60497000

CONTROL DATA CORPORATION

COBOL VERSION 4 INSTANT MANUAL

CDC® OPERATING SYSTEMS:
NOS
NOS/BE

REVISION RECORD

REVISION	DESCRIPTION
A	Original release.
(2-15-76)	
B	All technical corrections have been incorporated.
(1-15-80)	This is the last scheduled revision to this manual.
	It will be a regularly stocked item at LDS for a
	minimum of 12 months after this release.

ublication No.
60497000

VISION LETTERS I, O, Q
D X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

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	REV
Cover	—
Title Page	—
ii	B
iii/iv	B
v/vi	B
vii/viii	B
1 thru 44	B

PAGE	REV

PREFACE

This instant outlines the COBOL Version 4.7 language which operates under control of the following operating systems:

> NOS 1 for the CONTROL DATA® CYBER 170 Models 171, 172, 173, 174, 175, CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems.
>
> NOS/BE 1 for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems.

COBOL Version 4.7 is compatible with the American National Standard COBOL, X3.23-1968, and contains extensions to the standard. Extensions are indicated in this instant by shading.

This instant is written for a programmer familiar with the COBOL 4 language and the operating system on which the language is used.

More detailed information can be found in the publications listed below.

Publication	Publication Number
COBOL Version 4 Reference Manual	60496800
NOS Version 1 Reference Manual, Volume 1 of 2	60435400
NOS/BE Version 1 Reference Manual	60493800

CDC manuals can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

SPECIAL FEATURES

Mass storage input and output including indexed sequential, actual key, and direct access file processing

SORT verb sorts files within COBOL program

Automatic table search using index names and the SEARCH and SET statements

Report Writer produces printed reports automatically, or user may produce report page with LINAGE clause and WRITE statement

Full arithmetic facility including:
- 18-digit operands
- DIVIDE with REMAINDER
- COMPUTE with exponentiation
- CORRESPONDING option with ADD and SUBTRACT

Segmentation and overlay of object program

Inter-program communication with separately compiled COBOL programs as well as with FORTRAN or COMPASS programs

Access to COBOL source library

Memory dumps with restart at specified checkpoints

Remote interactive capability for remote terminal input/output

Multiple-index processing capability

Ability to trace paragraphs

EBCDIC file processing

CONTENTS

PROGRAM EFFICIENCY HINTS

To reduce keypunching:

Use abbreviations where permitted.

Use PIC clause rather than SIZE, CLASS, USAGE clauses.

To increase compilation efficiency:

Restrict data and paragraph names to 9 characters or less.

Eliminate unnecessary paragraph names.

Reduce forward references.

To increase execution efficiency:

Use same size sending and receiving fields.

Make table and item sizes a multiple of 10 characters.

Reduce subscripting.

Subscript with literals instead of variables.

Use COMPUTATIONAL-1 items or index-names as subscripts.

Use COMPUTATIONAL-1 items as arithmetic variables.

Restrict arithmetic items to 9 digits or less.

Use SYNCHRONIZED RIGHT clause for data frequently referenced.

Use SAME RECORD AREA to save moves; SAME AREA to save space.

NOTATION

[ ] Enclosed elements are optional.

$\left\{ \begin{matrix} \\ \\ \end{matrix} \right\}$ Only one element must be selected

... Repeat preceding bracketed material as needed.

{ } ... Entire phrase can be repeated.

COBOL words have preassigned meanings and appear in capitals.

COBOL words not underlined can be omitted.

Terms in small letters are words supplied by the programmer.

Punctuation and special characters are required where shown.

COBOL LANGUAGE ELEMENTS

Word	Sequence of up to 30 alphanumeric characters including embedded hyphens.
Identifier	Data name followed by qualifiers, subscripts, and indexes.
Literal	String of characters that represents a specific value; numeric literal can be a string of up to 18 digits 0-9, +, -, and decimal point; non-numeric literal can be a string of up to 255 alphanumeric characters and must be enclosed in quotes.
Identifier	Word that may be qualified or subscripted.
Literal	String of characters whose value is exactly represented by the characters; numeric literal may be 0-9, +, -, and decimal point; non-numeric literal must be enclosed in quotes, may be any alphanumeric character except quotes.
Statement	Procedure Division verb with associated options.
Sentence	One or more statements terminated by period.
Paragraph	Procedure Division sentences, Identification and Environment Division entries introduced by paragraph name, terminated by period.
Paragraph Name	Word terminated by period used to introduce paragraph; user defined in Procedure Division, pre-defined in Identification and Environment Divisions.
Section	Group of one or more paragraphs introduced by section header.
Section Header	Word followed by SECTION and terminated by period; user defined in Procedure Division, pre-defined in Environment and Data Divisions.
Entry	Unit of description in Data Division; must be terminated by period.

IDENTIFICATION DIVISION

```
{ ID.             }
{ IDENTIFICATION  }  DIVISION.
```

PROGRAM-ID. program-name.
[AUTHOR. [comment-entry.]]
[INSTALLATION. [comment-entry.]]
[DATE-WRITTEN. [comment-entry.]]
[DATE-COMPILED. [current-date supplied by compiler.]]
[SECURITY. [comment-entry.]]
[REMARKS. [comment-entry.]]

ENVIRONMENT DIVISION

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.†

format 1:

SOURCE-COMPUTER. COPY library-name

```
[ REPLACING { literal-1    } BY { literal-2    }
            { word-1       }    { word-2       }
            { identifier-1 }    { identifier-2 }

  [ { literal-3    } BY { literal-4    } ]... ]
  [ { word-3       }    { word-4       } ]
  [ { identifier-3 }    { identifier-4 } ]
```

format 2:

SOURCE-COMPUTER. computer-name.

format 1:

OBJECT-COMPUTER. COPY library-name

```
[ REPLACING { literal-1    } BY { literal-2    }
            { word-1       }    { word-2       }
            { identifier-1 }    { identifier-2 }

  [ { literal-3    } BY { literal-4    } ]... ]
  [ { word-3       }    { word-4       } ]
  [ { identifier-3 }    { identifier-4 } ]
```

†CONFIGURATION SECTION, SOURCE-COMPUTER, and OBJECT-COMPUTER are optional.

format 2:

```
OBJECT-COMPUTER. computer-name

    [SEGMENT-LIMIT IS priority-number]

    [ MEMORY SIZE integer { WORDS      } ]
                          { CHARACTERS }
                          { MODULES    }
```

format 1:

```
SPECIAL-NAMES. COPY library-name

[ REPLACING { literal-1  } BY { literal-2  }
            { word-1     }    { word-2     }
            { identifier-1 }  { identifier-2 }

          [ { literal-3  } BY { literal-4  } ]... ]
            { word-3     }    { word-4     }
            { identifier-3 }  { identifier-4 }
```

format 2:

```
SPECIAL-NAMES.

[ SWITCH integer-1

  { IS mnemonic-name-1                                  }
  {    [ON STATUS IS condition-name-1                   }
  {    [OFF STATUS IS condition-name-2] ]               }
  { IS mnemonic-name-2                                  }
  {    [OFF STATUS IS condition-name-3                  }
  {    [ON STATUS IS condition-name-4] ]                }
  { ON STATUS IS condition-name-5                       }  ...
  {    [OFF STATUS IS condition-name-6]                 }
  { OFF STATUS IS condition-name-7                      }
  {    [ON STATUS IS condition-name-8]                  } ]

  [ implementor-name-1 IS mnemonic-name-1
    implementor-name-2 IS mnemonic-name-2 . . .]

  [ non-numeric-literal-1 IS mnemonic-name-1
    non-numeric-literal-2 IS mnemonic-name-2 . . .]

  [CURRENCY SIGN IS literal]

  [DECIMAL-POINT IS COMMA]
```

```
[CONSOLE IS mnemonic-name]
[TERMINAL IS mnemonic-name]
[SUB-SCHEMA IS sub-schema-name]
```

INPUT-OUTPUT SECTION.

format 1:

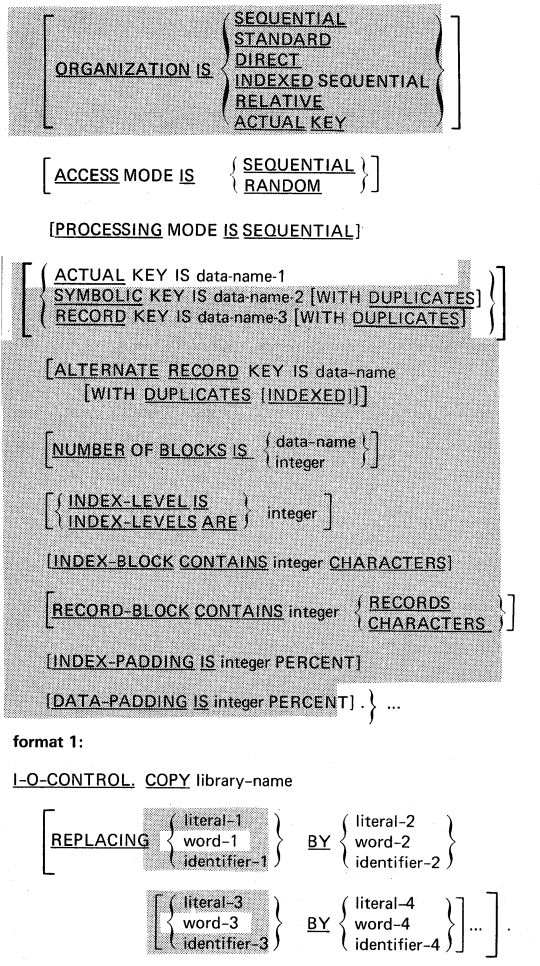
FILE-CONTROL. COPY library-name

```
[ REPLACING { literal-1  } BY { literal-2  }
            { word-1     }    { word-2     }
            { identifier-1 }  { identifier-2 }

            [ { literal-3  } BY { literal-4  } ... ] ]
            { word-3     }    { word-4     }
            { identifier-3 }  { identifier-4 }
```

format 2:

FILE-CONTROL.

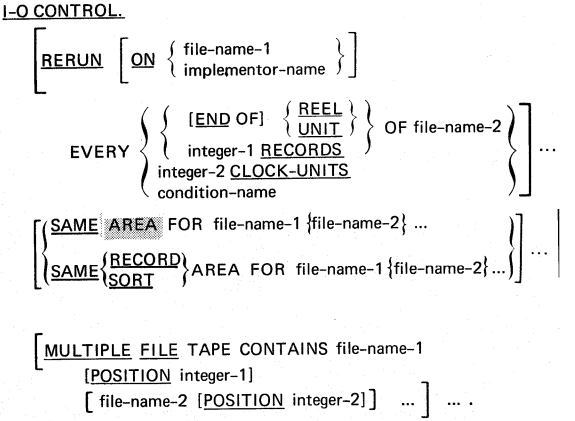
```
{SELECT [OPTIONAL] file-name-1 [RENAMING file-name-2]

 ASSIGN TO [integer] implementor-name-1 [implementor-name-2]...

   [OR implementor-name-3 [implementor-name-4] ...]

[ FOR MULTIPLE [{REEL}] ] [ERROR FILE IS file-name]
               [{UNIT}]

[RESERVE { NO      } ALTERNATE [{AREA }]]
         { integer }           [{AREAS}]

[{ {FILE-LIMIT IS  }  {data-name-1 }
  {FILE-LIMITS ARE }  {literal-1   }
   {THRU     }  {data-name-2 }
   {THROUGH  }  {literal-2   }
   [ {data-name-3 } {THRU    } {data-name-4 } ] ...
   {literal-3   }  {THROUGH }  {literal-4   }
  FILE-LIMIT IS {data-name-5 }
                {literal-5   } }]
```



format 1:

I-O-CONTROL. COPY library-name

format 2:


I-O CONTROL.
RERUN
ON
file-name-1
implementor-name
EVERY
END OF
REEL
UNIT
integer-1 RECORDS
integer-2 CLOCK-UNITS
condition-name
OF file-name-2
SAME AREA FOR file-name-1 file-name-2 ...
SAME RECORD SORT AREA FOR file-name-1 file-name-2 ...
MULTIPLE FILE TAPE CONTAINS file-name-1
[POSITION integer-1]
[file-name-2 [POSITION integer-2]] ...] ... .

PICTURE DESCRIPTION CODES

Data Characters

A	Alphabetic character
X	Alphanumeric character
9	Numeric character

Operation Symbols

S	Signed
V	Assumed decimal point location
P	Assumed decimal point scaling position

Replacement Characters

Z	Leading zeros replaced by blanks
*	Leading zeros replaced by * (check protection symbol)

Insertion Characters

$	Dollar sign; floating when more than one (dollar sign may be replaced by currency sign defined in SPECIAL-NAMES)
,	Comma
/	Slash (instead of comma)
.	Actual decimal point
B	Blank
0	Zero
-	Minus sign when item is negative, blank when positive; floating when more than one
+	Plus sign when item is positive, minus when negative; floating when more than one
CR	Credit symbol when item is negative, blank when positive
DB	Debit symbol when item is negative, blank when positive

	File Section			Common and Working Storage Sections				Constant Section			
	01	group	elem	77	01	group	elem	77	01	group	elem
REDEFINES	I										
SIZE		K	R	R		K	R	R		K	R
USAGE											
CLASS				R				R			
OCCURS	I			I	I			I	I		
POINT LOCATION	J	I			J	I			J	I	
SIGNED	J	I			J	I			J	I	
JUSTIFIED	J	I			J	I			J	I	
SYNCHRONIZED	J	I			J	I			J	I	
PICTURE	J	I			J	I			J	I	
Editing Clauses	J	I			J	I		I	J	I	I
COPY											
VALUE	K	K	C					V			V
FILLER	I			I	I			I	I		

C — Legal only in defining values for condition names

I — Illegal

R — Required if PICTURE is not used

blank — Optional

V — Required

J — Legal only on elementary 01 items

K — Documentary only

01 items can be elementary or group, depending on the program

DATA DIVISION

DATA DIVISION.
[FILE SECTION.]
[COMMON-STORAGE SECTION.]
[WORKING-STORAGE SECTION.]
[CONSTANT SECTION.]
[LINKAGE SECTION.]
[REPORT SECTION.]

File Description Entry (File Section Only)

format 1:

FD file-name COPY library-name

```
[ REPLACING { literal-1    } BY { literal-2    }
            { word-1       }    { word-2       }
            { identifier-1 }    { identifier-2 }
          [ { literal-3    } BY { literal-4    } ] ... ]
            { word-3       }    { word-4       }
            { identifier-3 }    { identifier-4 }
```

format 2:

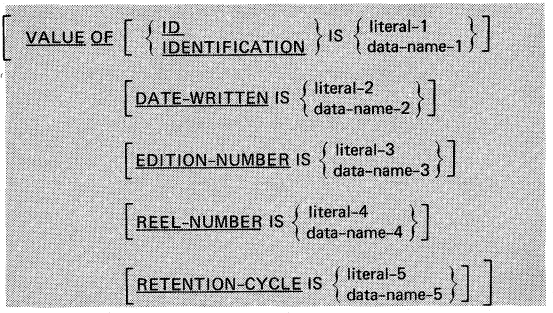
FD file-name

```
[ BLOCK CONTAINS [integer-1 TO] integer-2 { RECORDS    } ]
                                          { CHARACTERS }

{ [ DATA { RECORD IS  } data-name-1 [data-name-2] ... ] }
         { RECORDS ARE }                                 }
{ { REPORT IS  }                                         }
{ { REPORTS ARE } report-name-1 [report-name-2] ...      }
```

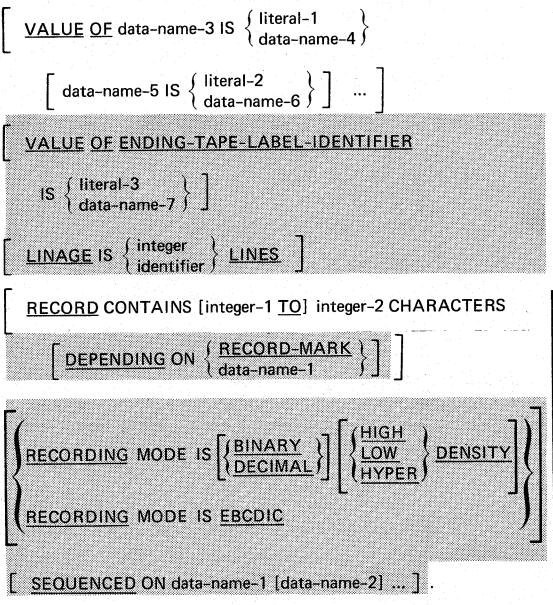
[FILE CONTAINS ABOUT integer RECORDS]

```
LABEL { RECORDS ARE } { STANDARD      }
      { RECORD IS   } { OMITTED       }
                      { data-name-1 [data-name-2] ... }
```

If label records are STANDARD:



If label records are a data-name:



Sort File Description Entry (File Section Only)

format 1:

```
SD file-name  COPY library-name
    [ REPLACING { literal-1 } BY { literal-2 }
                { word-1    }    { word-2    }
                { identifier-1 }  { identifier-2 }
          [ { literal-3 } BY { literal-4 } ] ... ]
          { word-3    }    { word-4    }
          { identifier-3 }  { identifier-4 }
```

format 2:

```
SD file-name
    [DATA {RECORD IS  } data-name-1 [data-name-2] ...]
          {RECORDS ARE}
    [RECORD CONTAINS [integer-1 TO] integer-2 CHARACTERS]
    [FILE CONTAINS ABOUT integer RECORDS ]
```

Record Description Entry (File, Common-Storage, Working-Storage, Constant and Linkage Sections)

format 1:

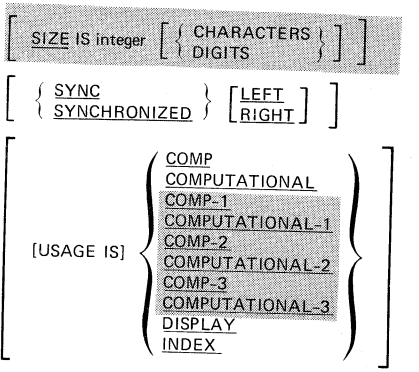
```
{01   }
{02-49} data-name  COPY library-name [FROM LIBRARY]
    REPLACING { word-1    } BY { word-2    }
              { identifier-1 }  { identifier-2 }
              { literal-1 }     { literal-2 }
        [ { word-3    } BY { word-4    } ]  ... ]
        { identifier-3 }  { identifier-4 }
        { literal-3 }     { literal-4 }
```

format 2:

```
level-number data-name-1 [REDEFINES identifier]
    COPY data-name-2 FROM SOURCE.
```

format 3:

```
level-number { data-name } [REDEFINES identifier]
             { FILLER   }

[ { BWZ              } ]
  { BLANK WHEN ZERO  }

[ { CHECK PROTECT                                  }
  { FLOAT { DOLLAR   } SIGN } [LEAVING integer PLACES]
  {       { CURRENCY }      }
  { ZERO SUPPRESS                                  } ]

[ [CLASS IS] { ALPHABETIC  } ]
             { NUMERIC     }
             { ALPHANUMERIC }
             { AN          } ]

[ { JUST      } RIGHT ]
  { JUSTIFIED }

[ OCCURS [integer-1 TO] integer-2 TIMES
    [DEPENDING ON data-name-1]
    [ { ASCENDING  } KEY IS data-name-2 [data-name-3] ... ] ...
      { DESCENDING }
    [INDEXED BY index-name-1 [index-name-2] ...] ]

[ { PIC     } IS character-string ]
  { PICTURE }

[ POINT LOCATION IS { LEFT  } integer PLACES ]
                    { RIGHT }

[ RANGE IS literal-1 [ { THRU    } literal-2 ] ]
                       { THROUGH }

[ { SIGNED          } ]
  { SIGN IS data-name }
```



[VALUE IS literal].

format 4:

66 data-name RENAMES identifier-1 [{ THRU / THROUGH } identifier-2].

format 5:

88 condition-name { VALUE IS / VALUES ARE } literal-1

[{ THRU / THROUGH } literal-2] [literal-3 [{ THRU / THROUGH } literal-4]]...

Report Description Entry (Report Section Only)

format 1:

RD report-name [WITH CODE mnemonic-name]

COPY library-name [REPLACING { literal-1 / word-1 / identifier-1 } BY

{ literal-2 / word-2 / identifier-2 } [{ literal-3 / word-3 / identifier-3 } BY { literal-4 / word-4 / identifier-4 }] ...]

format 2:

RD report-name [WITH CODE mnemonic-name]

```
[ { CONTROL IS  } { identifier-1 [identifier-2] ...                  } ]
[ { CONTROLS ARE} { FINAL                                            } ]
                  { FINAL identifier-1 [identifier-2] ...            }

[ PAGE { LIMIT IS  } integer-1 { LINE  }                             ]
       { LIMITS ARE}           { LINES }
    [HEADING integer-2] [FIRST DETAIL integer-3]
    [LAST DETAIL integer-4] [FOOTING integer-5]                      ]
```

Report Group Description Entry (Report Section Only)

format 1:

01 [data-name] COPY library-name [FROM LIBRARY]

```
[ REPLACING { literal-1    } BY { literal-2    }
            { word-1       }    { word-2       }
            { identifier-1 }    { identifier-2 }

            [ { literal-3    } BY { literal-4    }  ] ... ]
            { word-3       }    { word-4       }
            { identifier-3 }    { identifier-4 }
```

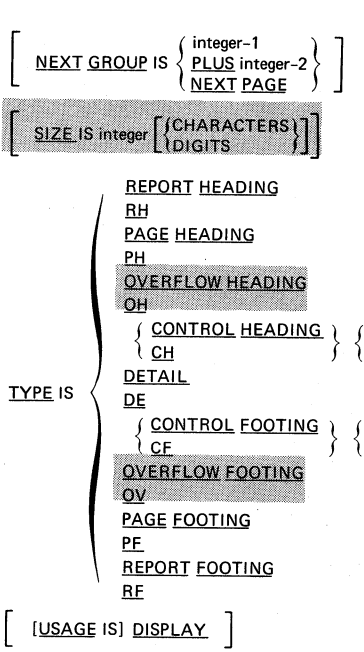
format 2:

01 data-name-1 [REDEFINES identifier]
COPY data-name-2 FROM SOURCE.

format 3:

01 [data-name]

```
[ [CLASS IS] { ALPHABETIC   } ]
             { NUMERIC      }
             { ALPHANUMERIC }
             { AN           }

[ LINE NUMBER IS { integer-1      } ]
                 { PLUS integer-2 }
                 { NEXT PAGE      }
```

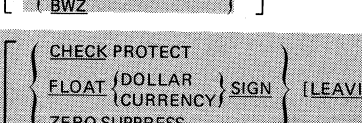


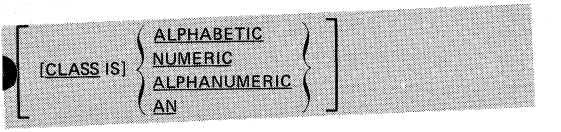
Report Element Description (Report Section Only)

TYPE clause allowed if level 01

NEXT GROUP clause allowed if level 01

level-number [data-name]





[COLUMN NUMBER IS integer]

[GROUP INDICATE]

[{ JUSTIFIED / JUST } RIGHT]

[LINE NUMBER IS { integer-1 / PLUS integer-2 / NEXT PAGE }]

[{ PIC / PICTURE } IS character-string]

[POINT LOCATION IS { LEFT / RIGHT } integer PLACES]

[RESET ON { identifier / FINAL }]

[{ SIGNED / SIGN IS data-name }]

[SIZE IS integer [{ CHARACTERS / DIGITS }]]

{ SOURCE IS { [SELECTED] identifier / LINE-COUNTER / PAGE-COUNTER / TODAYS-DATE }
SUM identifier-1 [identifier-2] ... [UPON data-name]
VALUE IS literal }

[[USAGE IS] DISPLAY].

USAGE SPECIFICATIONS

Element	Upper Limit
data-name	30 characters
literal: numeric non-numeric	18 digits 255 characters
PERFORM nesting	no limit
level numbers	01-49, 66, 77, 88, FD, RD, SD
OCCURS...DEPENDING ON	1 per record description
library copies	5 levels of nesting
ACCEPT items	80 characters; 40 characters from console
PICTURE clause	30 symbols
arithmetic operand	18 digits
GO TO statement	no limit
ALTER statement	100 procedure names
DISPLAY items	no limit
ENTER parameters	no limit
Total files, I/O devices, and reports	53
Total procedure names	depends on field length
Total external references	depends on field length
index names	9
report group	127 detail lines

VALID MOVE OPERATIONS

Source Field \ Rec. Field	Elem. Binary	Elem. Alpha	Elem. BCD Num.	Elem. AN	Elem. Edit Num.	Elem. Edit AN	Group AN
Elem. Binary	Num. Bin.	X	Conv. Num.	Conv.† AN	Conv. Edit	Conv.† AN-Edit	TD AN
Elem. Alpha	X	AN	TD AN	AN	X	AN-Edit	AN
Elem. BCD Num.	Conv. Bin.	TD AN	Num.	AN†	Edit†	AN-Edit	AN†
Elem. AN	X	TD AN	Num.	AN	Edit	AN-Edit	AN
Elem. Edit Num.	X	TD AN	X	AN	X	AN-Edit	AN
Elem. Edit AN	X	TD AN	X	AN	X	AN-Edit	AN
Group AN	TD AN	TD AN	TD AN	AN	X	AN-MOVES	AN
Group Binary & Mixed	TD AN	TD AN	TD AN	TD AN	X	TD AN-MOVES	TD AN
Zero	Num. Bin.	X	Num.	AN	Edit	AN-Edit	AN
Literal & Fig. Cons. AN	X	TD AN	X	AN	Edit	AN-Edit	AN
Literal Num.	Conv. Bin.	X	Num.	AN†	Edit	AN-Edit	AN

† Valid only when source is integer; others TD.

Any move to a binary or mixed group is treated as an alphanumeric move; a precautionary diagnostic is issued.

A move to a figurative constant or literal is illegal.

X	Illegal
AN	Alphanumeric
AN-Edit	Alphanumeric edited
Conv.	Conversion prior to move
Edit	Numeric edited
Num.	Numeric
Num. Bin.	Numeric binary
TD	Trivial diagnostic issued
AN-MOVES	Alphanumeric moves
Bin.	Binary

PROCEDURE DIVISION

```
PROCEDURE DIVISION [USING parameter-list]

[ DECLARATIVES.

    { section-name SECTION. declarative-sentence.

    { paragraph-name. { sentence. } ... } ... } ...

  END DECLARATIVES. ]

    { section-name SECTION [priority-number]

    { paragraph-name.   { sentence. } ... } ... } ...
```

```
ACCEPT identifier [ FROM { TIME      } ]
                         { DATE      }
                         { DAY       }
                         { mnemonic-name }
```

```
ADD { identifier-1 } [ { identifier-2 } ] ...
    { literal-1    }   { literal-2    }

    identifier-3 [ROUNDED]

    [ON SIZE ERROR imperative-statement]
```

```
ADD { identifier-1 } [ { identifier-2 } ] ...
    { literal-1    }   { literal-2    }

    TO identifier-3 [ROUNDED] [identifier-4 [ROUNDED]] ...

    [ON SIZE ERROR imperative-statement]
```

```
ADD { identifier-1 } { identifier-2 } [ { identifier-3 } ] ...
    { literal-1    } { literal-2    }   { literal-3    }

    GIVING identifier-4 [ROUNDED] [identifier-5 [ROUNDED]] ...

    [ON SIZE ERROR imperative-statement]
```

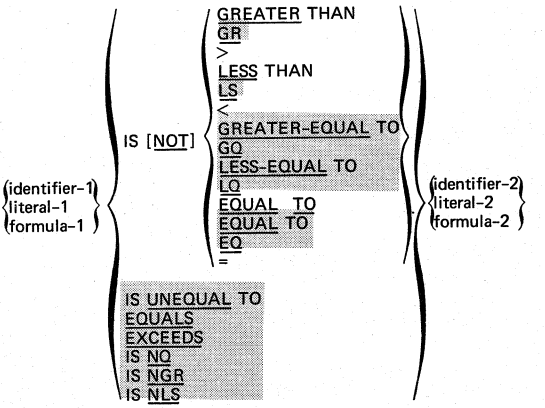
```
ADD { CORR          } identifier-1
    { CORRESPONDING }

    TO identifier-2 [ROUNDED] [identifier-3 [ROUNDED]] ...

    [ON SIZE ERROR imperative-statement]
```

```
ALTER procedure-name-1 TO [PROCEED TO] procedure-name-2
    [procedure-name-3 TO [PROCEED TO] procedure-name-4] ...

CALL program-name [USING parameter-list]

CLOSE file-name-1 [ { UNIT  } ]   [ WITH { NO REWIND } ]
                  [ { REEL  } ]          { LOCK      }
  [ file-name-2 [ { UNIT  } ]   [ WITH { NO REWIND } ] ]...
                [ { REEL  } ]          { LOCK      }

COMPUTE identifier-1 [ROUNDED] [identifier-2 [ROUNDED]] ...
    { FROM   }   { literal                  }
    { =      }   { arithmetic-expression    }
    { EQUALS }   { identifier-3             }
  [ON SIZE ERROR imperative-statement]

{ COPY    }   { identifier-1 }   [FROM LIBRARY]
{ INCLUDE }   { library-name }

[ REPLACING { literal-1    }   BY { literal-2    }
            { word-1       }      { word-2       }
            { identifier-2 }      { identifier-3 }
          [ { literal-3    }   BY { literal-4    } ]... ]
            { word-3       }      { word-4       }
            { identifier-4 }      { identifier-5 }

DELETE [LAST] RECORD FROM file-name
    INVALID KEY imperative-statement

DISPLAY { identifier-1 } [ { identifier-2 } ] ...
        { literal-1    }   { literal-2    }
  [UPON mnemonic-name]

DIVIDE { identifier-1 } INTO identifier-2 [ROUNDED]
       { literal      }
  [identifier-3 [ROUNDED]] ...
  [ON SIZE ERROR imperative-statement]
```

```
DIVIDE { identifier-1 } { BY   } { identifier-2 }
       { literal-1    } { INTO } { literal-2    }

    GIVING identifier-3 [ROUNDED] [identifier-4 [ROUNDED]]...

    [ON SIZE ERROR imperative-statement]

DIVIDE { identifier-1 } { BY   } { identifier-2 }
       { literal-1    } { INTO } { literal-2    }

    GIVING identifier-3 [ROUNDED]

    REMAINDER identifier-4

    [ON SIZE ERROR imperative-statement]

ENTER [language-name] routine-name [USING parameter-list].

ENTER COBOL.]

ENTER LINKAGE.]

ENTRY routine-name [USING parameter-list].

EXAMINE identifier

{ TALLYING { ALL      }                  literal-1
           { LEADING  }                    [REPLACING BY literal-2]
           { UNTIL FIRST }
{ REPLACING { ALL      }                 literal-3 BY literal-4
            { LEADING  }
            { [UNTIL] FIRST }

EXIT.

{ EXIT PROGRAM. }
{ RETURN.       }

GENERATE identifier

GO TO [procedure-name]

GO TO procedure-name-1 [procedure-name-2]...

    DEPENDING ON identifier
```

```
IF conditional-expression [THEN] { statement-1      }
                                 { NEXT SENTENCE    }

   [ [THEN] { OTHERWISE } { statement-2      } ]
            { ELSE      } { NEXT SENTENCE    }
```

Conditional expressions include:




```
{identifier} IS [NOT] { POSITIVE }
{formula   }          { NEGATIVE }
                      { ZERO     }

identifier IS [NOT] { NUMERIC      }
                    { ALPHABETIC   }

[NOT] { condition-name     }
      { switch-status-name }

INITIATE { report-name-1 [report-name-2] ... }
         { ALL                               }

MOVE { identifier-1 } TO identifier-2 [identifier-3] ...
     { literal      }

MOVE { CORR          } identifier-1 TO
     { CORRESPONDING }

     identifier-2 [identifier-3] ...
```

```
MULTIPLY { identifier-1 } BY identifier-2 [ROUNDED]
         { literal      }
    [identifier-3 [ROUNDED]] ...
    [ON SIZE ERROR imperative-statement]

MULTIPLY { identifier-1 } BY { identifier-2 }
         { literal-1    }    { literal-2    }
    GIVING identifier-3 [ROUNDED]
    [identifier-4 [ROUNDED]] ...
    [ON SIZE ERROR imperative-statement]

NOTE character-string.

                  EXTEND file-name-1 [file-name-2] ...
                  INPUT file-name-1 [ { REVERSED      } ]
                                     { WITH NO REWIND }
                      [ file-name-2 [ { REVERSED      } ] ] ...
                                    { WITH NO REWIND }
OPEN                  OUTPUT file-name-1   [WITH NO REWIND]
                             [file-name-2   [WITH NO REWIND]] ...
                    { INPUT-OUTPUT }
                    { I-O          } file-name-1 [file-name-2] ...

PERFORM procedure-name-1 [ { THRU    } procedure-name-2 ]
                         { THROUGH }
    [ { { identifier } TIMES }  ]
      { { integer    }       }
      { UNTIL condition      }
```

```
PERFORM procedure-name-1 [ { THRU      } procedure-name-2 ]
                         { THROUGH }

    VARYING { index-name-1 } FROM { literal-1      } BY
            { identifier-1 }      { index-name-2   }
                                  { identifier-2   }

    { literal-2    } UNTIL condition-1 [ AFTER { index-name-3 }
    { identifier-3 }                          { identifier-4 }

    FROM { literal-3    } BY { literal-4    } UNTIL condition-2
         { index-name-4 }    { identifier-6 }
         { identifier-5 }

    [ AFTER { index-name-5 } FROM { literal-5    } BY
            { identifier-7 }      { index-name-6 }
                                  { identifier-8 }

    { literal-6    } UNTIL condition-3 ] ]
    { identifier-9 }
```

```
READ file-name[NEXT] RECORD [INTO identifier]

{ AT END imperative-statement                                          }
{ [{MAJOR KEY IS data-name}]                                           }
{ [{KEY IS data-name      }] INVALID KEY imperative-statement          }
```

```
READ relation-name [NEXT] RECORD

{ AT END imperative-statement                                          }
{ [KEY IS data-name] INVALID KEY imperative-statement                  }
```

```
RELEASE record-name [FROM identifier]
```

```
RETURN file-name RECORD [INTO identifier]

    AT END imperative-statement
```

```
REWRITE [LAST] record-name [FROM identifier]
    INVALID KEY imperative-statement
```

```
SEARCH identifier-1 [ VARYING { index-name } ]
                              { identifier-2 }

    [AT END imperative-statement-1]
```

```
    WHEN condition-1 { imperative-statement-2 }
                     { NEXT SENTENCE          }

  [ WHEN condition-2 { imperative-statement-3 } ] ...
                     { NEXT SENTENCE          }

SEARCH ALL identifier [AT END imperative-statement-1]

    WHEN condition { imperative-statement-2 }
                  { NEXT SENTENCE          }

SEEK file-name RECORD [WITH KEY CONVERSION]

SET { index-name-1  [index-name-2] ... }
    { identifier-1  [identifier-2] ... }

    TO { index-name-3 }
       { identifier-3 }
       { literal      }

SET index-name-1 [index-name-2] ...

    { UP BY   } { identifier }
    { DOWN BY } { literal    }

SKIP { literal   } RECORDS ON file-name
     { data-name }

SORT file-name-1 ON { DESCENDING }
                    { ASCENDING  }

    KEY data-name-1 [data-name-2] ...

  [ ON { DESCENDING } KEY data-name-3 [data-name-4] ... ] ...
       { ASCENDING  }

{ INPUT PROCEDURE IS section-name-1                    }
{   [{ THRU      } section-name-2 ]                    }
{    { THROUGH   }                                    }
{ USING file-name-2                                    }
```

```
{ OUTPUT PROCEDURE IS section-name-3 }
{   [ { THRU      } section-name-4 ]  }
{     { THROUGH   }                 }
{ GIVING file-name-3                }
```

```
START file-name [ KEY { IS EQUAL TO          } data-name ] ]
                      { IS =                 }
                      { IS GREATER THAN      }
                      { IS >                 }
                      { IS NOT LESS THAN     }
                      { IS NOT <             }
    INVALID KEY imperative-statement
```

```
STOP { literal }
     { RUN     }
```

```
SUBTRACT { identifier-1 } [ { identifier-2 } ] ...
         { literal-1    }   { literal-2    }
    FROM identifier-3
    [ROUNDED] [identifier-4[ROUNDED]] ...
    [ON SIZE ERROR imperative-statement]
```

```
SUBTRACT { identifier-1 } [ { identifier-2 } ] ...
         { literal-1    }   { literal-2    }
    FROM { identifier-3 }
         { literal-3    }
    GIVING identifier-4 [ROUNDED]
    [identifier-5 [ROUNDED]] ...
    [ON SIZE ERROR imperative-statement]
```

```
SUBTRACT { CORR          } identifier-1 FROM
         { CORRESPONDING }
    identifier-2 [ROUNDED] [identifier-3 [ROUNDED]] ...
    [ON SIZE ERROR imperative-statement]
```

```
TERMINATE { report-name-1 [report-name-2] ... }
          { ALL                               }
```

```
USE AFTER STANDARD ERROR PROCEDURE ON
        { file-name-1 [file-name-2] ... }
        { INPUT                        }
        { OUTPUT                       }
        { INPUT-OUTPUT                 }
        { I-O                          }
        { EXTEND                       }

USE { BEFORE } STANDARD   [ BEGINNING ]
    { AFTER  }            [ ENDING    ]
          [ REEL ]
          [ FILE ]
          [ UNIT ]

LABEL { PROCEDURE  } ON { file-name-1[file-name-2] ... }
      { PROCEDURES }    { INPUT                       }
                        { OUTPUT                      }
                        { INPUT-OUTPUT                }
                        { I-O                         }

USE BEFORE REPORTING identifier-1 [identifier-2] ...

USE FOR HASHING ON { ALL                       }
                   { file-name-1[file-name-2] ... }

USE FOR DUPLICATE KEY ON { ALL                       }
                         { file-name-1[file-name-2] ... }

USE FOR KEY CONVERSION ON { ALL                       }
                          { file-name-1[file-name-2] ... }

WRITE record-name [FROM identifier-1]

    [ { BEFORE } ADVANCING { identifier-2 LINES } ]
      { AFTER  }           { integer LINES      }
                           { mnemonic-name      }

    [ AT { END-OF-PAGE } imperative-statement ]
         { EOP         }

WRITE [NEXT] record-name [FROM identifier]

    INVALID KEY imperative-statement
```

Parameters are used to select compilation options. All are optional and can be specified in any order. Each is separated from the other by a comma. The list can be enclosed in parentheses (as shown) or it can be separated from the word COBOL by a comma and terminated by a period.

```
COBOL.
COBOL (parameter-list)    [comments]
```

A (Blank Conversion)	A	treats leading blanks as zeros
B (Binary Output)	absent B B = LGO	relocatable binary file on file LGO
	B = fn	binary output on file fn
	B = 0	suppress binary output
BUF (Buffer Size)	BUF	selects buffer size by method of version 3.0 COBOL
C (Copy Default)	C	uses version 3.0 COPY mode; to copy from library, FROM LIBRARY must be specified
D (Execution Abort)	D	prevents execution of program if E diagnostic occurs
DB (Subscript Limit Check)	DB	checks if subscript item within range of OCCURS for that item
DB1 (Paragraph Trace)	DB1	allows a program's flow to be traced by paragraphs. Used in conjunction with trace directives: ENTER 'D.ONTR', ENTER 'D.OFFTR', or ENTER 'D.STPTR'.
E (EDITLIB)	E = prog	generates OVERLAY (COBCODE, 0, 0) statement; selects V parameter

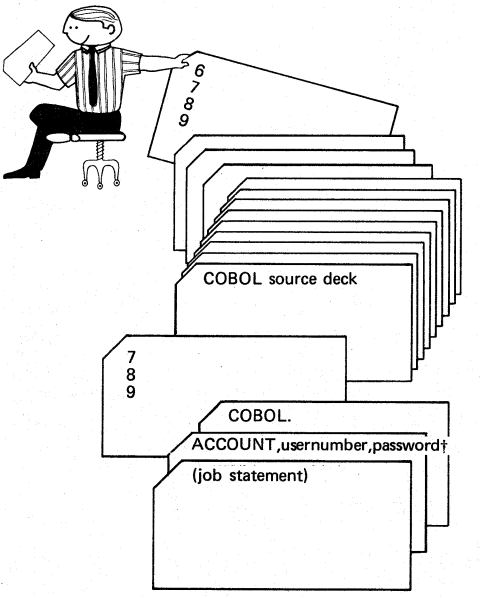
F (Computational Modification)	F	interprets COMPUTATIONAL items as COMPUTATIONAL-1
H (Sort File)	H	allocates buffer area for a sort
I (Source Input)	absent I I=INPUT	source input on file INPUT
	I=fn	source input on file fn
K (Sub-Schema File)	K=fn	sub-schema on fn
	absent K=O	no sub-schema files for this compile
L (List)	absent L L=OUTPUT	normal listing on file OUTPUT
	any combination of following options:	
	L=fn	output on file fn
	L=O	suppress list output
	LX	extended diagnostics
	LR	cross reference pointers
	LC	copy from library
	LO	object code in octal
	LM	data map
N (Non-ANSI Diagnostic)	N	diagnoses any non-ANSI feature
OB (Overlay Binary)	OB=fn	binary output from overlay segments put on file fn
P (ANSI Execution)	P	allows non-ANSI reserved words; selects N parameter
PD (Page Density)	PD=lines-per-inch	indicates number of lines per inch for COBOL output; must be 3, 4, 6, or 8

	absent	installation default is used
	PD	equivalent to PD=8
PS (Page Size)	PS=lines-per-page	indicates number of lines per page for COBOL output
	absent and PD specified	page size is truncated result of calculation: $PS=PS*\frac{\text{installation default for PS}}{\text{installation default for PD}}$
	absent and PD not specified	installation default is used
S (Source Library)	absent S S=COLIB	source library from file COLIB
	S=fn	source library from file fn
SUB (Subcompile)	SUB	suppresses all Data Division binary output except from Working-Storage Section and Constant Section
SUBM (Main Subcompile)	SUBM	indicates non-COBOL main program is loading a COBOL subprogram
T (Tape Sort)	T	sort requests tape sort
U (Alternate Collating Sequence)	U	uses alternate collating sequence
V (Sort Overlay)	V	saves loaded program, using NOGO
W (Initialize Overlays)	W	uses version 3.0 method of treating independent segments: they are available in last used state
Z (3.0 Compatibility)	Z	provides compatibility with version 3.0 COBOL; selects parameters C, and W

COBOL CODING FORMAT

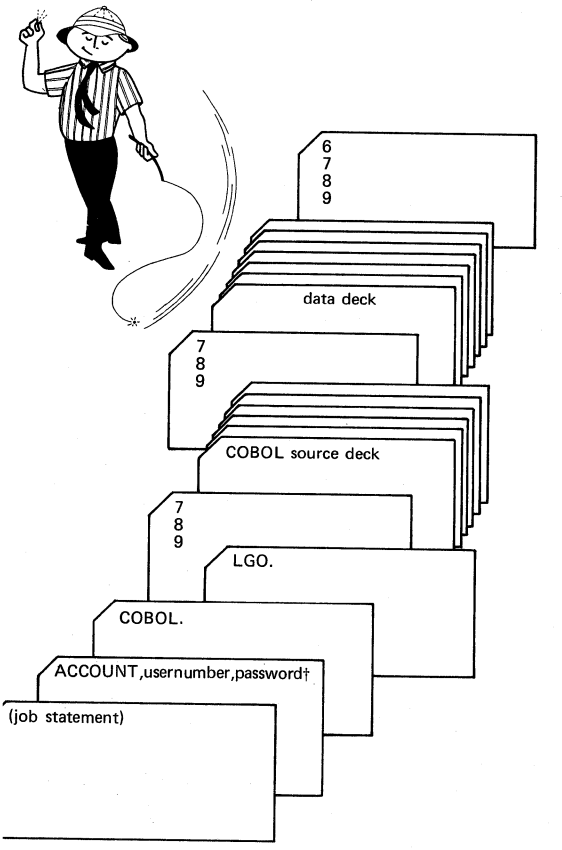
Column	Element
1 - 6	Sequence number
7	Hyphen, slash, or asterisk
8 - 11	Division header Section header Paragraph name Level indicators FD, SD, and RD Level numbers 01 and 77 Keywords DECLARATIVES and END DECLARATIVES
12 - 72	Level numbers 66, 88, and 02 through 49 Statements and clauses Continuation of statements and clauses
73 - 80	Program identification (optional)

Sequence number	Optional, checked by the processor if used.
Hyphen	Continuation line.
Slash	Comment line printed after page ejection.
Asterisk	Comment line printed immediately after preceding line.
Division header	Terminated by period, remainder of line is blank.
Section header	Followed by optional segment number, terminated by period; can only be followed on same line by USE or COPY sentence.
Paragraph name	Terminated by period, followed by at least space before text begins.
Level indicators FS, SD, and RD	Followed by at least one space and file name or report name.
Level numbers 01 and 77	Followed by at least one space and data name.
Keywords DECLARATIVES and END DECLARATIVES	Terminated by period, remainder of line is blank.
Level numbers 66, 88, and 02 through 49	Followed by at least one space and data name.
Statements and clauses	Can begin on a new line or follow a preceding statement or clause; separated by at least one space.



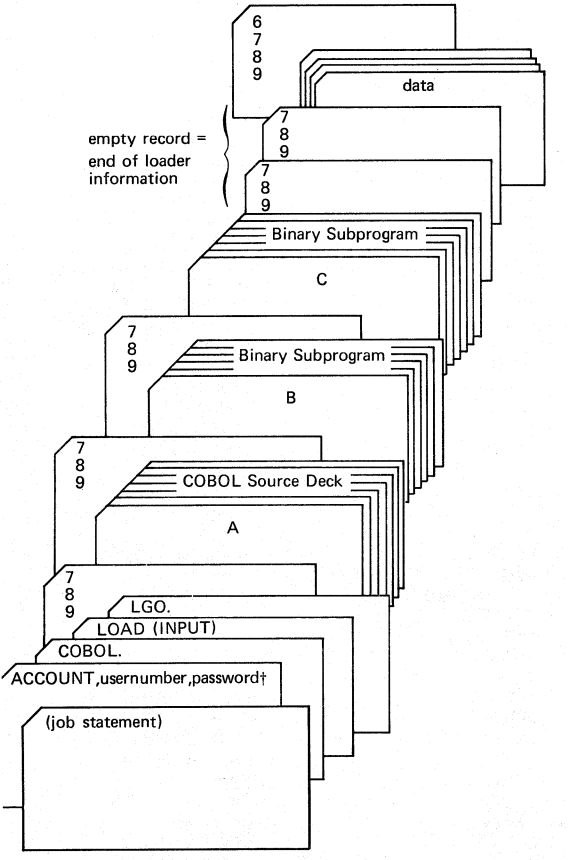
†If required by operating system

COMPILATION AND EXECUTION



† required by operating system

EXECUTION WITH SUBCOMPILED PROGRAM



If required by operating system

COBOL RESERVED WORD LIST

*Indicates word not implemented.

ABOUT
ACCEPT
ACCESS
ACTUAL
ADD
*ADDRESS
ADVANCING
AFTER
ALL
ALPHABETIC
ALPHANUMERIC
ALTER
ALTERNATE
AN
AND
*ANSIB
*APPLY
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR

*BCD
BEFORE
BEGINNING
BEGINNING-FILE-LABEL
BEGINNING-TAPE-LABEL
BINARY
*BITS
BLANK
BLOCK
BLOCKS
BWZ
BY

CALL
*CANCEL
*CD
CF
CH
CHARACTER
CHARACTERS
CHECK
CLASS
CLOCK-UNITS
CLOSE
COBOL
CODE
COLUMN
COMMA
COMMON-STORAGE
COMP
COMP-1
COMP-2
COMP-3
COMPUTATIONAL
COMPUTATIONAL-1
COMPUTATIONAL-2
COMPUTATIONAL-3
COMPUTE
CONFIGURATION
CONSOLE
CONSTANT
CONTAINS
CONTROL
CONTROLS
CONVERSION
COPY
CORR
CORRESPONDING
*COUNT
CREATE
CURRENCY

DATA
DATA-PADDING
DATE
DATE-COMPILED
DATE-WRITTEN
DAY
DE
DECIMAL
DECIMAL-POINT
DECLARATIVES
DELETE
*DELIMITER
*DELIMITED

DENSITY
DEPENDING
*DEPTH
DESCENDING
*DESTINATION
DETAIL
DIGIT
DIGITS
DIRECT
*DISABLE
DISPLAY
DIVIDE
DIVIDED
DIVISION
DOLLAR
DOWN
DUPLICATE
DUPLICATES

EBCDIC

ELSE
*EMI
*ENABLE
END
END-OF-PAGE
ENDING
ENDING-FILE-LABEL
ENDING-TAPE-LABEL
ENDING-TAPE-LABEL-IDENTIFIER
ENTER
ENTRY
ENVIRONMENT
EOP
EQ
EQUAL
EQUALS
ERROR
ERROR-CODE
*ESI
*ETI
EVERY
EXAMINE
EXCEEDS
EXIT
EXPONENTIATED
EXTEND
*EXTERNAL

FD
FILE
FILE-CONTROL
FILE-LABEL
FILE-LIMIT
FILE-LIMITS
FILLER
FINAL
*FIND
FIRST
FLOAT
FOOTING
FOR
*FORMAT
FROM

GENERATE
GIVING
GO
GQ
GR
GREATER
GREATER-EQUAL
GROUP

*HASHED
HASHED-VALUE
HASHING
HEADING
HIGH
HIGH-VALUE
HIGH-VALUES
*HOLD
HYPER

ID
IDENTIFICATION
IF
IN
INCLUDE
INDEX

INDEX-BLOCK
INDEX-LEVEL
INDEX-LEVELS
INDEX-PADDING
INDEXED
INDICATE
INITIATE
INPUT
INPUT-OUTPUT
INSTALLATION
INTO
INVALID
I-O
I-O-CONTROL
IS

JUST
JUSTIFIED

KEY
KEYS

LABEL
LAST
LEADING
LEAVING
LEFT
LESS
LESS-EQUAL
LIBRARY
LIMIT
LIMITS
LINAGE
LINAGE-COUNTER
LINE
LINE-COUNTER
LINES
LINKAGE
LOCATION
LOCK
LOW
LOW-VALUE
LOW-VALUES
*LOWER-BOUND
*LOWER-BOUNDS
LQ
LS

MAJOR
MEMORY
*MESSAGE
MINUS
MODE
MODULES
MOVE
MULTIPLE
MULTIPLIED
MULTIPLY

NEGATIVE
NEXT
NGR
NLS
NO
NOT
NOTE
NQ
NUMBER
NUMERIC

OBJECT-COMPUTER
OCCURS
OF
OFF
OH
OMITTED
ON
OPEN
OPTIONAL
OR
ORGANIZATION
OTHERWISE
OUTPUT
OV
OVERFLOW
*OWNER

PAGE
PAGE-COUNTER
PERCENT
PERFORM
PF
PH
PIC
PICTURE
PLACES
PLUS
POINT
POSITION
POSITIVE
*PREPARED
*PRINT-SWITCH
PRIORITY
PROCEDURE
PROCEDURES
PROCEED
*PROCESS
PROCESSING
PROGRAM
PROGRAM-ID
PROTECT

*QUEUE
QUOTE
QUOTES

RANDOM
RANGE
RD
READ
*RECEIVE
RECORD
RECORD-BLOCK
RECORD-MARK
RECORDING
RECORDS
REDEFINES
REEL

*REFERENCES
RELATIVE
RELEASE

REMAINDER
REMARKS
RENAMES
RENAMING
REPLACING
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RESET
RETURN
REVERSED
REWIND
REWRITE
RF
RH
RIGHT
ROUNDED
RUN

*SA
SAME
SD
SEARCH
SECTION
SECURITY
SEEK
SEGMENT-LIMIT
SELECT
SELECTED
*SEND
SENTENCE
SEQUENCED
SEQUENTIAL
SET
SIGN
SIGNED
SIZE
SKIP
SORT
SOURCE
SOURCE-COMPUTER

SPACE
SPACES
SPECIAL-NAMES
STANDARD
START
STATUS
STOP
*STRING
*SUB-QUEUE-1
*SUB-QUEUE-2
*SUB-QUEUE-3
SUB-SCHEMA
SUBTRACT
SUM
*SUPERVISOR
SUPPRESS
SYMBOLIC
SYNC
SYNCHRONIZED

*TABLE
TALLY
TALLYING
TAPE
TERMINAL
TERMINATE
*TEST
*TEXT
THAN
THEN
THROUGH
THRU
TIME
TIMES
TO
TODAYS-DATE
TYPE

UNEQUAL
UNIT
*UNSTRING
UNTIL
UP
UPON
*UPPER-BOUND
*UPPER-BOUNDS
USAGE
USE
USING

VALUE
VALUES
VARYING
*VOLUME

WHEN
WITH
*WORDS
WORKING-STORAGE
WRITE

ZERO
ZEROES
ZEROS

CDC COLLATING SEQUENCE

Char Set Collating Sequence		COBOL Character	Display Code	Hollerith Punch	
63	64			(026)	(029)
00	00	blank	55	no punch	no punch
01	01	≤*	74	8-5	12-8-4
	02	%	63	8-6	0-8-4
02	03	[*	61	8-7	8-5
03	04	→*	65	0-8-5	0-8-5
04	05	≡*	60	0-8-6	8-3
05	06	∧*	67	0-8-7	12
06	07	↑*	70	11-8-5	8-4
07	08	↓*	71	11-8-6	0-8-7
08	09	>	73	11-8-7	0-8-6
09	10	≥*	75	12-8-5	0-8-2
10	11	¬*	76	12-8-6	11-8-7
11	12	.	57	12-8-3	12-8-3
12	13	)	52	12-8-4	11-8-5
13	14	;	77	12-8-7	11-8-6
14	15	+	45	12	12-8-6
15	16	$	53	11-8-3	11-8-3
16	17	*	47	11-8-4	11-8-4
17	18	-	46	11	11
18	19	/	50	0-1	0-1
19	20	,	56	0-8-3	0-8-3
20	21	(	51	0-8-4	12-8-5
21	22	=	54	8-3	8-6
22	23	≠†	64	8-4	8-7
23	24	<	72	12-0	12-8-2
24	25	A	01	12-1	12-1
25	26	B	02	12-2	12-2
26	27	C	03	12-3	12-3
27	28	D	04	12-4	12-4
28	29	E	05	12-5	12-5
29	30	F	06	12-6	12-6
30	31	G	07	12-7	12-7

*Not in COBOL character set; might be present in data.

†COBOL quote character (") is output on printer as ≠ for some terminals.

CDC COLLATING SEQUENCE (Contd)

Char Set Collating Sequence		COBOL Character	Display Code	Hollerith Punch	
63	64			(026)	(029)
31	32	H	10	12-8	12-8
32	33	I	11	12-9	12-9
33	34	V	66	11-0	11-8-2
34	35	J	12	11-1	11-1
35	36	K	13	11-2	11-2
36	37	L	14	11-3	11-3
37	38	M	15	11-4	11-4
38	39	N	16	11-5	11-5
39	40	O	17	11-6	11-6
40	41	P	20	11-7	11-7
41	42	Q	21	11-8	11-8
42	43	R	22	11-9	11-9
43	44	]††	62	0-8-2	12-8-7
44	45	S	23	0-2	0-2
45	46	T	24	0-3	0-3
46	47	U	25	0-4	0-4
47	48	V	26	0-5	0-5
48	49	W	27	0-6	0-6
49	50	X	30	0-7	0-7
50	51	Y	31	0-8	0-8
51	52	Z	32	0-9	0-9
52	53	:*	00	8-2	8-2
53	54	0	33	0	0
54	55	1	34	1	1
55	56	2	35	2	2
56	57	3	36	3	3
57	58	4	37	4	4
58	59	5	40	5	5
59	60	6	41	6	6
60	61	7	42	7	7
61	62	8	43	8	8
62	63	9	44	9	9

*Not in COBOL character set; might be present in data.
††COBOL record mark.

ASCII COLLATING SEQUENCE

Char Set Collating Sequence		Character	Display Code	Hollerith Punch	
63	64			(026)	(029)
00	00	blank	55	no punch	no punch
01	01	!*	62	0-8-2	11-8-2
02	02	"	64	8-4	8-7
03	03	#	60	0-8-6	8-3
04	04	$	53	11-8-3	11-8-3
	05	%	63	8-6	0-8-4
05	06	&	67	0-8-7	12
06	07	'	61	8-5	12-8-2
07	08	(	51	0-8-4	12-8-5
08	09	)	52	12-8-4	11-8-5
09	10	*	47	11-8-4	11-8-4
10	11	+	45	12	12-8-6
11	12	,	56	0-8-3	0-8-3
12	13	-	46	11	11
13	14	.	57	12-8-3	12-8-3
14	15	/	50	0-1	0-1
15	16	0	33	0	0
16	17	1	34	1	1
17	18	2	35	2	2
18	19	3	36	3	3
19	20	4	37	4	4
20	21	5	40	5	5
21	22	6	41	6	6
22	23	7	42	7	7
23	24	8	43	8	8
24	25	9	44	9	9
25	26	:	00	8-2	8-2
26	27	;	77	12-8-7	11-8-6
27	28	<	74	8-5	8-4
28	29	=	54	8-3	8-6
29	30	>	73	11-8-7	0-8-6
30	31	?	71	11-8-6	0-8-7

ASCII COLLATING SEQUENCE (Contd)

Char Set Collating Sequence		Character	Display Code	Hollerith Punch	
63	64			(026)	(029)
31	32	@	70	11-8-5	8-5
32	33	A	01	12-1	12-1
33	34	B	02	12-2	12-2
34	35	C	03	12-3	12-3
35	36	D	04	12-4	12-4
36	37	E	05	12-5	12-5
37	38	F	06	12-6	12-6
38	39	G	07	12-7	12-7
39	40	H	10	12-8	12-8
40	41	I	11	12-9	12-9
41	42	J	12	11-1	11-1
42	43	K	13	11-2	11-2
43	44	L	14	11-3	11-3
44	45	M	15	11-4	11-4
45	46	N	16	11-5	11-5
46	47	O	17	11-6	11-6
47	48	P	20	11-7	11-7
48	49	Q	21	11-8	11-8
49	50	R	22	11-9	11-9
50	51	S	23	0-2	0-2
51	52	T	24	0-3	0-3
52	53	U	25	0-4	0-4
53	54	V	26	0-5	0-5
54	55	W	27	0-6	0-6
55	56	X	30	0-7	0-7
56	57	Y	31	0-8	0-8
57	58	Z	32	0-9	0-9
58	59	[	72	12-0	12-8-4
59	60	/	75	12-8-5	0-8-2
60	61	]	66	11-0	12-8-7
61	62	^	76	12-8-6	11-8-7
62	63	_	65	0-8-5	0-8-5

CORPORATE HEADQUARTERS, 8100 34th AVE. SO.
MINNEAPOLIS, MINN. 55440

SALES OFFICES AND SERVICE CENTERS
IN MAJOR CITIES THROUGHOUT THE WORLD