

60497000



COBOL VERSION 4  
INSTANT MANUAL

---

CONTROL DATA®  
CYBER 170 SERIES  
CYBER 70 SERIES  
MODELS 72, 73, 74  
6000 SERIES  
COMPUTER SYSTEMS



60497000



---

COBOL VERSION 4  
INSTANT MANUAL

---

CONTROL DATA®  
CYBER 170 SERIES  
CYBER 70 SERIES  
MODELS 72, 73, 74  
6000 SERIES  
COMPUTER SYSTEMS



# INTRODUCTION

This instant outlines the COBOL Version 4.5 language as designed to run under NOS/BE 1 and NOS 1 operating systems for the CONTROL DATA<sup>®</sup> CYBER 170, CYBER 70 Model 72, 73, 74, and 6000 Series Computer Systems.

COBOL Version 4.5 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.

## 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

## 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.

{ } Only one element must be selected

... Repeat preceding bracketed material as needed.

{ } ... Entire phrase may be repeated.

COBOL words have preassigned meanings and appear in capitals.

COBOL words not underlined may 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	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	Paragraphs may be included in sections introduced by section name.
Section Name	Word followed by SECTION and terminated by period; user defined in Procedure Division, pre-defined in Identification, Environment, and Data Divisions.
Entry	Unit of description in Data Division; must be terminated by period.

## IDENTIFICATION DIVISION

{ ID } DIVISION  
{ IDENTIFICATION }

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.<sup>†</sup>

format 1:

SOURCE-COMPUTER. 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:

SOURCE-COMPUTER. computer-name.

format 1:

OBJECT-COMPUTER. 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 } ] ... ]

<sup>†</sup> 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  
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:

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 mnemonic-name].

INPUT-OUTPUT SECTION.

format 1:

FILE-CONTROL. COPY library-name

$$\left[ \begin{array}{l} \text{REPLACING} \left\{ \begin{array}{l} \text{literal-1} \\ \text{word-1} \\ \text{identifier-1} \end{array} \right\} \text{ BY } \left\{ \begin{array}{l} \text{literal-2} \\ \text{word-2} \\ \text{identifier-2} \end{array} \right\} \\ \\ \left\{ \begin{array}{l} \text{literal-3} \\ \text{word-3} \\ \text{identifier-3} \end{array} \right\} \text{ BY } \left\{ \begin{array}{l} \text{literal-4} \\ \text{word-4} \\ \text{identifier-4} \end{array} \right\} \dots \end{array} \right].$$

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 } { UNIT } ] [ ERROR FILE IS file-name]

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

$$\left[ \left\{ \begin{array}{l} \text{FILE-LIMIT IS} \left\{ \begin{array}{l} \text{data-name-1} \\ \text{literal-1} \end{array} \right\} \\ \text{FILE-LIMITS ARE} \left\{ \begin{array}{l} \text{data-name-2} \\ \text{literal-2} \end{array} \right\} \\ \\ \left[ \left\{ \begin{array}{l} \text{data-name-3} \\ \text{literal-3} \end{array} \right\} \left\{ \begin{array}{l} \text{THRU} \\ \text{THROUGH} \end{array} \right\} \left\{ \begin{array}{l} \text{data-name-4} \\ \text{literal-4} \end{array} \right\} \right] \dots \\ \\ \text{FILE-LIMIT IS} \left\{ \begin{array}{l} \text{data-name-5} \\ \text{literal-5} \end{array} \right\} \end{array} \right\} \right]$$

[ ORGANIZATION IS { SEQUENTIAL  
STANDARD  
DIRECT  
INDEXED SEQUENTIAL  
RELATIVE  
ACTUAL KEY } ]

[ ACCESS MODE IS { SEQUENTIAL }  
RANDOM ]

[ PROCESSING MODE IS SEQUENTIAL ]

{ ACTUAL KEY IS data-name-1  
SYMBOLIC KEY IS data-name-2 [WITH DUPLICATES]  
RECORD KEY IS data-name-3 [WITH DUPLICATES] }

[ ALTERNATE RECORD KEY IS data-name-3  
[WITH DUPLICATES [INDEXED]] ]

[ NUMBER OF BLOCKS IS { data-name }  
integer ]

[ { INDEX-LEVEL IS  
INDEX-LEVELS ARE } integer ]

[ INDEX-BLOCK CONTAINS integer CHARACTERS ]

[ RECORD-BLOCK CONTAINS integer { RECORDS  
CHARACTERS } ]

[ INDEX-PADDING IS integer PERCENT ]

[ DATA-PADDING IS integer PERCENT ] . } ...

format 1:

I-O-CONTROL. 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:

I-O CONTROL.

[ RERUN [ ON { file-name-1  
          { implementor-name } } ] ]

EVERY { { [ END OF ] { REEL }  
          { UNIT } } OF file-name-2 }  
      { integer-1 RECORDS }  
      { integer-2 CLOCK-UNITS }  
      { condition-name } } ...

[ { 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)
- ,
- / Slash (instead of comma)
- .
- 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

# DATA SPECIFICATIONS

	File Section			Common and Working Storage Sections				Constant Section			
	01	g r o u p	e l e m	77	01	g r o u p	e l e m	77	01	g r o u p	e l e m
REDEFINES	I										
SIZE			R	R			R	R			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 may 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)

A Sort File Description (SD) entry may contain only DATA RECORD, RECORD CONTAINS, and FILE CONTAINS clauses; any or all may be omitted from an SD entry.

format 1:

$$\left\{ \begin{array}{l} \underline{SD} \\ \underline{FD} \end{array} \right\} \text{file-name } \underline{COPY} \text{ library-name}$$
$$\left[ \begin{array}{l} \underline{REPLACING} \left\{ \begin{array}{l} \text{literal-1} \\ \text{word-1} \\ \text{identifier-1} \end{array} \right\} \underline{BY} \left\{ \begin{array}{l} \text{literal-2} \\ \text{word-2} \\ \text{identifier-2} \end{array} \right\} \\ \left\{ \begin{array}{l} \text{literal-3} \\ \text{word-3} \\ \text{identifier-3} \end{array} \right\} \underline{BY} \left\{ \begin{array}{l} \text{literal-4} \\ \text{word-4} \\ \text{identifier-4} \end{array} \right\} \dots \end{array} \right].$$

format 2:

$$\left\{ \begin{array}{l} \underline{SD} \\ \underline{FD} \end{array} \right\} \text{file-name}$$
$$\left[ \underline{BLOCK CONTAINS} \text{ [integer-1 } \underline{TO}] \text{ integer-2 } \left\{ \begin{array}{l} \underline{RECORDS} \\ \underline{CHARACTERS} \end{array} \right\} \right]$$
$$\left[ \underline{DATA} \left\{ \begin{array}{l} \underline{RECORD IS} \\ \underline{RECORDS ARE} \end{array} \right\} \text{data-name-1 [data-name-2] ...} \right]$$
$$\left[ \left\{ \begin{array}{l} \underline{REPORT IS} \\ \underline{REPORTS ARE} \end{array} \right\} \text{report-name-1 [report-name-2] ...} \right]$$

[FILE CONTAINS ABOUT integer RECORDS]

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

If label records are STANDARD:

[ VALUE OF [ { ID  
IDENTIFICATION } IS { literal-1  
data-name-1 } ] ]  
[ DATE-WRITTEN IS { literal-2  
data-name-2 } ]  
[ EDITION-NUMBER IS { literal-3  
data-name-3 } ]  
[ REEL-NUMBER IS { literal-4  
data-name-4 } ]  
[ RETENTION-CYCLE IS { literal-5  
data-name-5 } ] ]

If label records are a data-name:

[ VALUE OF data-name-3 IS { literal-1  
data-name-4 } ]  
[ data-name-5 IS { literal-2  
data-name-6 } ] ... ]

[ VALUE OF ENDING-TAPE-LABEL-IDENTIFIER  
IS { literal-3  
data-name-7 } ] ]

[ LINAGE IS { integer  
identifier } LINES ] ]

[ RECORD CONTAINS [integer-1 TO] integer-2 CHARACTERS ]

[ DEPENDING ON { RECORD-MARK  
data-name-1 } ] ]

[ RECORDING MODE IS [ { BINARY  
DECIMAL } ] [ { HIGH  
LOW } DENSITY ] ]  
[ { EBCDIC } ] ]

[ SEQUENCED ON data-name-1 [data-name-2] ... ] ]



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

format 1:

{ 01 } data-name COPY library-name [FROM LIBRARY]  
 { 02-49 }

[ 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-1 [REDEFINES identifier] }  
 { FILLER }

[ { BWZ } ]  
 { BLANK WHEN ZERO }

[ { CHECK PROTECT }  
 { FLOAT DOLLAR SIGN } [LEAVING integer PLACES]  
 { FLOAT CURRENCY SIGN }  
 { 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] ... ] ...  
 [INDEXED BY index-name-1 [index-name-2] ...] ]

[ { PIC  
 PICTURE } IS character-string ]

[ POINT LOCATION IS { LEFT  
 RIGHT } integer PLACES ]

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

[ { SIGNED  
 SIGN IS data-name } ]

[ SIZE IS integer [ { CHARACTERS  
 DIGITS } ] ]

[ { SYNC  
 SYNCHRONIZED } { LEFT  
 RIGHT } ]

[ [USAGE IS] { COMP  
 COMPUTATIONAL  
 COMP-1  
 COMPUTATIONAL-1  
 COMP-2  
 COMPUTATIONAL-2  
 DISPLAY  
 INDEX } ]

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

[ PAGE { LIMIT IS  
LIMITS ARE } integer-1 { LINE  
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  
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:

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 } ]

[ NEXT GROUP IS { integer-1  
                          PLUS integer-2  
                          NEXT PAGE } ]

[ SIZE IS integer { CHARACTERS  
                          DIGITS } ]

TYPE IS { REPORT HEADING  
                  RH  
                  PAGE HEADING  
                  PH  
                  OVERFLOW HEADING  
                  OH  
                  { CONTROL HEADING } { identifier-1 }  
                  { CH } { FINAL }  
                  DETAIL  
                  DE  
                  { CONTROL FOOTING } { identifier-2 }  
                  { CF } { FINAL }  
                  OVERFLOW FOOTING  
                  OV  
                  PAGE FOOTING  
                  PF  
                  REPORT FOOTING  
                  RF }

[ USAGE IS DISPLAY ].

## Report Element Description (Report Section Only)

TYPE clause allowed if level 01

NEXT GROUP clause allowed if level 01

level-number [data-name]

[ { BLANK WHEN ZERO } ]  
[ { BWZ } ]

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

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

[ COLUMN NUMBER IS integer ]

[ GROUP INDICATE ]

[ { JUSTIFIED } RIGHT ]  
[ { JUST } ]

[ 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	255 characters/digits
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

Rec. Field Source 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- Edit	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



## 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 } ] ... TO  
      { literal-1 } [ { literal-2 } ]

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  
      { CORRESPONDING } } identifier-1

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 [language-name] routine-name

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 } library-name [FROM LIBRARY]  
{ INCLUDE }

[ 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 } ] ... ]

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-1 literal-1 formula-1 }	IS [NOT]	{ GREATER THAN GR > LESS THAN LS < GREATER-EQUAL TO GO LESS-EQUAL TO LO EQUAL TO EQUAL TO EQ = IS UNEQUAL TO EQUALS EXCEEDS IS NO IS NGR IS NLS	{ identifier-2 literal-2 formula-2 }

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

identifier IS [NOT] { NUMERIC  
ALPHABETIC }

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

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

MOVE { { CORR  
CORRESPONDING } identifier-1 }  
literal-1  
identifier-1 } TO

identifier-2[identifier-3] ...

MULTIPLY { identifier-1 } BY identifier-2 [ROUNDED]

[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.

OPEN { EXTEND file-name-1 [file-name-2] ...

{ INPUT file-name-1 [ { REVERSED } ]

[ file-name-2 [ { REVERSED } ] ] ...

{ OUTPUT file-name-1 [ WITH NO REWIND ]

[file-name-2 [ WITH NO REWIND ] ] ...

{ INPUT-OUTPUT } file-name-1 [file-name-2] ...

{ I-O }

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

[ THROUGH ]

[ { { identifier } TIMES } ]

[ { integer } ]

[ UNTIL condition ]

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

VARYING { index-name-1 } FROM { literal-1  
index-name-2 } BY  
identifier-1 } 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-5 } identifier-6 }

[ AFTER { index-name-5 } FROM { literal-5 }  
identifier-7 } index-name-6 } BY  
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 ] INVALID KEY imperative-statement }  
[ KEY IS data-name ]

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 }  
      { data-name } RECORDS ON file-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 } section-name-2 ] }  
  { USING file-name-2 }

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

START file-name KEY  $\left\{ \begin{array}{l} \text{IS EQUAL TO} \\ \text{IS } = \\ \text{IS GREATER THAN} \\ \text{IS } > \\ \text{IS NOT LESS THAN} \\ \text{IS NOT } < \end{array} \right\}$  data-name

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 ]

# COBOL CONTROL STATEMENT

Parameters are used to select compilation options. All are optional and may be specified in any order. Each is separated from the other by a comma. The list may be enclosed in parentheses (as shown) or it may 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	} relocatable binary file on file LGO
	B	
	B = 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, O, O) 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	INPUT assumed
	I=fn	
K (Sub-Schema File)	K=fn	sub-schema on file fn
	absent K=O	no sub-schema files for this compile
L (List)	absent L L=OUTPUT	normal listing on OUTPUT
	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
S (Source Library)	absent S S=COLIB	source library from file COLIB
	S=fn	

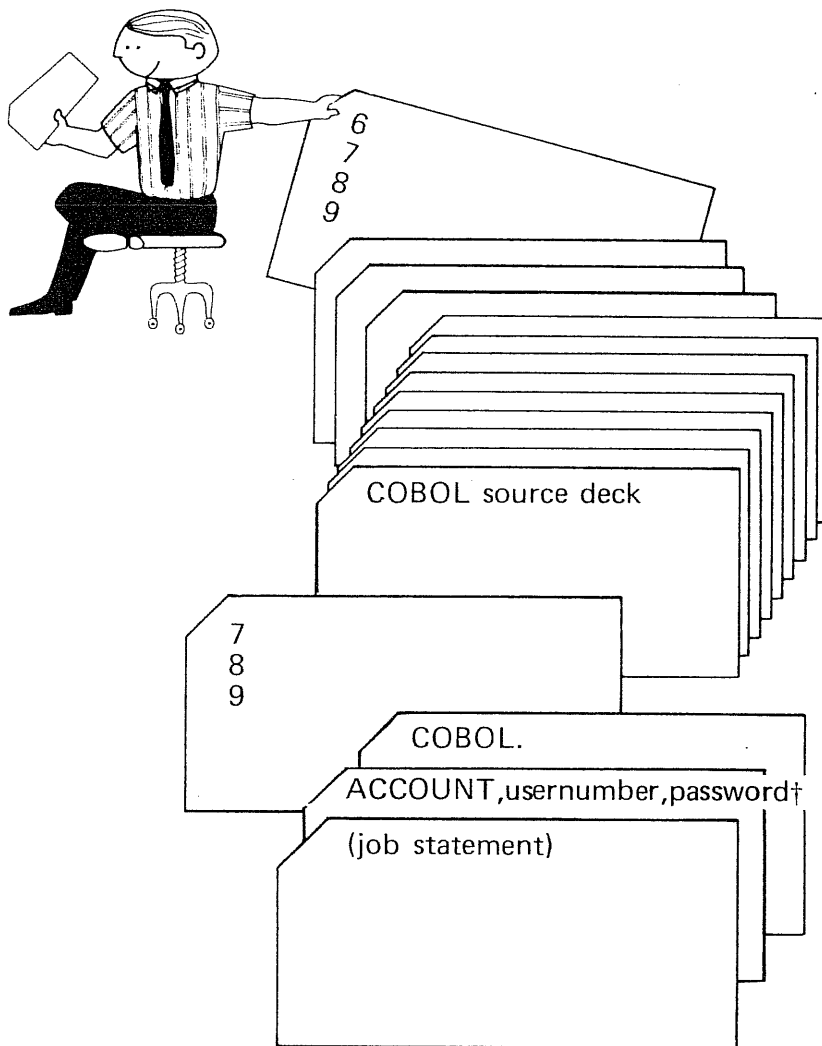
SUB (Subcompile)	SUB	suppresses all Data Division binary output except from Working and Constant Storage
SUBM (Main Subcompile)	SUB	indicates non-COBOL main program is loading a COBOL subprogram
T (Tape Sort)	T	sort requests tape sort
U (Alternate Collating)	U	uses alternate collating sequence
V (Sort 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	Division name Section name Paragraph name File description Record description level number
12	Record description data name First sentence of a paragraph File name Continuation of a data description or a sentence
73 – 80	Identification (optional)

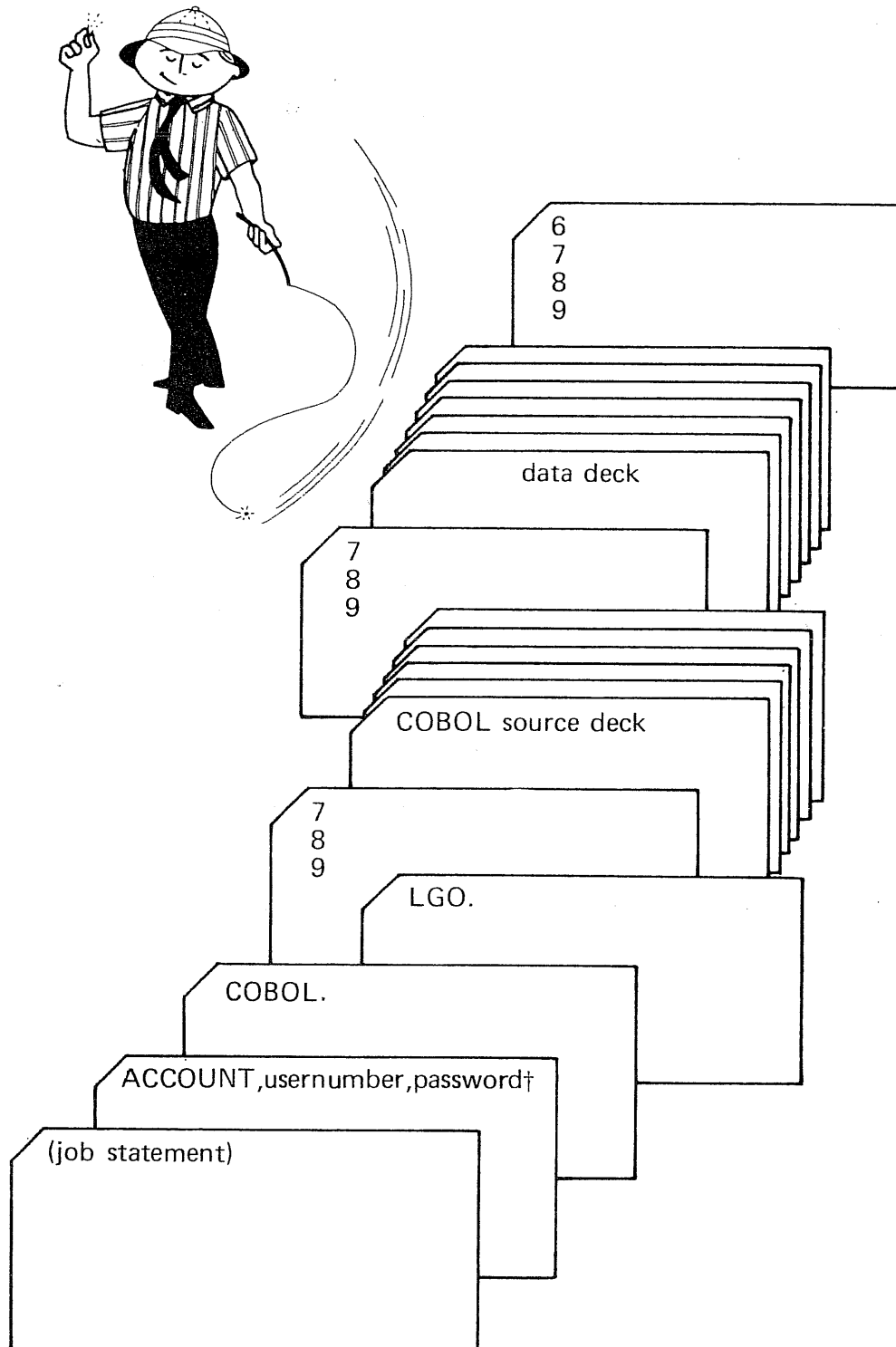
Sequence number	Optional, checked by the processor if used
Hyphen	Indicates continuation of a word or literal from the preceding line
Slash or asterisk	Remainder of line is treated as comment and skips to new page
Division name	Terminated by period, remainder of line is blank
Section name	Followed by optional priority number, terminated by period, remainder is blank
Paragraph name	Terminated by period, followed by at least one blank before text begins
File Description	FD or SD followed by file name and at least one blank
Record Description	Level number followed by at least one blank and data name
First Sentence	Begins in or after column 12. Spaces may be used freely to avoid splitting a word or literal. If a word or literal is split, a hyphen must appear in column 7 of the next line.

# COBOL COMPILATION



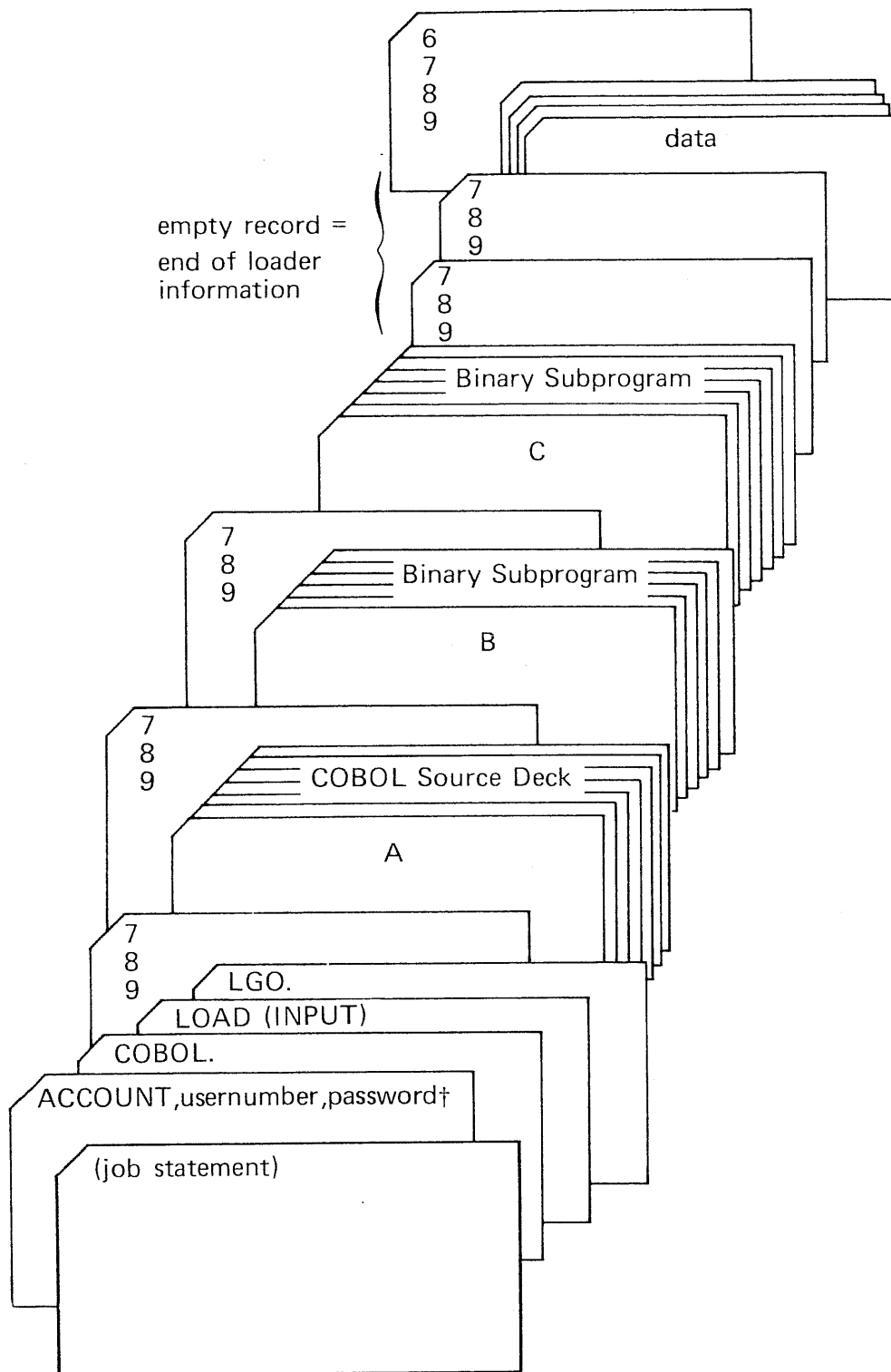
†If required by operating system

# COMPILATION AND EXECUTIO



†If required by operating system

# EXECUTION WITH SUBCOMPILED PROGRAM



† If required by operating system



# COBOL RESERVED WORD LIST

\*Indicates word not implemented.

ABOUT	CHARACTERS
ACCEPT	CHECK
ACCESS	CLASS
ACTUAL	CLOCK-UNITS
ADD	CLOSE
*ADDRESS	COBOL
ADVANCING	CODE
AFTER	COLUMN
ALL	COMMA
ALPHABETIC	COMMON-STORAGE
ALPHANUMERIC	COMP
ALTER	COMP-1
ALTERNATE	COMP-2
AN	COMP-3
AND	COMPUTATIONAL
*ANSIB	COMPUTATIONAL-1
*APPLY	COMPUTATIONAL-2
ARE	COMPUTATIONAL-3
AREA	COMPUTE
AREAS	CONFIGURATION
ASCENDING	CONSOLE
ASSIGN	CONSTANT
AT	CONTAINS
AUTHOR	CONTROL
	CONTROLS
	CONVERSION
*BCD	COPY
BEFORE	CORR
BEGINNING	CORRESPONDING
BEGINNING-FILE-LABEL	*COUNT
BEGINNING-TAPE-LABEL	CREATE
BINARY	CURRENCY
*BITS	
BLANK	DATA
BLOCK	DATA-PADDING
BLOCKS	DATE
BWZ	DATE-COMPILED
BY	DATE-WRITTEN
	DAY
CALL	DE
CANCEL	DECIMAL
*CD	DECIMAL-POINT
CF	DECLARATIVES
CH	DELETE
CHARACTER	*DELIMITER
	*DELIMITED

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

EBCDIC  
EDITION-NUMBER  
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

GO

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  
REEL-NUMBER  
\*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  
ZEROS  
ZEROS

# 64-CHARACTER SET COLLATING SEQUENCE

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

\*Not in COBOL character set; may be present in data

†COBOL quote character (") is output on printer as ≠

# 64-CHARACTER SET COLLATING SEQUENCE (cont)

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

\*Not in COBOL character set

††COBOL record mark

# ASCII COLLATING SEQUENCE

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



## ASCII COLLATING SEQUENCE (cont)

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

# 63-CHARACTER SET COLLATING SEQUENCE

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

\*Not in COBOL character set; may be present in data  
 †COBOL quote character (") is output on printer as ≠

# 63-CHARACTER SET COLLATING SEQUENCE (cont)

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

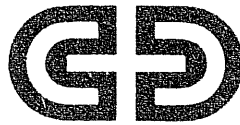
\*Not in COBOL character set  
††COBOL record mark

NOTES





CONTROL DATA  
CORPORATION



---

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

SALES OFFICES AND SERVICE CENTERS  
IN MAJOR CITIES THROUGHOUT THE WORLD