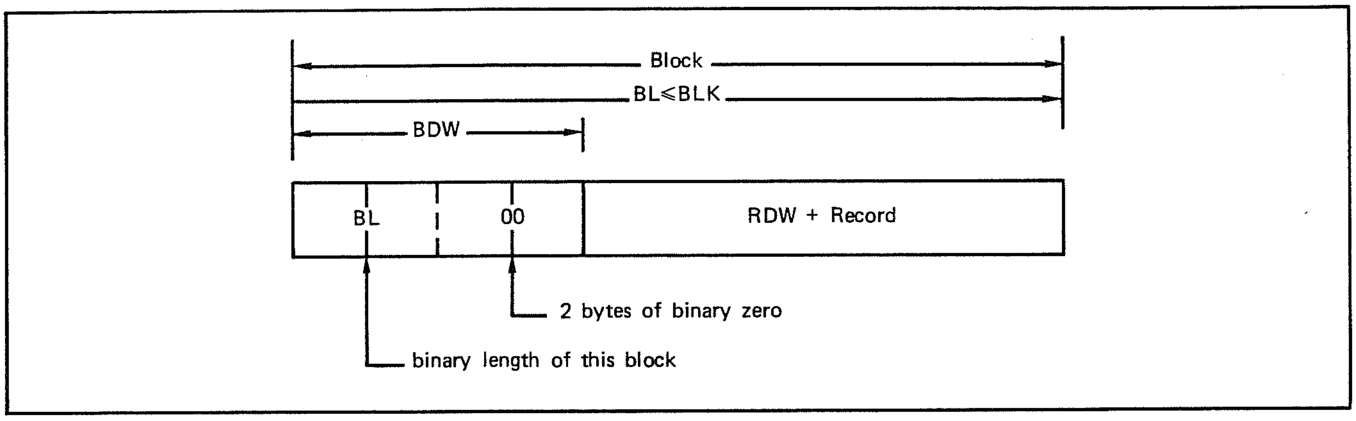


Figure D-4. V Format Block

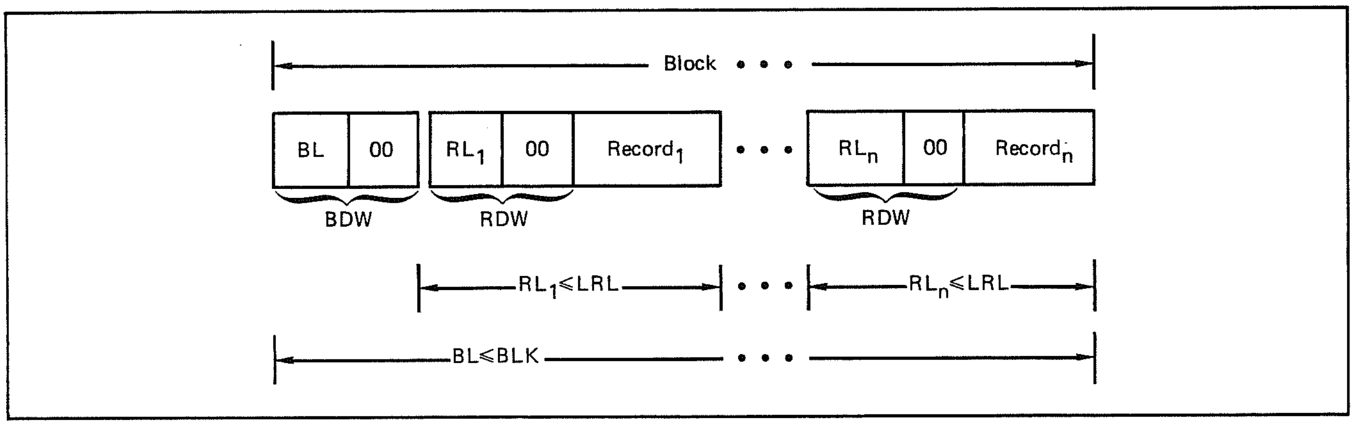


Figure D-5. VB Format Records

## VSB RECORD FORMAT

In the VSB record format, several record segments can occupy a block (as in VB). At most, one segment of a record can occur in a block. An attempt is made to fill the block to BLK; the attempt fails if the remaining space is less than 4 bytes, since at least one data byte must be written. If the attempt fails, the current block is ended and a new block is started. VSB format is shown in figure D-6.

## UNSPECIFIED RECORD FORMAT

The unspecified format is U. Undefined length records can be any nonzero length, up to LRL. Each record occupies exactly one block. U format is shown in figure D-7.

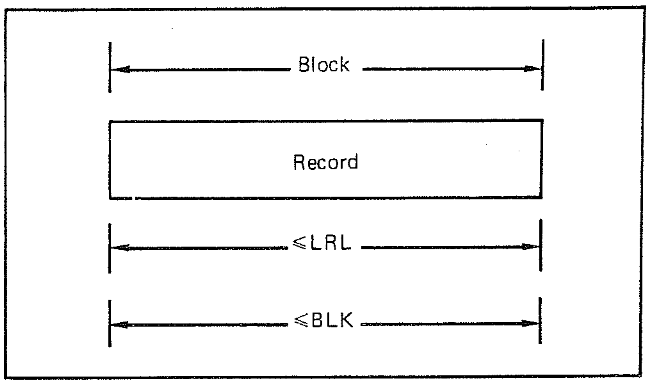


Figure D-7. U Format Record

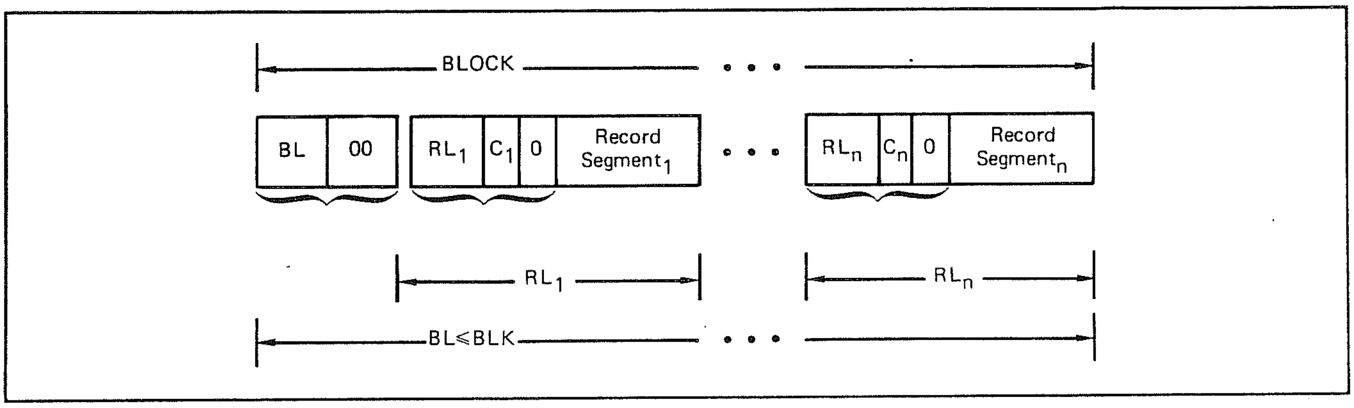


Figure D-6. VSB Format Records

# DATA FORMATS

An IBM tape file created by an IBM FORTRAN or COBOL source program can differ in data type from a CDC file created by a CDC FORTRAN or COBOL source program. This section shows the relationship between the various data type declarations, the storage allocations for the various data types, and the corresponding Tm values for conversion use with the various data types.

## IBM FORTRAN DATA FORMATS

IBM FORTRAN programs can utilize six types of constant data and four types of variable data. The constant type determines the number of 8-bit byte storage locations needed to represent the data. A default and optional length specification for each variable type determines the number of bytes reserved for that type. Table E-1 gives the associated length specifications for constants and variables.

An IBM tape file created by an IBM FORTRAN program can contain combinations of the following data types:

| | |
|---|---|
| 8-bit characters | (X) |
| half-word integers, 16 bits | (H) |
| whole-word integers, 32 bits | (W) |
| 32-bit floating-point | (F) |
| 64-bit floating-point | (L) |

The letters in parentheses are the corresponding T values as described in table 2-1 of section 2. An IBM FORTRAN program cannot create 64-bit double-word integers or 128-bit extended-precision floating-point data. All of the above can be processed as bit string data (B).

Table 2-1 of section 2 should also be consulted to determine the appropriate m value for each T selected. A more complete description of IBM data format types appears in appendix F.

Table E-2 shows the relationship between IBM FORTRAN data type declarations, byte storage allocation, and the corresponding Tm values for conversion use.

Table E-2 contains only examples of explicit type declarations. COMPLEX declarations do not appear since the complex variable is composed of two real data items and follows the conventions for REAL data types. The user determines conversion of LOGICAL data types.

TABLE E-1. IBM FORTRAN - CONSTANT AND VARIABLE SIZES

| Type | 8-Bit Bytes Allocated |
|---|---|
| Constant | |
| Integer | 2, 4, or 8 |
| Real | 4 (single-precision)<br>8 (double-precision)<br>16 (long-precision) |
| Complex | 8 (two single-precision)<br>16 (two double-precision) |
| Logical | 1 or 4 |
| Literal | 1 ≤ n (n is string length) |
| Packed Decimal | 1 byte for each 2 digits |
| Variable | |
| Integer | 4 (default)<br>2 or 8 (optional) |
| Real | 4 (default)<br>8 or 16 (optional) |
| Complex | 8 (default)<br>16 (optional) |
| Logical | 4 (default)<br>1 (optional) |

TABLE E-2. IBM FORTRAN - Tm VALUES

| IBM FORTRAN Type Declaration | IBM Boundary Alignment | T | 8-Bit Bytes Used[†] |
|---|---|---|---|
| Integer*2 | Half-word | H | 2 |
| Integer Integer*4 | Full-word | W | 4 |
| Integer*8 | Double-word | G | 8 |
| Real Real*4 | Full-word | F | 4 (Single-precision) |
| Real*8 Double-precision | Double-word | L | 8 (Double-precision) |
| Real*16 | Double-word | E | 16 |
| Logical | Full-word | Usage defined | 4 |
| Logical*1 | None | Usage defined | 1 |

[†]The value used for m is determined by the number of bytes used.

## CDC FORTRAN DATA FORMATS

The type of constant or variable determines the number of central memory words needed to represent it. Table E-3 shows the constant and variable types and the associated 6-bit byte length specifications for CDC FORTRAN Extended Version 4 and CDC FORTRAN Version 5.

Records written by a CDC FORTRAN program can contain combinations of the following data types. The letter in parentheses is the corresponding T value as determined from table 2-1 of section 2:

| | |
|---|---|
| 6-bit characters | (X) |
| whole-word integer, 60 bits | (I) |
| single-word floating-point, 60 bits | (E) |
| double-word floating-point, 120 bits | (D) |

All of the preceding data types can be processed as type B bit string data. Table 2-1 of section 2 should be consulted to determine the appropriate m value. While a whole-word integer is a data type, integers used in multiplication and division operations are truncated to 48 bits. More complete information on this subject appears in appendix D and in the appropriate FORTRAN reference manual.

Table E-4 shows the relationship between CDC FORTRAN data type declarations, storage allocation for each type, and corresponding Tm values for conversion use.

COMPLEX type declarations do not appear in the table; the complex variable is composed of two real data items and follows the conventions for REAL data types. The user determines the manner in which LOGICAL data items are to be converted.

## IBM COBOL DATA FORMATS

A tape file created by an IBM COBOL program can contain any combination of the following data types (the letters in parentheses are the T values given in table 2-1 of section 2):

| | |
|---|---|
| 8-bit characters | (X) |
| half-word integers, 16 bits | (H) |
| whole-word integers, 32 bits | (W) |
| double-word integers, 64 bits | (G) |
| 32-bit floating-point | (F) |
| 64-bit floating-point | (L) |
| packed decimal (internal decimal) | (P) |
| decimal signed numeric (external decimal) | (S) |

An IBM COBOL program cannot create 128-bit extended-precision floating-point data. Any of the preceding data types can be considered to be bit string data (B). To select the appropriate m value for each of the data types, refer to table 2-1 of section 2. A complete description of IBM data type formats is given in appendix F.

If IBM computational items are mixed with other elementary items in the data record description, slack bytes might be present on IBM tape files to assure the proper byte alignment for each COMPUTATIONAL, COMPUTATIONAL-1, and COMPUTATIONAL-2 item. For instance, the byte address of the first byte of a half-word binary item must be divisible by 2; the byte address of a full- or double-word binary item must be divisible by 4; the byte address of a COMPUTATIONAL-1 item must be divisible by 4; the byte address of a COMPUTATIONAL-2 item must be divisible by 8.

Table E-5 shows the relationship between IBM COBOL USAGE clauses, picture clauses, IBM byte storage allocation, and the corresponding 8-bit Tm values to be used in converting such data types.

TABLE E-3. CDC FORTRAN - CONSTANT AND VARIABLE SIZES

| Type | FORTRAN Extended 4<br>6-Bit Bytes Allocated | FORTRAN 5<br>6-Bit Bytes Allocated |
|---|---|---|
| | Constant | |
| Complex | 20 | 20 |
| Double-Precision | 20 (two real constants) | 20 (two real constants) |
| Hollerith | 1 n (n is string length) | 10 |
| Integer | 10 | 10 |
| Logical | 10 | 10 |
| Octal | 10 | 10 |
| Real | 10 | 10 |
| Boolean† | †† | 10 |
| Character | †† | 1 per character up to length of string; minimum string length 1, maximum string length $2^{15}-1$ |
| Hexadecimal | †† | 10 |
| | Variable | |
| Complex | 20 | 20 |
| Double-Precision | 20 | 20 |
| Integer | 10 | 10 |
| Logical | 10 | 10 |
| Real | 10 | 10 |
| Boolean† | †† | 10 |
| Character | †† | User defined |

†The use of boolean data types and operations (SHIFT, MASK, and so forth) should be avoided where possible because boolean data types can be processor dependent.

††Applies only to FORTRAN 5.

TABLE E-4. CDC FORTRAN - Tm VALUES

| Type Declaration | CDC Boundary Alignment | T | FORTRAN Extended 4 6-Bit Bytes Used† | FORTRAN 5 6-Bit Bytes Used† |
|---|---|---|---|---|
| Integer | Full-word | I | 10 | 10 |
| Real | Full-word | E | 10 | 10 |
| Double-precision | Double-word | D | 20 | 20 |
| Logical | Full-word | Usage defined | 20 | 20 |
| Boolean†† | Full-word | I or B60 | ††† | 10 |
| Character | Character | X | ††† | User defined |

†The value used for m is determined by the number of bytes used.

††The use of boolean data types and operations (SHIFT, MASK, and so forth) should be avoided where possible, because boolean data types can be processor dependent. Type character should be used instead, if working with character data.

†††Applies only to FORTRAN 5.

## CDC COBOL DATA FORMATS

FORM recognizes the following data types written by a CDC COBOL program:

| | |
|---|---|
| 6-bit characters | (X) |
| unnormalized floating-point, 60 bit | (U) |
| numeric display signed overpunch | (S) |
| numeric display, unsigned | (N) |
| numeric display, leading zeros suppressed | (Z) |

The letters in parentheses are the corresponding T values as described in table 2-1 of section 2 under CDC Format. A CDC COBOL program cannot create or process 60-bit integer data. Any of the above data types can be considered as bit string data (B). To choose the corresponding m value for the conversion items, refer to the right half of table 2-1 of section 2. A more complete description of the CDC internal format types appears in appendix F.

Table E-6 shows the relationship between CDC COBOL USAGE clauses, PICTURE clauses, CDC storage allocation, and the corresponding 8-bit Tm values used in processing such data types.

TABLE E-5. IBM COBOL - Tm VALUES

| IBM COBOL USAGE Category | Picture Format | IBM Boundary Alignment | T | IBM 8-Bit Bytes Used (m) |
|---|---|---|---|---|
| DISPLAY | Alphabetic | None | X | 1 per character or digit (except for V in external floating-point), limit 18 |
| DISPLAY | Alphanumeric | None | X | 1 per character or digit (except for V in external floating-point), limit 18 |
| DISPLAY | Report format | None | X | 1 per character or digit (except for V in external floating-point), limit 18 |
| DISPLAY | External floating-point form | None | X | 1 per character or digit (except for V in external floating-point), limit 18 |
| DISPLAY | Numeric unsigned | None | N or X | 1 per digit, limit 18 |
| DISPLAY | Numeric signed | None | S | 1 per digit, limit 18 |
| COMP (binary) | 1 to 4 digits | Half-word | H | 2 |
| COMP (binary) | 5 to 9 digits | Full-word | W | 4 |
| COMP (binary) | 10 to 18 digits | Full-word | G | 8 |
| COMP-1 (internal floating-point) | Not applicable | Full-word | F | 4 (short-precision) |
| COMP-2 (internal floating-point) | Not applicable | Double-word | L | 8 (long-precision) |
| COMP-3 (packed decimal) | Numeric unsigned or signed | None | P | 1 byte per 2 digits plus 1 byte for low-order digit and sign |

TABLE E-6. CDC COBOL 5 - Tm VALUES

| CDC COBOL Usage | Picture Format | Boundary Alignment | T | CDC 6-Bit Bytes Used in Storage Allocation (m) |
|---|---|---|---|---|
| DISPLAY | Alphabetic | None | X | 1 per character or digit (except for V) |
| DISPLAY | Alphanumeric | None | X | 1 per character or digit (except for V) |
| DISPLAY | Edited report form, leading zeros suppressed | None | X | 1 per character or digit (except for V) |
| COMPUTATIONAL or DISPLAY | Unsigned numeric | None | N<br>X | 1 per digit, limit 18 |
| COMPUTATIONAL or DISPLAY | Signed numeric | None | S | 1 per digit, limit 18 |
| DISPLAY | Edit, leading zeros suppressed | None | Z | 1 per digit or space, limit 18 |
| COMP-1 | 1 to 14 digits | Word | I | 10 |
| COMP-2 | 1 to 14 digits | Word | E | 10 |
| COMP-4 | 1 to 14 digits, unsigned numeric | Byte | B | Number of bits required to store the maximum decimal value that can be represented by the specified number of digits in the PICTURE clause, divided by 6 and rounded up |
| COMP-4 | 1 to 14 digits, signed numeric | Byte | B | Number of bits required by unsigned items plus 1, divided by 6 and rounded up |

# INTERNAL DATA REPRESENTATION F

IBM and CDC internal data representation is explained in this appendix.

## IBM DATA FORMATS

IBM data formats consist of character and numeric data formats. There are four forms of numeric data: fixed-point binary, floating-point hexadecimal, packed decimal, and decimal signed numeric.

### CHARACTER DATA

EBCDIC character codes are stored in IBM systems in 8-bit bytes. Bit numbers are assigned from high to low for each bit position within a byte. By IBM standards, the numbers assigned are E0 through E7, as shown in figure F-1.

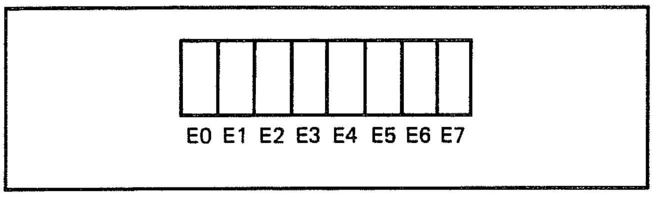


Figure F-1. IBM EBCDIC Bit Numbers

ASCII character codes are stored in IBM systems in 8-bit bytes. Bit numbers 1 through 8 are assigned to each bit position within a byte from low-order to high-order, as shown in figure F-2.

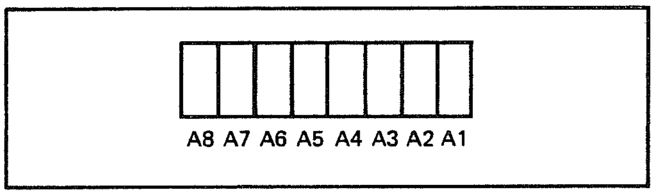


Figure F-2. IBM ASCII Bit Numbers

### NUMERIC DATA

In IBM systems, 8-bit bytes are grouped to represent numeric data. A double byte (16 bits) is referred to as a halfword; four bytes comprise a wholeword (fullword); eight bytes are a doubleword.

IBM uses four forms of numeric data: fixed-point binary, floating-point hexadecimal, packed decimal, and decimal signed numeric. The numeric data formats are shown in figure F-3.

#### Fixed-Point Binary

Fixed-point values can be written in halfword, fullword, or doubleword format consisting of a single sign bit followed by the binary field. On occasion, these formats are referred to as signed integer format. Negative values are represented in two's complement form.

#### Floating-Point Hexadecimal

Floating-point data occupies short, long, and extended-precision formats. Each form uses the first bit as the sign of the fraction, the next seven bits to represent a characteristic, and the remaining bits to represent the fraction expressed in hexadecimal digits. The value expressed is the product of the fraction and the number 16 raised to the power of the exponent.

The greatest precision is achieved when a floating-point number is normalized. The fraction part of the floating-point number has a nonzero, high-order, hexadecimal digit produced by shifting the fraction left until the high-order, hexadecimal digit is nonzero and reducing the characteristic by the number of hexadecimal digits shifted. A zero fraction cannot be normalized.

Normalization applies to hexadecimal digits; thus the three high-order bits of a normalized number can be zero.

#### Packed Decimal

Because decimal numbers may be expressed by four bits, it is possible to pack two 1-digit decimal values into one 8-bit byte. Variable length fields are used to contain packed decimal values; the rightmost four bits of the low-order byte of the field contain the sign of the value:

- The EBCDIC sign code generated is 1101 for minus and 1100 for plus.
- Packed decimal data is not the same as 8-bit character data and cannot be treated as such.

#### Decimal Signed Numeric

Also known as a Zoned Decimal Format, this representation is required for character set sensitive I/O devices. A Zoned Decimal Format number carries its sign in the leftmost four bits of the low-order byte. The zoned format is not used in decimal arithmetic operations.

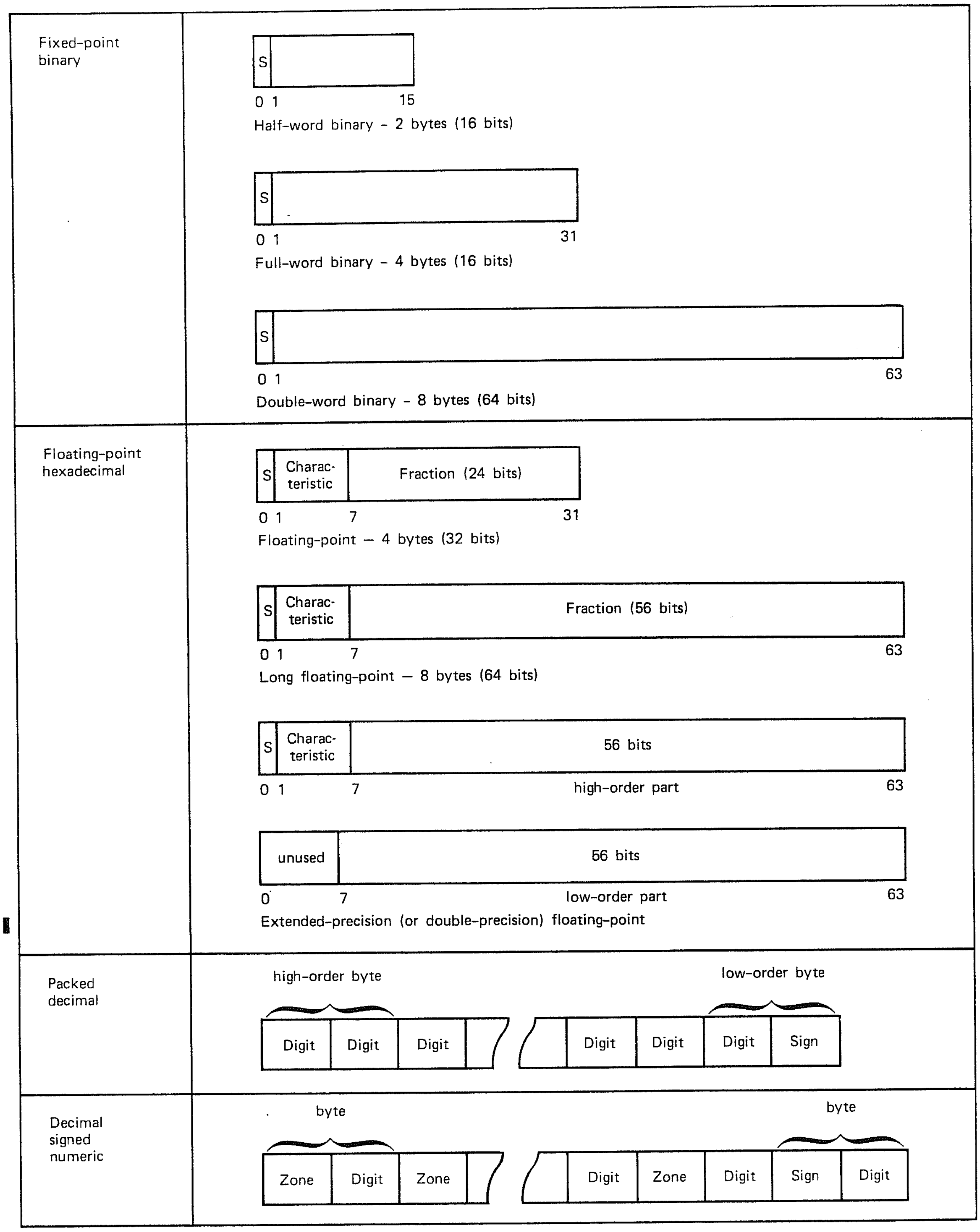
## CDC DATA FORMATS

CDC data formats consist of internal 6-bit display code, internal 8/12-bit ASCII code, and arithmetic data. The CDC arithmetic data types are integer and floating-point. CDC also stores data that is display code numeric sign overpunched.

### INTERNAL 6-BIT DISPLAY CODE

In the central memory unit of the CDC system, 6-bit coded information is represented in 6-bit bytes, ten bytes in a 60-bit word. Bytes are numbered as shown in figure F-4.

Within a word, 6-bit bytes are stored as shown in figure F-5.



Fixed-point binary
S
0 1 15
Half-word binary - 2 bytes (16 bits)
S
0 1 31
Full-word binary - 4 bytes (16 bits)
S
0 1 63
Double-word binary - 8 bytes (64 bits)
Floating-point hexadecimal
S Charac-teristic Fraction (24 bits)
0 1 7 31
Floating-point — 4 bytes (32 bits)
S Charac-teristic Fraction (56 bits)
0 1 7 63
Long floating-point — 8 bytes (64 bits)
S Charac-teristic 56 bits
0 1 7 high-order part 63
unused 56 bits
0 7 low-order part 63
Extended-precision (or double-precision) floating-point
Packed decimal
high-order byte
low-order byte
Digit Digit Digit
Digit Digit Digit Sign
Decimal signed numeric
byte
byte
Zone Digit Zone
Digit Zone Digit Sign Digit

Figure F-3. IBM Numeric Data Formats

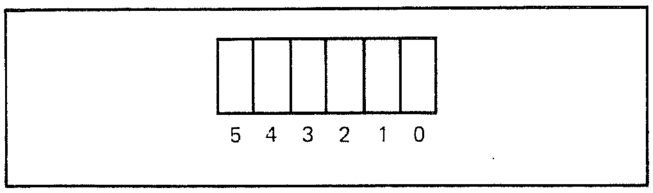


Figure F-4. CDC Display Code Bit Numbers

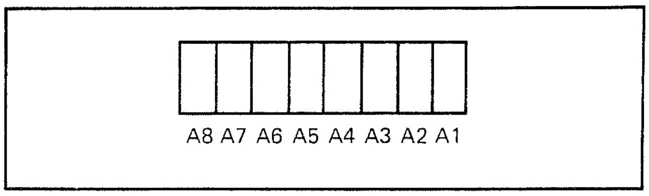


Figure F-6. CDC 8-Bit ASCII Bit Numbers

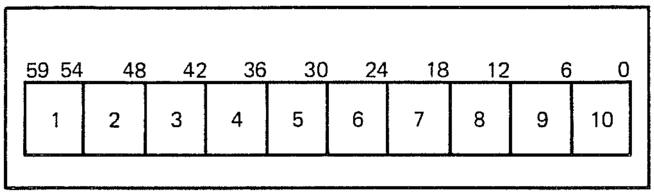


Figure F-5. CDC Display Code Byte Numbers

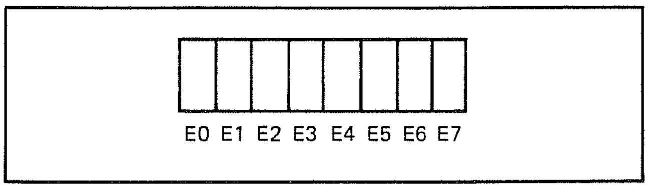


Figure F-7. CDC 8-Bit EBCDIC Bit Numbers

## INTERNAL REPRESENTATION OF 8-BIT DATA

The eight bits in a string representing an ASCII code are numbered from left to right (A8 through A1) as shown in figure F-6. The ASCII 8/12 data format consists of the ASCII 7-bit code right-justified in a 12-bit byte. Byte A8 is always set to zero. The eight bits in a string which represent EBCDIC code are numbered from left to right (E0 to E7) as shown in figure F-7.

The CDC 12-bit byte data format is shown in figure F-8. The unused four bits are set to zero when the byte is stored in a word and ignored when the character code is used.

In ASCII or EBCDIC codes, 8-bit data can be represented in central memory as 12-bit bytes, stored five bytes to a word. Character data must be aligned on the byte boundaries as shown in figure F-9.

## ARITHMETIC DATA

The CDC arithmetic data types are integer and floating-point. The data formats are shown in figure F-10.

### Integer

Integer data is stored in a 60-bit central memory word. The binary representation of the integer is right-justified in the word. The sign is in bit 59; the binary point is at the right of bit 0. Negative numbers are represented in one's complement notation.

### Floating-Point

Floating-point data is stored in either single-precision or double-precision format, as shown in figure F-10.

The binary point is considered to be to the right of the integer coefficient; therefore, the 48-bit integer coefficient is equivalent to a 14 digit value. The sign of the coefficient is in bit 59. Negative numbers are carried in one's complement notation. The 11-bit exponent carries a bias of $2^{10}$ (2000 octal). As the coefficient is stored in unnormalized form, the bias is removed when the word is normalized for computation and restored when the word is returned to floating-point format.

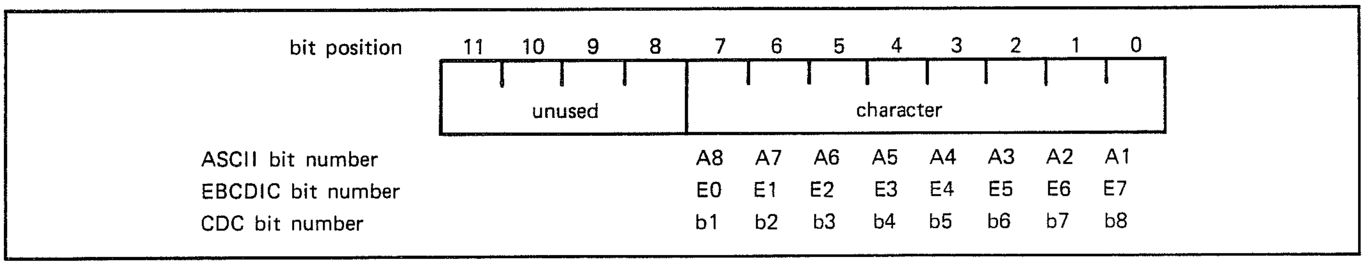


Figure F-8. CDC 12-Bit Byte Format

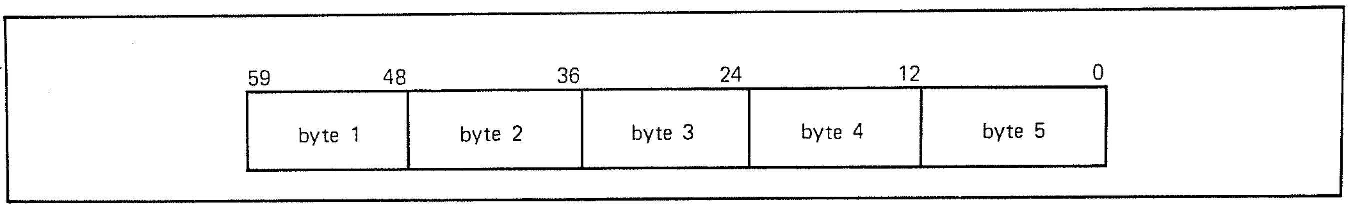


Figure F-9. CDC 5-Byte Word Format

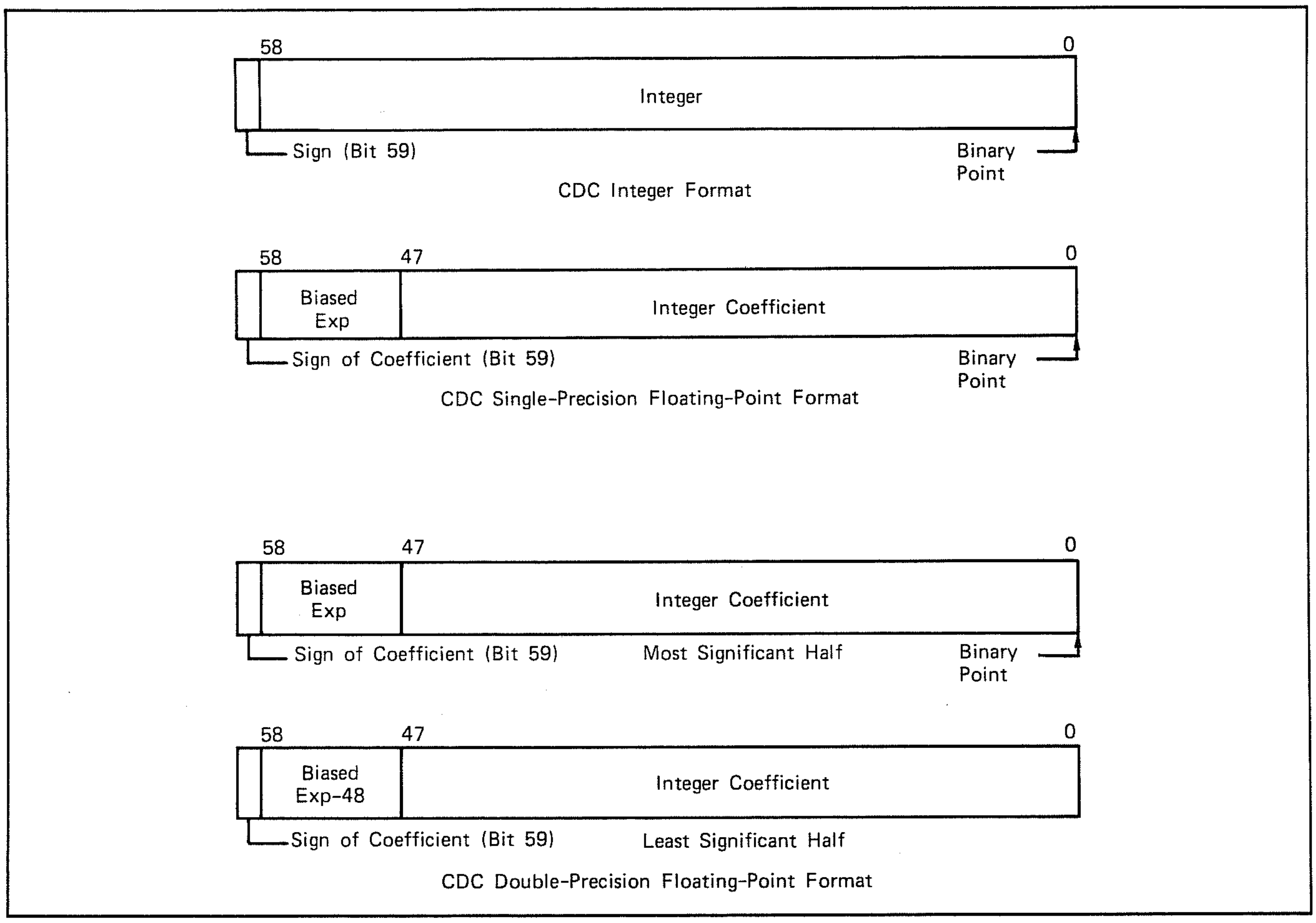


Figure F-10. CDC Numeric Data Formats

In double-precision format, two adjacent memory words (n and n+1) are used. The sign of the coefficient is carried in bit 59 of both words. The 96-bit integer coefficient is split, and the most significant 48 bits are stored in word n; the least significant 48 bits are in word n+1. The binary point is at the right of bit 0 in word n. Since the biased exponent of the least significant half of the coefficient is 48 less than the exponent of the most significant half, the two exponents are used to locate the true position of the binary point. If the exponent in word n represented 56, the exponent in word n+1 would be +8, indicating that the true position of the binary point is in the least significant half, eight bits to the right of the biased exponent in word n+1. Conversely, if the exponent in word n represented 32, the exponent in word n+1 would be -16, indicating that the true position of the binary point is in the significant half, 16 bits to the left of bit 0 in word n.

## Display Code Numeric Sign Overpunch

A string of display code decimal digits forms a display code sign overpunch number.

When the item is displayed in output or is received in input as a card image, the signed digit appears as specified in the second column of table F-1. When the item is to be received as input from a card, the signed digit must be punched as specified in the third column of table F-1. When input data is positive or unsigned, output data is the same as the input data. The negative sign is represented by a - overpunch in row 11; the positive sign by the absence of an overpunch or the presence of a + overpunch in row 12.

The sign overpunch numeric is CDC COBOL-defined, and insertion of the plus sign into the units digit is not automatic; therefore, all unsigned, signed, and positive numbers appear to have the same format. In the ∧ sign overpunch numeric, the 8-bit subroutines always insert the plus sign into a positive value and transform the units digit.

TABLE F-1. DISPLAY CODE NUMERIC SIGN OVERPUNCH REPRESENTATION

| Sign and Digit | Output Representation | | Hollerith Punch |
|---|---|---|---|
| | CDC | ASCII | |
| +9 | I | I | 12-9 |
| +8 | H | H | 12-8 |
| +7 | G | G | 12-7 |
| +6 | F | F | 12-6 |
| +5 | E | E | 12-5 |
| +4 | D | D | 12-4 |
| +3 | C | C | 12-3 |
| +2 | B | B | 12-2 |
| +1 | A | A | 12-1 |
| +0 | < | < | 12-0† |
| -0 | ∨ | ! | 11-0† |
| -1 | J | J | 11-1 |
| -2 | K | K | 11-2 |
| -3 | L | L | 11-3 |
| -4 | M | M | 11-4 |
| -5 | N | N | 11-5 |
| -6 | O | O | 11-6 |
| -7 | P | P | 11-7 |
| -8 | Q | Q | 11-8 |
| -9 | R | R | 11-9 |

†Under NOS, the 029 keypunch cannot be used to make the Hollerith punch patterns that represent +0 or -0.

# CONVERSION RULES G

Table G-1 shows conversion rules for the REF directive. Table G-2 shows conversion rules for the CON directive when converting from IBM format data to CDC format data. Table G-3 shows the conversion rules for the CON directive when converting from CDC format data to IBM format data. The numbers in the tables refer to notes following the tables. A hexadecimal-octal conversion table is provided at the end of this appendix.

TABLE G-1. CONVERSION RULES FOR REF DIRECTIVE

| Input Format \ Output Format | B | X | I | E | U | D | S | N | Z |
|---|---|---|---|---|---|---|---|---|---|
| B | 1 | 2 | 3,4 | 3,6 | 3,5 | 3,7 | 3,8 | 3,9 | 3,10 |
| X | 11 | 12 | 13,4 | 13,6 | 13,5 | 13,7 | 13,8 | 13,9 | 13,10 |
| I | 14 | 15 | 4 | 6 | 5 | 7 | 8 | 9 | 10 |
| E | 14 | 15 | 4 | 6 | 5 | 7 | 8 | 9 | 10 |
| U | 14 | 15 | 4 | 6 | 5 | 7 | 8 | 9 | 10 |
| D | 14 | 15 | 4 | 6 | 5 | 7 | 8 | 9 | 10 |
| S | 14,26 | 15,26 | 13,4 | 6,13,26 | 5,26 | 7,26 | 8,26 | 9,26 | 10,26 |
| N | 14 | 15 | 13,4 | 6,13 | 5 | 7 | 8 | 9 | 10 |
| Z | 14 | 15 | 13,4 | 6,13 | 5 | 7 | 8,26 | 9 | 10 |

TABLE G-2. CONVERSION RULES FOR CON DIRECTIVE: IBM TO CDC

| IBM \ CDC | B | X | I | U | E | D | S | N | Z |
|---|---|---|---|---|---|---|---|---|---|
| B | 1 | 2 | 3,4 | 3,5 | 3,6 | 3,7 | 3,8 | 3,9 | 3,10 |
| X | 11 | 12 | 13,4 | 13,5 | 13,6 | 13,7 | 13,8 | 13,9 | 13,10 |
| H | 14 | 15 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| W | 14 | 15 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| G | 14 | 15 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| F | 14 | 15 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| L | 14 | 15 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| E | 14 | 15 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| P | 14,25 | 15,25 | 4,25 | 5,25 | 6,25 | 7,25 | 8,25 | 9,25 | 10,25 |
| S | 14,26 | 15,26 | 4,26 | 5,26 | 6,26 | 7,26 | 8,26 | 9,26 | 10,26 |

TABLE G-3. CONVERSION RULES FOR CON DIRECTIVE: CDC TO IBM

| CDC \ IBM | B | X | H | W | G | F | L | E | P | S |
|---|---|---|---|---|---|---|---|---|---|---|
| B | 1 | 2 | 3,16 | 3,17 | 3,18 | 3,19,20 | 3,19,21 | 3,19,22 | 3,23,25 | 3,24,26 |
| X | 11 | 12 | 13,16 | 13,17 | 13,18 | 13,19,20 | 13,19,21 | 13,19,22 | 13,23,25 | 13,24,26 |
| I | 14 | 15 | 16 | 17 | 18 | 19,20 | 19,21 | 19,22 | 23,25 | 24,26 |
| U | 14 | 15 | 16 | 17 | 18 | 19,20 | 19,21 | 19,22 | 23,25 | 24,26 |
| E | 14 | 15 | 16 | 17 | 18 | 19,20 | 19,21 | 19,22 | 23,25 | 24,26 |
| D | 14 | 15 | 16 | 17 | 18 | 19,20 | 19,21 | 19,22 | 23,25 | 24,26 |
| S | 14 | 15 | 16 | 17 | 18 | 19,20 | 19,21 | 19,22 | 23,25 | 24,26 |
| N | 14 | 15 | 16 | 17 | 18 | 19,20 | 19,21 | 19,22 | 23,25 | 24,26 |
| Z | 14 | 15 | 16 | 17 | 18 | 19,20 | 19,21 | 19,22 | 23,25 | 24,26 |

## NOTES FOR CONVERSION RULES, TABLES G-1, G-2, G-3

1. Applies Bm to Bn. The leftmost [min,(m,n)] bits are copied from the source field to the destination field. If m>n, the rightmost (m-n) bits of the source field are ignored. If m<n, the rightmost (n-m) bits of the destination field are set to zero.

   BmB0 can be used to skip m bits of the source record. B0Bn can be used to zero an n-bit field in the destination area.

2. Applies Bm to Xn. The source field is copied to the destination field one bit at a time from the left. Convert each zero bit to the character 0, and each one bit to the character 1. If m>n, the rightmost (m-n) bits of the source field are ignored. If m<n, the rightmost (n-m) characters of the destination field are set to 0.

   B0Xn can be used to set an n-character field in the destination area to all 0s.

3. Applies Bm to Xn numeric. The source field is treated as an m-bit positive binary integer.

   A B0 source field is treated as zero.

4. Applies when converting to I. The source field is rounded (if necessary) to an integer, and the low-order 59 bits are the value. If the magnitude is $> 2^{59}$ (i.e., more than 59 bits required), an error condition is flagged.

5. Applies when converting to U. The source field is rounded (if necessary) to an integer, and the high-order 48 bits are the value. The result is kept as a single-precision floating-point number. However, this number is denormalized (COBOL COMP-1 definition), if necessary, to keep the biased exponent $\geq 2000_8$.

6. Applies when converting to E. The source field is rounded to 48-bit precision and the result is kept as a single-precision floating-point number.

7. Applies when converting to D. The source field is rounded to 96-bit precision and the result is kept as a double-precision floating-point number.

8. Applies when converting to Sn. The source field is rounded (if necessary) to an integer. If the magnitude is $>10^n$, high-order truncation occurs. The value is converted to a display code string of decimal digits, with leading zeros if necessary. The sign of the number is indicated by amending the low-order (units) digit as shown in figure G-1. The 11-0 and 12-0 card punch is not supported on NOS in a manner that translates as signed numeric overpunch.

| | |
|---|---|
| 0→< | (corresponds to 12-0 card punch) |
| Positive: | |
| 1-9→A-J | (corresponds to 12-1→12-9 card punch) |
| 0→v | (corresponds to 11-0 card punch) |
| Negative: | |
| 1-9→J-R | (corresponds to 11-1→11-9 card punch) |

Figure G-1. Amending of Low-Order Digits

9. Applies when converting to Nn. The source field is rounded (if necessary) to an integer. If the magnitude is $\geq 10^n$, an error is flagged. The value is converted to a display code string of decimal digits, with leading zeros if necessary. The sign of the field is lost (magnitude only saved).

10. Applies when converting to Zn. The source field is rounded ( if necessary) to an integer. If the magnitude is $\geq 10^n$, an error is flagged. The value is converted to a display code string of decimal digits. Leading zeros are suppressed and replaced by blanks. If the number is negative, a - replaces the rightmost blank. If the number is negative and no blanks are in the field, an error is flagged.

11. Applies Xm to Bn. The source field is copied to the destination field one character at a time from the left. Each 0 is converted to a single zero bit, and each 1 is converted to a single one bit. If any character besides 0 or 1 is encountered, an error is flagged. If m>n, the rightmost (m-n) characters of the source are ignored. If m<n, the rightmost (n-m) bits of the destination are set to zero.

12. Applies (string)m to (string)n. The source field is copied to the destination field from the left, with conversion according to appendixes A, B, and C if necessary. If the source field corresponds to card (Hollerith) input, and if the card punches for any character position are invalid, the eight-ones character (hexadecimal FF) is used for that position. If m>n, the rightmost (m-n) characters of the source string are ignored. If m<n, the right-most (n-m) characters of the destination are set to blanks.

    Using an n value of 0, m characters of the source can be skipped. Using an m value of 0 allows setting a destination field to all spaces.

13. Applies (string) to numeric type. A source character string which is to be converted to numeric type must have one of the following forms:

    n n.n .n n. n.nE$\pm$s .nE$\pm$s n.E$\pm$s nE$\pm$s

    where:

    | | |
    |---|---|
    | n | is a coefficient of $\leq 15$ decimal digits. |
    | s | is the exponent (base 10). |

    Numbers are kept to a precision of at least 96 bits. Spaces are ignored; they can be embedded anywhere within the field. If any other character appears in the field, or if the syntactic form is incorrect, an error is flagged. If the source field width is zero, the value is taken to be zero. If E is present, a decimal point also must be present.

14. Applies numeric to Bn. If necessary, the source field is converted to binary and rounded to integer form. The rightmost n bits, with sign extended, are moved to the destination field. The binary representation is in the form appropriate to the destination: two's complement for IBM format and one's complement for CDC format. If n bits are insufficient to contain the result, the rightmost n bits are placed in the destination field. If the source field width is zero, or the value is infinite or indefinite, the value is taken to be -0 (+0 if the destination is IBM format).

15. Applies numeric to (string)n. The conversion of numeric fields to alphanumeric (string) fields depends upon several factors, including the size of the destination field, magnitude and sign of the source field, and maximum precision of the source item. The receiving field format can be determined by the following subrules (headings refer to the source field format).

    <u>All</u>

    A. If the destination field width is zero, no conversion takes place.

    B. If the source item is indefinite or infinite, the destination field is filled as shown in table G-4.

    C. Determine the maximum precision, p, of the source item from table G-5.

    D. Set a variable, d, to n. If the source value is negative, set d to n-1 (d is the available destination field width).

    E. If the source value is an integer (units bit represented and fractional part equal to zero), follow the rules under Integer; otherwise, follow the rules under Floating-Point.

    <u>Integer</u>

    A. If the magnitude of the value is $\geq 10^d$, follow the rules under Type E.

    B. The value is converted to a decimal integer and placed in the destination field, right-justified. All leading zeros are replaced (except in the units position) by spaces.

    C. If the value is negative and n>1, a - is placed immediately to the left of the leftmost digit. Otherwise (must be -0 in a 1-character field) the 0 is replaced by a -.

TABLE G-4. DESTINATION FIELD FOR INFINITE OR INDEFINITE ITEMS

| Condition | 1 | 2 | 3 | 4 or More (Right-Justified in Field) |
|---|---|---|---|---|
| + | F | NF | INF | INF |
| - | F | -F | -NF | -INF |
| +? | D | ND | IND | IND |
| -? | D | -D | -ND | -IND |

TABLE G-5. MAXIMUM PRECISION OF SOURCE ITEMS

| Source Computer | Source Item Type | p (digits) | Number of Guaranteed Accurate Decimal Digits |
|---|---|---|---|
| IBM | H | 5 | 4 |
| | W | 10 | 9 |
| | G | 19 | 18 |
| | F | 8 | 7 |
| | L | 17 | 16 |
| | E | 34 | 33 |
| | Pm | 2m-1 | 2m-1 |
| | Sm | m | m |
| CDC | I | 18 | 17 |
| | U | 15 | 14 |
| | E | 15 | 14 |
| | D | 29 | 28 |
| | Sm | m | m |
| | Nm | m | m |
| | Zm | m | m |

Floating-Point

A. Determine r, the minimum number of digit positions required to use this representation as follows:

If $|\text{value}| \geq 1$, then r=K,

where $10^{K-1} \leq |\text{value}| < 10^{K}$

If $|\text{value}| < 1$, then r=K-1+min(p,d-5),

where $10^{-K} \leq |\text{value}| < 10^{-K+1}$

B. If $r \leq (d-1)$, proceed to step C; otherwise, follow steps under Type E.

C. The value is converted according to one of the following formats and the result string is placed, right-justified, in the destination field. The value is rounded, if necessary, to the indicated number of places.

(1) If $(x \geq 1)$ and $[(r=d-1) \text{ or } (r \geq p)]$:

d1d2...di. i=r

(2) If $(x \geq 1)$ and $(r<d-1)$ and $(r<p)$:

d1d2...di.di+1...dj i=r, j=min(p,d-1)

(3) If $(x<1)$ and $[(d-1)>(k-1+p)]$:

0.d1d2...dj j=K-1+p

(4) If $(x<1)$ and $[(d-1) \leq (k-1+p)]$:

.d1d2...dj j=d-1

D. If the value is negative, place a - immediately to the left of the leftmost nonblank character.

Type E

A. If $d>6$ and $|\text{value}| \geq .95 \times 10^{-99}$, proceed to step B. Otherwise, the receiving field is not wide enough to represent the value; the destination field is filled with all asterisks. If the value is negative, the leftmost * is replaced by a -.

B. The value is converted according to the following format:

d1.d2...djEeee j=min(p,d-5)

where eee is -99 to -01, +00 to +99, or 100 to 305.

If a negative exponent of less than -99 is required, the following format is used:

d1.d2...djE-nnn j-min(p,d-6)

Similar to FORTRAN scientific notation, the value is rounded to the indicated number of digits and placed, right-justified, in the destination field.

C. If the value is negative, a - is placed immediately to the left of the leftmost nonblank character.

16. Applies when converting to H. The source value is rounded, if necessary, to a two's complement integer, and the low-order 16 bits are taken as the value. If significance is lost, an error is flagged.

17. Applies when converting to W. The source value is rounded, if necessary, to a two's complement integer, and the low-order 32 bits are taken as the value. If significance is lost, an error is flagged.

18. Applies when converting to G. The source value is rounded, if necessary, to a two's complement integer, and the low-order 64 bits are taken as the value. If significance is lost, an error is flagged.

19. Applies when converting to F, L, E. The source value is converted to an IBM format floating-point number, rounded to the indicated number of bits. If the source magnitude is too large ($> \sim 5 \times 10^{75}$), the largest possible number is supplied. If the source magnitude is too small ($< \sim 5 \times 10^{-75}$) but not zero, the smallest nonzero number is supplied.

20. Applies when converting to F. The resultant value is a 4-byte field of 21- to 24-bit precision (IBM short floating-point).

21. Applies when converting to L. The resultant value is an 8-byte field of 53- to 56-bit precision (IBM long floating-point).

22. Applies when converting to E. The resultant value is a 16-byte field of 110- to 112-bit precision (IBM extended-precision floating-point). The low-order 14 to 16 bits might not be accurate, since only 96-bit precision is guaranteed.

23. Applies when converting to Pm. The source value is rounded, if necessary, to an integer and converted to packed decimal form. If $|\text{value}| \geq 10^{2m-1}$, overflow has occurred and an error is flagged.

24. Applies when converting to Sm. The source value is rounded, if necessary, to an integer and converted to a decimal string. If $|\text{value}| \geq 10^{m}$, overflow has occurred and an error is flagged.

25. For P fields, the sign of the field and low-order (units) numeric place are contained in the low-order (rightmost) byte as shown in figure G-2.

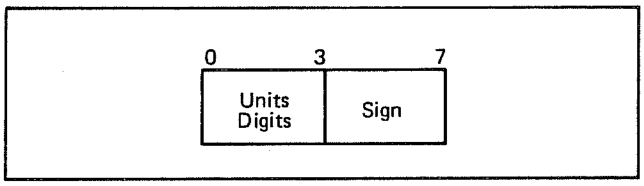


Figure G-2. Location of Sign and Numeric Place for P Fields

Valid sign codes used when a P field is read are as follows:

| Bit Pattern | Sign |
|---|---|
| 1010 | + |
| 1011 | - |
| 1100 | + |
| 1101 | - |
| 1110 | + |
| 1111 | + |

The EBCDIC sign code generated is 1101 for a negative sign and 1100 for a positive sign.

26. For S fields in EBCDIC data, the sign of the field and low-order (units) numeric places are contained in the low-order (rightmost) byte as shown in table G-6.

TABLE G-6. SIGN POSITION FOR EBCDIC FIELDS

| Low-Order Digit | Character Placed in That Position + (sign) | - |
|---|---|---|
| 0 | { | } |
| 1 | A | J |
| 2 | B | K |
| 3 | C | L |
| 4 | D | M |
| 5 | E | N |
| 6 | F | O |
| 7 | G | P |
| 8 | H | Q |
| 9 | I | R |

For S fields in ASCII data, the sign of the field and low-order (units) numeric digits are contained in the low-order byte as shown in table G-7.

TABLE G-7. SIGN POSITION FOR ASCII FIELDS

| Low-Order Digit | Character Placed in That Position + (sign) | - |
|---|---|---|
| 0 | @ | P |
| 1 | A | Q |
| 2 | B | R |
| 3 | C | S |
| 4 | D | T |
| 5 | E | U |
| 6 | F | V |
| 7 | G | W |
| 8 | H | X |
| 9 | I | Y |

## HEXADECIMAL-OCTAL CONVERSION

TABLE G-8. HEXADECIMAL-OCTAL CONVERSION TABLE

| | First Hexadecimal Digit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| Second Hexadecimal Digit 0 | 000 | 020 | 040 | 060 | 100 | 120 | 140 | 160 | 200 | 220 | 240 | 260 | 300 | 320 | 340 | 360 |
| 1 | 001 | 021 | 041 | 061 | 101 | 121 | 141 | 161 | 201 | 221 | 241 | 261 | 301 | 321 | 341 | 361 |
| 2 | 002 | 022 | 042 | 062 | 102 | 122 | 142 | 162 | 202 | 222 | 242 | 262 | 302 | 322 | 342 | 362 |
| 3 | 003 | 023 | 043 | 063 | 103 | 123 | 143 | 163 | 203 | 223 | 243 | 263 | 303 | 323 | 343 | 363 |
| 4 | 004 | 024 | 044 | 064 | 104 | 124 | 144 | 164 | 204 | 224 | 244 | 264 | 304 | 324 | 344 | 364 |
| 5 | 005 | 025 | 045 | 065 | 105 | 125 | 145 | 165 | 205 | 225 | 245 | 265 | 305 | 325 | 345 | 365 |
| 6 | 006 | 026 | 046 | 066 | 106 | 126 | 146 | 166 | 206 | 226 | 246 | 266 | 306 | 326 | 346 | 366 |
| 7 | 007 | 027 | 047 | 067 | 107 | 127 | 147 | 167 | 207 | 227 | 247 | 267 | 307 | 327 | 347 | 367 |
| 8 | 010 | 030 | 050 | 070 | 110 | 130 | 150 | 170 | 210 | 230 | 250 | 270 | 310 | 330 | 350 | 370 |
| 9 | 011 | 031 | 051 | 071 | 111 | 131 | 151 | 171 | 211 | 231 | 251 | 271 | 311 | 331 | 351 | 371 |
| A | 012 | 032 | 052 | 072 | 112 | 132 | 152 | 172 | 212 | 232 | 252 | 272 | 312 | 332 | 352 | 372 |
| B | 013 | 033 | 053 | 073 | 113 | 133 | 153 | 173 | 213 | 233 | 253 | 273 | 313 | 333 | 353 | 373 |
| C | 014 | 034 | 054 | 074 | 114 | 134 | 154 | 174 | 214 | 234 | 254 | 274 | 314 | 334 | 354 | 374 |
| D | 015 | 035 | 055 | 075 | 115 | 135 | 155 | 175 | 215 | 235 | 255 | 275 | 315 | 335 | 355 | 375 |
| E | 016 | 036 | 056 | 076 | 116 | 136 | 156 | 176 | 216 | 236 | 256 | 276 | 316 | 336 | 356 | 376 |
| F | 017 | 037 | 057 | 077 | 117 | 137 | 157 | 177 | 217 | 237 | 257 | 277 | 317 | 337 | 357 | 377 |
| Octal | 000 – 037 | | 040 – 077 | | 100 – 137 | | 140 – 177 | | 200 – 237 | | 240 – 277 | | 300 – 337 | | 340 – 377 | |

# SUMMARY OF KEYWORDS AND DIRECTIVES H

The user communicates with FORM by indicating the desired functions and associated options. Each function is specified by a directive. The INP, OUT, SEQ, PAG, and XEQ directives have the following general format:

aaa(lfn,p1,p2,p3,...,pn)

The CON, QAL, and REF directives have the following general format:

aaa(lfn,string)

The FORM directives are summarized in table H-1.

Options and associated values are described in table H-2.

Conversion strings, qualification strings, and reformat strings are described in figures H-1, H-2, and H-3.

TABLE H-1. SUMMARY OF DIRECTIVES

| Directive Mnemonic | Parameters |
|---|---|
| INP( | lfn, MAX=n, REW=r, HRL=ept, DCA=ept, LX=ept, RFM=f, BLK=n, LRL=n, COD=c, CX=ept, EX=ept, DX=ept, POS=±n, IRL=n) |
| OUT( | lfn,† MAX=n, REW=r, HRL=ept, CPA=ept, LX=ept, RFM=f, BLK=n, LRL=n, COD=c, CX=ept, EX=ept, DCT=ept, BGD=g, KEY=±d, RX=ept, IRL=n) |
| CON( | lfn, conversion-string) |
| QAL( | lfn, qualification-string) |
| REF( | lfn, reformat-string) |
| SEQ( | lfn, NBR=d, BET=n, ADD=n) |
| PAG( | lfn, FMT=a, PGL=n, TOP=n, TTL=l) |
| XEQ( | lfn, IX=ept, FEX=ept) |

Legend:

| | |
|---|---|
| a | 1 = single space, 2 = double space, A = use 1st character carriage control, D = dump format |
| c | A = ASCII, C = EBCDIC, E = EBCDIC |
| d | item descriptor (see section 2) |
| f | F = fixed, V = variable, U = unspecified, FB = fixed blocked, VB = variable blocked, VSB = variable spanned blocked |
| g | X = blank, Z = display code zero, B = binary zero, C = copy input record |
| l | literal ($....$ or *....*) |
| n | decimal integer |
| r | R = rewind, N = no rewind, U = rewind and return (disk) or unload (tape) |
| ept | entry point name |
| lfn | logical file name |

†logical file name lfn/R = rewind before use.

TABLE H-2. SUMMARY OF KEYWORDS AND DESCRIPTORS

| Format | Associated Directives | Keyword Meaning | Values | Default Value |
|---|---|---|---|---|
| ADD = n | SEQ | Increment BEG | 1 through $2^{48}$-1 | 1 |
| BEG = n | SEQ | Initial value of sequence number | 0 through $2^{48}$-1 | 1 |
| BGD = $\left\{\begin{matrix}X\\Z\\B\\C\end{matrix}\right\}$ | OUT | Output record background | X (blank)<br>Z (display code zero)<br>B (binary zero)<br>C (copy input record) | C |
| BLK = n | INP, OUT (with CON) | Block size in 8-bit bytes | 1 through 32760 | None |
| COD = $\left\{\begin{matrix}A\\C\\E\end{matrix}\right\}$ | OUT | IBM character code | A (ASCII)<br>C (EBCDIC)<br>E (EBCDIC) | E |
| CPA = ept | OUT | Record compression/ encryption exit | Applicable entry point name | None |
| CX = ept | INP, OUT | Conversion error exit | Applicable entry point name | None |
| DCA = ept | INP | Record decompression decryption exit | Applicable entry point name | None |
| DCT = ept | OUT | Display-collating conversion table | Applicable entry point name | None |
| DX = ept | INP | End-of-partition, end-of-section, end-of-data exit | Applicable entry point name | None |
| EX = ept | INP, OUT | Error exit | Applicable entry point name | None |
| FEX = ept | XEQ | Execution error exit | Applicable entry point name | None |
| FMT = a | PAG | Print format | 1 (single space)<br>2 (double space)<br>A (1st character carriage control)<br>D (dump format) | 1 |
| HRL = ept | INP | Key hashing exit | Applicable entry point name | None |
| IRL | INP, OUT (with CON) | Internal record length | 1 through BLK | External record length (LRL) for variable length records $\left[\frac{4*LRL}{3}\right]$ |

TABLE H-2. SUMMARY OF KEYWORDS AND DESCRIPTORS (Contd)

| Format | Associated Directives | Keyword Meaning | Values | Default Value |
|---|---|---|---|---|
| iTm | REF, QAL, CON | Describes a data field: | | |
| | | i starting character position | 1 through record size | None |
| | | T data type | X,I,E,D,U,B,S,N,Z (CDC) B,X,I,U,E,D,S,N,Z (IBM) | None |
| | | m field length | Number of characters or bits; omit if fixed-length | 1 |
| i/w | QAL, REF, CON | Initial bit position: | | |
| | | i initial 6- or 8-bit byte | 1 through 999 | None |
| | | w initial bit within byte | 1 through 6 or 8 | 1 |
| IX = ept | XEQ | Input record exit | Applicable entry point name | None |
| KEY = ±iTm | OUT | Actual key or indexed sequential key field description: | | |
| | | i starting character position | 1 through record size | None |
| | | T data type | X, F, I | None |
| | | m field length | Number of characters | 1 |
| | | + include key in record | | |
| | | - omit key from record | | |
| LRL = n | INP, OUT (with CON) | Record length (8-bit bytes) | 1 through BLK | None |
| LX = ept | INP, OUT | Nonstandard label exit | Applicable entry point name | None |
| MAX = n | INP, OUT | Max number of records | 1 through 16777215 | All records in file |
| NBR = iTm | SEQ | Sequence field description: | | |
| | | i starting character or bit position | 1 through record size | None |
| | | T data type | X or B | None |
| | | m field length | 1 through 10 (characters) 1 through 60 (bit) | 1 |

TABLE H-2. SUMMARY OF KEYWORDS AND DESCRIPTORS (Contd)

| Format | Associated Directives | Keyword Meaning | Values | Default Value |
|---|---|---|---|---|
| oln | REF, QAL, CON | Describes search criteria for variable length data fields in input record: | | |
| | | o character position where search is to begin | Max is record length | None |
| | | l delimiting literal for which a match is sought | Character string not to exceed 255 characters | None |
| | | n ordinal of the literal delimiter | $\pm$1 through $\pm$record length | None |
| PGL = n | PAG | Page line limit | 1 through 60 | 60 if FMT=1 specified 30 if FMT=2 specified |
| POS = $\left\{\begin{matrix}+n\\-n\end{matrix}\right\}$ | INP | Initial position partition skip count | $\pm$1 through $\pm$2047 | No skipping |
| REW = $\left\{\begin{matrix}R\\U\\N\end{matrix}\right\}$ | INP, OUT | Rewind option | R (rewind)<br>U (unload)<br>N (no rewind) | N |
| RFM = $\left\{\begin{matrix}F\\FB\\U\\V\\VB\\VSB\end{matrix}\right\}$ | INP, OUT (with CON) | IBM record blocking | F (fixed)<br>FB (fixed blocked)<br>U (unspecified)<br>V (variable)<br>VB (variable blocked)<br>VSB (variable spanned blocked) | F |
| RX = ept | OUT | Reformatting error exit | Applicable entry point name | None |
| TOP = n | PAG | Lines in top margin of page | 2 through 60 | 2 |
| TTL = lit | PAG | Contents of page title | Character string not to exceed 115 characters | Blank |

```
conversion string:

     conversion-specification [; conversion-specification] . . .

conversion-specification:

     [selector-expression:] conversion-item [, conversion-item] . . .

conversion-item:

     [repeat-count]  { simple-item-conversion }
                     { (conversion-string)    }

simple-item-conversion:
   {Tm1[Tm2]}

repeat-count:

     decimal integer indicating the number of times the conversion item is to be executed.
```

Figure H-1. Conversion String Format

```
qualification string:

       [NOT]  { selector-expression  } [ { AND }  [NOT] { selector-expression  } ] . . .
              { qualification-string } [ { OR  }        { qualification-string } ]
```

Figure H-2. Qualification String Format

```
reformat string:
  reformat-specification [;reformat-specification] . . .

reformat-specification:
  [selector-expression:] reformat-item [, reformat-item] . . .

reformat-item:
 {                 simple-reformat        }
 { repeat-count  { (simple-reformat) }    }
 {               { (reformat-string) }    }

simple-reformat:
 { iTm=iTm     }
 { iTm=literal }
 { iTm=KEY     }
 { iTm=KEYA    }
 { iTm         }

repeat-count:
  decimal integer specifying the number of times the refor-
  mat item is to be repeated.
```

Figure H-3. Reformat String Format

# MAINTAINING 8-BIT SIGNIFICANCE

Maintaining 8-bit significance in data converted from IBM 360/370 tape files is necessary when such files contain character codes not included in the CDC 64-character graphic set. Lowercase characters and several special characters, such as @ ? ! and # are included in the IBM EBCDIC and ASCII character sets but are not in the CDC 64-character set. These characters can be processed by FORM, but such characters cannot be printed on CDC equipment. The user must decide the necessity of maintaining special character codes. In many cases, the uppercase equivalent will suffice. Appendix A contains standard CDC character sets.

In a number of cases, 8-bit significance need not be maintained. Files containing only characters that appear in the CDC 64-character set will convert to CDC 6-bit display codes. Files containing packed decimal data, in which each digit occupies four bits, will convert to CDC 6-bit numeric display fields. IBM files containing binary arithmetic data can be converted to CDC binary arithmetic or display numeric data per user specification. Accuracy is maintained for IBM data that does not exceed CDC double-precision format. When double-precision significance is exceeded, accuracy can be maintained by using bit image conversion; however, user routines must be provided to process such data.

# FORM/8-BIT SUBROUTINE COMPARISON J

FORM utilizes the 8-bit subroutines to perform IBM/CDC conversions; hence, FORM has most of the capabilities of the 8-bit subroutines, including the capability of maintaining 8-bit significance in converted data. FORM handles the same character sets as the 8-bit subroutines. Differences include management of print files, ease of usage, and minor functional differences.

FORM has the same conversion capabilities as the 8-bit subroutines, including identical conversion string syntax. However, conversion strings can be changed from record to record when using 8-bit subroutines, whereas FORM record specifications are set once per output file and can be varied from record to record only by comparing a record field to a literal quantity. FORM has an automatic data reformatting capability (REF directive) that allows the user to reorder data fields and insert literals; in the 8-bit subroutines, data fields in a record are only reformatted sequentially. FORM has a record selection capability (QAL directive) that allows the user to select certain records for inclusion in an output file; when the 8-bit subroutines process a file, all input records are included in the output file. FORM is primarily file oriented and can be used to process random files; the 8-bit subroutines are primarily record oriented and their use is restricted to sequential files.

Print files are handled differently by FORM than by the 8-bit subroutines. FORM allows automatic organization of print file functions, such as paging, line spacing, and titling, through the PAG directive. Also, FORM print files can contain only CDC display code characters. Print files processed by the 8-bit subroutines must be organized by the user, but any characters in the 95 graphic ASCII set can be used. A utility program, COPY8P, is provided in the 8-bit subroutines to print automatically IBM print files using the CDC 595-6 Print Train.

Other minor functional differences between FORM and the 8-bit subroutines include the following:

- FORM provides automatic sequencing of records through the SEQ directive; the 8-bit subroutines have no such provision.
- The 8-bit subroutines process IBM card files as free-form binary input and output; FORM does not.

# FORM HANDLING OF VARIABLE LENGTH RECORDS

Files containing variable length records consist of records with differing record lengths. Figure K-1 shows the data formats and some sample records for EMPFIL1, a file containing variable length records. The longest records (type A) in the file contain the following information:

| | |
|---|---|
| Social security number | (9 characters) |
| Employee name | (27 characters) |
| Employee number | (6 characters) |
| Yearly wage (if salaried) | (6 characters) |
| Hourly rate (if nonsalaried) | (5 characters) |
| Sex | (1 character) |
| Date of birth | (6 characters) |
| Division number | (4 characters) |
| Job description | (21 characters) |
| Managerial position (if any) | (18 characters) |

The shortest records (type B) in EMPFIL1 contain all of the preceding information excluding the 18-character field that describes the managerial position of the employee. This field is absent if the employee does not hold a managerial position.

When handling variable length records, FORM keeps track of the input and output record lengths. Determination of input record length is dependent on the source of the input record. Determination of output record length is dependent on the occurrence of a reformat in the record and the type of output record written by FORM.

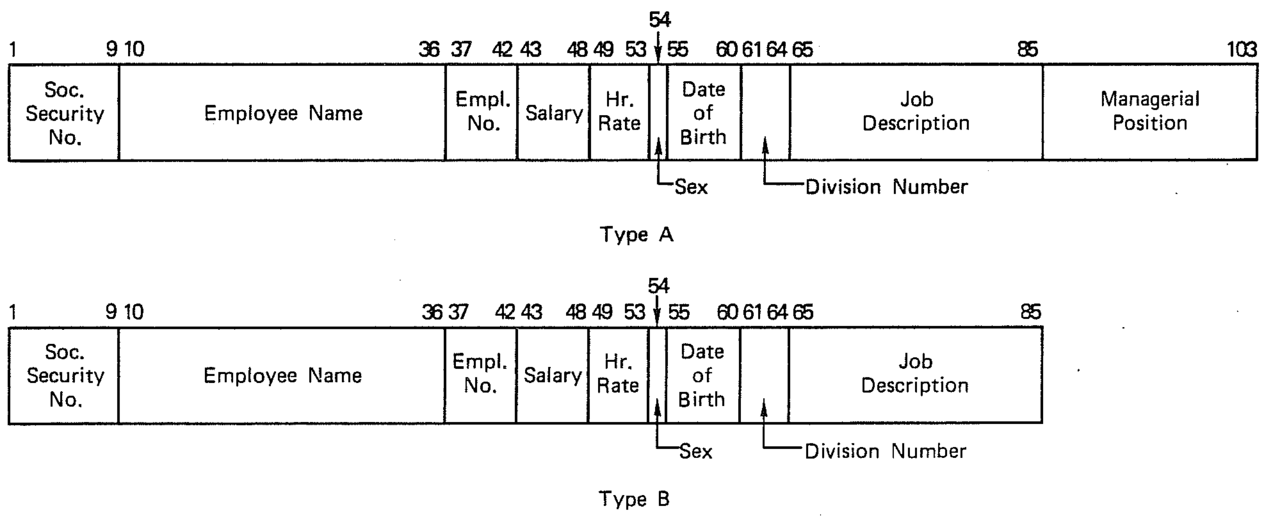
## DETERMINATION OF INPUT RECORD LENGTH

FORM receives input from three sources:

- CYBER Record Manager (CRM) files
- IX exit return parameters
- IBM tape format files

### CYBER RECORD MANAGER FILES

If the input file is a CRM file, the input record length is taken directly from the file information table (FIT) after each read. The record length must be less than or equal to the maximum record length (MRL).



```
934627900JOHNSON, P. T.           9Q7734 35000     F0401305320ENGINEER,          MANAGER
572396245ABERCROMBIE, M. L .      6Q4436 27000     M0128445321PROGRAMMER
562325798STANFIELD, R. J.         7Q4993       7.50F0906495913SECRETARY
439784231SHRADER, F. G.           3Q4453 40000     M0909326312PROD ENGINEER,     HEAD ENGINEER
693276941MILLER, S. P.            4Q7769 37950     F0621514392RESEARCH SCIENTIST
673942183CHRISTMANO, P. A.        5Q3378 45000     F1020396118STRUCTURAL ENGINEER, HEAD MANAGER
572394671CHANDLER, A. P.          3Q4770       8.50M0719547391MACHINE OPERATOR
```

Figure K-1. Data Format and Sample Records From Empfil1

## IX FILES

If input is provided via the IX exit, the values for the address and length of the input record must be assigned within the function and passed to FORM. The input record length is taken from the IX return parameters. The maximum possible record length must be declared by the internal record length (IRL) parameter in the INP directive.

## IBM FORMAT FILES

If input is an IBM format file to be converted by the CON directive, the input record length is dependent on the tape record length as well as on the conversions desired. To provide an area large enough for the conversion, the IRL parameter should be specified in the INP directive. (If IRL is not specified, IRL=(4*LRL)/3 rounded up is used.)

IRL specifies the maximum record length for IBM tape files in 6-bit bytes, and functions like MRL for CRM files. When IBM tape is converted by the CON directive, FORM keeps track of the converted record size. This size, as long as it is less than or equal to IRL, is used as the length of the input record. If the converted record size is larger than IRL, IRL is used for input record length.

If the IBM tape file record type is V, VB, VS, or VSB, the tape record length can be shorter than the longest logical record (LRL). If the tape length is shorter than LRL, conversions are terminated when the end of the input record is reached. No source field in a conversion item can extend beyond the end-of-record, with the following exception:

> An X or B field can extend beyond the end-of-record if the field is being reformatted to consist of X or B items. The source item m value is reduced to the remaining record size and the destination m value is unchanged.

## EFFECT OF REFORMATTING ON OUTPUT RECORD LENGTH

When a reformat is performed with the parameter BGD equal to C, the output record length is initialized to either the input record length or the MRL value, whichever is smaller, for the particular file. If any other BGD is specified, the output record length is initialized to zero.

During the reformat, the output record length is updated whenever a reformat item places data beyond the current output record length value. The updated output record length reflects the last character written to the output record being built.

Assume the following REF directive is used:

```
REF(EMPFIL3,100(x=x))
```

FORM examines each character and determines whether the character is valid data. Those characters that lie within the input record length are valid data. Assuming the actual record length of each input record is unknown, a program can be written to determine the length of each record. The FMT parameter of the PAG directive can be used to print out the record length of each record. MRL in the FILE control statement can be set to the longest possible record length of any record in the file. Figure K-2 shows the job deck for such a program. Figure K-3 shows EMPFIL3, the output resulting from a reformat of EMPFIL1.

```
NOS:

JOB statement
USER statement
CHARGE statement
GET,EMPFIL1.
FILE(EMPFIL1,BT=C,RT=Z,MRL=200)
FILE(EMPFIL3,BT=C,RT=Z,MRL=200)
FORM.
SAVE,EMPFIL3.
REWIND,EMPFIL3.
COPYSBF,EMPFIL3,OUTPUT.
7/8/9
INP(EMPFIL1)
OUT(EMPFIL3,BGD=X)
REF(EMPFIL3,100(X=X))
PAG(EMPFIL3,FMT=D)
6/7/8/9

NOS/BE:

JOB statement
ACCOUNT statement
ATTACH,EMPFIL1.
REQUEST,EMPFIL3,*PF.
FILE(EMPFIL1,BT=C,RT=Z,MRL=200)
FILE(EMPFIL3,BT=C,RT=Z,MRL=200)
FORM.
CATALOG,EMPFIL3,ID=MYID.
REWIND,EMPFIL3.
COPYSBF,EMPFIL3,OUTPUT.
7/8/9
INP(EMPFIL1)
OUT(EMPFIL3,BGD=X)
REF(EMPFIL3,100(X=X))
PAG(EMPFIL3,FMT=D)
6/7/8/9
```

Figure K-2. Example of a Reformat Where BGD=X

If a reformat item specifies a source field outside of input record length (but within MRL or IRL) the output record length is not updated and the input field is treated as though it were zero filled or blank filled according to the default value shown in table K-1.

A new file, EMPFIL4, is to be created. Figure K-4 shows the data format of type B records for both EMPFIL1 and EMPFIL4. After the reformat, EMPFIL4 contains the following information:

| | |
|---|---|
| Employee number | (6 characters) |
| Employee name | (26 characters) |
| Social security number | (9 characters) |
| Managerial position (for A type records) Blank field (for B type records) | (18 characters) |
| Blank field | (3 characters) |
| Sex | (1 character) |
| Blank field | (3 characters) |
| Job description | (20 characters) |

```
RECORD 1                 92 CHARS
93462790OJOHNSON, P. T.                 9Q7734 35000      F0401305320ENGINEER,            MANAGER
RECORD 2                 74 CHARS
572396245ABERCROMBIE, M. L .            6Q4436 27000      M0128445321PROGRAMMER
RECORD 3                 73 CHARS
562325798STANFIELD, R. J.               7Q4993        7.50F0906495913SECRETARY
RECORD 4                 98 CHARS
439784231SHRADER, F. G.                 3Q4453 40000      M0909326312PROD ENGINEER,       HEAD ENGINEER
RECORD 5                 82 CHARS
693276941MILLER, S. P.                  4Q7769 37950      F0621514392RESEARCH SCIENTIST
RECORD 6                 97 CHARS
673942183CHRISTMANO, P. A.              5Q3378 45000      F1020396118STRUCTURAL ENGINEER, HEAD MANAGER
RECORD 7                 82 CHARS
572394671CHANDLER, A. P.                3Q4770        8.50M0719547391MACHINE OPERATOR
```

Figure K-3. Example of Job Decks for Reformatting Where BGD=X

TABLE K-1. DEFAULT, PADDING AND ERROR MESSAGES FOR ALL DATA TYPES

| Data Type | Default | Padding of Error Message |
|---|---|---|
| B | Binary zero | Binary zero |
| X | Blanks | Blanks |
| H | Binary zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| W | Binary zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| G | Binary zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| F | Binary zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| L | Binary zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| E | Binary zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| P | Positive zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| S | Character zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| I | Binary zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| U | Binary zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| D | Binary zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| N | Display coded zero | FIELD EXTENDS BEYOND END-OF-RECORD |
| Z | Display coded zero | FIELD EXTENDS BEYOND END-OF-RECORD |

The job decks for both the NOS and NOS/BE operating systems are shown in figure K-5. Figure K-6 shows the FORM directives, EMPFIL4, and the dayfile for the reformat.

Source items extending beyond the input record length are invalid unless the source and destination items are each an X or B type. If the source and destination items are not X or B types, FORM terminates with an error message indicating a reference beyond the end-of-record as shown in table K-1. If the source and destination items are an X or B type, the source item length is reduced to the remaining input record length, the destination item length is unchanged, and output record length is updated.

## DETERMINATION OF OUTPUT RECORD LENGTH

If no reformat is performed, the output record length (which is determined for each record in each output file) is set to the value of the input record length or the value of the MRL (whichever is smaller) for the particular file. FORM output files are either CRM files or IBM format files.

For CRM files, output record length is the record length given to CRM when the record is written by FORM. File structure must be adequately described through a FILE control statement so that the output record length determined by FORM is compatible with the CRM record type for the particular file.

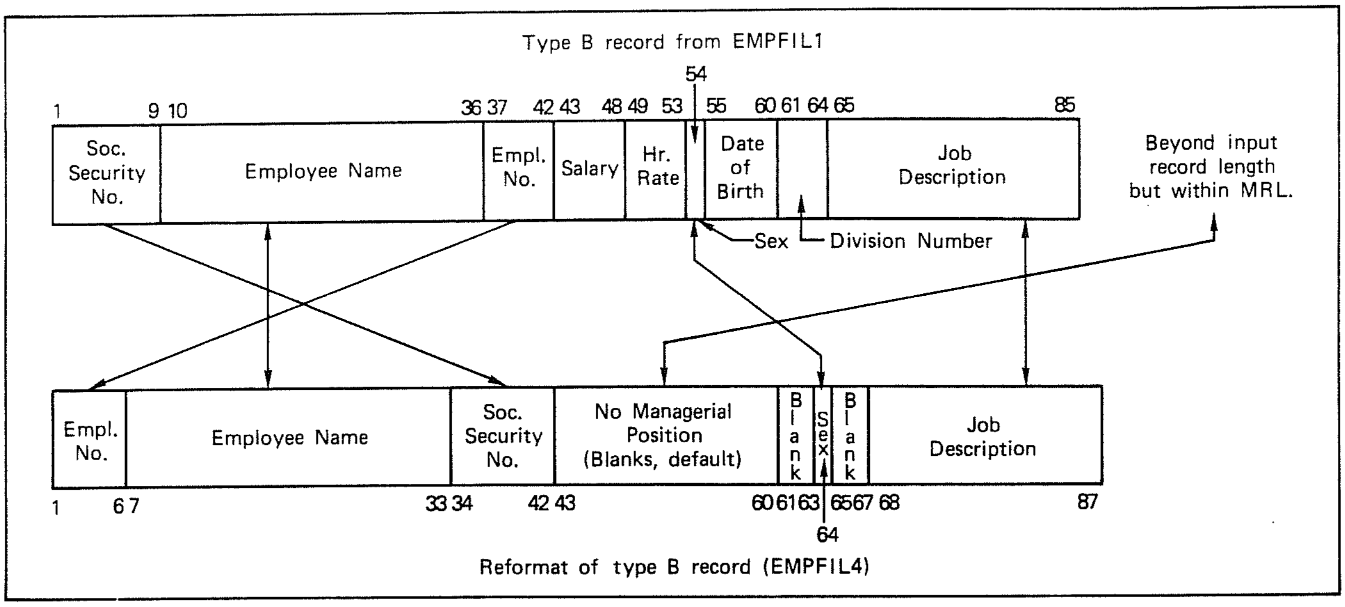


Figure K-4. Data Formats of Type B Records in EMPFIL1 and EMPFIL4

```
NOS:


JOB statement
USER statement
CHARGE statement
GET,EMPFIL1.
FILE(EMPFIL1,BT=C,RT=Z,MRL=120)
FILE(EMPFIL4,BT=C,RT=Z,MRL=120)
FORM.
REWIND,EMPFIL4.
COPYSBF,EMPFIL4,OUTPUT.
7/8/9
INP(EMPFIL1)
OUT(EMPFIL4,BGD=X)
REF(EMPFIL4,X6=37X6,X27=10X27,X9=1X9,X18=86X18,
    X3=$   $,X1=54X1,X3=$   $,X20=65X20)
PAG(EMPFIL4,FMT=D)
6/7/8/9

NOS/BE:


JOB statement
ACCOUNT statement
ATTACH,EMPFIL1,ID=MYID.
FILE(EMPFIL1,BT=C,RT=Z,MRL=120)
FILE(EMPFIL4,BT=C,RT=Z,MRL=120)
FORM.
REWIND,EMPFIL4.
COPYSBF,EMPFIL4,OUTPUT.
7/8/9
INP(EMPFIL1)
OUT(EMPFIL4,BGD=X)
REF(EMPFIL4,X6=37X6,X27=10X27,X9=1X9,X18=86X18,
    X3=$   $,X1=54X1,X3=$   $,X20=65X20)
PAG(EMPFIL4,FMT=D)
6/7/8/9
```

Figure K-5. Example of Job Decks for Reformatting

```
INP(EMPFIL1)
OUT(EMPFIL4,BGD=X)
REF(EMPFIL4,X6=37X6,X27=10X27,X9=1X9,X18=86X18,
    X3=$   $,X1=54X1,X3=$   $,X20=65X20)
PAG(EMPFIL4,FMT=D)



SUMMARY
 INPUT FILE - EMPFIL1          7 RECORDS READ
OUTPUT FILE - EMPFIL4          7 RECORDS WRITTEN
END OF RUN.


1
```

```
ORECORD 1               87 CHARS
 9Q7734JOHNSON, P. T.                934627900MANAGER            F    ENGINEER,
ORECORD 2               87 CHARS
 6Q4436ABERCROMBIE, M. L .           572396245                   M    PROGRAMMER
ORECORD 3               87 CHARS
 7Q4993STANFIELD, R. J.              562325798                   F    SECRETARY
ORECORD 4               87 CHARS
 3Q4453SHRADER, F. G.                439784231HEAD ENGINEER      M    PROD ENGINEER,
ORECORD 5               87 CHARS
 4Q7769MILLER, S. P.                 693276941                   F    RESEARCH SCIENTIST
ORECORD 6               87 CHARS
 5Q3378CHRISTMANO, P. A.             673942183HEAD MANAGER       F    STRUCTURAL ENGINEER,
ORECORD 7               87 CHARS
 3Q4770CHANDLER, A. P.               572394671                   M    MACHINE OPERATOR
```

```
 ACMABVI. 80/04/23.(22) SVL SN112 NOS


13.26.21.EMP4.
13.26.21.UCCR, 7631,       0.014KCDS.
13.26.21.USER,CDSXXXX,.
13.26.21.CHARGE,5912,693XXXX.
13.26.21.GET,EMPFIL1.
13.26.22.FILE(EMPFIL1,BT=C,RT=Z,MRL=120)
13.26.22.FILE(EMPFIL4,BT=C,RT=Z,MRL=120)
13.26.22.FORM.
13.26.25.REWIND,EMPFIL4.
13.26.25.COPYSBF,EMPFIL4,OUTPUT.
13.26.25. EOI ENCOUNTERED.
13.26.25.UEAD,      0.002KUNS.
13.26.25.UEPF,      0.010KUNS.
13.26.25.UEMS,      0.749KUNS.
13.26.25.UECP,      0.219SECS.
13.26.25.AESR,      2.426UNTS.
13.44.27.UCLP, 7634,       0.256KLNS.
```

Figure K-6. Example of Reformatting of EMPFIL1 to Form EMPFIL4

For IBM format files, output record length controls the size of the record to be converted by the output CON directive. Conversion stops when the entire output record has been used. No source field in a conversion item can extend beyond output record length, with the following exception:

An X or B field can extend beyond the end-of-record if the field is being reformatted to consist of X or B items. The source item m value is reduced to the remaining record size. The destination m value remains unchanged.

# COMPARISON OF FORM 1.2 AND FORM 1.0

FORM 1.2 is a file copy and conversion utility written to replace FORM 1.0. FORM 1.2 has most of the capabilities of FORM 1.0 to reformat and to convert data fields. FORM 1.2 has the capability of maintaining 8-bit significance in converted data; FORM 1.0 does not have this capability. FORM 1.2 is primarily record oriented; FORM 1.0 is primarily file oriented. FORM 1.2 interfaces with Basic Access Methods 1.5 and Advanced Access Methods 2; FORM 1.0 interfaces with CYBER Record Manager 1.0.

External differences between FORM 1.2 and FORM 1.0 are shown in table L-1.

TABLE L-1. EXTERNAL DIFFERENCES BETWEEN FORM 1.2 AND FORM 1.0

| External Feature | FORM 1.2 | FORM 1.0 |
|---|---|---|
| Directive Order | INP/OUT directives required; must appear before other directives.<br><br>XEQ, XEQ (FIN) directive optional. | INP/OUT directives optional; can be in any order.<br><br>XEQ directive required. |
| File Information Table Default Values | None. | FO=SQ<br>BT=C<br>RT=Z<br>MRL=140 |
| File Formats | Initial and extended format for indexed sequential, actual key, and direct access files. | Initial format for indexed sequential, actual key, and direct access files. |
| Interfaces | CYBER Record Manager Basic Access Methods (1.5), CYBER Record Manager Advanced Access Methods (2).<br><br>Not supported. | CYBER Record Manager (1.0).<br><br>CREATE utility. |
| Output Record | Fixed length records: output record length=MRL.<br><br>Variable length records: see appendix K. | Equates to length of input record.<br><br>Not supported. |
| Owncode | File-error oriented. | Run-oriented. |
| Performance | Small minimum field length (< 20K) required.<br><br>More features available in qualification and reformat directives, but execution time is slower. | Large minimum field length (> 60K) required.<br><br>Fewer features available in qualification and reformat directives, but execution time is faster. |
| Runs Per Call | Single run per call only.<br><br>More than one IBM type file handled per run. | Multiruns per call.<br><br>Only one IBM type file handled per run. |

Functional differences between FORM 1.2 and FORM 1.0 include the following:

- FORM 1.2 uses triple-precision arithmetic in conversions; FORM 1.0 uses single-precision arithmetic in conversions.
- FORM 1.2 does not print a title page; FORM 1.0 prints a title page in block letters.
- FORM 1.2 reads to end-of-information, end-of-record, end-of-section, or end-of-partition; FORM 1.0 reads one partition only.
- FORM 1.2 has no user callable version; FORM 1.0 has a user callable version.

Other functional differences between FORM 1.2 and FORM 1.0 are shown in tables L-2 and L-3.

TABLE L-2. DIFFERENCES IN USAGE OF DIRECTIVE PARAMETERS BETWEEN FORM 1.2 AND FORM 1.0

| Directive | Parameter | FORM 1.2 | FORM 1.0 |
|---|---|---|---|
| INP | BLK | Block size: 32767, required for IBM tape files. | Record format, required for IBM tape files. |
| | CX | Conversion error exit. | Not supported. |
| | DCA | Record decompression/ decryption exit. | Not supported. |
| | DX | End-of-partition, end-of-section, end-of-data exit. | Not supported. |
| | EX | Error exit. | Not supported. |
| | HRL | Key hashing exit. | Not supported. |
| | IDS | Not supported. | Alternative CON directive, required for IBM tape files. |
| | IRL | Record size (6-bit), required for IBM tape files. | Not supported. |
| | LRL | Record size (8-bit), required for IBM tape files. | Not supported. |
| | LX | Label exit. | Not supported. |
| | MAX | Maximum number of records: 16777215. | Maximum number of records: 8388607. |
| | POS | Initial position partition skip count. | Initial position record skip count. |
| | RECFM | Record format, required for IBM tape files. | Not supported. |
| | RX | Reformatting error exit. | Not supported. |
| | SIZ | Not supported. | Block size: 16383, required for IBM tape files. |
| NON | LBL | Not supported. | Label value. |
| | LEN | Not supported. | Label length. |
| | ORD | Not supported. | File skip count. |
| OUT | BLK | Block size: 32767, required for IBM tape files. | Record format, required for IBM tape files. |

TABLE L-2. DIFFERENCES IN USAGE OF DIRECTIVE PARAMETERS BETWEEN FORM 1.2 AND FORM 1.0 (Contd)

| Directive | Parameter | FORM 1.2 | FORM 1.0 |
|---|---|---|---|
| OUT (Contd) | CPA | Record compression/ encryption exit. | Not supported. |
| | CX | Conversion error exit. | Not supported. |
| | DCT | Display-collating conversion table exit. | Not supported. |
| | EX | Error exit. | Not supported. |
| | HRL | Key hashing exit. | Not supported. |
| | IDS | Not supported. | Alternative CON directive, required for IBM tape files. |
| | IRL | Record size (6-bit), required for IBM tape files. | Not supported. |
| | LRL | Record size (8-bit), required for IBM tape files. | Not supported. |
| | LX | Label exit. | Not supported. |
| | MAX | Maximum number of records: 16777215. | Maximum number of records: 8388607. |
| | RECFM | Record format, required for IBM tape files. | Not supported. |
| | SIZ | Not supported. | Block size: 16383, required for IBM tape files. |
| PRT/PAG† | FMT | Print formats 1, 2, D, A supported; format 3 not supported. | Print formats 1, 2, 3, D, A supported. |
| | PGL | Page size. | Number of print lines equals page size/f. |
| | TOP | Margin size. | Title line; margin is n-1 lines. |
| | TTL | Prints title for all FMT values. | Ignores title when FMT=D. |
| REF | KEY, KEYA | Actual key or indexed sequential key field descriptor. | Not supported. |
| SEQ | RDX | Not supported. | Sequence radix: 2, 8, 10 or 16 possible. |
| XEQ | AX | Not supported. | After input record read exit. |
| | DX | Not supported. | End-of-data exit. |
| | EX | Not supported. | Parity error exit. |

TABLE L-2. DIFFERENCES IN USAGE OF DIRECTIVE PARAMETERS BETWEEN FORM 1.2 AND FORM 1.0 (Contd)

| Directive | Parameter | FORM 1.2 | FORM 1.0 |
|---|---|---|---|
| XEQ (Cont) | FEX | Execution error exit. | Not supported. |
| | IX | Input record supply exit. | Input record supply exit. |
| | KX | Not supported. | After record output exit. |
| | LX | Not supported. | Nonstandard label exit. |
| | NX | Not supported. | Unqualified record exit. |
| | QX | Not supported. | Qualified record exit. |

†Directive name PAG applies only to FORM 1.2.

TABLE L-3. DIFFERENCES BETWEEN FORM 1.2 AND FORM 1.0 DIRECTIVE SPECIFICATIONS

| Directive | Specification | FORM 1.2 | FORM 1.0 |
|---|---|---|---|
| CON | Alternatives | Nested alternatives possible. | Single level alternatives only; selected by IDS in INP statement. |
| | Condition Nesting | Seven levels of nesting possible. | Only one level of nesting possible. |
| | Data Types | All type combinations legal. | Many type combinations legal. |
| | String | Maximum length of replacement literals is 80 characters; maximum length of comparison literals is 10 characters. | Maximum length of all literals is 255 characters. |
| QAL | Condition Nesting | Seven levels of nesting possible. | Two levels of nesting possible. |
| | Data Types | All type combinations legal. | Many type combinations illegal. |
| REF | Alternatives | Multilevel alternative reformat specifications possible. | Single level alternative reformat specification possible; selected by IDS in INP directive. |
| | Data Type Combinations | All type combinations legal; allows checking of same type transfers. | Many type combinations illegal; allows no checking of same type transfers. |
| | iTm | Default ml value is 1. | Default ml value is m2 size. |
| | i/w | Character/bit. | Not supported. |
| | w/p | Not supported. | Word/bit. |
| SEQ | Data Types | All types legal. | Only X and B types legal. |

# FORM HANDLING OF ASCII DATA

FORM can be used to reformat ASCII data. NOS 2 and NOS/BE process ASCII data differently. Differences between operating system handling of character sets, codes, input, and output include the following:

- The 6-bit and 12-bit ASCII codes can be used in combination under NOS 2 only.
- The 12-bit ASCII code used by NOS 2 is not the same as the 12-bit ASCII code used by NOS/BE; the former uses 12 bits but the latter uses the 8 rightmost bits of a 12-bit byte.
- 5/7/9 cards are allowed only under NOS 2.

A character set differs from a code set. A character set is a set of graphic and/or control characters. A code set is a set of codes used to represent each character within a character set. Characters exist outside the computer system and communication network; codes are received, stored, retrieved, and transmitted within the computer system and network.

Character codes can be manipulated as coded data or as binary data. Coded data is converted from one code set representation to another as the coded data enters or leaves the computer system. Binary data is not converted. The distinction between coded data and binary data is important when reading or punching cards and when reading or writing magnetic tape. Only coded data can be properly reproduced as characters on a line printer.

## NOS 2

NOS 2 supports the 6/12 ASCII code set. The 6-bit ASCII codes represent the uppercase ASCII character set. The 12-bit character codes represent the ASCII lowercase character set, which includes special symbols and device control characters. The 12-bit codes begin with either $74_8$ or $76_8$, which signal NOS 2 that the next six bits contain ASCII data. The $74_8$ or $76_8$ is followed by a 6-bit code that represents lowercase ASCII data. The 12-bit codes are $7401_8$, $7402_8$, $7404_8$, $7407_8$, and $7601_8$ through $7677_8$. All other 12-bit codes ($74xx_8$ and $7600_8$) are undefined.

The ASCII 8/12 data format consists of the ASCII 7-bit code (as defined by ANSI Standard X3.4-1977) right-justified in a 12-bit byte. Assuming that the bits are numbered from the right starting with 0, bits 0 through 6 contain the ASCII code, bits 7 through 10 are reserved for CDC usage, and bit 11 is used to distinguish the ASCII code from the end-of-line indicator for NUL (ASCII $00_{16}$) only.

## NOS 2 INTERACTIVE FACILITY

When in normal interactive mode (specified by the IAF NORMAL command), the Interactive Facility (IAF) assumes that the ASCII graphic 64-character set is used and translates all input and output to or from display code.

When in ASCII interactive mode (specified by the IAF ASCII command), IAF assumes that the ASCII 128-character set is used and translates all input and output to or from 6/12 display code.

The IAF user can convert a 6/12 code file to an 8/12 ASCII code file that is compatible with NOS/BE by using the NOS FCOPY control statement. The resulting 12-bit ASCII file can be routed to a line printer but cannot be output through IAF. FORM does not recognize ASCII 6/12 data. The FCOPY utility must be used before ASCII 6/12 data is presented to FORM.

Figure M-1 shows DATA12, an ASCII 6/12 file containing records consisting of uppercase and lowercase letters. The FCOPY control statement is used to convert DATA612 to FORMFIL, a file containing ASCII 8/12 data. The job deck for the reformat is shown in figure M-2.

```
J. Brown    ,1422 East St.   ,Charge No. 1111
S. Apple    ,3434 Cherry St.,Charge No. 2211
R. Redi     ,7896 Algo Ave. ,Charge No. 1660
S. Sneade   ,234 Redding Dr.,Charge No. 1429
```

Figure M-1. DATA612, an ASCII 6/12 File

```
JOB command
USER command
CHARGE command
GET,DATA612.
FCOPY(P=DATA612,N=FORMFIL)
FILE,FORMFIL,BT=C,RT=Z,MRL=300.
FILE,DATA6,BT=C,RT=Z,MRL=300.
FORM.
REWIND,DATA6.
COPYSBF,DATA6,OUTPUT.
7/8/9
INP(FORMFIL)
OUT(DATA6,BGD=X)
REF(DATA6,300(X=A))
6/7/8/9
```

Figure M-2. Job Decks for the Reformat of an ASCII File

The following REF directive is used:

REF(DATA6,300(X=A))

FORM examines each character and determines whether the character is valid data. To determine the correct index for an 8/12 ASCII file, the maximum record length of the file must be multiplied by 2. The maximum record length of FORMFIL is 150. Any other format of the REF directive would convert the zero byte delimiter of Z type records to display code M. DATA6, the file resulting from the reformat, is shown in figure M-3. DATA6 uses the 6-bit display code character set.

```
J. BROWN   ,1422 EAST ST.   ,CHARGE NO. 1111
S. APPLE   ,3434 CHERRY ST.,CHARGE NO. 2211
R. REDI    ,7896 ALGO AVE.  ,CHARGE NO. 1660
S. SNEADE  ,234 REDDING DR.,CHARGE NO. 1429
```

Figure M-3. DATA6, a File Containing Records Coded in Display Code

## NOS 2 LOCAL BATCH

Card decks consisting of characters from the 64-character ASCII subset can be processed as input, but cannot be punched out unless the operating system is installed in ASCII mode (IP CSET).

To create an input file in 80-column binary format from an ASCII deck under the NOS 2 operating system, the deck must be preceded and followed by cards with a 5/7/9 punch in column 1 and a 4/5/6/7/8/9 punch in column 2. The deck must be in a record by itself (preceded and followed by a 7/8/9 card).

A file that is created using this procedure is punched by specifying a disposition code of P8 on a ROUTE or DISPOSE control statement. For more detailed information consult the NOS 2 Reference Set, Volume 3, System Commands.

If binary information is to reside in either the file INPUT or the file PUNCH, a FILE control statement must be used to override the default block type and record type for these files. The FILE statement must specify F type records and C type blocks.

### MAGNETIC TAPE

Coded data to be copied from mass storage (via the COPYCR or COPYCF control statement) to magnetic tape is assumed to be represented in display code.

#### 7-Track Tape

NOS converts the coded data to external BCD code when writing a coded 7-track tape. However, if the COPYCR or COPYCF control statement is not used, the CM parameter of the FILE control statement determines the conversion mode when the file is processed by CYBER Record Manager. CM=NO indicates that no conversion is to take place. CM=YES indicates that all coded data is to be converted to external BCD code.

#### 9-Track Tape

The CV parameter of the ASSIGN, BLANK, LABEL, or REQUEST control statement determines the conversion mode for 9-track tapes. CV=AS and CV=US indicate ASCII conversion of the tape label (if any) and ASCII conversion of the data if the data is in coded form. CV=EB indicates ECBDIC/display code conversion. For unlabeled 9-track internal (I) or sytem internal (SI) format tapes, conversion is always forced to ASCII. The CM=YES parameter of the FILE control statement also ensures that conversion takes place if coded data is present.

If lowercase ASCII is read from a 9-track coded tape via the COPYCR or COPYCF control statement, the coded data is converted to the uppercase 6-bit display code equivalent. Any character lacking a display code equivalent is converted to a blank. To read and write lowercase ASCII characters, the user must assign the tape in binary mode. Either the COPYBR or COPYBF control statement or the CM=NO parameter of the FILE control statement must be used. The binary data must then be converted by either a user-written program or by FORM.

Data on a 9-track tape is ASCII 8/8 data. FORM can be used to convert ASCII 8/8 data to ASCII 8/12 data. The conversion must be done byte by byte. FORM can be used to insert the 4 leftmost bits of the ASCII 8/12 data format before the ASCII 8/8 data.

## NOS/BE

NOS/BE supports the 12-bit ASCII code set composed of the ASCII 7-bit code (as defined by ANSI Standard X3.4-1977) right-justified in a 12-bit byte. Assuming that the bits are numbered from the right starting with 0, bits 0 through 6 contain the ASCII code, bits 7 through 10 contain zeros, and bit 11 distinguishes the 12-bit ASCII $0000_8$ code from the end-of-line byte. The 12-bit codes are $0001_8$ through $0177_8$ and $4000_8$.

Figure M-4 shows an ASCII file (DATA812) that was created by a BASIC program. The file is reformatted under NOS/BE to form a file (DATA6) coded in display code (6/6). The input directives and dayfile for the reformat are shown in figure M-5. Figure M-6 shows DATA6.

```
THIS PROGRAM IS USED TO ILLUSTRATE
THE ASCII CONVERSION CAPABILITY OF
THE FORM UTILITY.
THE EXECUTION RESULTING FROM THE PROGRAM
IS GENERATED IN ASCII 8/12 MODE AND
PLACED INTO FILE DATA812.
THEN FORM IS INVOKED TO CONVERT THE
ASCII 8/12 DATA FILE INTO ANOTHER ONE
WHICH CONTAINS DISPLAY CODE [6-BIT]
INFORMATION.
THE ASCII 8/12 FILE IS SEPARATELY SENT
TO THE ASCII PRINTER.
this line will appear as lowercase
alphabetic information on the
ascii printer.
alphabet  abcdefghijklmnopqrstuvwxyz
```

Figure M-4. An ASCII 8/12 File

### INTERCOM TERMINALS

Display code is the default character set when communicating through a terminal. COMPASS users and FORTRAN users (via the CALL CONNEC subprogram) can elect to use ASCII with a 64-, 95-, or 256-character set selected. Additional information is provided in the INTERCOM reference manual and the appropriate compiler reference manual.

When card decks are read from remote batch devices through INTERCOM, the ability to select alternate keypunch code translations depends on the remote terminal equipment. Remote batch terminal line printer and punched card character set support is also described in the INTERCOM reference manual.

```
ACCOUNT.  014G,5912,693A415,BOBBIE.
ATTACH,NEW812,ID=MYID.
FILE NEW812,FO=SQ,RT=Z,BT=C,OF=N,CF=R,FL=200,ERL=0.
FILE DATA6,FO=SQ,RT=Z,BT=C,OF=N,CF=R,FL=200,ERL=0.
FORM I=INPUT,L=OUTPUT.
DISPOSE,DATA6,*PE.
REWIND,INPUT.
COPYSBF,INPUT,OUTPUT.
INP(NEW812,REW=N)
OUT(DATA6,REW=N,NOSEC,BGD=B)
REF(DATA6,82(X=A))
```

Figure M-5. Reformat of an ASCII 8/12 File Under NOS/BE

```
THIS PROGRAM IS USED TO ILLUSTRATE
THE ASCII CONVERSION CAPABILITY OF
THE FORM UTILITY.
THE EXECUTION RESULTING FROM THE PROGRAM
IS GENERATED IN ASCII 8/12 MODE AND
PLACED INTO FILE DATA812.
THEN FORM IS INVOKED TO CONVERT THE
ASCII 8/12 DATA FILE INTO ANOTHER ONE
WHICH CONTAINS DISPLAY CODE [6-BIT]
INFORMATION.
THE ASCII 8/12 FILE IS SEPARATELY SENT
TO THE ASCII PRINTER.
THIS LINE WILL APPEAR AS LOWERCASE
ALPHABETIC INFORMATION ON THE
ASCII PRINTER.
ALPHABET   ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Figure M-6. Output Resulting From a FORM Run Under NOS/BE

## NOS/BE LOCAL BATCH

To create an input file in free-form binary format from an ASCII deck under the NOS/BE system, the deck must be preceded and followed by cards with punches in all 12 rows of both column 1 and column 2 (or any other column as long as the cards are identical). The deck must be in a record by itself (preceded and followed by a 7/8/9 card).

A file created using this procedure is punched by specifying a disposition code of P80C on a ROUTE control statement or P8 on a DISPOSE control statement. For more detailed information consult the NOS/BE reference manual.

If binary information is to reside in either the file INPUT or the file PUNCH, a FILE control statement must be used to override the default block type and record type for these files. The FILE statement must specify F type records and C type blocks.

## MAGNETIC TAPE

Coded data to be copied from mass storage (via the COPYCR or COPYCF control statement) to magnetic tape is assumed to be represented in display code.

### 7-Track Tape

NOS/BE converts the coded data to external BCD code when writing a coded 7-track tape. However, if the COPYCR or COPYCF control statement is not used, the CM parameter of the FILE control statement determines the conversion mode provided CYBER Record Manager processes the file (as with FORM). CM=NO indicates that conversion is to take place. CM=YES indicates that all coded data is to be converted to external BCD code.

### 9-Track Tape

The N parameter of the ASSIGN, BLANK, LABEL, or REQUEST control statement determines the conversion mode for 9-track tapes. N=AS indicates ASCII conversion of the tape label (if any) and ASCII conversion of the data if the data is in coded form. N=EB indicates ECBDIC/display code conversion. For unlabeled 9-track internal (I) or sytem internal (SI) format tapes, conversion is always forced to ASCII. The CM=YES parameter of the FILE control statement also ensures that conversion takes place if coded data is present.

If lowercase ASCII is read from a 9-track coded tape via the COPYCR or COPYCF control statement, the coded data is converted to the uppercase 6-bit display code equivalent. Any character lacking a display code equivalent is converted to a blank. To read and write lowercase ASCII characters the user must assign the tape in binary mode with the N parameter omitted. Either the COPYBR or COPYBF control statement or the CM=NO parameter of the FILE control statement must be used. The binary data must then be converted either by a user-written program or by FORM.

Data on a 9-track tape is ASCII 8/8 data. FORM can be used to convert ASCII 8/8 data to ASCII 8/12 data. The conversion must be done byte by byte. FORM can be used to insert the 4 leftmost bits of the ASCII 8/12 data format before the ASCII 8/8 data.

# COLLATING SEQUENCES

Table N-1 shows the CDC character set collating sequence. Table N-2 shows the ASCII collating sequence.

TABLE N-1. CDC CHARACTER SET COLLATING SEQUENCE

| Collating Sequence Decimal/Octal | | CDC Graphic | Display Code | External BCD | Collating Sequence Decimal/Octal | | CDC Graphic | Display Code | External BCD |
|---|---|---|---|---|---|---|---|---|---|
| 00 | 00 | blank | 55 | 20 | 32 | 40 | H | 10 | 70 |
| 01 | 01 | ≤ | 74 | 15 | 33 | 41 | I | 11 | 71 |
| 02 | 02 | % | 63 † | 16 † | 34 | 42 | ∨ | 66 | 52 |
| 03 | 03 | [ | 61 | 17 | 35 | 43 | J | 12 | 41 |
| 04 | 04 | → | 65 | 35 | 36 | 44 | K | 13 | 42 |
| 05 | 05 | ≡ | 60 | 36 | 37 | 45 | L | 14 | 43 |
| 06 | 06 | ∧ | 67 | 37 | 38 | 46 | M | 15 | 44 |
| 07 | 07 | ↑ | 70 | 55 | 39 | 47 | N | 16 | 45 |
| 08 | 10 | ↓ | 71 | 56 | 40 | 50 | O | 17 | 46 |
| 09 | 11 | > | 73 | 57 | 41 | 51 | P | 20 | 47 |
| 10 | 12 | ≥ | 75 | 75 | 42 | 52 | Q | 21 | 50 |
| 11 | 13 | ¬ | 76 | 76 | 43 | 53 | R | 22 | 51 |
| 12 | 14 | . | 57 | 73 | 44 | 54 | ] | 62 | 32 |
| 13 | 15 | ) | 52 | 74 | 45 | 55 | S | 23 | 22 |
| 14 | 16 | ; | 77 | 77 | 46 | 56 | T | 24 | 23 |
| 15 | 17 | + | 45 | 60 | 47 | 57 | U | 25 | 24 |
| 16 | 20 | $ | 53 | 53 | 48 | 60 | V | 26 | 25 |
| 17 | 21 | * | 47 | 54 | 49 | 61 | W | 27 | 26 |
| 18 | 22 | - | 46 | 40 | 50 | 62 | X | 30 | 27 |
| 19 | 23 | / | 50 | 21 | 51 | 63 | Y | 31 | 30 |
| 20 | 24 | , | 56 | 33 | 52 | 64 | Z | 32 | 31 |
| 21 | 25 | ( | 51 | 34 | 53 | 65 | : | 00 † | none† |
| 22 | 26 | = | 54 | 13 | 54 | 66 | 0 | 33 | 12 |
| 23 | 27 | ≠ | 64 | 14 | 55 | 67 | 1 | 34 | 01 |
| 24 | 30 | < | 72 | 72 | 56 | 70 | 2 | 35 | 02 |
| 25 | 31 | A | 01 | 61 | 57 | 71 | 3 | 36 | 03 |
| 26 | 32 | B | 02 | 62 | 58 | 72 | 4 | 37 | 04 |
| 27 | 33 | C | 03 | 63 | 59 | 73 | 5 | 40 | 05 |
| 28 | 34 | D | 04 | 64 | 60 | 74 | 6 | 41 | 06 |
| 29 | 35 | E | 05 | 65 | 61 | 75 | 7 | 42 | 07 |
| 30 | 36 | F | 06 | 66 | 62 | 76 | 8 | 43 | 10 |
| 31 | 37 | G | 07 | 67 | 63 | 77 | 9 | 44 | 11 |

†In installations using the 63-graphic set, the % graphic does not exist. The : graphic is display code 63, External BCD code 16.

TABLE N-2. ASCII CHARACTER SET COLLATING SEQUENCE

| Collating Sequence Decimal/Octal | | ASCII Graphic Subset | Display Code | ASCII Code | Collating Sequence Decimal/Octal | | ASCII Graphic Subset | Display Code | ASCII Code |
|---|---|---|---|---|---|---|---|---|---|
| 00 | 00 | blank | 55 | 20 | 32 | 40 | @ | 74 | 40 |
| 01 | 01 | ! | 66 | 21 | 33 | 41 | A | 01 | 41 |
| 02 | 02 | " | 64 | 22 | 34 | 42 | B | 02 | 42 |
| 03 | 03 | # | 60 | 23 | 35 | 43 | C | 03 | 43 |
| 04 | 04 | $ | 53 | 24 | 36 | 44 | D | 04 | 44 |
| 05 | 05 | % | 63† | 25 | 37 | 45 | E | 05 | 45 |
| 06 | 06 | & | 67 | 26 | 38 | 46 | F | 06 | 46 |
| 07 | 07 | ' | 70 | 27 | 39 | 47 | G | 07 | 47 |
| 08 | 10 | ( | 51 | 28 | 40 | 50 | H | 10 | 48 |
| 09 | 11 | ) | 52 | 29 | 41 | 51 | I | 11 | 49 |
| 10 | 12 | * | 47 | 2A | 42 | 52 | J | 12 | 4A |
| 11 | 13 | + | 45 | 2B | 43 | 53 | K | 13 | 4B |
| 12 | 14 | , | 56 | 2C | 44 | 54 | L | 14 | 4C |
| 13 | 15 | - | 46 | 2D | 45 | 55 | M | 15 | 4D |
| 14 | 16 | . | 57 | 2E | 46 | 56 | N | 16 | 4E |
| 15 | 17 | / | 50 | 2F | 47 | 57 | O | 17 | 4F |
| 16 | 20 | 0 | 33 | 30 | 48 | 60 | P | 20 | 50 |
| 17 | 21 | 1 | 34 | 31 | 49 | 61 | Q | 21 | 51 |
| 18 | 22 | 2 | 35 | 32 | 50 | 62 | R | 22 | 52 |
| 19 | 23 | 3 | 36 | 33 | 51 | 63 | S | 23 | 53 |
| 20 | 24 | 4 | 37 | 34 | 52 | 64 | T | 24 | 54 |
| 21 | 25 | 5 | 40 | 35 | 53 | 65 | U | 25 | 55 |
| 22 | 26 | 6 | 41 | 36 | 54 | 66 | V | 26 | 56 |
| 23 | 27 | 7 | 42 | 37 | 55 | 67 | W | 27 | 57 |
| 24 | 30 | 8 | 43 | 38 | 56 | 70 | X | 30 | 58 |
| 25 | 31 | 9 | 44 | 39 | 57 | 71 | Y | 31 | 59 |
| 26 | 32 | : | 00† | 3A | 58 | 72 | Z | 32 | 5A |
| 27 | 33 | ; | 77 | 3B | 59 | 73 | [ | 61 | 5B |
| 28 | 34 | < | 72 | 3C | 60 | 74 | \ | 75 | 5C |
| 29 | 35 | = | 54 | 3D | 61 | 75 | ] | 62 | 5D |
| 30 | 36 | > | 73 | 3E | 62 | 76 | ^ | 76 | 5E |
| 31 | 37 | ? | 71 | 3F | 63 | 77 | _ | 65 | 5F |

†In installations using a 63-graphic set, the % graphic does not exist. The : graphic is display code 63.

# INDEX

COMMENT SHEET

MANUAL TITLE: FORM Version 1 Reference Manual

PUBLICATION NO.: 60496200

REVISION: E

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments on the back (please include page number references).

_____ Please reply _____ No reply necessary

FOLD FOLD

CUT ALONG LINE

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MN.

POSTAGE WILL BE PAID BY ADDRESSEE

CONTROL DATA

**Technical Publications**
SVLF45
5101 Patrick Henry Drive
Santa Clara, CA 95054-1111

FOLD FOLD

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.
FOLD ON DOTTED LINES AND TAPE

NAME:

COMPANY:

STREET ADDRESS:

CITY/STATE/ZIP:

TAPE TAPE