decsystem10

COBOL
POCKET REFERENCE

Version 6 Cobol
Version 7 Libol

digital

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 –

Relational operator	Meaning
IS [NOT] GREATER THAN IS [NOT] > THAN	Greater than, not greater than
IS [NOT] LESS THAN IS [NOT] < THAN	Less than, not less than
IS [NOT] EQUAL (EQUALS) TO IS [NOT] = TO	Equal to, not equal to

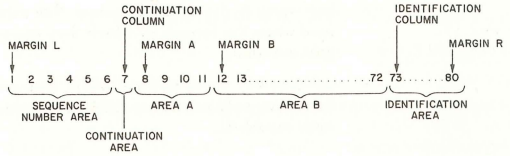
LOGICAL OPERATORS	MEANING
OR	Entire condition is true if either or both of the simple conditions are true.
AND	Entire condition is true if both of the simple conditions are true.

LINE AND SOURCE PROGRAM FORMAT

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.

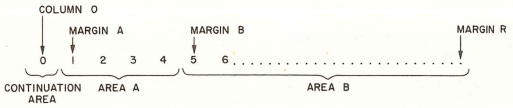
The continuation area occupies one character position in the continuation column.

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.

STANDARD FORMAT



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

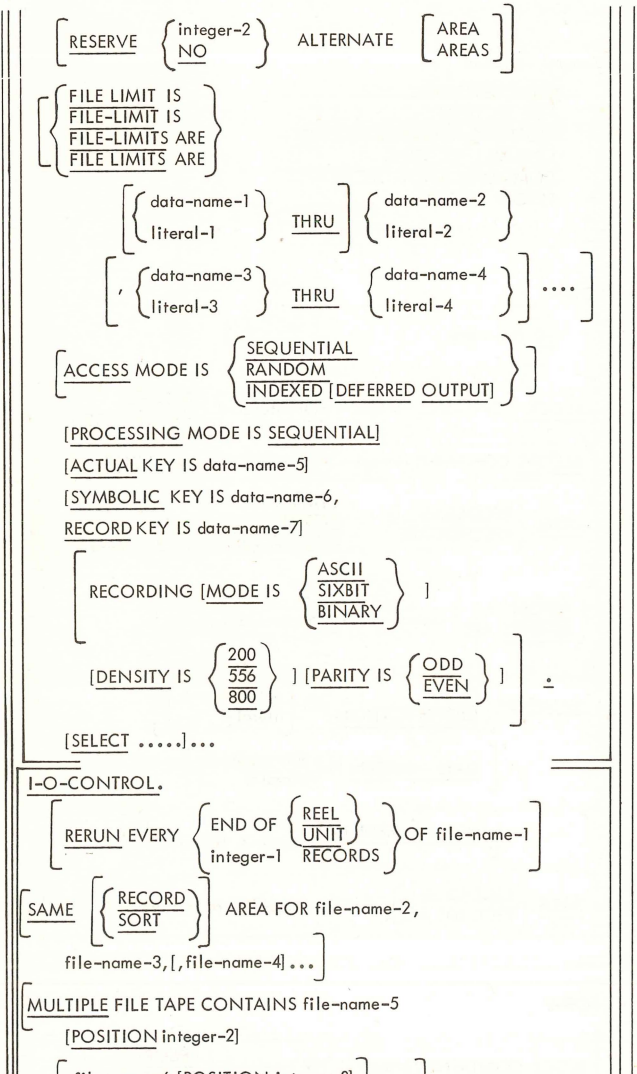
General structure

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

General structure

```
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]...]
[SWITCH (m) {IS mnemonic-name-4
                [ON STATUS IS condition-name-1]
                [OFF STATUS IS condition-name-2]
            | 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
NO
ALTERNATE
AREA
AREAS
FILE LIMIT IS
FILE-LIMIT IS
FILE-LIMITS ARE
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
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 .....]...
I-O-CONTROL.
RERUN EVERY
END OF
REEL
UNIT
RECORDS
integer-1
OF file-name-1
SAME
RECORD
SORT
AREA FOR file-name-2,
file-name-3,[,file-name-4]...
MULTIPLE FILE TAPE CONTAINS file-name-5
[POSITION integer-2]
,file-name-6 [POSITION integer-3]

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 / CHARACTERS } ]
[ RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS ]
[ LABEL { RECORD IS / RECORDS ARE } { STANDARD / OMITTED / 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:

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

RECORD CONTAINS
FORM:

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

REPORT
FORM:

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

SD file-name
FORM:

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

VALUE OF
FORM:

```
[ VALUE OF [ { ID             } IS { data-name-1 } ]
             { IDENTIFICATION }    { literal-1  }
[ DATE-WRITTEN IS { data-name-2 } ]
                  { literal-2   }
[ USER-NUMBER  IS { data-name-3                  } ] ]
                  { literal-3, literal-4         }
```

DATA DESCRIPTION ENTRY
FORM:

```
level-number { data-name-1 }
             { FILLER      }

[ REDEFINES data-name-2 ]
```

```
[{PICTURE}  IS  picture-string]
 {PIC    }

[                 {COMPUTATIONAL  }]
[                 {COMP           }]
[                 {COMPUTATIONAL-1}]
[                 {COMP-1         }]
[ [USAGE IS]      {DISPLAY        }]
[                 {DISPLAY-6      }]
[                 {DISPLAY-7      }]
[                 {INDEX          }]
[                 {DATA BASE-KEY  }]

 [{SYNCHRONIZED}  {LEFT }]
 [{SYNC        }  {RIGHT}]

 [{JUSTIFIED}          {RIGHT}]
 [{JUST     }          {LEFT }]

  [BLANK WHEN ZERO]

  [VALUE IS literal-1]

  [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] ...]  ] .
```

RENAMES ENTRY

```
66 data-name-1  RENAMES data-name-2  [THRU data-name-3]  .
```

CONDITION NAME ENTRY

```
88 condition-name  {VALUE IS  }   literal-1 [THRU literal-2]
                   {VALUES ARE}
    [,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 { VALUE IS  } literal-1 [ THRU literal-2 ]
                  { VALUES ARE}

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

DATA-NAME/FILLER
FORM:

```
level-number { data-name }
             { FILLER    }
```

JUSTIFIED
FORM:

```
[ { JUSTIFIED } { RIGHT } ]
  { JUST      } { LEFT  }
```

LEVEL-NUMBER
FORM:

```
level-number { data-name }
             { FILLER    }
```

OCCURS
FORM:

```
[ 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 ] ... ] ]
```

PICTURE
FORM:

```
[ { PICTURE } IS picture-string ]
  { PIC     }
```

REDEFINES

FORM:

```
level-number  data-name-1  REDEFINES data-name-2
```

RENAMES

FORM:

```
66 data-name-1  RENAMES  data-name-2  [ THRU  data-name-3 ] .
```

SYNCHRONIZED

FORM:

```
[ { SYNCHRONIZED } ]  [ { LEFT  } ]
[ { SYNC        } ]   [ { RIGHT } ]
```

USAGE

FORM:

```
[ [ USAGE IS ] { COMPUTATIONAL   } ]
               { COMP            }
               { COMPUTATIONAL-1 }
               { COMP-1          }
               { DISPLAY         }
               { DISPLAY-6       }
               { DISPLAY-7       }
               { INDEX           }
               { DATABASE-KEY    }
```

VALUE

FORM:

```
FORMAT 1:  [ VALUE IS literal ]

FORMAT 2:  [ { VALUE IS  } literal-1 [ THRU literal-2 ]
             { VALUES ARE }
             [ , literal-3 [ THRU literal-4 ] ]  ... ]
```

REPORT SECTION

REPORT DESCRIPTION (RD)

FORM:

```
RD report-name

  [ CODE mnemonic-name ]

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

CONTROL
FORM:

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

PAGE-LIMIT
FORM:

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

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

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

CODE
FORM:

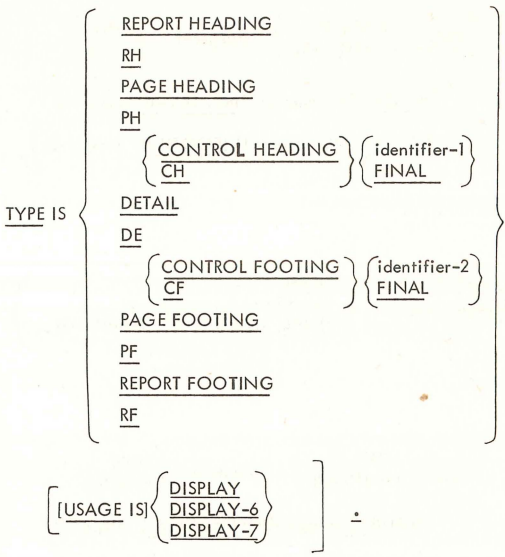
```
CODE  mnemonic-name
```

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      }
```



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 } { identifier-n }               │
                  │      { CH              } { FINAL        }               │
                  │ DETAIL                                                  │
TYPE IS           │ DE                                                      │
                  │      { CONTROL FOOTING } { identifier-n }               │
                  │      { CF              } { 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 { identifier-1 } [ , { identifier-2 } ]...
    { literal-1    }     { literal-2    }
      TO identifier-m [ROUNDED]
    [ , identifier-n [ROUNDED] ] ...
    [ON SIZE ERROR statement-1 [,statement-2] ... .]
```

Option 2

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

       GIVING identifier-m [ROUNDED]
     [ , identifier-n [ROUNDED] ]  ...
     [ ON SIZE ERROR statement-1 [, statement-2] ... . ]
```

Option 3

```
ADD  { CORRESPONDING }  identifier-1 TO identifier-2
     { CORR          }

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

ALTER STATEMENT

FORM:

```
ALTER procedure-name-1 TO PROCEED TO procedure-name-2

[, procedure-name-3 TO PROCEED TO procedure-name-4] ...
```

CALL STATEMENT

FORM:

```
CALL  { program-name }  [ USING  identifier-1  [, identifier-2] ... ]
      { entry-name   }
```

CLOSE STATEMENT

FORM:

```
CLOSE  file-name  [ { REEL   } ] [ WITH { NO REWIND } ]
                  { UNIT   }           { LOCK      }
                                       { DELETE    }

  [ , file-name-1  [ { REEL   } ] [ WITH { NO REWIND } ] ]  ...
                   { UNIT   }           { LOCK      }
                                        { DELETE    }
```

COMPUTE STATEMENT

FORM:

```
COMPUTE identifier-1 [ROUNDED] { EQUALS   }
                               { EQUAL TO }
                               { =        }
```

```
{ identifier-2 }
{ literal-1    }
{ arithmetic-expression }

[ 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    } ] ...
        { identifier-1 }     { identifier-2 }

        [UPON mnemonic-name] [WITH NO ADVANCING]
```

DIVIDE STATEMENT

FORM:

Option 1

```
DIVIDE { identifier-1 } INTO identifier-2 [ ROUNDED ]
       { literal-1    }

       [ REMAINDER identifier-4 ]

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

Option 2

```
DIVIDE { identifier-2 } BY identifier-1 [ ROUNDED ]
       { literal-2    }

       [ REMAINDER identifier-4 ]

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

Option 3

```
DIVIDE { identifier-1 } INTO { identifier-2 } GIVING identifier-3
       { literal-1    }     { literal-2    }

       [ ROUNDED ] [ REMAINDER identifier-4 ]

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

Option 4

```
DIVIDE { identifier-2 } BY { identifier-1 } GIVING identifier-3
       { literal-2    }    { literal-1    }

     [ ROUNDED ] [ REMAINDER identifier-4 ]

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

ENTER STATEMENT

FORM:

```
ENTER { MACRO      } program-name
      { FORTRAN-IV }
      { FORTRAN    }

    [ USING { identifier-1     } ]
            { literal-1        }
            { procedure-name-1 }

  [ , { identifier-2     } ]   ... ]
      { literal-2        }
      { procedure-name-2 }
```

ENTRY STATEMENT

FORM:

```
ENTRY entry-name [ USING identifier-1 [,identifier, identifier-2] ... ] .
```

EXAMINE STATEMENT

FORM:

```
EXAMINE identifier

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

{ REPLACING { ALL       } literal-1 BY literal-2 }
            { LEADING   }
            { [UNTIL] FIRST }
```

EXIT STATEMENT

FORM:

```
paragraph-name. EXIT.
```

EXIT PROGRAM STATEMENT

FORM:

```
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
     { literal-1    }
[ , identifier-3 ] ...
```

Option 2

```
MOVE { CORRESPONDING } identifier-1 TO identifier-2
     { CORR          }
```

MULTIPLY STATEMENT

FORM:

Option 1

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

Option 2

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

NOTE STATEMENT

FORM:

```
NOTE character- string .
```

OPEN STATEMENT

FORM:

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

PERFORM STATEMENT

FORM:

Option 1

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

Option 2

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

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    } UNTIL condition-1
       { identifier-3 }
  [ AFTER VARYING identifier-4 FROM { literal-3    }
                                    { identifier-5 }
      BY { literal-4    } UNTIL condition-2
         { identifier-6 }
  [ AFTER VARYING identifier-7 FROM { literal-5    }
                                    { identifier-8 }
      BY { literal-6    } UNTIL condition-3 ] ]
         { identifier-9 }
```

READ STATEMENT

FORM:

```
READ file-name RECORD
[ INTO identifier ] { AT END      } statement-1
                    { INVALID KEY }
[ ,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]... } ]... .
                     { NEXT SENTENCE                    }
```

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    } { identifier-3 }
                                     { UP BY } { 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 } GIVING identifier-n [ ROUNDED ]
     { literal-m    }

[ identifier-p [ROUNDED] ] ...

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

Option 3

```
SUBTRACT { CORRESPONDING } identifier-1 FROM identifier-2
         { CORR          }

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

TERMINATE STATEMENT

FORM:

```
TERMINATE report-name-1 [,report-name-2] ... .
```

TRACE STATEMENT

FORM:

```
TRACE { ON  }
      { OFF }
```

UNSTRING STATEMENT

FORM:

```
UNSTRING identifier-1
[ DELIMITED BY [ALL] { identifier-2 }
                     { literal-1    }
[ , OR [ALL] { identifier-3 } ] ... ]
             { literal-2    }
INTO identifier-4 [, DELIMITER IN identifier-5]
[, COUNT IN identifier-6]
[, identifier-7 [, DELIMITER IN identifier-8]
[, COUNT IN identifier-9]] ...
[WITH POINTER identifier-10] [TALLYING IN identifier-11]
[; ON OVERFLOW statement-1]
```

USE STATEMENT

FORM:

Format 1

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

Format 2

```
USE { BEFORE } STANDARD [ { BEGINNING } ]
    { AFTER  }           { ENDING    }
```

```
[ { REEL  }  ]
[ { FILE  }  ] LABEL PROCEDURE ON { file-name-1  }
[ { UNIT  }  ]                    { INPUT       }
                                  { OUTPUT      }
                                  { I-O         }
                                  { INPUT-OUTPUT }  .
```

Format 3

```
USE BEFORE REPORTING identifier-1 .
```

WRITE STATEMENT

FORM:

Option 1

```
WRITE record-name-1 [ FROM identifier-1 ]

[ { BEFORE } ADVANCING { identifier-2 LINES }  ]
[ { AFTER  }           { 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 { word-2       }
                      { identifier-1 }
                      { procedure-name-1 }

  [ , word-3 BY { word-4       }  ]  ...  ]  .
                { identifier-2 }
                { procedure-name-2 }
```

UTILITY PROGRAMS

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

LIBARY

The command string to LIBARY is:

```
. R LIBARY
*FILE1 .LIB,FILE2.LST=FILE3.LIB
```

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

TABLE I
Commands for Positioning Files

Command	Function
INSERT library-name	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.
INSERT library-name, dev:file.ext [ppn]	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.
DELETE library-name	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.
REPLACE library-name	The program does a DELETE followed by an INSERT.
REPLACE library-name, dev:file.ext [ppn]	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.
CORRECT library-name	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.

TABLE I (CONT.)
Commands for Positioning Files

Command	Function
END	The remainder of the input file is copied to the scratch file, and the output file is created, and the program then terminates.
RESTART	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.
EXTRACT library-name dev:file.ext [ppn]	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.

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

Command	Function
Dnnnnnn	The input file is copied to the scratch file until nnnnnn, the specified line, is encountered. That line is then skipped.
Innnnnn COBOL-statement	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.
Rnnnnnn COBOL-statement	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.

SORT

The command string to SORT is:

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 III

Switch	Meaning
/A	The file is recorded in ASCII mode.
/Bn	A block contains n records; n is a decimal number. If the /B switch is omitted, it is assumed that the file is unblocked.
/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. If this parameter is omitted, a numeric field is assumed to have an operational sign (S). 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.)

Switch	Meaning
/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 or /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., /Tdev$_1$.dev$_2$...).

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 lebels 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 produce 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 IV
COBOL Switch Summary

Switch	Action by Compiler
A	Allows the listing of the code generated (the source program is listed whenever a listing file is specified).
C	Produces a cross-reference listing of all user-defined items in the source program.
E	Checks the program for errors but does not generate code.
H	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.
I	Forces the compiler to suppress generation of a starting address for a main program.
J	Forces the compiler to generate a starting address for a subprogram.
L	Uses the preceding file descriptor as a library file whenever it encounters the COPY verb. 1. This switch is legal only with source files. 2. The file descriptor is not part of the main program. 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.
M	Prints a map showing the parameters of each user-defined item (e.g., data-names and procedure-names).
N	The source errors are not typed on the user's terminal.

TABLE IV (CONT.)
COBOL Switch Summary

Switch	Action by Compiler
P	Indicates production mode. Trace calls are not generated and user symbols are suppressed.
R	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.
S	The source file is in conventional format (with sequence numbers in columns 1-6 and with comments starting in column 73).
W	Rewinds the device before reading or writing. (This is valid for magnetic tape only.)
Z	Clears the directory of the device before writing. (This is valid for output DECtape only.)

EXAMPLE: *MYPROG.REL,MYPROG.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.

A

ACCEPT
ACCESS
ACTUAL
ADD
*ADDRESS
ADVANCING
AFTER
ALL
ALPHABETIC
ALTER
ALTERNATE
AND
**ANY
ARE
AREA
AREAS
ASCENDING
**ASCII
ASSIGN
AT
AUTHOR

B

BEFORE
BEGINNING
**BINARY
BLANK
BLOCK
BY

C

**CALL
**CANCEL
CD
CF
CH
**CHANNEL
CHARACTERS
CLASS
*CLOCK-UNITS
CLOSE
COBOL
CODE
COLUMN
COMMA
COMMUNICATION
COMP
**COMP-1
**COMPILE
COMPUTATIONAL
**COMPUTATIONAL-1
COMPUTE
CONFIGURATION
**CONSOLE
CONTAINS
CONTROL
CONTROLS
COPY
CORR
CORRESPONDING
COUNT
CURRENCY
**CURRENT

D

DATA
**DATABASE-KEY
DATE
DATE-COMPILED
DATE-WRITTEN
**DBKEY
DE
DECIMAL-POINT
DECLARATIVES
**DECSYSTEM-10
**DECSYSTEM10
**DEFERRED
**DELETE
DELIMITED
DELIMITER
**DENSITY
DEPENDING
DEPTH
DESCENDING
DESTINATION
DETAIL
DISABLE
DISPLAY
**DISPLAY-6
**DISPLAY-7
DIVIDE
DIVISION
DOWN
**DUP
**DUPLICATE

E

**EBCDIC
EGI
ELSE
EMI
**EMPTY
ENABLE
END
ENDING
ENTER
**ENTRY
ENVIRONMENT
**EPI
EQUAL
EQUALS
ERROR
ESI
**EVEN
EVERY
EXAMINE
**EXCL
**EXCLUSIVE
EXIT

F

FD
FILE
FILE-CONTROL
FILE-LIMIT
FILE-LIMITS
FILLER
FINAL
**FIND
FIRST
FOOTING
FOR
**FORTRAN-IV
**FORTRAN
FROM

G

GENERATE
**GET
GIVING
GO
**GOBACK
GREATER
GROUP

H

HEADING
HIGH-VALUE
HIGH-VALUES

I

I-O
I-O-CONTROL
**ID
IDENTIFICATION
IF
IN
INDEX
INDEXED
INDICATE
INITIAL
INITIATE
INPUT
INPUT-OUTPUT
**INSERT
INSTALLATION
INTO
INVALID
**INVOKE
IS

J

JUST
JUSTIFIED

K

KEY
KEYS

L

LABEL
LAST
LEADING
LEFT
LENGTH
LESS
LIMIT
LIMITS
LINE
LINE-COUNTER
LINES
**LINKAGE
LOCK
LOW-VALUE
LOW-VALUES

M

**MACRO
**MEMBER
MEMORY
MESSAGE
MODE
**MODIFY
MODULES
MOVE
MULTIPLE
MULTIPLY

N

NEGATIVE
NEXT
NO
NOT
NOTE
NUMBER
NUMERIC

O

OBJECT-COMPUTER
OCCURS
**ODD
OF
OFF
OMITTED
ON
**ONLY
OPEN
OPTIONAL
OR
OUTPUT
OVERFLOW
**OWNER

P

PAGE
PAGE-COUNTER
**PARITY
**PDP-10
PERFORM
PF
PH
PIC
PICTURE
PLUS
POINTER
POSITION
POSITIVE
**PRIOR
**PRIVACY
PROCEDURE
PROCEED
PROCESSING
**PROGRAM
PROGRAM-ID
**PROT
**PROTECTED

Q

QUEUE
QUOTE
QUOTES

R

RANDOM
RD
READ
RECEIVE
RECORD
**RECORDING
RECORDS
REDEFINES
REEL
**RELATIVE
RELEASE
REMAINDER
REMARKS
**REMOVE
RENAMES
REPLACING
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RESET
**RETR
**RETRIEVAL
RETURN
*REVERSED
REWIND
**REWRITE
RF
RH
RIGHT
ROUNDED
RUN
**RUN-UNIT

S

SAME
**SCHEMA
SD
SEARCH
SECTION
SECURITY
SEEK
SEGMENT
SEGMENT-LIMIT
SELECT
**SELECTIVE
SEND
SENTENCE
SEQUENTIAL
SET
**SETS
*SIGN
**SIXBIT
SIZE
SORT
SOURCE
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STATUS
STOP
**STORE
STRING
SUB-QUEUE-1
SUB-QUEUE-2
SUB-QUEUE-3
**SUB-SCHEMA
SUBTRACT
SUM
**SUPPRESS
**SWITCH
SYMBOLIC
SYNC
SYNCHRONIZED

T

TABLE
TALLY
TALLYING
TAPE
TERMINAL
TERMINATE
TEXT
THAN
THROUGH
THRU
TIME
TIMES
TO
**TODAY
**TRACE
TYPE

U

UNIT
UNSTRING
UNTIL
UP
**UPDATE
**UPDATES
UPON
USAGE
**USAGE-MODE
USE
**USER-NUMBER
USING

V

VALUE
VALUES
VARYING

W

WHEN
WITH
**WITHIN
WORDS
WORKING-STORAGE
WRITE

Z

ZERO
ZEROES
ZEROS

TABLE V

The following table gives the collating sequences for ASCII (DISPLAY-7) and SIXBIT (DISPLAY-6) fields as used in condition comparisons.

SIXBIT	Character	ASCII 7-Bit	SIXBIT	Character	ASCII 7-Bit
00	Space	040	60	P	120
01	!	041	61	Q	121
02	"	042	62	R	122
03	#	043	63	S	123
04	$	044	64	T	124
05	%	045	65	U	125
06	&	046	66	V	126
07	'	047	67	W	127
10	(	050	70	X	130
11	)	051	71	Y	131
12	*	052	72	Z	132
13	+	053	73	[	133
14	,	054	74	\	134
15	-	055	75	]	135
16	.	056	76	↑	136
17	/	057	77	←	137
20	0	060	-	`	140
21	1	061	-	a	141
22	2	062	-	b	142
23	3	063	-	c	143
24	4	064	-	d	144
25	5	065	-	e	145
26	6	066	-	f	146
27	7	067	-	g	147
30	8	070	-	h	150
31	9	071	-	i	151
32	:	072	-	j	152
33	;	073	-	k	153
34	<	074	-	l	154
35	=	075	-	m	155
36	>	076	-	n	156
37	?	077	-	o	157
40	@	100	-	p	160
41	A	101	-	q	161
42	B	102	-	r	162
43	C	103	-	s	163
44	D	104	-	t	164
45	E	105	-	u	165
46	F	106	-	v	166
47	G	107	-	w	167
50	H	110	-	x	170
51	I	111	-	y	171
52	J	112	-	z	172
53	K	113	-	{	173
54	L	114	-	¦	174
55	M	115	-	}	175
56	N	116	-	~	176
57	O	117	-	Delete	177

Created by

DECsystem-10 Software Documentation Group

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.