COBOL
POCKET REFERENCE

Version 6 Cobol
Version 7 Libol
CONVENTIONS AND EXPRESSIONS

CONVENTIONS

BRACES () - indicate that a choice must be made from one of the enclosed lines.

BRACKETS [] - indicate an optional feature.

ELLIPSIS ... - indicates that the information contained within the preceding pair of braces or brackets can be repeated at the programmer's option.

LOWER CASE CHARACTERS - values, names, and other parameters supplied by the user.

UPPER CASE CHARACTERS (UNDERSCORED) - key words in the COBOL lexicon that must be used when the formats of which they are a part are used.

UPPER CASE CHARACTERS (NOT UNDERSCORED) - optional key words in the COBOL lexicon that serve only to make a COBOL statement more readable.

EXPRESSIONS

Arithmetic expressions - the following symbols:

+ addition
- subtraction
* multiplication
/ division
** exponentiation
† exponentiation

Relational operators -

<table>
<thead>
<tr>
<th>Relational operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS [NOT] GREATER THAN</td>
<td>Greater than, not greater than</td>
</tr>
<tr>
<td>IS [NOT] ≥ THAN</td>
<td></td>
</tr>
<tr>
<td>IS [NOT] LESS THAN</td>
<td>Less than, not less than</td>
</tr>
<tr>
<td>IS [NOT] ≤ THAN</td>
<td></td>
</tr>
<tr>
<td>IS [NOT] EQUAL (EQUALS) TO</td>
<td>Equal to, not equal to</td>
</tr>
<tr>
<td>IS [NOT] = TO</td>
<td>MEANING</td>
</tr>
</tbody>
</table>

LOGICAL OPERATORS

OR

AND

Entire condition is true if either or both of the simple conditions are true.

Entire condition is true if both of the simple conditions are true.
A line in a COBOL source program is defined to be a string of adjacent character positions terminated by the first occurrence of a line termination character.

There are two source program formats which are acceptable to DECSYSTEM-10 COBOL: conventional and standard. Unless the "/S" switch is included in the command string or the special sequence numbers created by LINED are detected, the compiler will assume that the source program is written in the standard format.

**CONVENTIONAL FORMAT**

Margin L designates the leftmost (first) character position of a line.

The continuation column designates the seventh character position relative to the left margin.

Margin A designates the eighth character position relative to the left margin.

Margin B designates the twelfth character position relative to the left margin.

The identification column designates the seventy-third character position relative to the left margin.

Margin R designates the rightmost (eightieth) character position of a line.

The sequence number area is a six-character field beginning at margin L that normally contains a sequence number. The compiler ignores this field.

The continuation area occupies one character position in the continuation column.

Area A occupies four character positions beginning at margin A. All division-names, section-names, and paragraph-names must begin in area A. In the DATA DIVISION, the FD entry must begin in area A and level-number entries can begin in area A, but are not required to.

Area B occupies 61 character positions beginning at margin B and ending at column 72. All remaining entries begin in area B.

The identification area occupies eight character positions beginning at the identification column and ending at margin R.
COLUMN 0

MARGIN A

0 1 2 3 4

CONTINUATION
AREA

MARGIN B

5 6

AREA A

MARGIN R

AREA B

Column 0 designates a character position that is not counted by the compiler. It is only used for comment or continuation.

Margin A designates the first character position.

Margin B designates the fifth character position relative to Margin A (not column 0). To reach margin B, the user should type horizontal tab.

Margin R designates the rightmost character position of a line.

The continuation area occupies one character position in column 0.

Area A occupies four character positions beginning at margin A. All division-names, section-names, and paragraph-names must begin in area A. In the DATA DIVISION, the FD entry must begin in area A and level-number entries can begin in area A, but are not required to begin there.

Area B occupies up to 101 character positions, beginning at margin B. All remaining entries begin in area B. On an interactive terminal, the user can reach margin B by typing horizontal-tab anywhere in area A (or in column 0). Area B is terminated by a line-feed, form feed or vertical tab usually preceded by a carriage return.

PROGRAM STRUCTURE

IDENTIFICATION DIVISION

General structure

IDENTIFICATION DIVISION.

[PROGRAM-ID., [program-name] [comment paragraph .]]

[AUTHOR, comment paragraph .]

[INSTALLATION, comment paragraph .]

[DATE-WRITTEN, comment paragraph .]

[DATE-compiled, comment paragraph .]

[SECURITY, comment paragraph .]

[REMARKS, comment paragraph .]
ENVIRONMENT DIVISION

CONFIGURATION SECTION.

SOURCE-COMPUTER. (comment-paragraph).

OBJECT-COMPUTER. (DECSYSTEM-10)

PDP-10

(MEMORY SIZE integer-1

{CHARACTERS

WORDS

MODULES

(SEgment-Limit IS integer-2).

SPECIAL-NAMES. (CONSOLE IS mnemonic-name-1)

CHANNEL (m) IS mnemonic-name-2

[CHANNEL (n) IS mnemonic-name-3]...

IS mnemonic-name-4

[ON STATUS IS condition-name-1]

[OFF STATUS IS condition-name-2]

SWITCH (m)

{ON STATUS IS condition-name-1

[OFF STATUS IS condition-name-2]

OFF STATUS IS condition-name-2

[ON STATUS IS condition-name-1]

SWITCH (n) ....]...

[literal-1 IS mnemonic-name-5]

[CURRENCY SIGN IS literal-2]

[DECIMAL-POINT IS COMMA]

INPUT-OUTPUT SECTION.

FILE-CONTROL. SELECT [OPTIONAL] file-name

ASSIGN TO device-name-1 [,device-name-2]...

[FOR MULTIPLE {REEL

UNIT} ]
RESERVE \{ integer-2 \} ALTERNATE \{ AREA AREAS \}

FILE LIMIT IS \{ FILE-LIMIT IS \{ FILE-LIMITS ARE \}
\{ data-name-1 \} \{ literal-1 \} THRU \{ data-name-2 \} \{ literal-2 \}
\{ data-name-3 \} \{ literal-3 \} THRU \{ data-name-4 \} \{ literal-4 \} \ldots

ACCESS MODE IS \{ SEQUENTIAL RANDOM INDEXED DEFERRED OUTPUT \}

[PROCESSING MODE IS SEQUENTIAL]

[ACTUAL KEY IS data-name-5]

[SYMBOLIC KEY IS data-name-6,]

RECORD KEY IS data-name-7

RECORDING [MODE IS \{ ASCII SIXBIT BINARY \}]

[DENSITY IS \{ 200 556 800 \}] [PARITY IS \{ ODD EVEN \}]

SELECT \ldots \ldots

I-O-CONTROL.

RERUN EVERY \{ END OF \{ REEL UNIT RECORDS \} \} OF file-name-1

SAME \{ RECORD SORT \} AREA FOR file-name-2,
file-name-3, [\{ file-name-4 \} \ldots]

MULTIPLE FILE TAPE CONTAINS file-name-5

[POSITION integer-2]

file-name-6 [POSITION integer-3] \ldots
DATA DIVISION

General Structure

DATA DIVISION, 
FILE SECTION, 
WORKING-STORAGE SECTION, 
LINKAGE SECTION, 
COMMUNICATION SECTION, 
REPORT SECTION.

DATA DIVISION STATEMENTS

FILE DESCRIPTION

FORM:

FD file-name

[ BLOCK CONTAINS [integer-1 TO integer-2] RECORDS ARE [ RECORDS ARE ] ]

[ RECORD CONTAINS [integer-3 TO integer-4] CHARACTERS ]

[ LABEL RECORD IS OMITTED ]

[ STANDARD record-name-1 [, record-name-2] ... ]

[ REPORT IS REPORTS ARE report-name-1 [, report-name-2] ... ]

[ VALUE OF ID IDENTIFICATION IS data-name-1, literal-1 ]

[ DATE-WRITTEN IS data-name-2, literal-2 ]

[ USER-NUMBER IS data-name-3, literal-3, literal-4 ]

[ DATA RECORD IS RECORDS ARE record-name-3 [, record-name-4] ... ]

BLOCK CONTAINS

FORM:

[ BLOCK CONTAINS [integer-1 TO integer-2] RECORD(S) CHARACTERS ]

DATA RECORD

FORM:

[ DATA RECORD IS RECORDS ARE record-name-1 [, record-name-2] ... ]
FD filename
FORM:

FD filename

LABEL
FORM:

\[
\text{LABEL} \left\{ \begin{array}{l}
\text{RECORD IS} \\
\text{RECORDS ARE}
\end{array} \right. \left\{ \begin{array}{l}
\text{OMMITTED} \\
\text{STANDARD} \\
\text{record-name-1} \\
\text{,record-name-2} \\
\ldots
\end{array} \right. \]
\]

RECORD CONTAINS
FORM:

\[
\text{RECORD} \text{ CONTAINS} \left[ \text{integer-1 TO } \right. \text{integer-2 CHARACTERS} \right]
\]

REPORT
FORM:

\[
\left\{ \begin{array}{l}
\text{REPORT IS} \\
\text{REPORTS ARE}
\end{array} \right. \left[ \begin{array}{l}
\text{report-name-1} \\
\text{,report-name-2} \\
\ldots
\end{array} \right]
\]

SD file-name
FORM:

\[
\text{SD} \text{ file-name} \left\{ \begin{array}{l}
\text{DATA} \\
\text{RECORD IS} \\
\text{RECORDS ARE}
\end{array} \right. \left[ \begin{array}{l}
\text{record-name-1} \\
\text{,record-name-2} \\
\ldots
\end{array} \right]
\]

\[
\text{RECORD CONTAINS} \left[ \text{integer-1 TO } \right. \text{integer-2 CHARACTERS} \right]
\]

VALUE OF
FORM:

\[
\left\{ \begin{array}{l}
\text{VALUE OF} \\
\text{ID}\text{ENTIFICATION IS}
\end{array} \right. \left[ \begin{array}{l}
\text{data-name-1} \\
\text{literal-1}
\end{array} \right]
\]

\[
\text{DATE-WRITTEN IS} \left[ \begin{array}{l}
\text{data-name-2} \\
\text{literal-2}
\end{array} \right]
\]

\[
\text{USER-NUMBER IS} \left[ \begin{array}{l}
\text{data-name-3} \\
\text{literal-3, literal-4}
\end{array} \right]
\]

DATA DESCRIPTION ENTRY
FORM:

\[
\begin{array}{l}
\text{level-number} \left[ \begin{array}{l}
\text{data-name-1} \\
\text{FILLER}
\end{array} \right]
\end{array}
\]

\[
\text{REDEFINES} \text{ data-name-2}
\]
(PICTURE) IS picture-string

COMPUTATIONAL
COMP
COMPUTATIONAL-1
COMP-1
DISPLAY
DISPLAY-6
DISPLAY-7
INDEX
DATA BASE-KEY

SYNCHRONIZED
SYNC

LEFT

JUSTIFIED
JUST

RIGHT

BLANK WHEN ZERO

VALUE IS literal-1

OCCURS [integer-1 TO] integer-2 TIMES

DEPENDING ON data-name-1

ASCENDING

DESCENDING

KEY is data-name-2 [,data-name-3] ...]

INDEXED BY index-name-1 [,index-name-2] ...

RENAME ENTRY

66 data-name-1 RENAMES data-name-2 (THRU data-name-3)

CONDITION NAME ENTRY

88 condition-name

VALUE IS
VALUES ARE literal-1 [THRU literal-2]

, literal-3 [THRU literal-4] ...}

RECORD DESCRIPTIONS
FORM:

01 data-name
BLANK WHEN ZERO
FORM:

[BLANK WHEN ZERO]

CONDITION-NAME (Level-88)
FORM:

88 condition-name \( \{ \text{VALUE IS} \) \text{VALUES ARE} \) literal-1 \( \text{THRU} \) literal-2

\( \), literal-3 \( \text{THRU} \) literal-4 \( \quad \ldots \quad \)

DATA-NAME/FILLER
FORM:

level-number \( \{ \text{data-name} \) FILLER \}

JUSTIFIED
FORM:

\( \{ \text{JUSTIFIED} \) \( \text{RIGHT} \) \}

LEVEL-NUMBER
FORM:

level-number \( \{ \text{data-name} \) FILLER \}

OCCURS
FORM:

[OCCURS (integer-1 TO integer-2 TIMES

[DEPENDING ON data-name-1]

[ASCENDING DESCENDING] KEY IS data-name-2 \( \), data-name-3 \} \ldots \}

INDEXED BY index-name-1 \( \), index-name-2 \} \ldots \}

PICTURE
FORM:

[PICTURE PIC \} IS picture-string \}
REDEFINES
FORM:
level-number data-name-1 REDEFINES data-name-2

RENUMES
FORM:
66 data-name-1 RENUMES data-name-2 [THRU data-name-3]

SYNCHRONIZED
FORM:
\[
\begin{align*}
\text{SYNCHRONIZED} & \quad \text{LEFT} \\
\text{SYNC} & \quad \text{RIGHT}
\end{align*}
\]

USAGE
FORM:
\[
\begin{align*}
\text{USAGE IS}\{ \\
\text{COMP} \\
\text{COMPUTATIONAL-1} \\
\text{COMP-1} \\
\text{DISPLAY} \\
\text{DISPLAY-6} \\
\text{DISPLAY-7} \\
\text{INDEX} \\
\text{DATABASE-KEY}
\end{align*}
\]

VALUE
FORM:
\[
\begin{align*}
\text{FORMAT 1:} & \quad \text{VALUE IS literal} \\
\text{FORMAT 2:} & \quad \text{VALUES ARE} \quad \text{literal-1 [THRU literal-2]} \\
& \quad \text{, literal-3 [THRU literal-4]} \quad \ldots
\end{align*}
\]

REPORT SECTION

REPORT DESCRIPTION (RD)
FORM:
RD report-name
\[
\begin{align*}
\text{CODE mnemonic-name} & \\
\text{CONTROL IS} & \quad \text{FINAL identifier-1 [, identifier-2] \ldots} \\
\text{CONTROLS ARE} & \quad \text{FINAL, identifier-1 [, identifier-2] \ldots}
\end{align*}
\]
[PAGE {LIMIT IS LIMITS ARE} integer-1 {LINE LINES}

[HEADING integer-2] [FIRST DETAIL integer-3]

[LAST DETAIL integer-4] [FOOTING integer-5] ]

CONTROL FORM:

{CONTROL IS CONTROLS ARE}

FINAL identifier-1 [, identifier-2] ...

FINAL, identifier-1 [, identifier-2] ...

PAGE-LIMIT FORM:

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

[HEADING integer-2] [FIRST DETAIL integer-3]

[LAST DETAIL integer-4] [FOOTING integer-5] ]

CODE FORM:

CODE mnemonic-name

REPORT GROUP DESCRIPTION

Option 1

01 [data-name-1]

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

[NEXT GROUP IS {integer-3 PLUS integer-4} NEXT PAGE]
REPORT HEADING
RH
PAGE HEADING
PH
{CONTROL HEADING identifier-1 FINAL}
CONTROLL FOOTING
CF
{identifier-2 FINAL}
PAGE FOOTING
PF
REPORT FOOTING
RF

[USAGE IS]
{DISPLAY DISPLAY-6 DISPLAY-7}

Option 2

level-number [data-name-1]

[BLANK WHEN ZERO]
[COLUMN NUMBER IS integer-1]

[GROUP INDICATE]

{JUSTIFIED JUST} RIGHT

LINE NUMBER IS {integer-2 PLUS integer-3 NEXT PAGE}

{PICTURE PIC} IS character-string

RESET ON {identifier-1} FINAL

SOURCE IS identifier-2
SUM identifier-3 identifier-4 ... [UPON data-name-2]
VALUE IS literal-1

[USAGE IS] {DISPLAY DISPLAY-6 DISPLAY-7}
COLUMN NUMBER
FORM:

COLUMN NUMBER IS integer

GROUP INDICATE
FORM:

GROUP INDICATE

LINE NUMBER
FORM:

LINE NUMBER IS (integer-1
PLUS integer-2
NEXT PAGE)

NEXT GROUP
FORM:

NEXT GROUP IS (integer-1
PLUS integer-2
NEXT PAGE)

RESET
FORM:

RESET ON (identifier-1
FINAL)

SOURCE
FORM:

SOURCE IS identifier

SUM
FORM:

SUM identifier-1 [,identifier-2]...[UPON data-name-1]
TYPE FORM:

REPORT HEADING

RH

PAGE HEADING

PH

CONTROL HEADING

CH

identifier-n

FINAL

DE

CONTROL FOOTING

CF

identifier-n

FINAL

PAGE FOOTING

PF

REPORT FOOTING

RF

PROCEDURE DIVISION

General Structure

The first entry in the PROCEDURE DIVISION of a source program must be the division header:

PROCEDURE DIVISION [USING identifier-1 [,identifier-2] ...]

The next entry must be either the DECLARATIVES header or a paragraph-name or section-name.

PROCEDURE DIVISION STATEMENTS

ACCEPT STATEMENT

FORM:

ACCEPT identifier-1 [,identifier-2] ...[FROM mnemonic-name]

ADD STATEMENT

FORM:

Option 1

ADD

\[
\left\{ \text{identifier-1} \right\} \left\{ \text{literal-1} \right\} \ldots
\]

TO identifier-m [ROUNDED]

\[
\left\{ \text{identifier-2} \right\} \left\{ \text{literal-2} \right\} \ldots
\]

\[
\left[ \text{identifier-n} \right] \left[ \text{ROUNDED} \right] \ldots
\]

\[
\text{ON SIZE ERROR} \text{statement-1} [, \text{statement-2} \ldots]
\]
Option 2

```
ADD \{identifier-1\}, \{identifier-2\}, \{identifier-3\}...

GIVING identifier-n [ROUNDED]

\{identifier-n [ROUNDED]\}...

ON SIZE ERROR statement-1 [,statement-2] ...
```
ON SIZE ERROR statement-1 [, statement-2] ... 

DELETE STATEMENT
FORM:
DELETE record-name INVALID KEY statement-1 [, statement-2] ... 

DISPLAY STATEMENT
FORM:
DISPLAY \{ literal-1 \} [', \{ literal-2 \}] ... 
[UPON mnemonic-name] [WITH NO ADVANCING]

DIVIDE STATEMENT
FORM:
Option 1
DIVIDE \{ identifier-1 \} INTO identifier-2 [ROUNDED] 
[REMAINDER identifier-4] 
[ON SIZE ERROR statement-1 [, statement-2] ... ]

Option 2
DIVIDE \{ identifier-2 \} BY identifier-1 [ROUNDED] 
[REMAINDER identifier-4] 
[ON SIZE ERROR statement-1 [, statement-2] ... ]

Option 3
DIVIDE \{ identifier-1 \} INTO \{ identifier-2 \} GIVING identifier-3 
[ROUNDED] [REMAINDER identifier-4] 
[ON SIZE ERROR statement-1 [, statement-2] ... ]
Option 4

\[
\text{DIVIDE} \left\{ \text{identifier-2} \right\} \text{ BY } \left\{ \text{identifier-1} \right\} \text{ GIVING } \text{identifier-3} \\
\text{ [ ] } \text{ROUTED } \left\{ \text{REMAINDER} \text{ identifier-4} \right\} \\
\text{ [ ] ON SIZE ERROR statement-1 [ , statement-2 ] ... } \\
\]

\text{ENTER STATEMENT FORM:}

\[
\text{ENTER} \left\{ \text{MACRO FORTRAN-IV} \text{ FORTRAN} \right\} \text{ program-name} \\
\text{ [ ] USING } \left\{ \text{identifier-1} \right. \\
\text{ [ ] literal-1} \\
\text{ [ ] procedure-name-1} \\
\text{ [ , identifier-2} \\
\text{ [ ] literal-2} \\
\text{ [ ] procedure-name-2} \right\} ... \\
\]

\text{ENTRY STATEMENT FORM:}

\[
\text{ENTRY entry-name} \left[ \text{USING identifier-1 [ , identifier, identifier-2] ... } \right] . \\
\]

\text{EXAMINE STATEMENT FORM:}

\[
\text{EXAMINE identifier} \\
\text{ [ ] TALLYING } \left\{ \text{ALL} \right. \\
\text{ [ ] LEADING} \\
\text{ [ ] UNTIL FIRST} \right\} \text{ literal-1} \left[ \text{REPLACING BY literal-2} \right] \\
\text{ [ ] REPLACING } \left\{ \text{ALL} \right. \\
\text{ [ ] LEADING} \\
\text{ [ ] UNTIL FIRST} \right\} \text{ literal-1} \text{ BY literal-2} \\
\]

\text{EXIT STATEMENT FORM:}

\[
\text{paragraph-name. EXIT.} \\
\]

\text{EXIT PROGRAM STATEMENT FORM:}

\[
\text{EXIT PROGRAM.} \\
\]
GENERATE STATEMENT
FORM:

GENERATE identifier

GO TO STATEMENT
FORM:

Option 1

GO TO \[ procedure-name-1 \]

Option 2

GO TO procedure-name-1, procedure-name-2

\[, procedure-name-3 \] ...

DEPENDING ON identifier

GOBACK STATEMENT
FORM:

GOBACK

IF STATEMENT
FORM:

IF conditional expression

\{ statement-1 \[, statement-2 \] \} ...

NEXT SENTENCE

ELSE \{ statement-3 \[, statement-4 \] \} ...

NEXT SENTENCE

INITIATE STATEMENT
FORM:

INITIATE report-name-1 \[, report-name-2 \] ...

MOVE STATEMENT
FORM:

Option 1

MOVE \{ identifier-1 \} TO identifier-2

\[, identifier-3 \] ...
MOVE \{ CORRESPONDING \} identifier-1 TO identifier-2

MULTIPLY STATEMENT
FORM:

Option 1

MULTIPLY \{ \text{identifier-1} \} \text{BY} \text{identifier-2} \{ \text{ROUNDED} \}
\text{ON SIZE ERROR} \text{statement-1 [,statement-2] ... *}

Option 2

MULTIPLY \{ \text{identifier-1} \} \text{BY} \{ \text{identifier-2} \}
\text{GIVING} \text{identifier-3} \{ \text{ROUNDED} \}
\text{ON SIZE ERROR} \text{statement-1 [,statement-2] ... *}

NOTE STATEMENT
FORM:

\text{NOTE} \text{character-string *}

OPEN STATEMENT
FORM:

\{ \text{INPUT} \text{OUTPUT} \} \text{file-name-1 [WITH NO REWIND]}
\text{OPEN} \{ \text{file-name-2 [WITH NO REWIND]} \text{...}
\text{I-O} \text{INPUT-OUTPUT} \} \text{file-name-3 [,file-name-4] ... *}

PERFORM STATEMENT
FORM:

Option 1

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

Option 2

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

{identifier-1}

integer-1 TIMES

Option 3

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

UNTIL condition-1

Option 4

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

VARYING identifier-1 FROM {literal-1}

identifier-2

BY { literal-2

identifier-3 } UNTIL condition-1

AFTER VARYING identifier-4 FROM { literal-3

identifier-5 }

BY { literal-4

identifier-6 } UNTIL condition-2

AFTER VARYING identifier-7 FROM { literal-5

identifier-8 }

BY { literal-6

identifier-9 } UNTIL condition-3

READ STATEMENT
FORM:

READ file-name RECORD

{ INTO identifier }{ AT END

INVALID KEY } statement-1

..., statement-2 } ...

RELEASE STATEMENT
FORM:

RELEASE record-name { FROM identifier }

RETURN STATEMENT
FORM:

RETURN file-name RECORD { INTO identifier } AT END

statement-1 { ..., statement-2 } ...
REWRITE STATEMENT
FORM:

REWRITE record-name [ FROM identifier ]

INVALID KEY statement-1 [,statement-2] ... .

SEARCH STATEMENT
FORM:

Option 1

SEARCH identifier-1 [VARYING identifier-2] [AT END statement-1 [,statement-2] ...]

WHEN condition-1 (statement-3 [,statement-4] ...)

NEXT SENTENCE

[ , WHEN condition-2 (statement-5 [, statement-6] ... ) ... ]

Option 2

SEARCH ALL identifier-1 [AT END statement-1 [,statement-2] ...]

WHEN condition-1 (statement-3 [,statement-4] ...)

NEXT SENTENCE

SEEK STATEMENT
FORM:

SEEK file-name RECORD

SET STATEMENT
FORM:

SET identifier-1 [,identifier-2] ... { TO UP BY } { identifier-3 }

{ literal-1 }

DOWN BY

SORT STATEMENT
FORM:

SORT file-name-1 ON { ASCENDING ) KEY data-name-1

{ DESCENDING }[ ,data-name-2] ... { ON ASCENDING } KEY data-name-3

{ DESCENDING }[ ,data-name-4] ... }

INPUT PROCEDURE IS procedure-name-1 [THRU procedure-name-2 ]

USING file-name-2

OUTPUT PROCEDURE IS procedure-name-3 [THRU procedure-name-4 ]

GIVING file-name-3
STOP STATEMENT
FORM:

STOP \{ literal RUN \} 

STRING STATEMENT
FORM:

STRING \{ identifier-1 \} [ identifier-2 ] [ literal-1 ] [ literal-2 ] ... 

DELIMITED BY \{ identifier-3 \} [ literal-3 ] SIZE 

[ identifier-4 ] [ identifier-5 ] [ literal-4 ] [ literal-5 ] ... 

DELIMITED BY \{ identifier-6 \} [ literal-6 ] SIZE 

INTO identifier-7 [ WITH POINTER identifier-8 ] [; ON OVERFLOW statement-1 ]

SUBTRACT STATEMENT
FORM:

Option 1

SUBTRACT \{ identifier-1 \} [ identifier-2 ] \{ literal-1 \} [ literal-2 ] ... 

FROM identifier-m [ ROUNDED ] [ , identifier-n ] [ ROUNDED ] ... 

[ ON SIZE ERROR statement-1 [ , statement-2 ] ... ]

Option 2

SUBTRACT \{ identifier-1 \} [ identifier-2 ] \{ literal-1 \} [ literal-2 ] ... 

FROM \{ identifier-m \} [ literal-m ] GIVING identifier-n [ ROUNDED ] 

[ identifier-p [ ROUNDED ] ... ] 

[ ON SIZE ERROR statement-1 [ , statement-2 ] ... ]
Option 3

\[
\text{SUBTRACT} \left\{ \text{CORRESPONDING} \right\} \text{identifier-1 FROM identifier-2}
\]

\[
\left[ \text{ROUNDED} \right] \left[ \text{ON SIZE ERROR} \right] \text{statement-1}
\]

\[
\left[ \text{, statement-2} \right] \ldots *
\]

**TERMINATE STATEMENT**

FORM:

**TERMINATE** report-name-1 \[\text{, report-name-2} \] \[\ldots *\]

**TRACE STATEMENT**

FORM:

**TRACE** \{ \text{ON} \}

\{ \text{OFF} \}

**UNSTRING STATEMENT**

FORM:

**UNSTRING** identifier-1

\[
\text{DELIMITED BY [ALL]} \left\{ \text{identifier-2} \right\}
\]

\[
\left[ \text{, OR [ALL]} \right] \left\{ \text{identifier-3} \right\}
\]

\[
\left\{ \text{, literal-1} \right\}
\]

\[
\left\{ \text{, literal-2} \right\}
\]

\[
\ldots
\]

\[
\text{INTO identifier-4 [ , DELIMITER IN identifier-5]}
\]

\[
\left[ , \text{COUNT IN identifier-6} \right]
\]

\[
\left[ , \text{identifier-7 [ , DELIMITER IN identifier-8]}
\right]
\]

\[
\left[ , \text{COUNT IN identifier-9}] \ldots \right.
\]

\[
\left[ \text{WITH POINTER identifier-10} [ \text{, TALLYING IN identifier-11]}
\right]
\]

\[
\left[ ; \text{ON OVERFLOW} \text{ statement-1} \right]
\]

**USE STATEMENT**

FORM:

Format 1

**USE AFTER STANDARD ERROR PROCEDURE ON** \{ \text{file-name-1 [OPEN]} \}

\{ \text{INPUT OUTPUT} \}

\{ I-O \}

\{ INPUT-OUTPUT \}

Format 2

**USE** \{ \text{BEFORE} \}

\{ \text{AFTER} \}

**STANDARD** \{ \text{BEGINNING} \}

\{ \text{ENDING} \}
Format 3

USE BEFORE REPORTING identifier-1

WRITE STATEMENT
FORM:

Option 1

WRITE record-name-1 FROM identifier-1

BEFORE ADVANCING identifier-2 LINES integer-1 LINES mnemonic-name

Option 2

WRITE record-name-1 FROM identifier-1 INVALID KEY

statement-1 [, statement-2] ... *

COBOL LIBRARY
COPY STATEMENT
FORM:

COPY library-name

REPLACING word-1 BY identifier-1 procedure-name-1

[, word-3 BY identifier-2 procedure-name-2] ... *
**UTILITY PROGRAMS**

The LIBRARY, SORT, ISAM, RERUN AND COBDT utility programs aid the user in accomplishing COBOL-oriented tasks.

**LIBRARY**

The command string to LIBRARY is:

```plaintext
* R LIBRARY
  FILE1.LIB, FILE2.LST=FILE3.LIB
```

The six commands to position the input and scratch files are described in Table 1.

**TABLE 1**

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT library-name</td>
<td>The input file is copied to the scratch file, starting at the current position of the files, until a source routine with a name alphabetically greater than the one specified is encountered. The new name is inserted in the Fine Table, and the program awaits another command.</td>
</tr>
<tr>
<td>INSERT library-name, dev:file,ext [ppn]</td>
<td>The entire file is inserted into the library with the name indicated by library-name. The file must be ASCII. If there are line numbers in the file, they are included in the file. If there are no line numbers, they are added to the lines, starting with 10 and incrementing by 10.</td>
</tr>
<tr>
<td>DELETE library-name</td>
<td>The input file is copied to the scratch file until the source routine with the name specified is encountered. The input file is then positioned after that source routine.</td>
</tr>
<tr>
<td>REPLACE library-name</td>
<td>The program does a DELETE followed by an INSERT.</td>
</tr>
<tr>
<td>REPLACE library-name, dev:file,ext [ppn]</td>
<td>The file named library-name is replaced with the specified file. The new file must be ASCII. If there are line numbers in the file, they are included in the file. If there are no line numbers in the file, they are added to the lines, starting with 10 and incrementing by 10.</td>
</tr>
<tr>
<td>CORRECT library-name</td>
<td>The input file is copied to the scratch file until a source routine with the name specified is encountered. Typing /N after the CORRECT command causes new line numbers to be applied to the output version of the source language routine.</td>
</tr>
</tbody>
</table>
## TABLE I (CONT.)
Commands for Positioning Files

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td>The remainder of the input file is copied to the scratch file, and the output file is created, and the program then terminates.</td>
</tr>
<tr>
<td>RESTART</td>
<td>The remainder of the input file is copied to the scratch file. The scratch file then becomes the input file, and a new scratch file is started. This command allows the user to update routines out of library-name order.</td>
</tr>
<tr>
<td>EXTRACT library-name</td>
<td>A file with the specified name and extension on the specified device is created from the file named library-name. If the /N switch is included after the file descriptor, line numbers are put on the lines of the output file. If the /N switch is not included, the file will not have line numbers.</td>
</tr>
<tr>
<td>dev:file.ext [ppn]</td>
<td></td>
</tr>
</tbody>
</table>

The three commands to alter the contents of a source file are described in Table II.

## TABLE II
Commands for Altering Contents of Source File

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dnnnnnn</td>
<td>The input file is copied to the scratch file until nnnnnn, the specified line, is encountered. That line is then skipped.</td>
</tr>
<tr>
<td>Innnnnn COBOL-statement</td>
<td>The input file is copied until either a line having a larger line-number or a new source language routine is encountered. The COBOL-statement is inserted at that point.</td>
</tr>
<tr>
<td>Rnnnnnn COBOL-statement</td>
<td>The input file is copied until the specified line is encountered. The COBOL-statement with that line-number is replaced by the statement in the command.</td>
</tr>
</tbody>
</table>

**SORT**

The command string to SORT is:

```plaintext
dev:outfil.ext/sw_1/sw_2.../sw_n = dev:infil.ext/sw_1/sw_2.../sw_n
```

The switches to the SORT program are shown in Table III.
<table>
<thead>
<tr>
<th>Switch</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>/A</td>
<td>The file is recorded in ASCII mode.</td>
</tr>
<tr>
<td>/Bn</td>
<td>A block contains n records; n is a decimal number. If the /B switch is omitted, it is assumed that the file is unblocked.</td>
</tr>
</tbody>
</table>
| /Kabcm.n | /K defines the sort key according to the following parameters:  
|         | a = S  The field has an operational sign.  
|         | a = U  The field has no operational sign; its magnitude is used.  
|         | b = X  The field is alphanumeric.  
|         | b = C  The field is COMPUTATIONAL.  
|         | b = F  The field is COMPUTATIONAL-1 (floating point).  
|         | b = N  The field is numeric display.  
|         | If this parameter is omitted, the field is assumed to be alphanumeric if the sign parameter (a) is also omitted.  
|         | If this parameter is omitted and the sign parameter is included, the field is assumed to be numeric display (b=N). Data formats are described in Chapter 5, the USAGE clause.  
|         | c = A  The field is to be sorted in ascending order.  
|         | c = D  The field is to be sorted in descending order.  
|         | If this parameter is omitted, the field is sorted in ascending order.  
|         | m      is the starting byte or position of one field (e.g., the starting column on a card).  
|         | n      is the size of the field in either bytes or digits, depending on the context.  
|         | More than one key can be entered with the /K switch, providing the keys are separated from one another by commas (e.g., /Kabcm,n,abcm,n...). The keys are sorted in the order that they are entered in the command string. |
### TABLE III (CONT.)

<table>
<thead>
<tr>
<th>Switch</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| /Lam   | /L specifies the labeling convention.  
|        | a = S  The labels are standard.  
|        | a = O  The labels are omitted.  
|        | a = N  The labels are nonstandard.  
|        | m specifies the size of a nonstandard label in bytes.  
|        | If the /L switch is omitted, it is assumed that the labels are omitted unless a file name is specified or a directory device is used. In the latter cases, standard labels are assumed. |
| /Rm    | /R indicates the size of the largest input record, where m is the size of the record in bytes. |
| /S     | The file is recorded in SIXBIT mode. |
| /Tdev  | /T indicates that the specified device is to be used as a scratch device during the sort. More than one device can be specified, providing the devices are separated by commas (e.g., /Tdev1,dev2,...). |

---

RERUN

The command string to run the RERUN program is:

```
  R RERUN
```

ISAM

ISAM performs three basic functions. The command string to build an indexed sequential file from a sequential file is:

```
  R ISAM
  *dev1:indfil.ext[ppn1],dev2:datfil.ext=dev3:seqfil.ext[ppn2]/B
```

The command string to maintain an indexed sequential file is:

```
  R ISAM
  *dev1:indfil.ext[ppn1],dev2:datfil.ext=infil.ext[ppn2]/M
```

The command string to pack an indexed sequential file is:

```
  R ISAM
  *dev1:seqfil.ext[ppn1]=dev2:indfil.ext[ppn2]/P
```
The command string to ignore errors when packing an indexed sequential file is:

```
*R ISAM
/dev1:seqfil.ext[ppn1]=dev2:indfil.ext[ppn2]/P/I
```

The command string to read magnetic tape labels when building an indexed sequential file is:

```
*R ISAM
/dev1:indfil.ext[ppn],dev2:datfil.ext=MTAn:seqfil.ext/B/L
```

The command string to write magnetic tape labels when packing an indexed sequential file is:

```
*R ISAM
/MTAn:seqfil.ext=dev1:indfil.ext[ppn]/P/L
```

**COBDDT**

COBDDT is loaded and started by three methods:

```
*LOAD %"LOCALS"dev:prognm, SYS:COBDDT
  _START
```

or

```
*R LINK
/LOCALS dev:prognm, SYS:COBDDT/GO
  _START
```

or

```
*DEBUG dev:prognm
```

The commands to COBDDT are listed below.

**ACCEPT COMMAND**

FORM: ACCEPT

```
ACCEPT data-name
```

**BREAK COMMAND**

FORM: BREAK paragraph-name

**CLEAR COMMAND**

FORM: CLEAR paragraph-name

**DISPLAY COMMAND**

FORM: DISPLAY

```
DISPLAY data-name
```

**HISTORY BEGIN COMMAND**

FORM: HISTORY BEGIN

**HISTORY INITIALIZE COMMAND**

FORM: HISTORY INITIALIZE
HISTORY REPORT COMMAND
FORM: HISTORY REPORT

MODULE COMMAND
FORM: MODULE program-name

PROCEED COMMAND
FORM: PROCEED
    PROCEED n
WHERE: "n" is a number

STOP COMMAND
FORM: STOP

TRACE COMMAND
FORM: TRACE ON
    TRACE OFF

WHERE COMMAND
FORM: WHERE

RUNNING THE COBOL COMPILER DIRECTLY

The command to run the COBOL compiler is:

    _R COBOL

A command to the compiler is of the general form:

    _*RELFIL,LSTFIL = SRC1,SRC2,...

EXAMPLE:

    _*MYPROG.REL, MYPROG.LST=MYPROG.CBL,MYPROG.CBL

A "-" can be used to exclude an output file. For example: to pro-
duce no listing file use:

    _*, - = MYPROG

to produce no binary file use:

    *-, - = MYPROG

to produce no output file

    *-, -, = MYPROG

The switches to the COBOL compiler are shown in Table IV.
<table>
<thead>
<tr>
<th>Switch</th>
<th>Action by Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Allows the listing of the code generated (the source program is listed whenever a listing file is specified).</td>
</tr>
<tr>
<td>C</td>
<td>Produces a cross-reference listing of all user-defined items in the source program.</td>
</tr>
<tr>
<td>E</td>
<td>Checks the program for errors but does not generate code.</td>
</tr>
<tr>
<td>H</td>
<td>Types a description of COBOL command strings and lists the switches. When this switch is used, the other parts of the command string are ignored.</td>
</tr>
<tr>
<td>I</td>
<td>Forces the compiler to suppress generation of a starting address for a main program.</td>
</tr>
<tr>
<td>J</td>
<td>Forces the compiler to generate a starting address for a subprogram.</td>
</tr>
<tr>
<td>L</td>
<td>Uses the preceding file descriptor as a library file whenever it encounters the COPY verb.</td>
</tr>
<tr>
<td></td>
<td>1. This switch is legal only with source files.</td>
</tr>
<tr>
<td></td>
<td>2. The file descriptor is not part of the main program.</td>
</tr>
<tr>
<td></td>
<td>3. More than one descriptor may have the /L switch. If the first source file is not a library file, the file LIBRARY.LIB is used (if present on the DSK) until the /L file is described.</td>
</tr>
<tr>
<td>M</td>
<td>Prints a map showing the parameters of each user-defined item (e.g., data-names and procedure-names).</td>
</tr>
<tr>
<td>N</td>
<td>The source errors are not typed on the user's terminal.</td>
</tr>
</tbody>
</table>
### TABLE IV (CONT.)
**COBOL Switch Summary**

<table>
<thead>
<tr>
<th>Switch</th>
<th>Action by Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P</strong></td>
<td>Indicates production mode. Trace calls are not generated and user symbols are suppressed.</td>
</tr>
<tr>
<td><strong>R</strong></td>
<td>A two-segment object program is produced. The high segment will contain the resident sections of the Procedure Division; the low segment will contain all else. When the object program is loaded with the linking loader, LIBOL will be added to the high segment.</td>
</tr>
<tr>
<td><strong>S</strong></td>
<td>The source file is in conventional format (with sequence numbers in columns 1-6 and with comments starting in column 73).</td>
</tr>
<tr>
<td><strong>W</strong></td>
<td>Rewinds the device before reading or writing. (This is valid for magnetic tape only.)</td>
</tr>
<tr>
<td><strong>Z</strong></td>
<td>Clears the directory of the device before writing. (This is valid for output DECtape only.)</td>
</tr>
</tbody>
</table>

**EXAMPLE:** *MYPROG.REL,MYPLOG.LST=MYPROG.CBL/E*

### MONITOR COMMANDS TO RUN THE COBOL COMPILER

Compilation of COBOL source program files can be performed by use of the **compile**, **load**, **execute**, and **debug** commands.

**COMPILE COMMAND**

FORM: _COMPILE filename,.ext  
**EXAMPLE:** _COMPILE TEST,.CBL

**LOAD COMMAND**

FORM: _LOAD filename,.ext  
**EXAMPLE:** _LOAD TEST,.CBL

**EXECUTE COMMAND**

FORM: _EXECUTE filename,.ext  
**EXAMPLE:** _EXECUTE TEST,.CBL

**DEBUG COMMAND**

FORM: _DEBUG filename,.ext  
**EXAMPLE:** _DEBUG TEST,.CBL

### COBOL Reserved Words

In the listing below, words preceded by no symbols are standard COBOL reserved words that are also reserved in DECsystem-10 COBOL. Words preceded by a single * are ANSI standard reserved COBOL, words that are not reserved in DECsystem-10 COBOL, but should be avoided for compatibility with other COBOL compilers. Words preceded by ** are reserved in DECsystem-10 COBOL but not in the ANSI standard.
<table>
<thead>
<tr>
<th>SIXBIT</th>
<th>Character</th>
<th>ASCII 7-Bit</th>
<th>SIXBIT</th>
<th>Character</th>
<th>ASCII 7-Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Space</td>
<td>040</td>
<td>60</td>
<td>P</td>
<td>120</td>
</tr>
<tr>
<td>01</td>
<td>!</td>
<td>041</td>
<td>61</td>
<td>Q</td>
<td>121</td>
</tr>
<tr>
<td>02</td>
<td>&quot;</td>
<td>042</td>
<td>62</td>
<td>R</td>
<td>122</td>
</tr>
<tr>
<td>03</td>
<td>#</td>
<td>043</td>
<td>63</td>
<td>S</td>
<td>123</td>
</tr>
<tr>
<td>04</td>
<td>$</td>
<td>044</td>
<td>64</td>
<td>T</td>
<td>124</td>
</tr>
<tr>
<td>05</td>
<td>%</td>
<td>045</td>
<td>65</td>
<td>U</td>
<td>125</td>
</tr>
<tr>
<td>06</td>
<td>&amp;</td>
<td>046</td>
<td>66</td>
<td>V</td>
<td>126</td>
</tr>
<tr>
<td>07</td>
<td>'</td>
<td>047</td>
<td>67</td>
<td>W</td>
<td>127</td>
</tr>
<tr>
<td>10</td>
<td>(</td>
<td>050</td>
<td>70</td>
<td>X</td>
<td>130</td>
</tr>
<tr>
<td>11</td>
<td>)</td>
<td>051</td>
<td>71</td>
<td>Y</td>
<td>131</td>
</tr>
<tr>
<td>12</td>
<td>*</td>
<td>052</td>
<td>72</td>
<td>Z</td>
<td>132</td>
</tr>
<tr>
<td>13</td>
<td>+</td>
<td>053</td>
<td>73</td>
<td>[</td>
<td>133</td>
</tr>
<tr>
<td>14</td>
<td>,</td>
<td>054</td>
<td>74</td>
<td>\</td>
<td>134</td>
</tr>
<tr>
<td>15</td>
<td>-</td>
<td>055</td>
<td>75</td>
<td>]</td>
<td>135</td>
</tr>
<tr>
<td>16</td>
<td>*</td>
<td>056</td>
<td>76</td>
<td>↑</td>
<td>136</td>
</tr>
<tr>
<td>17</td>
<td>/</td>
<td>057</td>
<td>77</td>
<td>←</td>
<td>137</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>060</td>
<td>-</td>
<td>~</td>
<td>140</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>061</td>
<td>-</td>
<td>a</td>
<td>141</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>062</td>
<td>-</td>
<td>b</td>
<td>142</td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>063</td>
<td>-</td>
<td>c</td>
<td>143</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>064</td>
<td>-</td>
<td>d</td>
<td>144</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>065</td>
<td>-</td>
<td>e</td>
<td>145</td>
</tr>
<tr>
<td>26</td>
<td>6</td>
<td>066</td>
<td>-</td>
<td>f</td>
<td>146</td>
</tr>
<tr>
<td>27</td>
<td>7</td>
<td>067</td>
<td>-</td>
<td>g</td>
<td>147</td>
</tr>
<tr>
<td>30</td>
<td>8</td>
<td>070</td>
<td>-</td>
<td>h</td>
<td>150</td>
</tr>
<tr>
<td>31</td>
<td>9</td>
<td>071</td>
<td>-</td>
<td>i</td>
<td>151</td>
</tr>
<tr>
<td>32</td>
<td>:</td>
<td>072</td>
<td>-</td>
<td>j</td>
<td>152</td>
</tr>
<tr>
<td>33</td>
<td>;</td>
<td>073</td>
<td>-</td>
<td>k</td>
<td>153</td>
</tr>
<tr>
<td>34</td>
<td>&lt;</td>
<td>074</td>
<td>-</td>
<td>l</td>
<td>154</td>
</tr>
<tr>
<td>35</td>
<td>=</td>
<td>075</td>
<td>-</td>
<td>m</td>
<td>155</td>
</tr>
<tr>
<td>36</td>
<td>&gt;</td>
<td>076</td>
<td>-</td>
<td>n</td>
<td>156</td>
</tr>
<tr>
<td>37</td>
<td>?</td>
<td>077</td>
<td>-</td>
<td>o</td>
<td>157</td>
</tr>
<tr>
<td>40</td>
<td>@</td>
<td>100</td>
<td>-</td>
<td>p</td>
<td>160</td>
</tr>
<tr>
<td>41</td>
<td>A</td>
<td>101</td>
<td>-</td>
<td>q</td>
<td>161</td>
</tr>
<tr>
<td>42</td>
<td>B</td>
<td>102</td>
<td>-</td>
<td>r</td>
<td>162</td>
</tr>
<tr>
<td>43</td>
<td>C</td>
<td>103</td>
<td>-</td>
<td>s</td>
<td>163</td>
</tr>
<tr>
<td>44</td>
<td>D</td>
<td>104</td>
<td>-</td>
<td>t</td>
<td>164</td>
</tr>
<tr>
<td>45</td>
<td>E</td>
<td>105</td>
<td>-</td>
<td>u</td>
<td>165</td>
</tr>
<tr>
<td>46</td>
<td>F</td>
<td>106</td>
<td>-</td>
<td>v</td>
<td>166</td>
</tr>
<tr>
<td>47</td>
<td>G</td>
<td>107</td>
<td>-</td>
<td>w</td>
<td>167</td>
</tr>
<tr>
<td>50</td>
<td>H</td>
<td>110</td>
<td>-</td>
<td>x</td>
<td>170</td>
</tr>
<tr>
<td>51</td>
<td>I</td>
<td>111</td>
<td>-</td>
<td>y</td>
<td>171</td>
</tr>
<tr>
<td>52</td>
<td>J</td>
<td>112</td>
<td>-</td>
<td>z</td>
<td>172</td>
</tr>
<tr>
<td>53</td>
<td>K</td>
<td>113</td>
<td>-</td>
<td>{</td>
<td>173</td>
</tr>
<tr>
<td>54</td>
<td>L</td>
<td>114</td>
<td>-</td>
<td>}</td>
<td>174</td>
</tr>
<tr>
<td>55</td>
<td>M</td>
<td>115</td>
<td>-</td>
<td>~</td>
<td>175</td>
</tr>
<tr>
<td>56</td>
<td>N</td>
<td>116</td>
<td>-</td>
<td>Delete</td>
<td>176</td>
</tr>
<tr>
<td>57</td>
<td>O</td>
<td>117</td>
<td>-</td>
<td></td>
<td>177</td>
</tr>
</tbody>
</table>
For additional information on the COBOL Language, refer to the DECsystem-10 COBOL Language Manual, DEC-10-KC1E-D.

Order number DEC-10-OCRCA-A-D. For single copies of this card, contact a DEC sales office. Multiple copies may be ordered in lots of 25 from the Software Distribution Center, Digital Equipment Corporation, 200 Forest Street, Marlboro, Ma. 01752.