MICRO FOCUS

Micro Focus RM/COBOL

RM/COBOL Syntax Summary

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
http://www.microfocus.com

© Copyright 2017-2020 Micro Focus or one of its affiliates.

The only warranties for products and services of Micro Focus and its affiliates and licensors (“Micro Focus”) are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Revised 2020-05-07 for version 12.17

Contents

RM/COBOL Commands 1

Compile Command 1
Runtime Command 3
Debug Command 4

RM/COBOL Language Syntax 9

Source Program General Format 9
Identification Division General Format 9
Environment Division General Format 10
File Control Entry General Formats 12
Data Division General Format 14
file-description-entry 15
sort-merge-file-description-entry 15
record-description-entry 16
77-level-description-entry 16
data-description-entry 16
communication-description-entry 19
screen-description-entry 20
Procedure Division General Formats 23
Procedure Division Verbs 24
ACCEPT Statement 24
ADD Statement 26
ALTER Statement 27
CALL Statement 27
CALL PROGRAM Statement 28
CANCEL Statement 28
CLOSE Statement 29
COMPUTE Statement 29
CONTINUE Statement 29
DELETE Statement 29
DELETE FILE Statement 29
DISABLE Statement 30
DISPLAY Statement 30
DIVIDE Statement 32
ENABLE Statement 33
ENTER Statement 33
EVALUATE Statement 34
EXIT Statement 34
GOBACK Statement 35
GO TO Statement 35
IF Statement 35
INITIALIZE Statement 36
INSPECT Statement 36

MERGE Statement 38
MOVE Statement 38
MULTIPLY Statement 38
OPEN Statement 39
PERFORM Statement 39
PURGE Statement 41
READ Statement 41
RECEIVE Statement 41
RELEASE Statement 42
RETURN Statement 42
REWRITE Statement 42
SEARCH Statement 42
SEND Statement 43
SET Statement 44
SORT Statement 45
START Statement 46
STOP Statement 46
STRING Statement 47
SUBTRACT Statement 47
UNLOCK Statement 48
UNSTRING Statement 48
USE Statement 49
WRITE Statement 49
END PROGRAM Header General Format 50
COPY and REPLACE Statement General Formats 50
General Formats for Conditions 51
General Formats for Qualification 53
Miscellaneous Formats 54
Sentence 54
Statement Sequence 54
Subscripting 55
Reference Modification 55
Identifier 55
Special Registers 55
Figurative Constants 56
Concatenation Expression 56
Constant-Expression 57
PICTURE Character-String (Data Categories) 57
PICTURE Symbols 60
LIKE Pattern Grammar 63
Directives 69
Program Structure 70
General Format for Nested Source Programs 70
General Format for *nested-source-program* 70
General Format for a Sequence of Source Programs 71
COBOL Words 71
Reserved Words (A - B) 72
Reserved Words (C) 72
Reserved Words (D) 73
Reserved Words (E) 73
Reserved Words (F - I) 74
Reserved Words (J - N) 75
Reserved Words (O - Q) 75
Reserved Words (R) 76
Reserved Words (S) 76
Reserved Words (T - Z) 77

Unused Reserved Words....77
Context-Sensitive Words....78
Nonreserved System-Names....80

RM/COBOL Language Examples....83

ACCEPT Statement Examples....83
Add Statement Example....87
Alter Statement Example....88
CALL Statement Example....89
CALL Program Statement Example....91
CANCEL Statement Example....92
CLOSE Statement Example....93
COMPUTE Statement Example....94
CONTINUE Statement Example....96
DELETE Statement Example....96
DELETE FILE Statement Example....98
DISABLE Statement Example....99
DISPLAY Statement Examples....99
DIVIDE Statement Example....102
ENABLE Statement Example....103
ENTER Statement Example....103
EVALUATE Statement Example....104
EXIT Statement Example....105
GOBACK Statement Example....107
GO TO Statement Example....107
IF Statement Example....108
INITIALIZE Statement Example....109
INSPECT Statement Example....110
MERGE Statement Example....115
MOVE Statement Example....116
MULTIPLY Statement Example....117
OPEN Statement Example....117
PERFORM Statement Example....119
PURGE Statement Example....120
READ Statement Examples....121
RECEIVE Statement Example....124
RELEASE Statement Example....125
RETURN Statement Example....126
REWRITE Statement Example....127
SEARCH Statement Example....128
SEND Statement Example....131
SET Statement Example....132
SORT Statement Example....133
START Statement Example....134
STOP Statement Example....136
STRING Statement Example....136
SUBTRACT Statement Example....137
UNLOCK Statement Example....138
UNSTRING Statement Example....139
USE Statement Example....140
WRITE Statement Examples....140

Index....143

RM/COBOL Commands

Compile Command

The format of the Compile Command is as follows:

```
rmcobol filename [[()] [ option ] ... [)comment]]
```

filename is the name of the source file to be compiled.

option specifies a compiler option, described below. A tilde (~) preceding the option character negates the option. Options may be specified in either uppercase or lowercase letters. If an option is repeated in a command, the last occurrence of the option is used. Each option may be preceded by a hyphen. If any option is preceded by a hyphen, then a leading hyphen must precede all options. When assigning a value to an option, the equal sign is optional if leading hyphens are used.

comment is used to annotate the command.

A summary of the options for the Compile Command is shown in the following table. (For further information, see Chapter 6: *Compiling* of the *RM/COBOL User's Guide*.)

Option	Description
A	Directs the compiler to generate the allocation map in the listing.
B	Defines as binary sequential those sequential files not explicitly declared to be line sequential in their file control entries.
C[=*n*]	Suppresses the inclusion of copied text, replaced text, replacement text, or COPY statement text in the listing. *n* can be 0 to 15. Specifying C is equivalent to C=1.
D	Directs RM/COBOL to compile all source programs as if the WITH DEBUGGING MODE clause appeared in each compiled program.

Option	Description
E	Suppresses the inclusion of the source program component in the listing except for lines associated with diagnostic messages.
F={(*keyword-list*)|*keyword*}	Directs the compiler to flag occurrences of these language elements: COM1 INTERMEDIATE COM2 OBSOLETE EXTENSION SEG1 HIGH SEG2 If leading hyphens are used, the parentheses are optional.
G=*pathname*	Designates a file to be used as the primary compiler configuration.
H=*pathname*	Designates a file as a supplement to the compiler configuration.
K	Suppresses the banner message and the terminal error listing.
L[=*pathname*]	Directs the compiler to produce a listing file and optionally specify the directory in which to place the listing file.
M	Directs the compiler to suppress automatic input conversion for Format 1 and 3 ACCEPT statements with numeric operands and to suppress right justification of justified operands. Direct the compiler to suppress automatic output conversion for numeric fields of Format 3 DISPLAY statements.
N	Suppresses the generation of an object program.
O=*pathname*	Specifies the directory pathname where the object file will be placed.
P	Directs the compiler to write a copy of the listing to the printer.
Q	Directs the compiler to eliminate debugging information from generated object programs.
R	Directs the compiler to generate a sequential number in the first six columns of source records as they appear on the listing.
S	Directs the compiler to assume a separate sign when the SIGN clause is not specified for a DISPLAY usage, signed numeric data item (that is, for a data item whose character-string within a PICTURE clause begins with S).
T	Directs the compiler to write a copy of the listing to the standard output device.

Option	Description
U[={**B**|**D**|**P**}]	Directs the compiler to assume an alternative usage for data items described as COMP or COMPUTATIONAL. • The U Option specified alone or as U=B directs the compiler to assume BINARY usage for data items described as COMP or COMPUTATIONAL. • The U=D Option directs the compiler to assume DISPLAY usage for items described as COMP or COMPUTATIONAL. • The U=P Option directs the compiler to assume PACKED-DECIMAL usage for items described as COMP or COMPUTATIONAL.
V	Defines as line sequential those sequential files not explicitly declared to be binary sequential in their file control entries.
W=*n*	Specifies the amount of memory (in kilobytes) that the compiler should use for its internal table storage. *n* can be a decimal number from 32 to 524288.
X	Directs the compiler to generate a cross reference map in the listing.
Y[=*n*]	Directs the compiler to output the symbol table and debug line table to the object program file. *n* can be 0 to 3. Specifying Y is equivalent to Y=1.
Z=*version*	Indicates the object version of the RM/COBOL runtime you want to use. *version* can be 9 through 15.
2	Directs the compiler to accept source programs created for the RM/COBOL 2.*n* compiler.
7	Specifies the semantic rules under which the program is to be compiled as conforming to the American National Standard COBOL 1974.

Runtime Command

The format of the Runtime Command is as follows:

```
runcobol filename [ option ] ...
```

filename is the name of the main program of the run unit.

option specifies a runtime system option, described below. Options may be specified in either uppercase or lowercase letters. Each option may be preceded by a hyphen. If any option is preceded by a hyphen, then a leading hyphen must precede all options. When assigning a value to an option, the equal sign is optional if leading hyphens are used.

A summary of the options for the Runtime Command is shown in the following table. (For further information, see Chapter 7: *Running* of the *RM/COBOL User's Guide*.)

Option	Description
A=[*delim*] [*string*] [*delim*]	Passes an argument to the main program. The delimiter characters are optional if *string* does not contain spaces.
B=*n*	Specifies a maximum buffer size for use with the ACCEPT and DISPLAY statements.
C=*pathname*	Designates a file to be used as the primary runtime configuration file.
D	Invokes the RM/COBOL Interactive Debugger.
F=*fillchar*	Uses *fillchar* instead of space to preset read-write memory upon program load.
I	Collects RM/COBOL program instrumentation data.
K	Suppresses the banner message and the STOP RUN message.
L=*pathname*	Designates RM/COBOL non-COBOL subprogram libraries.
M	Directs that level 2 ANSI semantics are to be used for Format 1 ACCEPT and DISPLAY statements.
P[=Y|N]	Directs that the runtime window be persistent or not persistent after the COBOL program terminates on Windows. (The P option is for Windows only; the P option is not valid or meaningful on UNIX.)
Q=[*delim*] [*string*] [*delim*]	Specifies the value used to initialize the SYMBOLIC QUEUE, SYMBOLIC SUB-QUEUE-1, SYMBOLIC SUB-QUEUE-2, and SYMBOLIC SUB-QUEUE-3 area in a CD FOR INITIAL INPUT record area or the SYMBOLIC TERMINAL area in a CD FOR INITIAL I-O record area. The delimiter characters are optional if *string* does not contain spaces.
S=*n . . . n*	Sets (or resets) the initial value of switches in the RM/COBOL program.
T=*n*	Specifies the amount of memory (*n* bytes) to be used for a sort operation.
V	Directs that a trace of support modules loaded by the RM/COBOL runtime system be displayed.
X=*pathname*	Designates a file as a supplement to the runtime configuration.

Debug Command

A summary of the options for the Debug Command are shown in the following table. (For further information on the Debug commands, see Chapter 9: *Debugging* of the *RM/COBOL User's Guide*.)

Note In the Address-Size formats for the D, M, T, and U commands, *base* is one of the following:

- **U** *arg-num* for a formal argument, and *arg-num* is the formal argument number.
- **B** *item-num* for a based linkage item, and *item-num* is the based linkage item number.
- **G** for the GIVING formal argument.
- **X** *ext-num* for an external data item, and *ext-num* is the external item number.

Command	Description
A (Address Stop)	Sets a single-time breakpoint at a specific procedure division statement, paragraph, or section, and resumes program execution from the current location. **A** [*line* [+ *intraline*] [, [*prog-name*] [, [*count*]]]]
B (Breakpoint)	Sets a multi-time breakpoint at a specific procedure division statement, paragraph, or section, or displays all currently active breakpoints when the optional command operand is omitted. **B** [*line* [+ *intraline*] [, [*prog-name*] [, [*count*]]]]
C (Clear)	Clears an active breakpoint that has been set with the A or B Commands or clears all active breakpoints when the optional command operand is omitted. **C** [*line* [+ *intraline*] [, [*prog-name*]]]
D (Display)	Displays the value of a specified data item on the screen. **Identifier Format** **D** *name-1* [{ IN | OF } *name-2*] … [*script*] [*refmod*] [, { *type* | { * | & } [*type*] }] [*# alias*] **Address-Size Format** **D** [*base* :] *address* [+ *occur-size* * *occur-num*] …, *size* , [*type*] [*# alias*] **Alias Format** **D** *# alias*
E (End)	Ends debugging and resumes program execution. **E**
L (Line Display)	Specifies a line on the monitor screen at which command input echoes and Debug responses are to be displayed. **L** [*line-display*]

Command	Description
M (Modify)	Modifies the value of a specified data item. **Identifier Format** **M** *name-1* [{ IN | OF } *name-2*] … [*script*] [*refmod*] [, { *type* | { * | & } [*type*] }] [*# alias*] , *value* **Address-Size Format** M [*base* :] *address* [+ *occur-size* * *occur-num*] …, *size* , [*type*] [*# alias*] , *value* **Alias Format** **M** *# alias* , *value*
Q (Quit)	Quits debugging and program execution; control is returned to the operating system immediately as if a STOP RUN statement had been executed. **Q**
R (Resume)	Resumes program execution at the current location or at a specific procedure division statement, paragraph, or section specified in the command. **R** [*statement-address*]
S (Step)	Steps to the start of the next statement, paragraph, or section a specified number of times while tracing execution at each statement step. If P and S are omitted, a statement step is done. P specifies a step to next paragraph. S specifies a step to next section. A single step is done if *count* is omitted. **S** [**P** | **S**][*count*]
T (Trap)	Monitors the value of a specified data item, and suspends execution whenever a change in that value occurs; that is, activates a data trap or displays all activated data traps. **Identifier Format** **T** *name-1* [{ IN | OF } *name-2*] … [*script*] [*refmod*] [, { *type* | { * | & } [*type*] }] [*# alias*] **Address-Size Format** **T** [*base* :] *address* [+ *occur-size* * *occur-num*] …, *size* , [*type*] [*# alias*] **Alias Format** **T** *# alias* **Display All Traps Format** **T**

Command	Description
U (Untrap)	Clears some or all currently activated data traps. **Identifier Format** **U** *name-1* [{ IN | OF } *name-2*] … [*script*] [*refmod*] [, { *type* | { * | & } [*type*] }] **Address-Size Format** **U** [*base* :] *address* [+ *occur-size* * *occur-num*] …, *size* , [*type*] **Alias Format** **U** *# alias* **Clear All Traps Format** **U**

RM/COBOL Language Syntax

Source Program General Format

identification-division
[*environment-division*]
[*data-division*]
[*procedure-division*]
[*nested-source-program*]···
[*end-program-header*]

Identification Division General Format

{ IDENTIFICATION / ID } DIVISION.

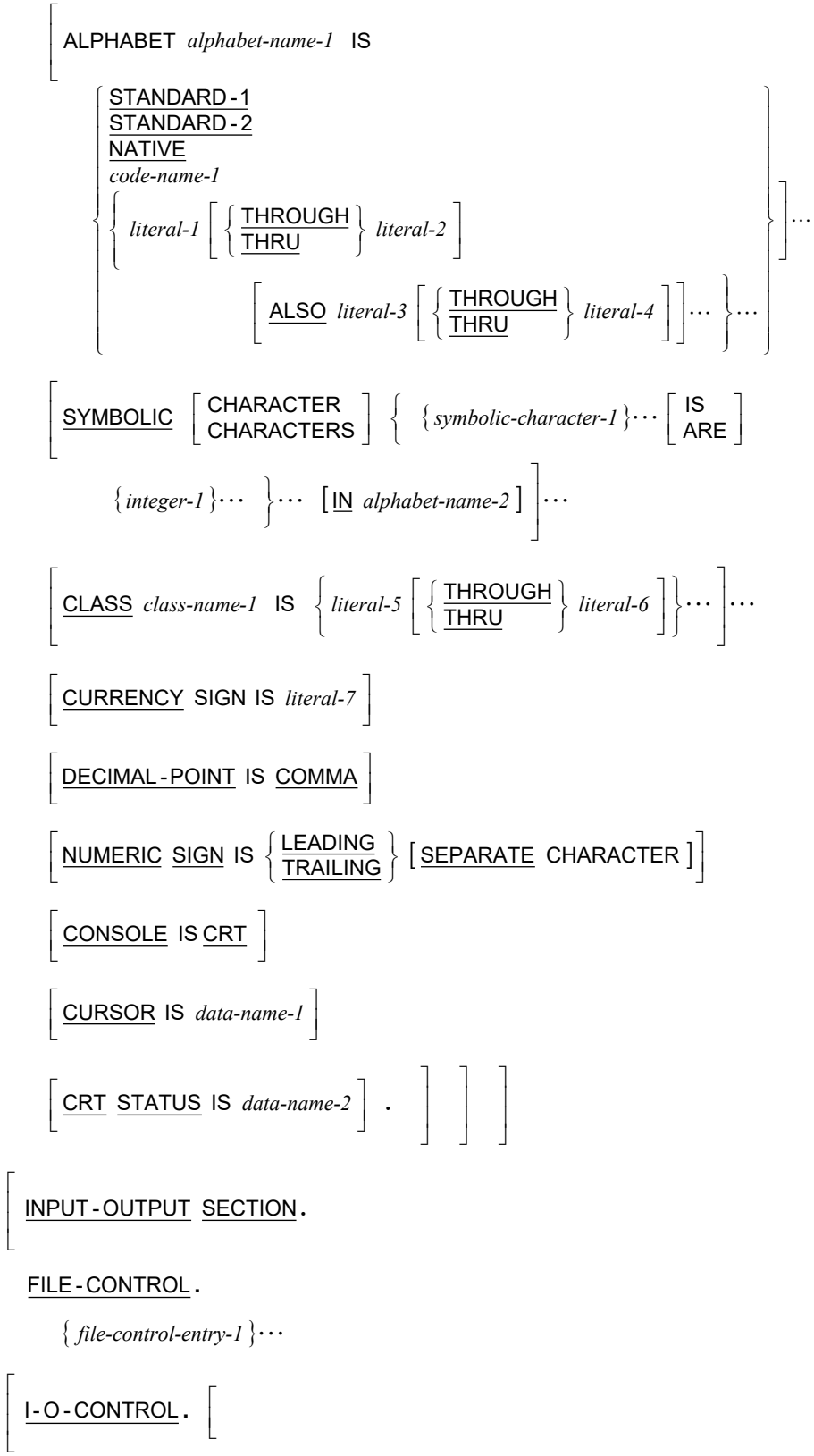
PROGRAM-ID. { *program-name-1* / *literal-1* } [IS { COMMON / INITIAL } PROGRAM].

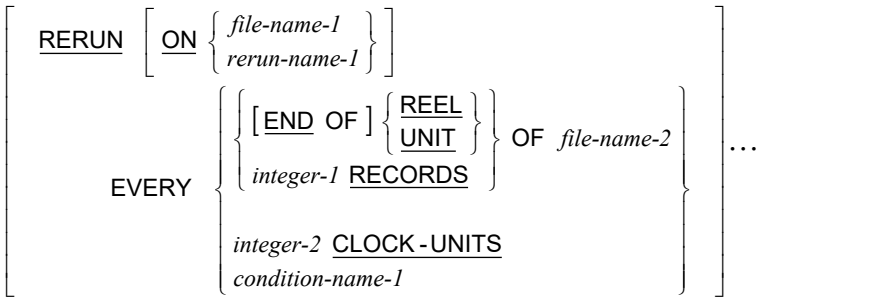
[AUTHOR. [*comment-entry-1*]···]
[INSTALLATION. [*comment-entry-2*]···]
[DATE-WRITTEN. [*comment-entry-3*]···]
[DATE-COMPILED. [*comment-entry-4*]···]
[SECURITY. [*comment-entry-5*]···]

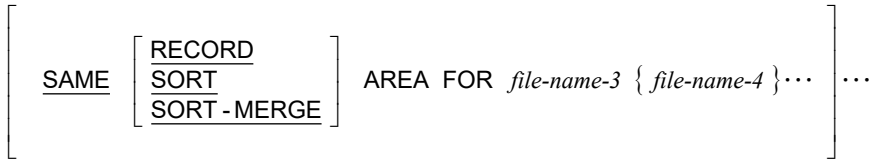
```
[ REMARKS. [ comment-entry-6 ]... ]
```

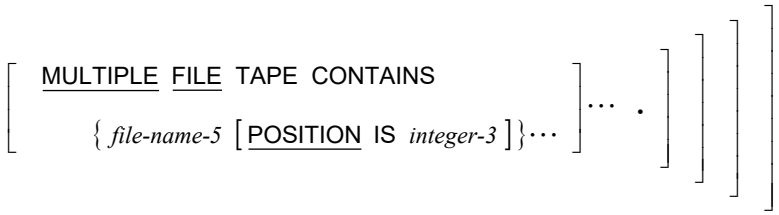
Environment Division General Format

```
[ ENVIRONMENT DIVISION. ]

[ CONFIGURATION SECTION. ]

[ SOURCE-COMPUTER. [ computer-name-1
    [ WITH DEBUGGING MODE ]. ] ]

[ OBJECT-COMPUTER. [ computer-name-2
    [ MEMORY SIZE integer-1 { WORDS | CHARACTERS | MODULES } ]
    [ PROGRAM COLLATING SEQUENCE IS alphabet-name-1 ]
    [ SEGMENT-LIMIT IS segment-number-1 ] . ] ]

[ SPECIAL-NAMES. [
    [ switch-name-1 { IS mnemonic-name-1 [ { ON STATUS IS condition-name-1 | OFF STATUS IS condition-name-2 } ]
                    | { ON STATUS IS condition-name-1 | OFF STATUS IS condition-name-2 } ] ...
    feature-name-1 IS mnemonic-name-2
    low-volume-I-O-name-1 IS mnemonic-name-3 ] ]
```


ALPHABET alphabet-name-1 IS
STANDARD-1
STANDARD-2
NATIVE
code-name-1
literal-1 THROUGH THRU literal-2
ALSO literal-3 THROUGH THRU literal-4
SYMBOLIC CHARACTER CHARACTERS
symbolic-character-1 IS ARE
integer-1
IN alphabet-name-2
CLASS class-name-1 IS literal-5 THROUGH THRU literal-6
CURRENCY SIGN IS literal-7
DECIMAL-POINT IS COMMA
NUMERIC SIGN IS LEADING TRAILING SEPARATE CHARACTER
CONSOLE IS CRT
CURSOR IS data-name-1
CRT STATUS IS data-name-2
INPUT-OUTPUT SECTION.
FILE-CONTROL.
file-control-entry-1
I-O-CONTROL.

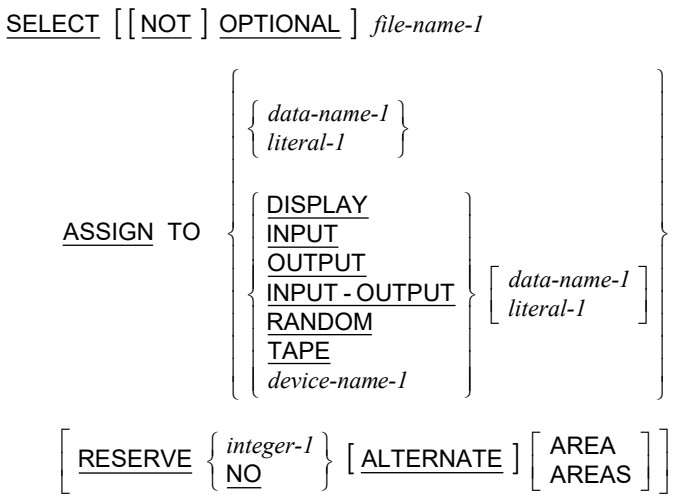


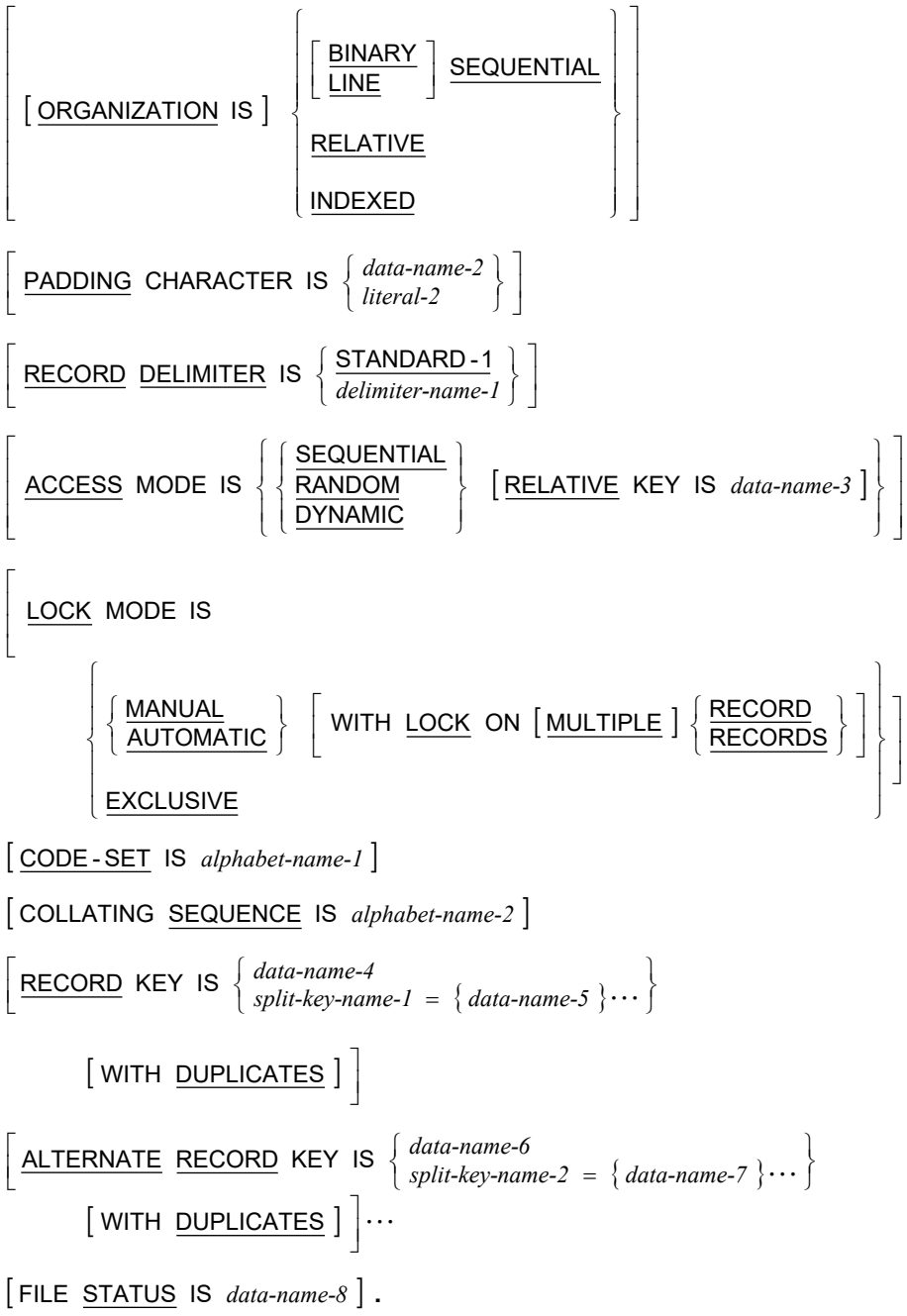




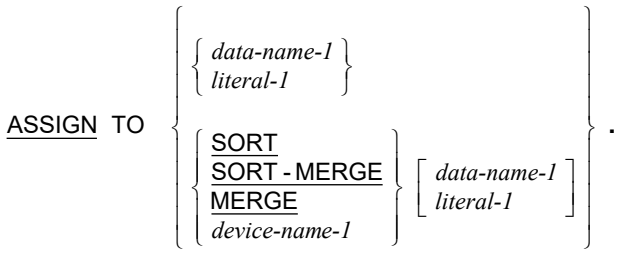
File Control Entry General Formats

file-control-entry

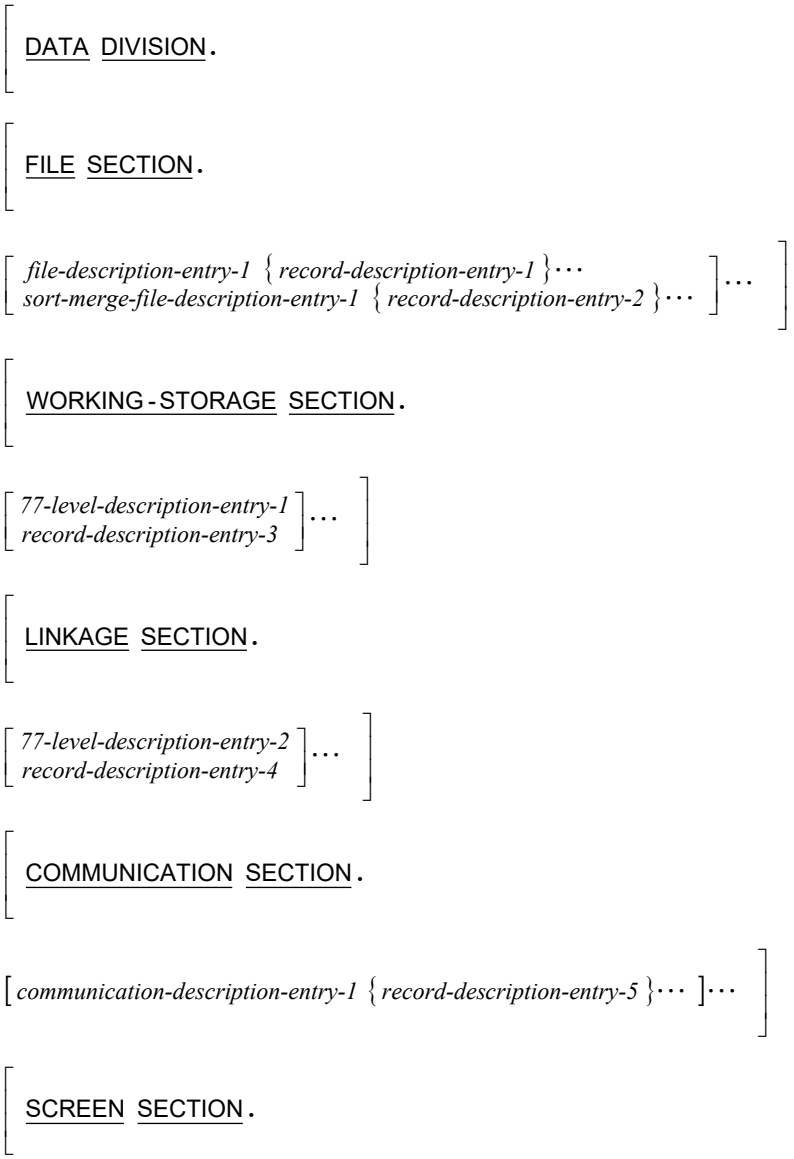




sort-merge-file-control-entry



Data Division General Format



```
[ screen-description-entry-1 ]... ] ]
```

file-description-entry

```
FD file-name-1

    [ IS EXTERNAL ]

    [ IS GLOBAL ]

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

    [ RECORD { CONTAINS [ integer-3 TO ] integer-4 CHARACTERS
             / IS VARYING IN SIZE
                 [ [ FROM integer-5 ] [ TO integer-6 ] CHARACTERS ]
                 [ DEPENDING ON data-name-1 ] } ]

    [ LABEL { RECORD IS / RECORDS ARE } { STANDARD / OMITTED } ]

    [ VALUE OF { { LABEL / label-name-1 } IS { data-name-2 / literal-1 } }... ]

    [ DATA { RECORD IS / RECORDS ARE } { data-name-3 }... ]

    [ LINAGE IS { data-name-4 / integer-7 } LINES [ WITH FOOTING AT { data-name-5 / integer-8 } ]
          [ LINES AT TOP { data-name-6 / integer-9 } ] [ LINES AT BOTTOM { data-name-7 / integer-10 } ] ]

    [ CODE-SET IS alphabet-name-1 ] .
```

sort-merge-file-description-entry

```
SD file-name-1
```

```
[ RECORD { CONTAINS [ integer-3 TO ] integer-4 CHARACTERS                          } ]
         { IS VARYING IN SIZE                                                      }
         {     [ [ FROM integer-5 ] [ TO integer-6 ] CHARACTERS ]                  }
         {     [ DEPENDING ON data-name-1 ]                                        }

[ DATA { RECORD IS  } { data-name-3 }... ] .
       { RECORDS ARE }
```

record-description-entry

```
{ data-description-entry-1 }...
```

77-level-description-entry

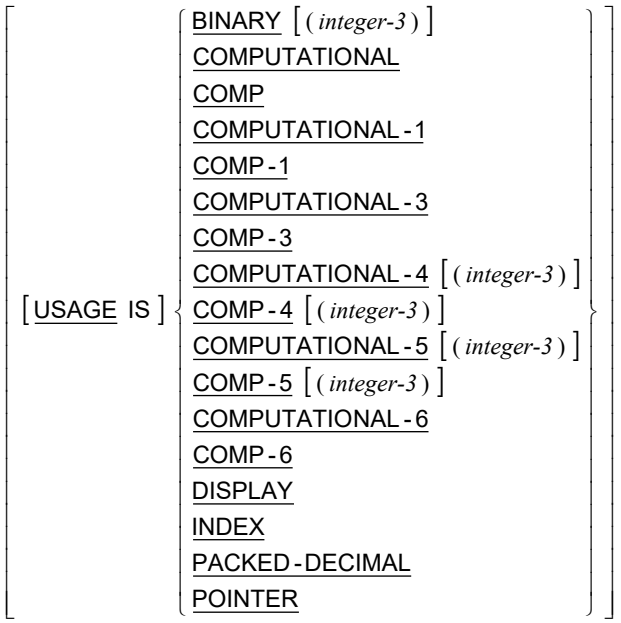
```
data-description-entry-2
```

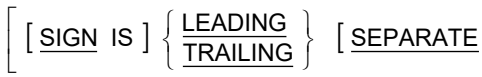
data-description-entry

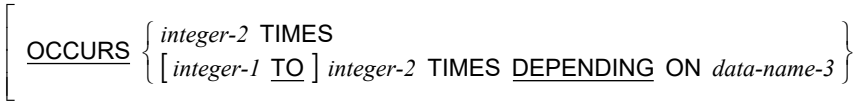
See also PICTURE Character-String (Data Categories) on pages 57 and PICTURE Symbols on page 60.

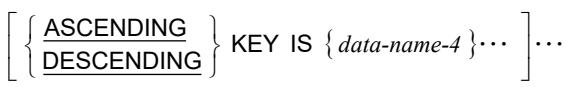
Format 1: Data-Name Full Declaration

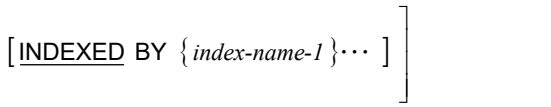
```
level-number-1 [ data-name-1 ]
               [ FILLER      ]

    [ REDEFINES data-name-2 ]
    [ IS EXTERNAL ]
    [ IS GLOBAL ]
    [ { PICTURE } IS character-string-1 ]
    [ { PIC     }                       ]
```

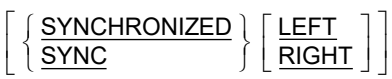












[BLANK WHEN ZERO]

[SAME AS *data-name-5*]

[VALUE IS *literal-1*].

Format 2: Data-Name Renames

66 *data-name-1*

```
RENAMES data-name-2 [ { THROUGH } data-name-3 ] .
                      { THRU    }
```

Format 3: Condition-Name Declaration

```
88 condition-name-1

{ VALUE  IS  }  { literal-1 [ { THROUGH } literal-2 ] }...
{ VALUES ARE }  {             { THRU    }           }
                {                                   }
                { relational-operator literal-1     }

[ WHEN SET TO FALSE IS literal-3 ] .
```

Format 4: Constant-Name Declaration

```
78 constant-name-1

VALUE IS { literal-1          } .
         { constant-expression-1 }
```

communication-description-entry

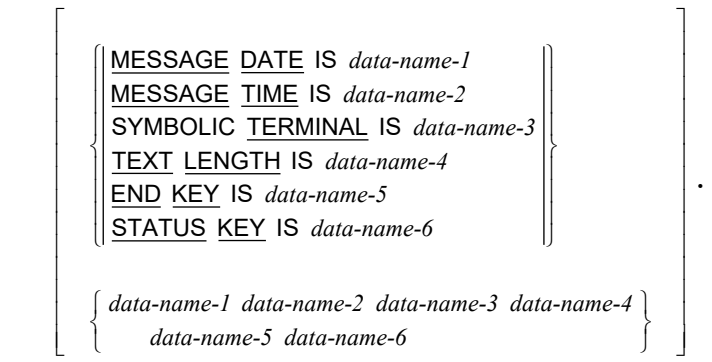
Format 1: Input CD

```
CD  cd-name-1  FOR [ INITIAL ] INPUT
    [
      { SYMBOLIC QUEUE IS data-name-1
        SYMBOLIC SUB-QUEUE-1 IS data-name-2
        SYMBOLIC SUB-QUEUE-2 IS data-name-3
        SYMBOLIC SUB-QUEUE-3 IS data-name-4
        MESSAGE DATE IS data-name-5
        MESSAGE TIME IS data-name-6
        SYMBOLIC SOURCE IS data-name-7
        TEXT LENGTH IS data-name-8
        END KEY IS data-name-9
        STATUS KEY IS data-name-10
        MESSAGE COUNT IS data-name-11 }

      { data-name-1 data-name-2 data-name-3 data-name-4
          data-name-5 data-name-6 data-name-7 data-name-8
          data-name-9 data-name-10 data-name-11 }
    ] .
```

Format 2: Output CD

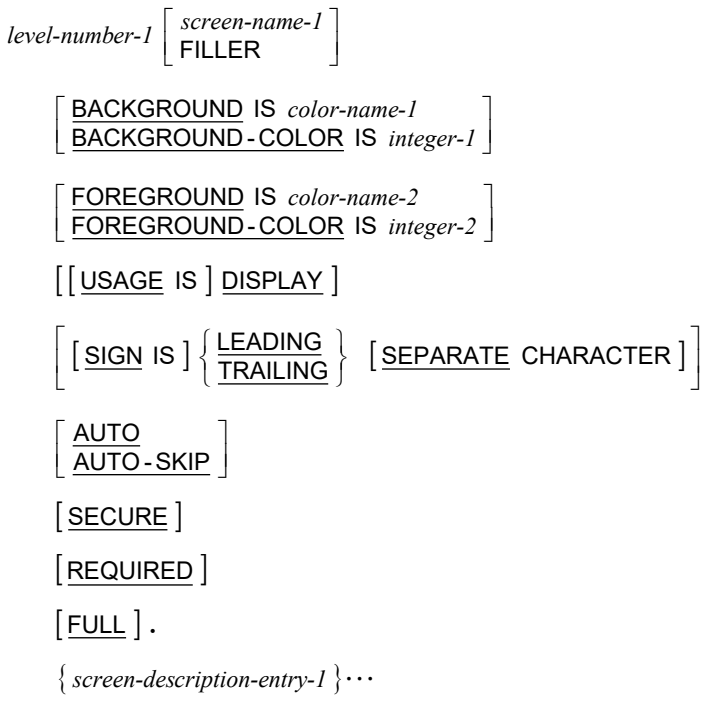
```
CD  cd-name-1  FOR OUTPUT
    [ DESTINATION COUNT IS data-name-1 ]
    [ TEXT LENGTH IS data-name-2 ]
    [ STATUS KEY IS data-name-3 ]
    [ DESTINATION TABLE OCCURS integer-1 TIMES
          [ INDEXED BY { index-name-1 }... ] ]
    [ ERROR KEY IS data-name-4 ]
    [ SYMBOLIC DESTINATION IS data-name-5 ] .
```

Format 3: Input-Output CD

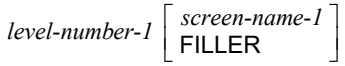


screen-description-entry

Format 1: Screen Group



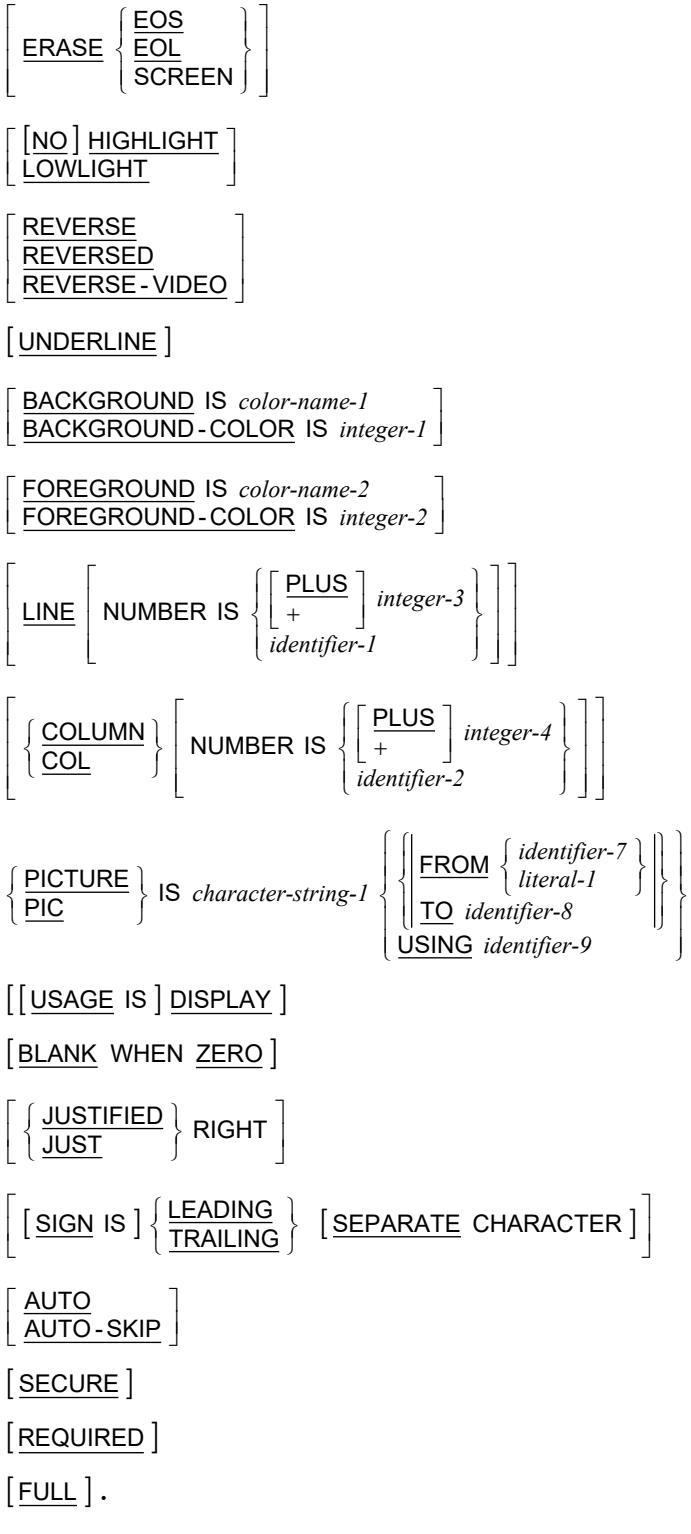
Format 2: Screen Literal



```
[ BELL ]
[ BEEP ]

[ BLANK { SCREEN   } ]
[       { LINE     } ]
[       { REMAINDER } ]

[ BLINK ]

[ ERASE { EOS    } ]
[       { EOL    } ]
[       { SCREEN } ]

[ [NO] HIGHLIGHT ]
[ LOWLIGHT       ]

[ REVERSE         ]
[ REVERSED        ]
[ REVERSE-VIDEO   ]

[ UNDERLINE ]

[ BACKGROUND IS color-name-1 ]
[ BACKGROUND-COLOR IS integer-1 ]

[ FOREGROUND IS color-name-2 ]
[ FOREGROUND-COLOR IS integer-2 ]

[ LINE [ NUMBER IS { [ PLUS ] integer-3 } ] ]
[                  { [ +    ]            } ]
[                  { identifier-1         } ]

[ { COLUMN } [ NUMBER IS { [ PLUS ] integer-4 } ] ]
[ { COL    } [           { [ +    ]            } ] ]
[                        { identifier-2         } ]

[ [ VALUE IS ] literal-1 ] .
```

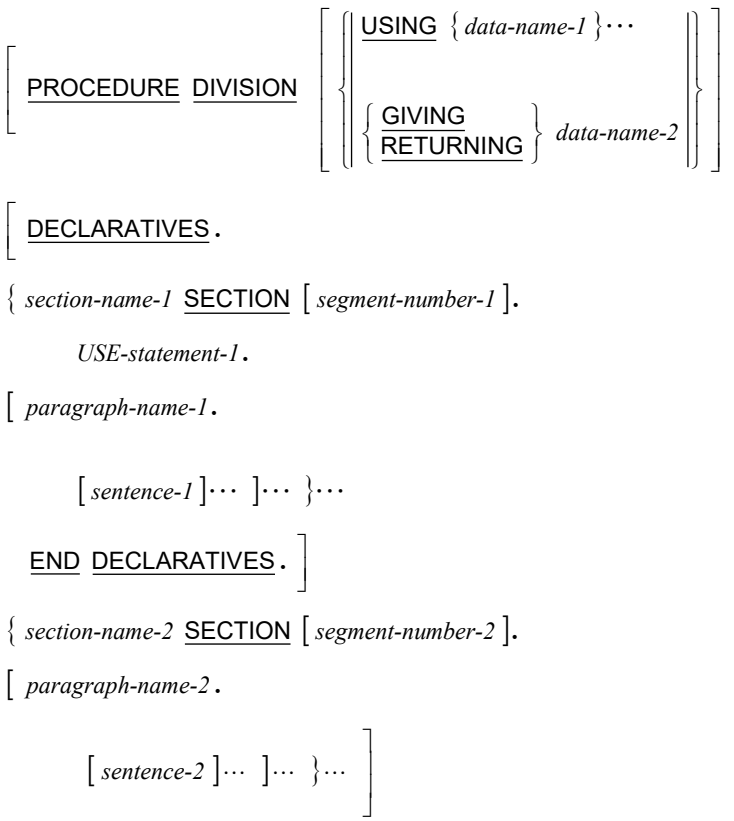
Format 3: Screen Field

```
level-number-1 [ screen-name-1 ]
               [ FILLER        ]

[ BELL ]
[ BEEP ]

[ BLANK { SCREEN   } ]
[       { LINE     } ]
[       { REMAINDER } ]

[ BLINK ]
```

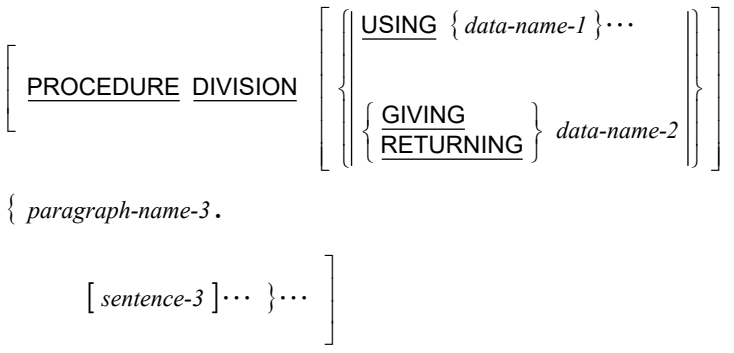

[ERASE { EOS / EOL / SCREEN }]
[[NO] HIGHLIGHT / LOWLIGHT]
[REVERSE / REVERSED / REVERSE-VIDEO]
[UNDERLINE]
[BACKGROUND IS color-name-1 / BACKGROUND-COLOR IS integer-1]
[FOREGROUND IS color-name-2 / FOREGROUND-COLOR IS integer-2]
[LINE [NUMBER IS { [PLUS / +] integer-3 / identifier-1 }]]
[{ COLUMN / COL } [NUMBER IS { [PLUS / +] integer-4 / identifier-2 }]]
{ PICTURE / PIC } IS character-string-1 { { [FROM { identifier-7 / literal-1 }] [TO identifier-8] } / USING identifier-9 }
[[USAGE IS] DISPLAY]
[BLANK WHEN ZERO]
[{ JUSTIFIED / JUST } RIGHT]
[[SIGN IS] { LEADING / TRAILING } [SEPARATE CHARACTER]]
[AUTO / AUTO-SKIP]
[SECURE]
[REQUIRED]
[FULL] .

Procedure Division General Formats

Format 1: Declaratives or Sections



Format 2: Paragraphs



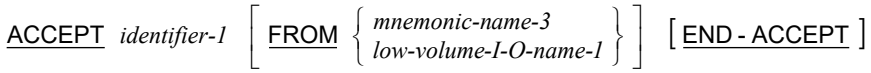
Procedure Division Verbs

This section presents the syntax of each Procedure Division statement. For detailed information on the syntax and meaning of each Procedure Division statement, see the *RM/COBOL Language Reference Manual*.

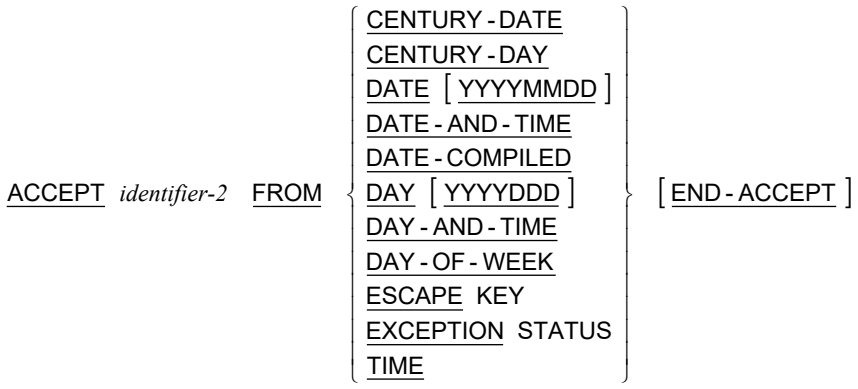
Examples illustrating the RM/COBOL language syntax for the procedure division verbs begin on page 83.

ACCEPT Statement

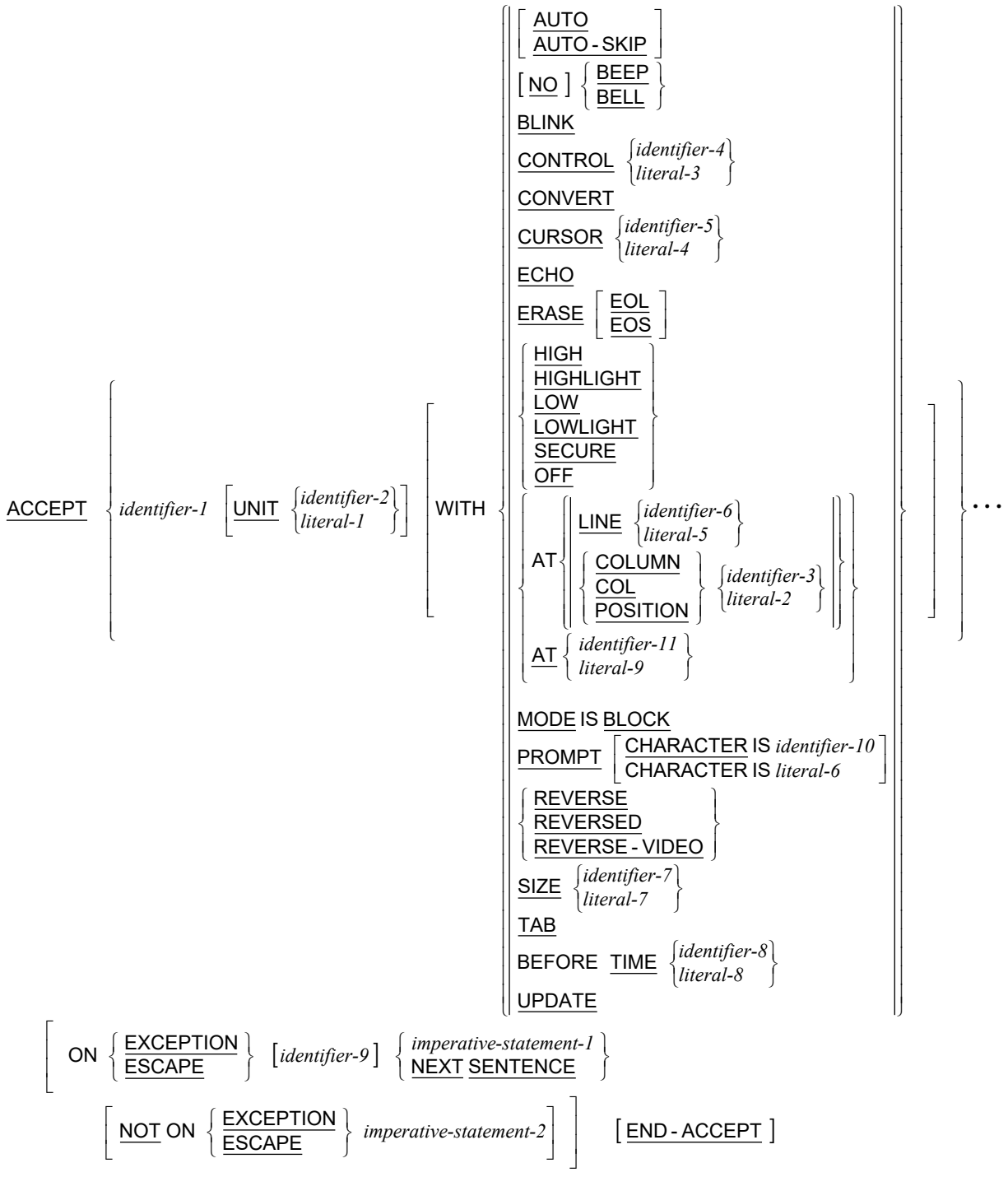
Format 1: Accept From System-Name



Format 2: Accept From Implicit Definition



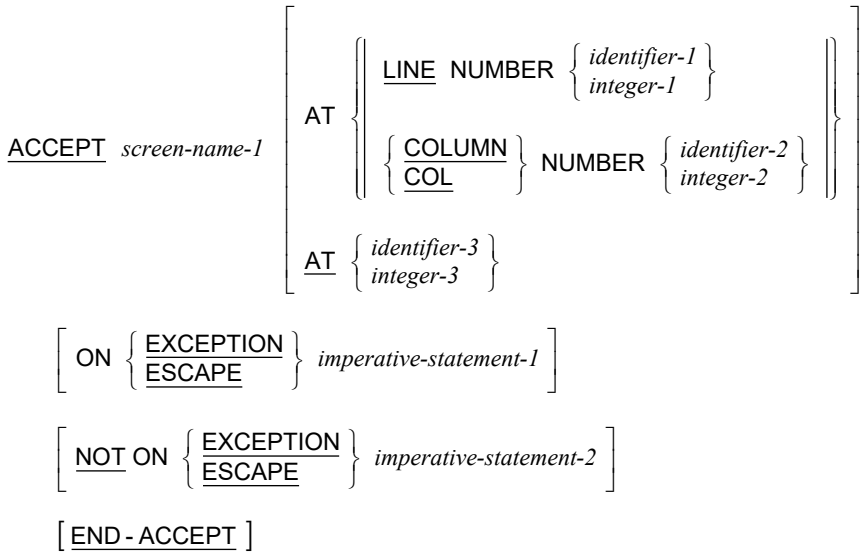
Format 3: Accept Terminal I-O



Format 4: Accept Input CD Message Count

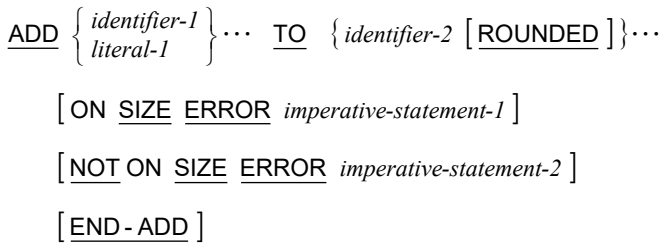
ACCEPT *cd-name-1* MESSAGE COUNT [END-ACCEPT]

Format 5: Accept Screen-Name

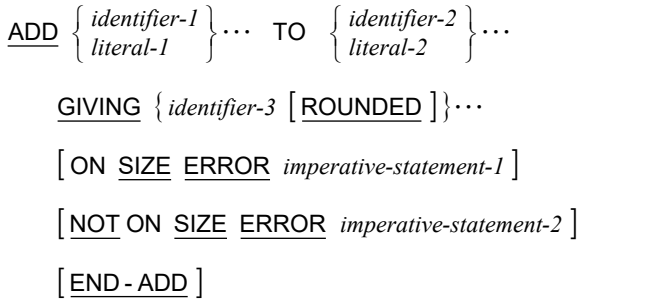


ADD Statement

Format 1: Add…To



Format 2: Add…Giving



Format 3: Add Corresponding

```
ADD { CORRESPONDING } identifier-1 TO identifier-2 [ ROUNDED ]
    { CORR         }
    [ ON SIZE ERROR imperative-statement-1 ]
    [ NOT ON SIZE ERROR imperative-statement-2 ]
    [ END-ADD ]
```

ALTER Statement

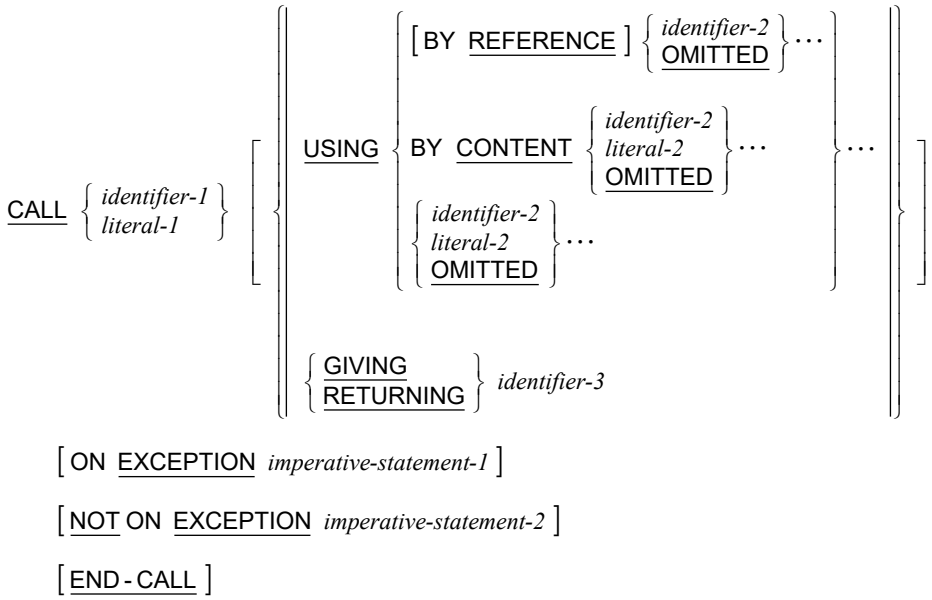
```
ALTER { procedure-name-1 TO [ PROCEED TO ] procedure-name-2 } ...
```

CALL Statement

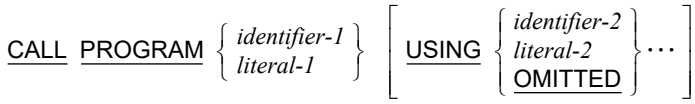
Format 1: Call…On Overflow

```
                                                                                                  [ BY REFERENCE ] { identifier-2 } ...
                                                                                                                   { OMITTED    }
                                                                                                  [ BY CONTENT ] { identifier-2 } ...
                                                                                                                 { literal-2    }
                                                                                                                 { OMITTED      }
CALL { identifier-1 } [ [ USING { [ BY REFERENCE ] { identifier-2 } ... [ BY CONTENT ] { identifier-2 } ... } ... ] ]
     { literal-1    }   [       { [ BY CONTENT ] { literal-2    } ...                   { literal-2    }       }     ]
                                                   { OMITTED    }                       { OMITTED      }
                                                                                                  { identifier-2 } ...
                                                                                                  { literal-2    }
                                                                                                  { OMITTED      }
                        [ { GIVING    } identifier-3 ]
                        [ { RETURNING } ]
    [ ON OVERFLOW imperative-statement-1 ]
    [ END-CALL ]
```

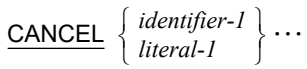
Format 2: Call…On Exception



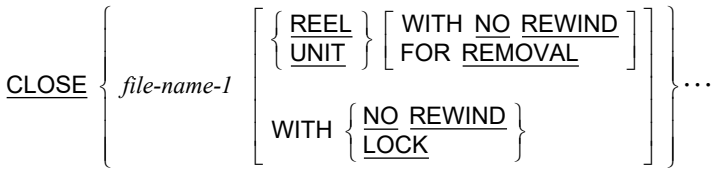
CALL PROGRAM Statement



CANCEL Statement



CLOSE Statement



COMPUTE Statement

```
COMPUTE { identifier-1 [ ROUNDED ] }... = arithmetic-expression-1
    [ ON SIZE ERROR imperative-statement-1 ]
    [ NOT ON SIZE ERROR imperative-statement-2 ]
    [ END-COMPUTE ]
```

CONTINUE Statement

```
CONTINUE
```

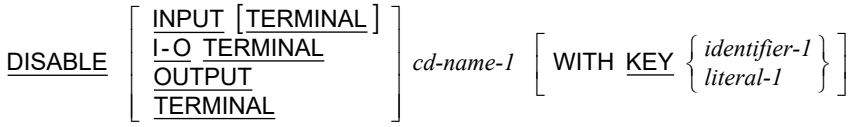
DELETE Statement

```
DELETE file-name-1 RECORD
    [ INVALID KEY imperative-statement-1 ]
    [ NOT INVALID KEY imperative-statement-2 ]
    [ END-DELETE ]
```

DELETE FILE Statement

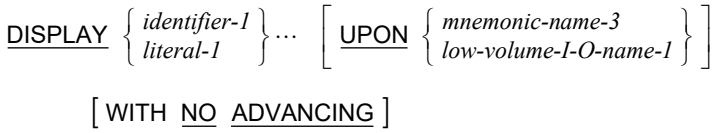
```
DELETE FILE { file-name-2 }... [ END-DELETE ]
```

DISABLE Statement

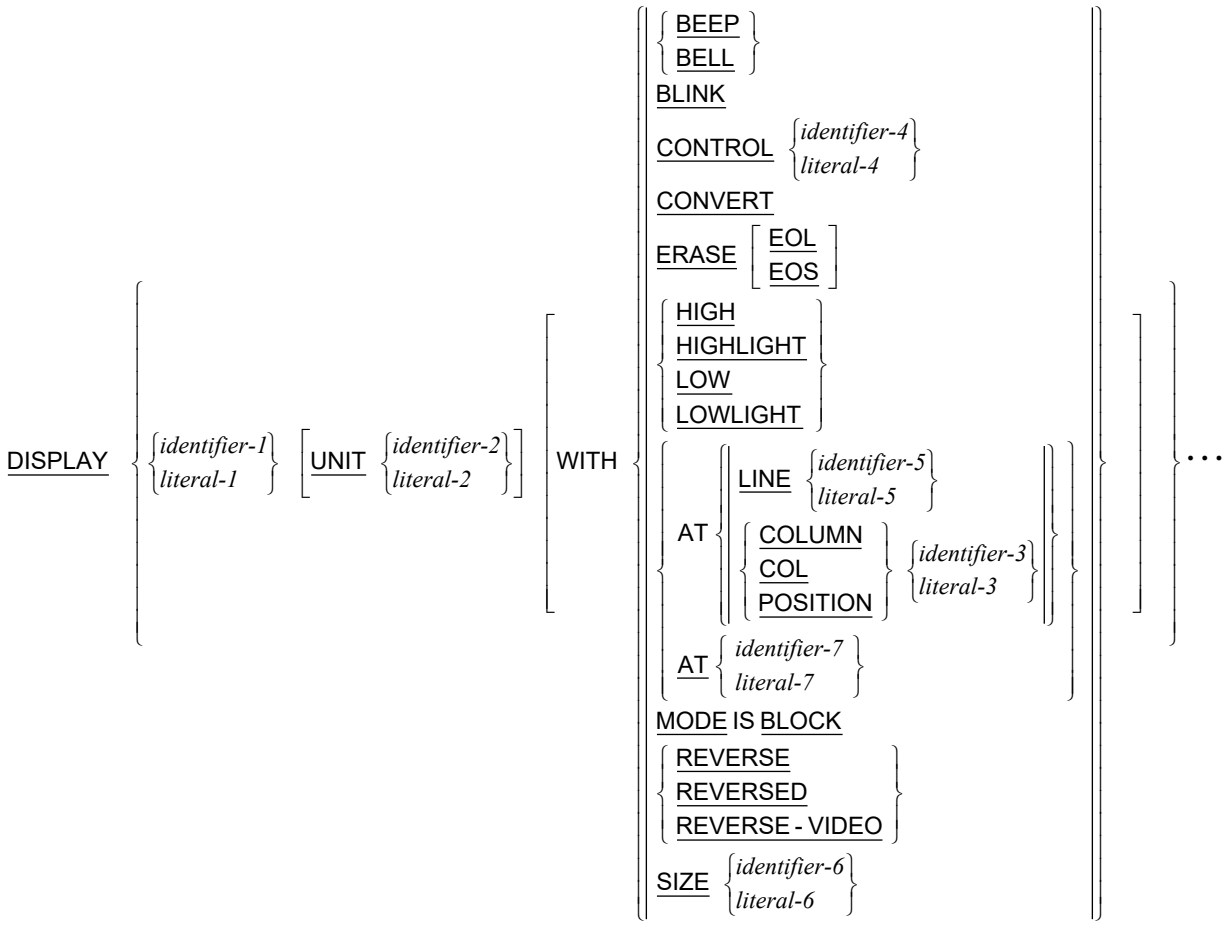


DISPLAY Statement

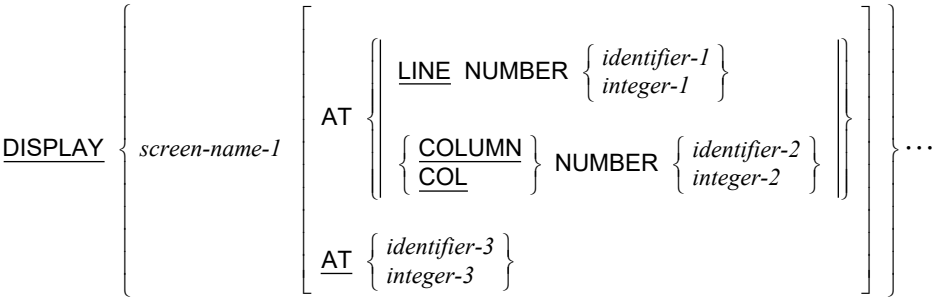
Format 1: Display Upon System-Name



Format 2: Display Terminal I-O



Format 3: Display Screen-Name



DIVIDE Statement

Format 1: Divide…Into

```
DIVIDE { identifier-1 } INTO { identifier-2 [ ROUNDED ] }...
       { literal-1    }
    [ ON SIZE ERROR imperative-statement-1 ]
    [ NOT ON SIZE ERROR imperative-statement-2 ]
    [ END-DIVIDE ]
```

Format 2: Divide…Into…Giving

```
DIVIDE { identifier-1 } INTO { identifier-2 }
       { literal-1    }      { literal-2    }
    GIVING { identifier-3 [ ROUNDED ] }...
    [ ON SIZE ERROR imperative-statement-1 ]
    [ NOT ON SIZE ERROR imperative-statement-2 ]
    [ END-DIVIDE ]
```

Format 3: Divide…By…Giving

```
DIVIDE { identifier-2 } BY { identifier-1 }
       { literal-2    }    { literal-1    }
    GIVING { identifier-3 [ ROUNDED ] }...
    [ ON SIZE ERROR imperative-statement-1 ]
    [ NOT ON SIZE ERROR imperative-statement-2 ]
    [ END-DIVIDE ]
```

Format 4: Divide…Into…Giving…Remainder

```
DIVIDE { identifier-1 } INTO { identifier-2 }
       { literal-1    }      { literal-2    }
    GIVING identifier-3 [ ROUNDED ] REMAINDER identifier-4
    [ ON SIZE ERROR imperative-statement-1 ]
    [ NOT ON SIZE ERROR imperative-statement-2 ]
    [ END-DIVIDE ]
```

Format 5: Divide…By…Giving…Remainder

```
DIVIDE { identifier-2 } BY { identifier-1 }
       { literal-2    }    { literal-1    }
    GIVING identifier-3 [ ROUNDED ] REMAINDER identifier-4
    [ ON SIZE ERROR imperative-statement-1 ]
    [ NOT ON SIZE ERROR imperative-statement-2 ]
    [ END-DIVIDE ]
```

ENABLE Statement

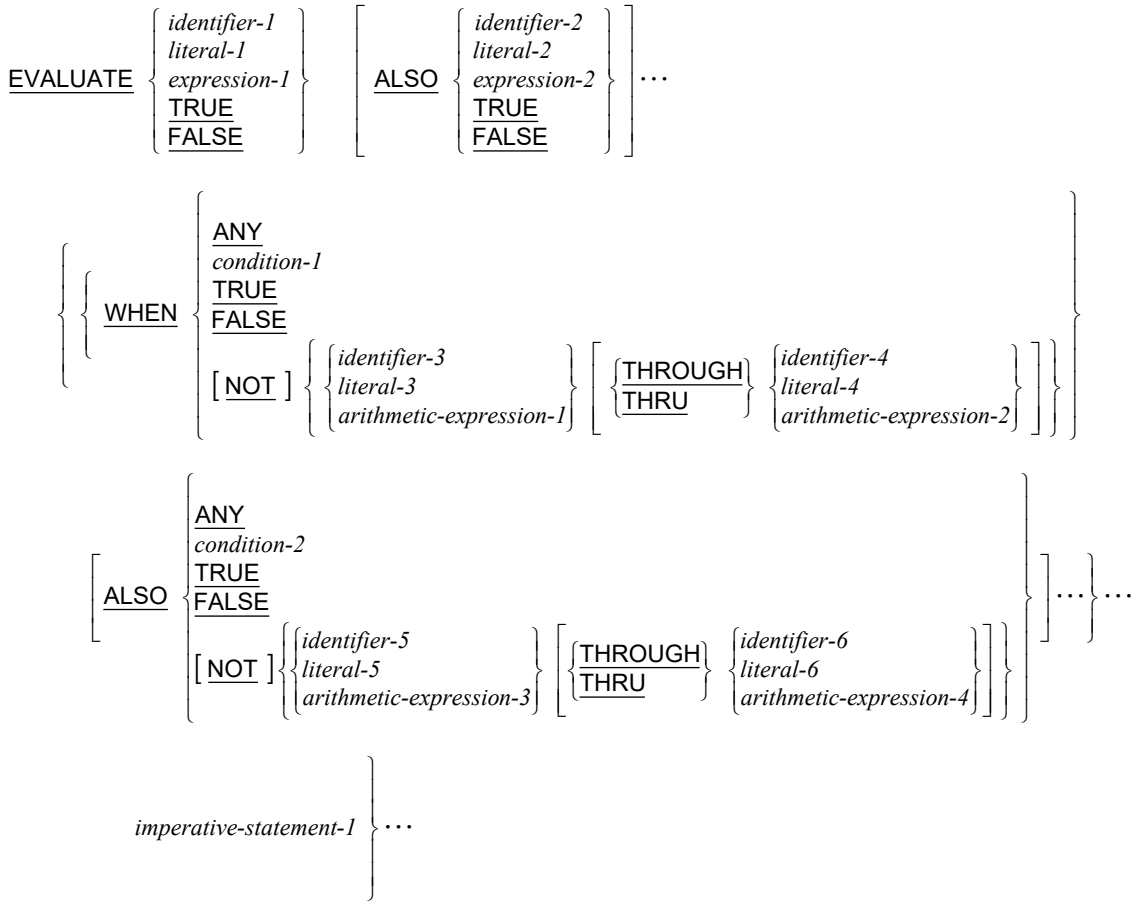
```
ENABLE [ INPUT [ TERMINAL ]  ]
       [ I-O TERMINAL        ]  cd-name-1 [ WITH KEY { identifier-1 } ]
       [ OUTPUT              ]               { literal-1    }
       [ TERMINAL            ]
```

ENTER Statement

```
ENTER language-name-1 [ routine-name-1 ]
```

Note The sentence ENTER COBOL must follow the last statement of the other language in order to indicate to the compiler where a return to COBOL source language takes place. It must be followed by a separator space. However, RM/COBOL does not currently support any other language embedded within a COBOL program. The ENTER statement is supported for compatibility with some dialects of COBOL that require an ENTER LINKAGE sentence preceding a CALL statement and an ENTER COBOL sentence immediately following a CALL statement.

EVALUATE Statement



EXIT Statement

Format 1: Exit Paragraph

```
EXIT
```

Format 2: Exit Program

```
EXIT PROGRAM
```

Format 3: Exit Perform

```
EXIT PERFORM [ CYCLE ]
```

Format 4: Exit Paragraph/Section

```
EXIT { PARAGRAPH }
     { SECTION   }
```

GOBACK Statement

```
GOBACK
```

GO TO Statement

Format 1: Go To (Alterable)

```
GO TO [ procedure-name-1 ]
```

Format 2: Go To (Non-Alterable)

```
GO TO procedure-name-1
```

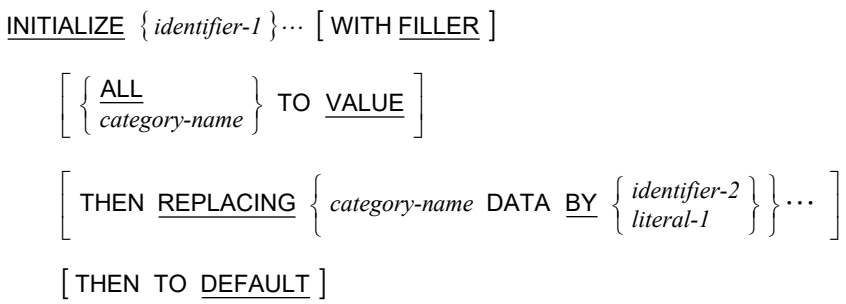
Format 3: Go To…Depending On

```
GO TO { procedure-name-1 }... DEPENDING ON identifier-1
```

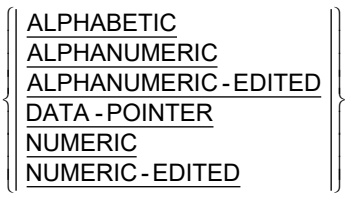
IF Statement

```
IF condition-1 THEN { statement-1      }
                    { NEXT SENTENCE    }

    [ ELSE { statement-2      } ]
           { NEXT SENTENCE    }
    [ END-IF ]
```

INITIALIZE Statement

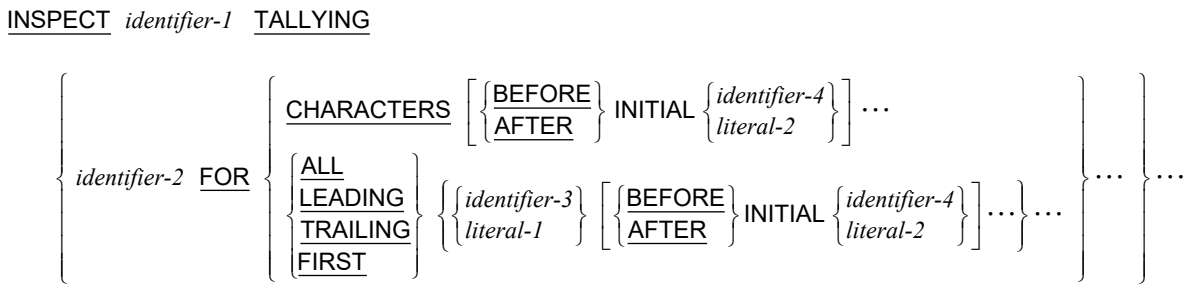


where *category-name* is:

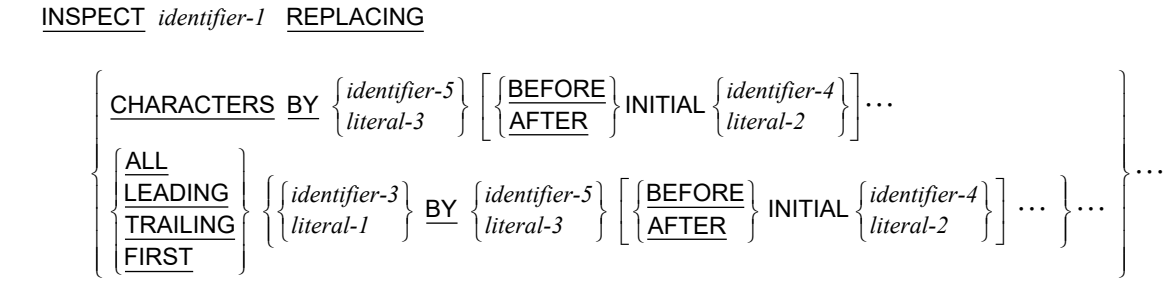


INSPECT Statement

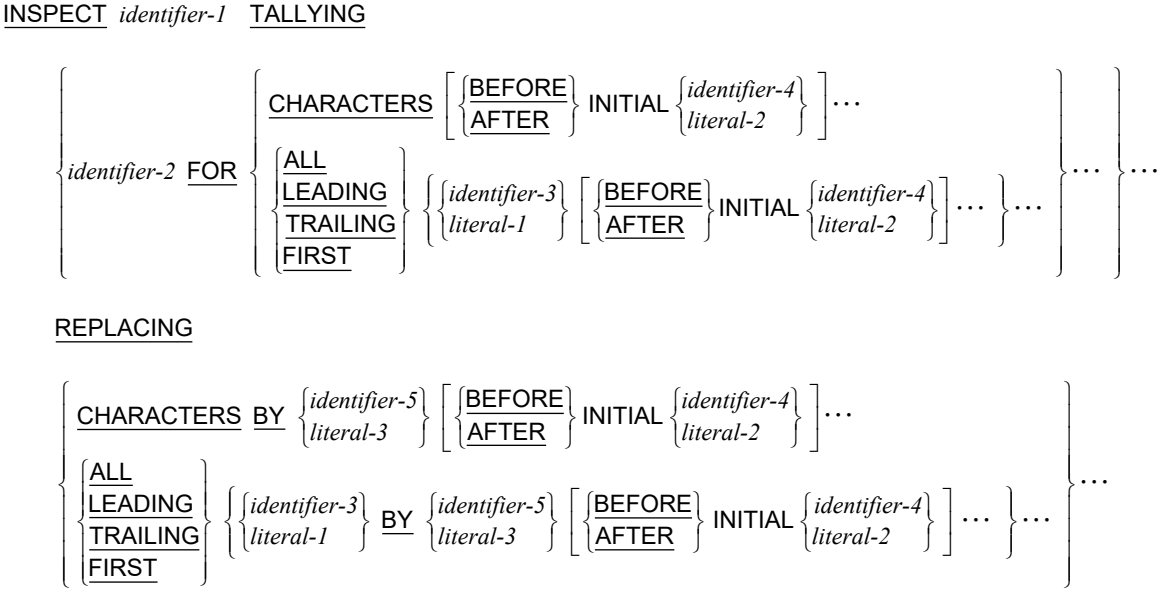
Format 1: Inspect…Tallying



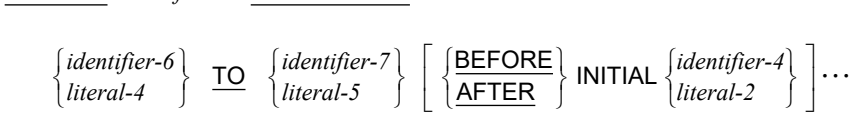
Format 2: Inspect…Replacing



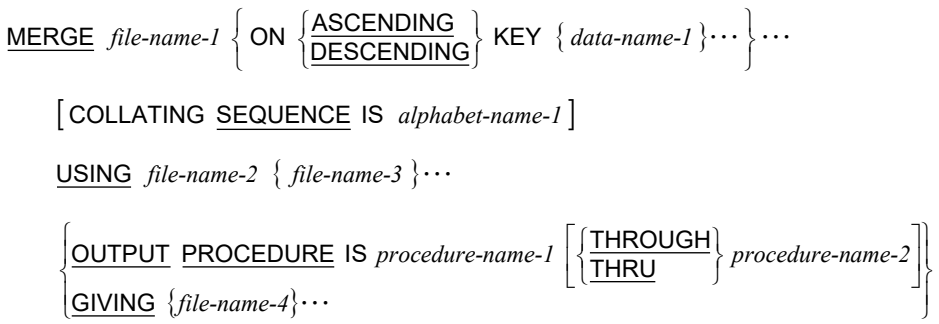
Format 3: Inspect…Tallying…Replacing



Format 4: Inspect…Converting

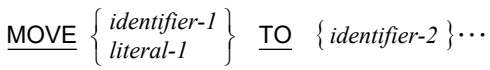


MERGE Statement

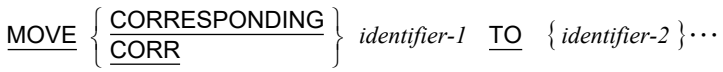


MOVE Statement

Format 1: Move…To

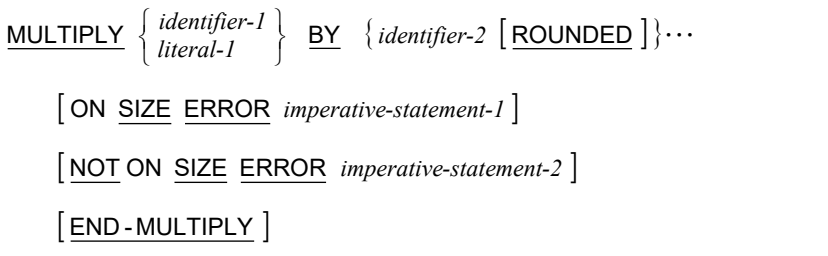


Format 2: Move Corresponding



MULTIPLY Statement

Format 1: Multiply…By



Format 2: Multiply…Giving

```
MULTIPLY { identifier-1 } BY { identifier-2 }
         { literal-1    }    { literal-2    }
    GIVING { identifier-3 [ ROUNDED ] }...
    [ ON SIZE ERROR imperative-statement-1 ]
    [ NOT ON SIZE ERROR imperative-statement-2 ]
    [ END-MULTIPLY ]
```

OPEN Statement

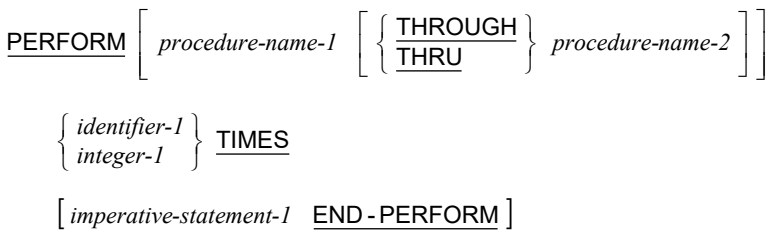
```
OPEN [ EXCLUSIVE ]
    { INPUT { file-name-1 [ WITH LOCK ] [ REVERSED          ] }...
            {                          [ WITH NO REWIND     ] }
      OUTPUT { file-name-2 [ WITH LOCK ] [ WITH NO REWIND ] }...
      I-O { file-name-3 [ WITH LOCK ] }...
      EXTEND { file-name-4 [ WITH LOCK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

PERFORM Statement

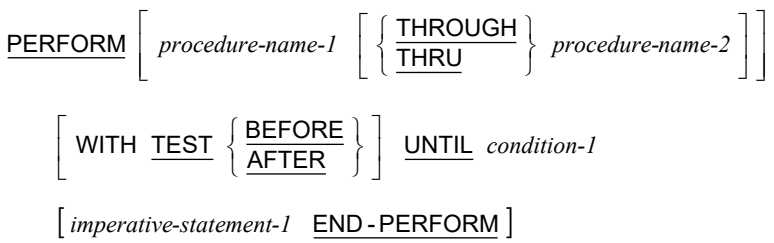
Format 1: Perform (Once)

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

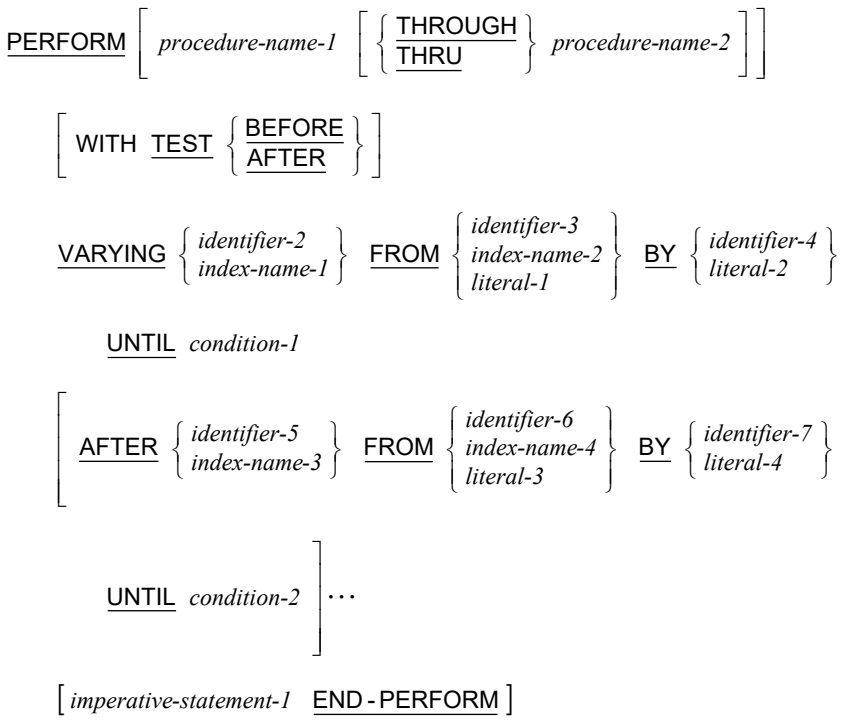
Format 2: Perform…Times



Format 3: Perform…Until



Format 4: Perform…Varying



PURGE Statement

```
PURGE cd-name-1
```

READ Statement

Format 1: Read Sequential Access

```
READ file-name-1 [ NEXT | PREVIOUS ] RECORD [ { WITH [ NO ] LOCK | INTO identifier-1 } ]
    [ AT END imperative-statement-1 ]
    [ NOT AT END imperative-statement-2 ]
    [ END-READ ]
```

Format 2: Read Random Access

```
READ file-name-1 RECORD [ { WITH [ NO ] LOCK | INTO identifier-1 } ]
    [ KEY IS { data-name-1 | split-key-name-1 } ]
    [ INVALID KEY imperative-statement-1 ]
    [ NOT INVALID KEY imperative-statement-2 ]
    [ END-READ ]
```

RECEIVE Statement

```
RECEIVE cd-name-1 { MESSAGE | SEGMENT } INTO identifier-1
    [ NO DATA imperative-statement-1 ]
    [ WITH DATA imperative-statement-2 ]
    [ END-RECEIVE ]
```

RELEASE Statement

```
RELEASE record-name-1 [ FROM { identifier-1 } ]
                             { literal-1    }
```

RETURN Statement

```
RETURN file-name-1 RECORD [ INTO identifier-1 ]
    [ AT END imperative-statement-1 ]
    [ NOT AT END imperative-statement-2 ]
    [ END-RETURN ]
```

REWRITE Statement

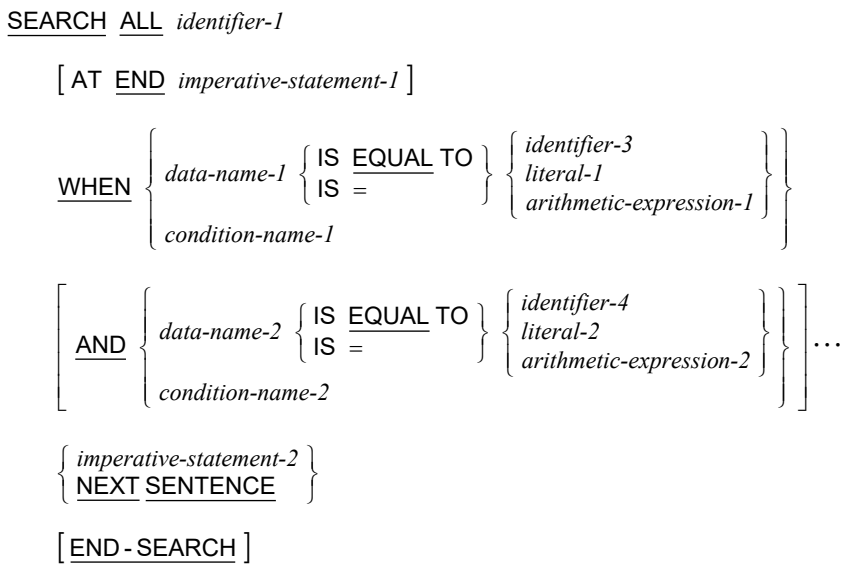
```
REWRITE record-name-1 [ FROM { identifier-1 } ]
                             { literal-1    }
    [ INVALID KEY imperative-statement-1 ]
    [ NOT INVALID KEY imperative-statement-2 ]
    [ END-REWRITE ]
```

SEARCH Statement

Format 1: Search (Serial)

```
SEARCH identifier-1 [ VARYING { identifier-2 } ]
                              { index-name-1 }
    [ AT END imperative-statement-1 ]
    { WHEN condition-1 { imperative-statement-2 } }...
                       { NEXT SENTENCE         }
    [ END-SEARCH ]
```

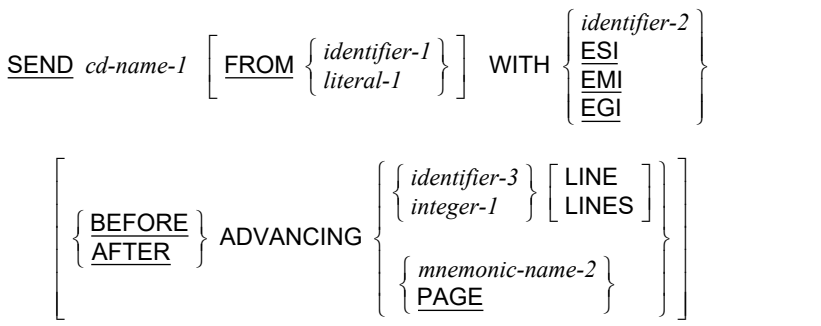
Format 2: Search All (Binary)



SEND Statement

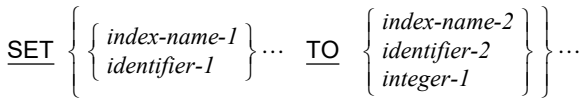
Format 1: Send (Simple)

Format 2: Send (Advancing/Replacing)



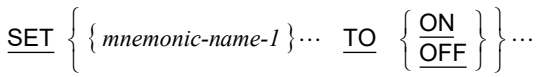
SET Statement

Format 1: Set Index

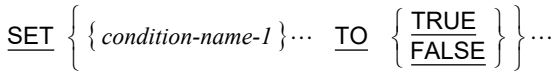


Format 2: Set Index Up/Down

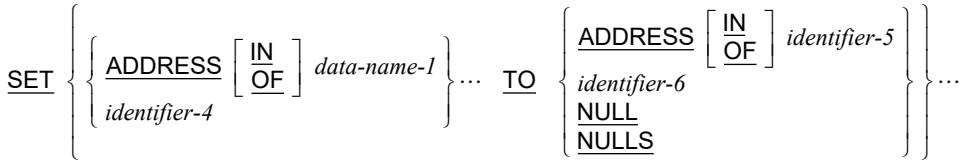
Format 3: Set Switch On/Off



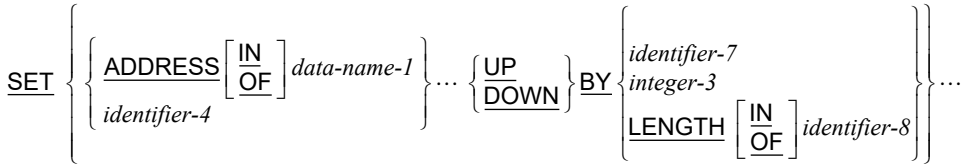
Format 4: Set Condition-Name True/False



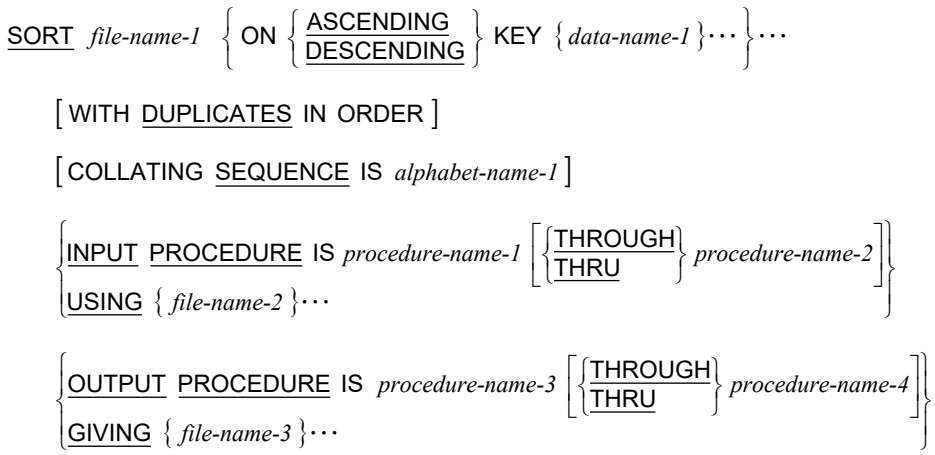
Format 5: Set Pointer



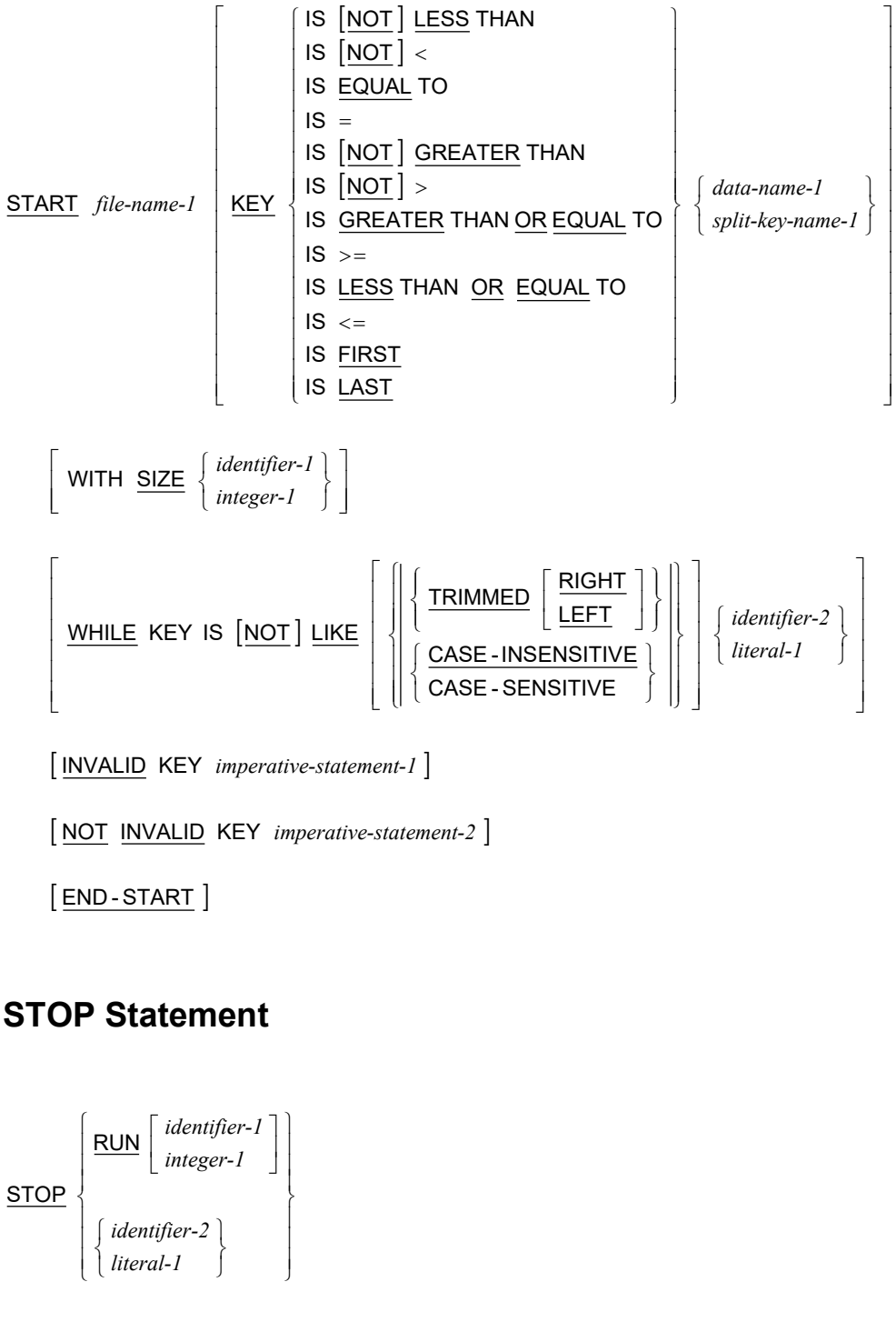
Format 6: Set Pointer Up/Down



SORT Statement

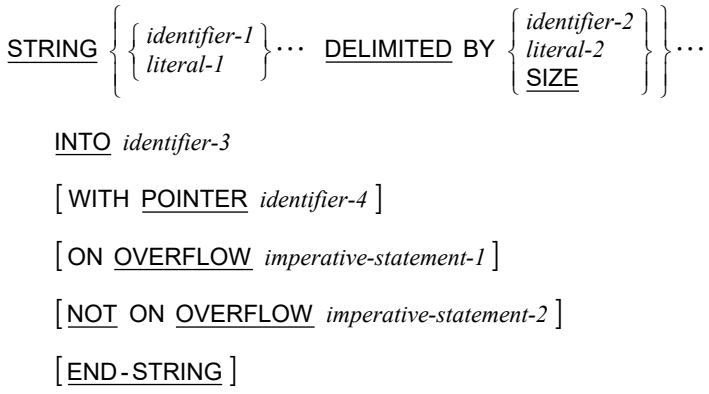


START Statement



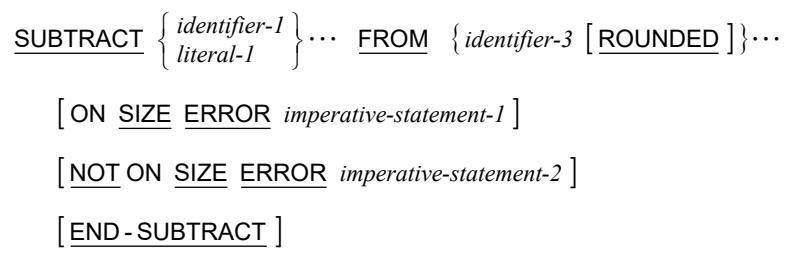
STOP Statement

STRING Statement

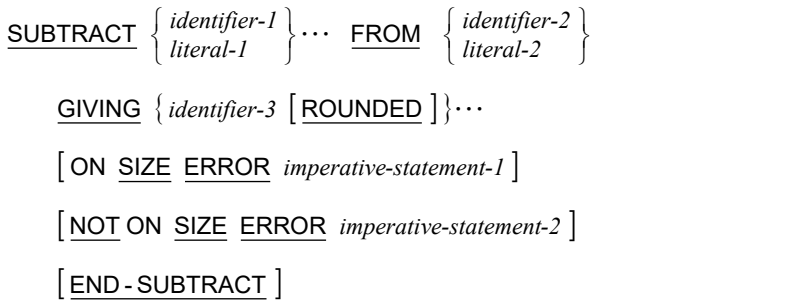


SUBTRACT Statement

Format 1: Subtract…From



Format 2: Subtract…Giving



Format 3: Subtract Corresponding

```
SUBTRACT { CORRESPONDING } identifier-1 FROM identifier-2 [ ROUNDED ]
         { CORR          }

    [ ON SIZE ERROR imperative-statement-1 ]

    [ NOT ON SIZE ERROR imperative-statement-2 ]

    [ END-SUBTRACT ]
```

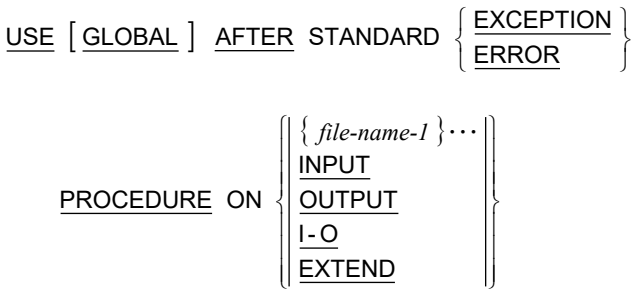
UNLOCK Statement

```
UNLOCK file-name-1 [ RECORD  ]
                   [ RECORDS ]
```

UNSTRING Statement

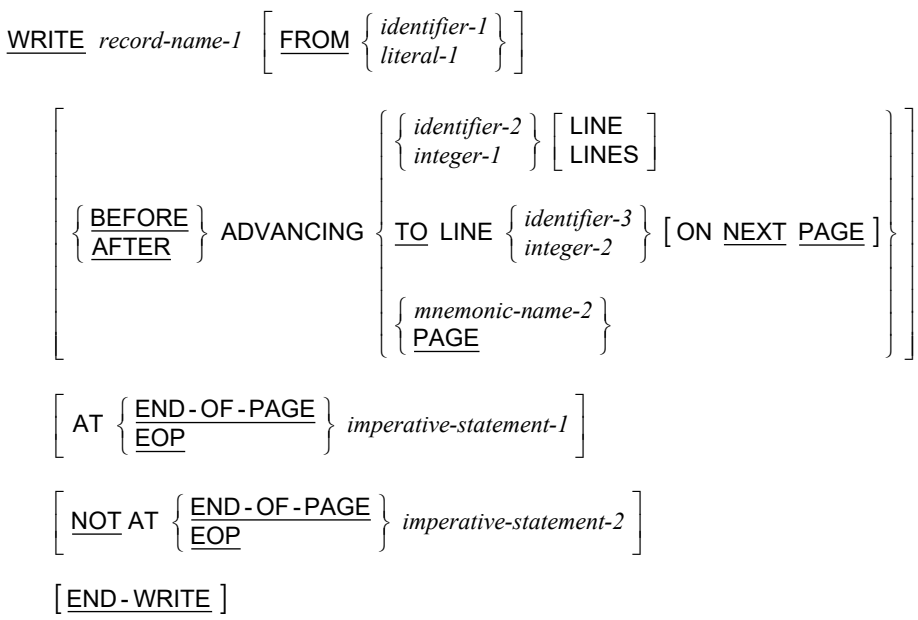
```
UNSTRING identifier-1

    [ DELIMITED BY [ ALL ] { identifier-2 } [ OR [ ALL ] { identifier-3 } ]... ]
                           { literal-1    }             { literal-2    }

    INTO { identifier-4 [ DELIMITER IN identifier-5 ] [ COUNT IN identifier-6 ] }...

    [ WITH POINTER identifier-7 ]

    [ TALLYING IN identifier-8 ]

    [ ON OVERFLOW imperative-statement-1 ]

    [ NOT ON OVERFLOW imperative-statement-2 ]

    [ END-UNSTRING ]
```

USE Statement

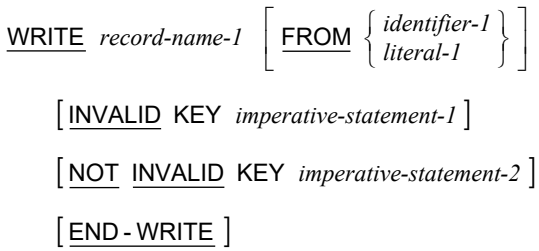


WRITE Statement

Format 1: Write Sequential File



Format 2: Write Relative and Indexed File



END PROGRAM Header General Format

END PROGRAM [*program-name-1* | *literal-1*].

COPY and REPLACE Statement General Formats

The REPLACE statement and the REPLACING phrase of the COPY statement replace entire text words in the source. Sometimes it is desirable to replace a portion of a word.

Parentheses may be used to demarcate portions of words to be replaced because the left and right parenthesis characters are always treated as text word separators (the hyphen is not a text word separator) and replacement does not add additional spaces.

For example, suppose you wish to replace the first part of each identifier (before the initial hyphen). That is, you wish that the statement

```
COPY FDMASTER REPLACING ==FILENAME== BY ==WS==.
```

would, for the copy file containing

```
01 FILENAME-REC.
   02 FILENAME-ITEM1 ... .
   02 FILENAME-ITEM2 ... .
```

replace each occurrence of `FILENAME`. Unfortunately, this would not occur. The text words in the copy file are `FILENAME-REC`, `FILENAME-ITEM1`, and `FILENAME-ITEM2`, none of which match the replacing key text word `FILENAME` specified in the COPY statement REPLACING phrase.

The solution is to use parentheses in the COPY statement REPLACING phrase

```
COPY FDMASTER REPLACING ==(FILENAME)== BY ==WS==.
```

and in the copy file

```
01 (FILENAME)-REC.
   02 (FILENAME)-ITEM1 ... .
   02 (FILENAME)-ITEM2 ... .
```

The parentheses separate the names into multiple text words, which are then replaced as desired. Since no additional spaces are inserted, the replacement yields a single COBOL word in the resultant source program that is compiled.

```
COPY { text-name-1 }  [ { IN }  { library-name-1 } ]  [ SUPPRESS PRINTING ]
     { literal-1   }  [ { OF }  { literal-2      } ]

[ REPLACING { { == pseudo-text-1 == }  BY { == pseudo-text-2 == } }... ]
            { { identifier-1        }     { identifier-2        }
            { { literal-3           }     { literal-4           }
            { { word-1              }     { word-2              }

[END-COPY]
```

Format 1: Begin or Change Replacement

```
REPLACE { == pseudo-text-1 ==  BY  == pseudo-text-2 == }... [END-REPLACE]
```

Format 2: End Replacement

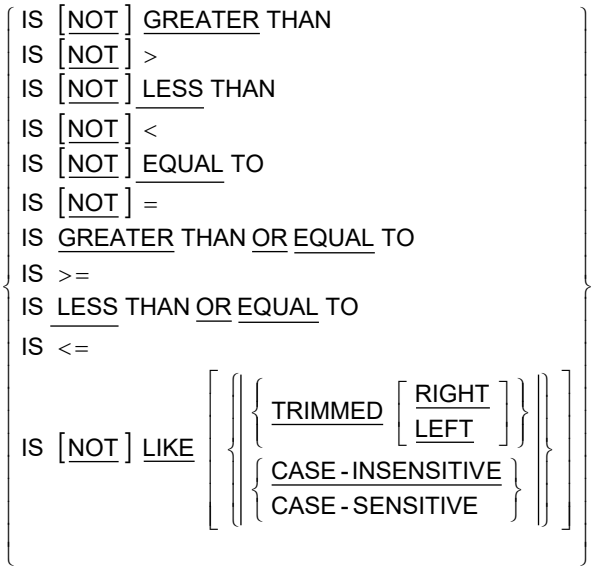
```
REPLACE OFF [END-REPLACE]
```

General Formats for Conditions

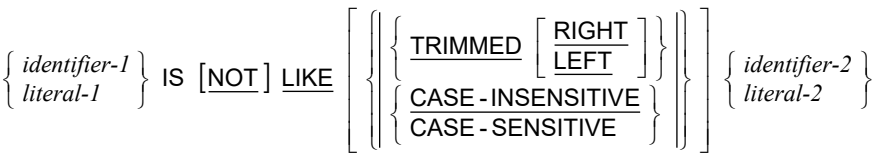
Relation Condition

```
{ identifier-1           }  relational-operator  { identifier-2           }
{ literal-1              }                      { literal-2              }
{ arithmetic-expression-1}                      { arithmetic-expression-2}
{ index-name-1           }                      { index-name-2           }
```

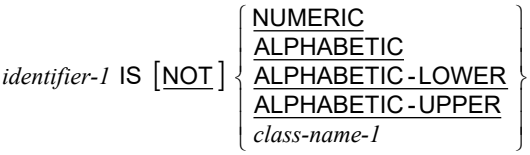
Relational Operator



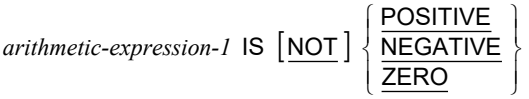
LIKE Condition (special case of a relation condition)



Class Condition



Sign Condition



Condition-Name Condition

condition-name-1

Switch-Status Condition

condition-name-2

Negated Condition

NOT *condition-1*

Combined Condition

condition-2 { { AND / OR } *condition-3* }...

Abbreviated Combined Relation Condition

relation-condition-1 { { AND / OR } [NOT] [*relational-operator*] *object-1* }...

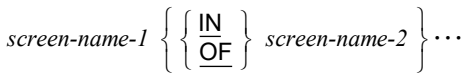
General Formats for Qualification

Format 1: Qualification for Data-Names, Index-Names and Condition-Names

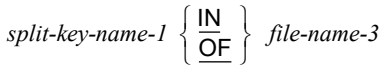
{ *data-name-1* / *index-name-1* / *condition-name-1* } { { { IN / OF } *data-name-2* }... [{ { IN / OF } { *file-name-1* / *cd-name-1* }] / { IN / OF } { *file-name-1* / *cd-name-1* } }

Format 2: Qualification for LINAGE-COUNTER

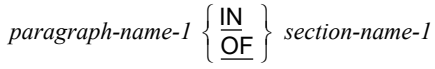
Format 3: Qualification for Screen-Names



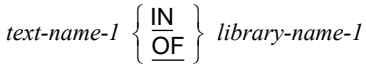
Format 4: Qualification for Split-Key-Names



Format 5: Qualification for Paragraph Names



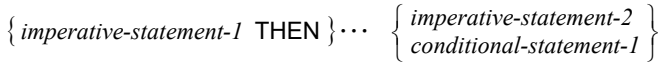
Format 6: Qualification for Text-Names (COPY)



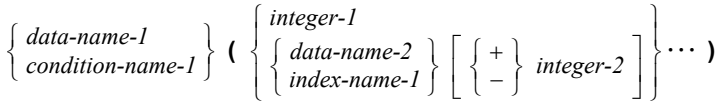
Miscellaneous Formats

Sentence

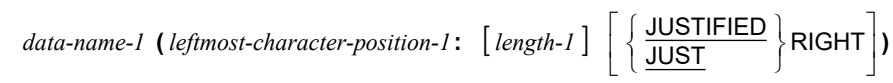
Statement Sequence



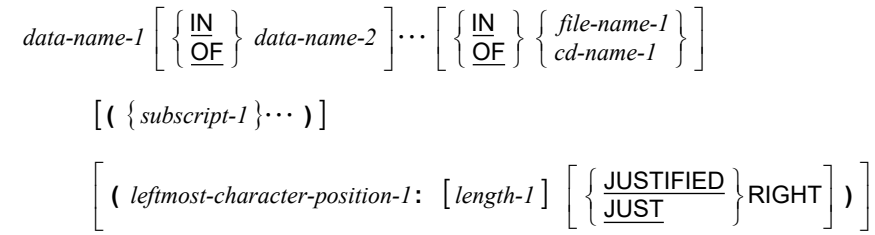
Subscripting



Reference Modification



Identifier



Special Registers

ADDRESS [IN / OF] *identifier-1*

COUNT [IN / OF] *data-name-1*

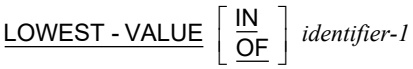
COUNT-MAX [IN / OF] *data-name-1*

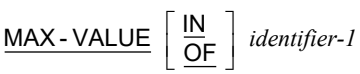
COUNT-MIN [IN / OF] *data-name-1*

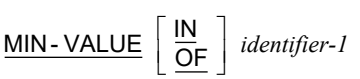
HIGHEST-VALUE [IN / OF] *identifier-1*

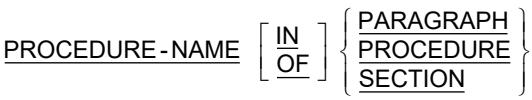
INITIAL-VALUE [IN / OF] *data-name-1*

LENGTH [IN / OF] { *identifier-1* / *literal-1* }









PROGRAM-ID

RETURN-CODE

WHEN-COMPILED

Figurative Constants

[ALL] HIGH-VALUE
[ALL] HIGH-VALUES

[ALL] LOW-VALUE
[ALL] LOW-VALUES

[ALL] NULL
[ALL] NULLS

[ALL] QUOTE
[ALL] QUOTES

[ALL] SPACE
[ALL] SPACES

[ALL] ZERO
[ALL] ZEROES
[ALL] ZEROS

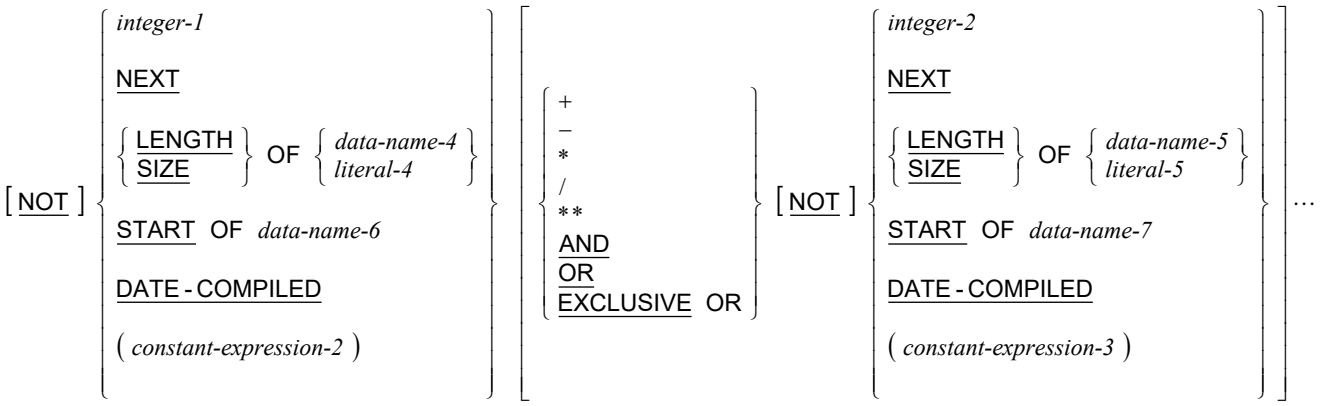
ALL *literal-1*

[ALL] *symbolic-character-1*

Concatenation Expression

literal-1 & *literal-2*

Constant-Expression



PICTURE Character-String (Data Categories)

The five categories of data that can be described with a PICTURE clause are defined as follows. Note that the additional data categories, **index data** and **data pointer**, also exist, but do not use a PICTURE clause in their data description entry. An index data item is described with the USAGE IS INDEX clause. A data pointer data item is described with the USAGE IS POINTER clause.

Note The additional data categories, index data and data pointer, also exist, but do not use a PICTURE clause in their data description entry. An index data item is described with the USAGE IS INDEX clause. A data pointer data item is described with the USAGE IS POINTER clause.

Alphabetic

Its PICTURE character-string can contain only the symbol **A**. The contents of an alphabetic data item when represented in standard data format must be one or more alphabetic characters ("a" through "z", "A" through "Z", and space). See the examples on page 58.

Alphanumeric

Its PICTURE character-string is restricted to certain combinations of the symbols **A**, **X** and **9**, and the item is treated as if the character-string contained all symbols **X**. The PICTURE character-string must contain at least one symbol **X** or a combination of the symbols **A** and **9**. A PICTURE character-string that contains all symbols **A** or all symbols **9** does not define an alphanumeric data item, since such character-strings define an alphabetic or numeric data item, respectively. The contents of an alphanumeric data item when represented in standard data format must be one or more characters in the character set of the computer. See the examples on page 58.

Alphanumeric edited

Its PICTURE character-string is restricted to certain combinations of the following symbols: **A**, **X**, **9**, **B**, **0**, and slash (**/**). The PICTURE character-string must contain at least one symbol **A** or **X** and at least one symbol **B**, **0**, or slash (**/**). The contents of an alphanumeric edited data item when represented in standard data format must be two or more characters in the character set of the computer. See the examples on page 58.

Numeric

Its PICTURE character-string can contain only the symbols **9**, **P**, **S**, and **V**. Its PICTURE character-string must contain at least one symbol **9** and not more than thirty symbols **9**. Each symbol **9** specifies a digit position. If unsigned, the contents of a numeric data item when

represented in standard data format must be one or more numeric characters. If signed, a numeric data item may also contain a "+", "–", or other representation of an operational sign. The actual in-memory contents of a numeric data item are not standard data format when the usage is other than DISPLAY as specified by a USAGE clause that applies to the data description entry or when the data item is signed, but without the SEPARATE CHARACTER phrase in a SIGN clause that applies to the data description entry. See the examples on page 59.

Numeric edited

Its PICTURE character-string is restricted to certain combinations of the following symbols: **B**, slash (**/**), **P**, **V**, **Z**, **0**, **9**, comma (**,**), period (**.**), asterisk (*****), minus (**–**), plus (**+**), **CR**, **DB**, and the currency symbol (the symbol **$** or the symbol specified in the CURRENCY SIGN clause of the SPECIAL-NAMES paragraph). The allowable combinations are determined from the order of precedence of symbols and the editing rules. The number of digit positions that can be represented in the PICTURE character-string must range from one to thirty, inclusive. The character-string must contain at least one symbol **0**, **B**, slash, **Z**, asterisk, plus, minus, comma, period, **CR**, **DB**, or the currency symbol. The contents of each of the character positions in a numeric edited data item must be consistent with the corresponding PICTURE symbol. See the examples on page 59.

Alphabetic PICTURE Character-String Examples

```
A           *> 1-character alphabetic item
AAAAA       *> 5-character alphabetic item
A(5)        *> 5-character alphabetic item
AAAA(4)A    *> 8-character alphabetic item
```

Alphanumeric PICTURE Character-String Examples

```
X           *> 1-character alphanumeric item
A9          *> 2-character alphanumeric item
AX9         *> 3-character alphanumeric item
X(5)        *> 5-character alphanumeric item
XXX9(4)A    *> 8-character alphanumeric item
X(80)       *> 80-character alphanumeric item
```

Alphanumeric-Edited PICTURE Character-String Examples

```
XX/BB/00         *> 8-character alphanumeric edited item
XX/990/0BB       *> 10-character alphanumeric edited item
X(4)BA(4)B9(4)   *> 14-character alphanumeric edited item
```

Numeric PICTURE Character-String Examples

```
*> Unsigned integers:
 9          1-digit numeric integer (1,0)
 99         2-digit numeric integer (2,0)
 9(6)       6-digit numeric integer (6,0)
 9(30)     30-digit numeric integer (30,0)
 9(6)V      6-digit numeric integer (6,0)
 9(6)PPV    6-digit numeric integer (2 right scaling)
 9(8)P(4)   8-digit numeric integer (4 right scaling)

*> Unsigned non-integer numbers:
 V9         1-digit numeric fraction (1,1)
 VPP9(4)    4-digit numeric fraction (4,6)
 P(6)9(2)   2-digit numeric fraction (2,8)
 9(4)V9(5)  9-digit numeric (9,5)

*> Signed integers:
S9          1-digit numeric integer (1,0)
S99         2-digit numeric integer (2,0)
S9(6)       6-digit numeric integer (6,0)
S9(30)     30-digit numeric integer (30,0)
S9(6)V      6-digit numeric integer (6,0)
S9(6)PPV    6-digit numeric integer (2 right scaling)
S9(8)P(4)   8-digit numeric integer (4 right scaling)

*> Signed non-integer numbers:
SV9         1-digit numeric fraction (1,1)
SVPP9(4)    4-digit numeric fraction (4,6)
SP(6)9(2)   2-digit numeric fraction (2,8)
S9(4)V9(5)  9-digit numeric (9,5)
```

Numeric-Edited PICTURE Character-String Examples

```
*> Simple insertion editing (comma, space (B), zero, slash):
999,999,999 *> 9-digit (size 11) numeric edited item (9,0)
99,999BB    *> 5-digit (size 8) numeric edited item (5,0)
99/00/99    *> 4-digit (size 8) numeric edited item (4,0)

*> Special insertion editing (explicit decimal point):
9(5).99     *> 7-digit (size 8) numeric edited item (7,2)
999,999.99  *> 8-digit (size 10) numeric edited item (8,2)
9,999.9999  *> 8-digit (size 10) numeric edited item (8,4)

*> Fixed insertion editing (sign or currency):
9(5)CR      *> 5-digit (size 7) numeric edited item (5,0)
99DB        *> 2-digit (size 4) numeric edited item (2,0)
9(5)+       *> 5-digit (size 6) numeric edited item (5,0)
999.99-     *> 5-digit (size 7) numeric edited item (5,2)
+9(18)      *> 18-digit (size 19) numeric edited item (18,0)
-9(6)V99    *> 8-digit (size 9) numeric edited item (8,2)
$9(4).99    *> 6-digit (size 10) numeric edited item (6,2)

*> Floating insertion editing (sign or currency):
```

```
+++9         *> 3-digit (size 4) numeric edited item (3,0)
-(8)9        *> 8-digit (size 9) numeric edited item (8,0)
-(3).-(4)    *> 6-digit (size 8) numeric edited item (6,4)
$(5)9        *> 5-digit (size 6) numeric edited item (5,0)
$(6)         *> 5-digit (size 6) numeric edited item (5,0)

*> Zero suppression editing (spaces (Z) or asterisk (*)):
Z(5)         *> 5-digit (size 5) numeric edited item (5,0)
Z(5)9        *> 6-digit (size 6) numeric edited item (6,0)
Z(5).ZZ      *> 7-digit (size 8) numeric edited item (7,2)
ZZZ,ZZZ,ZZ9 *> 9-digit (size 11) numeric edited item (9,0)
*(5)         *> 5-digit (size 5) numeric edited item (5,0)
***9.99      *> 6-digit (size 7) numeric edited item (6,2)
***,**9.99   *> 8-digit (size 10) numeric edited item (8,2)
*(5).**      *> 7-digit (size 8) numeric edited item (7,2)
```

PICTURE Symbols

The functions of the symbols used in a PICTURE character-string to describe an elementary data item are as follows:

PICTURE Symbol	Description
A	Each symbol **A** in the character-string represents a character position that can contain only an alphabetic character ("a" through "z", "A" through "Z", and space). Each symbol **A** is counted in the size of the data item described by the PICTURE character-string.
B	Each symbol **B** in the character-string represents a character position into which the character space will be inserted when the data item is the receiving item of an elementary MOVE statement. Each symbol **B** is counted in the size of the data item described by the PICTURE character-string.
P	Each symbol **P** in the character-string indicates an assumed decimal scaling position and is used to specify the location of an assumed decimal point when the point is not within the number that appears in the data item. The scaling position symbol **P** is not counted in the size of the data item described by the PICTURE character-string, but each symbol **P** is counted in determining the maximum number (30) of digit positions in numeric and numeric edited data items. The symbol **P** may appear only as a contiguous string in the leftmost or rightmost digit positions within a PICTURE character-string. Since the scaling position symbol **P** implies an assumed decimal point (to the left of the symbols **P** if they are the leftmost digit positions and to the right of the symbols **P** if they are the rightmost digit positions), the assumed decimal point symbol **V** is redundant either to the left or right of the symbols **P**, respectively, within such a PICTURE character-string. The symbol **P** and the insertion symbol period (**.**) cannot both occur in the same PICTURE character-string.

PICTURE Symbol	Description
S	The symbol **S** is used in the character-string to indicate the presence, but neither the representation nor, necessarily, the position of an operational sign. The symbol **S** must be written as the leftmost character in the PICTURE character-string. The symbol **S** is not counted in determining the size (in terms of standard data format characters) of the data item described by the PICTURE character-string unless the entry contains or is subject to a SIGN clause that specifies the SEPARATE CHARACTER phrase. The symbol **S** in the PICTURE character-string and the BLANK WHEN ZERO clause may not occur in the same data description entry.
V	The symbol **V** is used in a character-string to indicate the location of the assumed decimal point and may appear only once in any single PICTURE character-string. The symbol **V** does not represent a character position and therefore is not counted in the size of the data item described by the PICTURE character-string. When the assumed decimal point is to the right of the rightmost symbol in the string representing a digit position or scaling position or is to the left of scaling positions that represent the leftmost digit positions, the symbol **V** is redundant. The symbol **V** and the insertion symbol period (**.**) cannot both occur in the same PICTURE character-string.
X	Each symbol **X** in the character-string is used to represent a character position that contains any allowable character from the character set of the computer. Each symbol **X** is counted in the size of the data item described by the PICTURE character-string.
Z	Each symbol **Z** in a character-string may only be used to represent the leftmost leading numeric character positions that will be replaced by space characters when the contents of those character positions are leading zeroes and the data item is the receiving item of an elementary MOVE statement. Each symbol **Z** is counted in the size of the item described by the PICTURE character-string and in determining the maximum number (30) of digit positions allowed in a numeric edited data item. If the symbol **Z** is used to the right of the decimal point in a character-string, then all digit positions in that character-string must be described with the symbol **Z**. If the symbol **Z** represents all the digit-positions in the character-string, then the described data item is blank when zero, even if the BLANK WHEN ZERO clause is not specified.
9	Each symbol **9** in the character-string represents a character position that contains a numeric character. Each symbol **9** is counted in the size of the item described by the PICTURE character-string and in determining the maximum number (30) of digit positions in a numeric or numeric edited data item.
0	Each symbol **0** in the character-string represents a character position into which the character zero ("0") will be inserted when the data item is the receiving item of an elementary MOVE statement and removed when a numeric edited data item is the sending item in an elementary MOVE statement with a numeric or numeric edited receiving data item. Each symbol **0** is counted in the size of the data item described by the PICTURE character-string. The symbol **0** does not represent a digit position in a numeric edited data item.
/	Each symbol slash (**/**) in the character-string represents a character position into which a character slash ("/") will be inserted when the data item is the receiving item of an elementary MOVE statement. Each symbol slash (**/**) is counted in the size of the data item described by the PICTURE character-string.

PICTURE Symbol	Description
,	Each symbol comma (**,**) in the character-string represents a character position into which a character comma (",") will be inserted when the data item is the receiving item of an elementary MOVE statement. Each symbol comma (**,**) is counted in the size of the data item described by the PICTURE character-string.
.	When the symbol period (**.**) appears in the character-string it is an editing symbol that represents the decimal point for alignment purposes and, in addition, represents a character position into which the character period (".") will be inserted. The symbol period is counted in the size of the data item described by the PICTURE character-string. The symbols **P** and **V** cannot occur with a symbol period (**.**) in the same PICTURE character-string. **Note** For a given program the functions of the period and comma are exchanged if the DECIMAL-POINT IS COMMA clause is stated in the SPECIAL-NAMES paragraph. In this exchange the rules for the period apply to the comma and the rules for the comma apply to the period wherever they appear in a PICTURE character-string.
+, **–**, **CR**, **DB**	These symbols are used as editing sign control symbols. When used, they represent the character position into which the editing sign control symbol will be placed. The symbols are mutually exclusive in any one PICTURE character-string and each character used in the symbol is counted in determining the size of the data item described by the PICTURE character-string. If the symbols plus or minus occur more than once (a floating sign control symbol), then one less than the total number of these symbols is counted in determining the maximum number (30) of digit positions allowed in a numeric edited data item. If a floating symbol plus or minus is used to the right of the decimal point in a character-string, then all digit positions in that character-string must be described with the symbol plus or minus, respectively. If a floating plus or minus symbol string represents all the digit-positions in the character-string, then the described data item is blank when zero, even if the BLANK WHEN ZERO clause is not specified.
*****	Each symbol asterisk (*****) in the character-string represents a leading numeric character position into which a character asterisk ("*") will be placed when that position contains a leading zero and the data item is the receiving item of an elementary MOVE statement. Each symbol asterisk (*****) is counted in the size of the data item described by the PICTURE character-string and in determining the maximum number (30) of digit positions allowed in a numeric edited data item. If the symbol asterisk (*****) is used to the right of the decimal point in a character-string, then all digit positions in that character-string must be described with the symbol asterisk (*). The symbol asterisk in the PICTURE character-string and the BLANK WHEN ZERO clause may not occur in the same data description entry. If the symbol asterisk represents all the digit-positions in the character-string, then, when zero, the described data item is all asterisks (ALL "*"), except that, if the character-string contains the symbol period (**.**), a period (".") will occur at the specified location in the data item.

PICTURE Symbol	Description
cs	The currency symbol in a character-string is represented by either the currency sign (the symbol **$**) or by the single character specified in the CURRENCY SIGN clause in the SPECIAL-NAMES paragraph. The currency symbol in the character-string represents a character position into which a currency symbol is to be placed when the data item is the receiving item of an elementary MOVE statement. Each currency symbol is counted in the size of the data item described by the PICTURE character-string. If the currency symbol occurs more than once (a floating currency symbol), then one less than the total number of currency symbols is counted in determining the maximum number (30) of digit positions allowed in a numeric edited data item. If the currency symbol is used to the right of the decimal point in a character-string, then all digit positions in that character-string must be described with the currency symbol. If a floating currency symbol string represents all the digit-positions in the character-string, then the described data item is blank when zero, even if the BLANK WHEN ZERO clause is not specified.

LIKE Pattern Grammar

The grammar for a regular expression that specifies the pattern for a LIKE condition is as follows:

```
 [1] regExp          ::= branch ( '|' branch )*

 [2] branch          ::= piece*

 [3] piece           ::= atom quantifier?

 [4] quantifier      ::= [?*+] | ( '{' quantity '}' )

 [5] quantity        ::= quantRange | quantMin | QuantExact

 [6] quantRange      ::= QuantExact ',' QuantExact

 [7] quantMin        ::= QuantExact ','

 [8] QuantExact      ::= [0-9]+

 [9] atom            ::= Char | charClass | ( '(' regExp ')' )

[10] Char            ::= [^.\?*+()|#x5B#x5D]

[11] charClass       ::= charClassEsc | charClassExpr

[12] charClassExpr   ::= '[' charGroup ']'
```

```
[13] charGroup      ::= posCharGroup | negCharGroup |charClassSub

[14] posCharGroup   ::= ( charRange | charClassEsc )+

[15] negCharGroup   ::= '^' posCharGroup

[16] charClassSub   ::= ( posCharGroupND | negCharGroupND )
                          '-' charClassExpr

[17] negCharGroupND ::= '^' posCharGroupND

[18] posCharGroupND ::= ( XmlCharRef | XmlChar | charClassEsc )+

[19] XmlCharRef     ::= ( '&#' [0-9]+ ';' ) |
                          ( '&#x' [0-9a-fA-F]+ ';' )

[20] XmlChar        ::= [^\#x2D#x5B#x5D]

[21] charRange      ::= seRange | XmlCharRef | XmlCharIncDash

[22] seRange        ::= charOrEsc '-' charOrEsc

[23] charOrEsc      ::= XmlChar | SingleCharEsc

[24] XmlCharIncDash ::= [^\#x5B#x5D]

[25] charClassEsc   ::= ( SingleCharEsc | MultiCharEsc |
                          catEsc | complEsc )

[26] SingleCharEsc  ::= '\' [nrt\|.?*+(){}#x2D#x5B#x5D#x5E]

[27] catEsc         ::= '\p{' charProp '}'

[28] complEsc       ::= '\P{' charProp '}'

[29] charProp       ::= IsCategory | IsBlock

[30] IsCategory     ::= Letters | Marks | Numbers |
                          Punctuation | Separators |
                          Symbols | Others

[31] Letters        ::= 'L' [ultmo]?

[32] Marks          ::= 'M' [nce]?

[33] Numbers        ::= 'N' [dlo]?
```

```
[34] Punctuation    ::= 'P' [cdseifo]?

[35] Separators     ::= 'Z' [slp]?

[36] Symbols        ::= 'S' [mcko]?

[37] Others         ::= 'C' [cfon]?

[38] IsBlock        ::= 'Is' [a-zA-Z [0-9a-zA-Z#x2D]*

[39] MultiCharEsc   ::= '.' | ( '\' [sSiIcCdDwW] )
```

Note that in the grammar, quoted characters, for example '|', in a rule indicate that the literal character itself may appear in a regular expression derived from the rule.

In the grammar, certain unquoted characters have special meaning as follows:

* zero or more occurrences are allowed (Kleene closure)

\+ one or more occurrences are allowed (positive closure)

? zero or no occurrences are allowed (optional)

[] any of the class of characters contained between the brackets. A hyphen is used to represent a range of characters, unless the hyphen is the first or last character in the class, in which case it represents a hyphen character in the class.

[^] any character other than the class of characters between the brackets and following the ^. For example, [^0-9] means any character other than a decimal digit.

Note These characters have similar meaning when used in an actual pattern regular expression, but their use in the grammar is distinct from their occurrence in a pattern. For example, grammar rule 4 shows that the ?, *, and + characters may be used in a pattern by giving the grammar class expression [?*+].

In the grammar, some characters are represented by the hexadecimal representation #x*hh*, where *hh* specifies the two hexadecimal digits for the code-point of the desired character.

Here are some examples of patterns that may be used for a LIKE condition.

Pattern	Meaning
Box	The string "Box".
\s*(dog|cat)\s*	Zero or more white space characters followed by the string "dog" or the string "cat" followed by zero or more white space characters.
([Cc]at|[Tt]ext) box	The strings "Cat box", "cat box", "Text box", or "text box".
[0-9]+\.[0-9]{1,5}	One or more decimal digits followed by a decimal point followed by 1 to 5 decimal digits.
\d+\.\d+\D*	One or more decimal digits followed by a decimal point followed by one or more decimal digits followed by zero or more characters other than decimal digits.

Pattern	Meaning
\d{1,3}(,\d{3})*\.\d+	One to three decimal digits followed by zero or more occurrences of three decimal digits with a leading comma followed by a decimal point followed by one or more decimal digits.
.*Butter.*	Zero or more of any character followed by the string "Butter" followed by zero or more of any character.
(cat)?box	The string "cat box" or the string "box".
\p{Ll}{3,}	Three or more lower-case letter characters.
.*[\p{Lu}-[M-P]]+	Zero or more of any character followed by one or more of any character in the class of upper-case letters excluding M, N, O, and P.

The following quantifier equivalences occur in a regular expression.

Short Quantifier	Equivalent Quantifier	Meaning
?	{0,1}	Zero or one (optional)
*	{0,}	Zero or more (Kleene closure)
+	{1,}	One or more (positive closure)

The following XML entity references are recognized in a regular expression and converted to the corresponding character.

Entity Reference	Character	Description
&	&	ampersand
'	'	apostrophe
<	<	less than sign
>	>	greater than sign
"	"	double quote

These XML entity references are recognized in addition to XML character references. XML character references specify a particular code-point with the forms &#*d*, where *d* is the decimal value of the code-point, or &#x*h*, where *h* is the hexadecimal value of the code-point, per rule 19 of the grammar.

The following escape sequences represent a single character in a regular expression.

Escape Sequence	Character
\n	newline (
)
\r	return (
)
\t	horizontal tab ( )
\\	\
|	|

Escape Sequence	Character
\.	.
\-	-
\^	^
\?	?
*	*
\+	+
\{	{
\}	}
\(	(
\)	)
\[	[
\]	]

The following escape sequences represent multiple characters; that is, a character class, in a regular expression.

Escape Sequence	Equivalent Character Class	Meaning
.	[^\n\r]	Any character except newline or return.
\s	[\t\n\r]	White space.
\S	[^\s]	Not whitespace.
\i	[\p{L}_:]	Initial name characters (of XML).
\I	[^\i]	Not initial name characters (of XML).
\c	[\i\d\.·-]	Name characters (of XML). **Note** The B7h code point in Unicode is the “MIDDLE DOT” extender character and is classified as a name character. Therefore, XML name characters include this code point value.
\C	[^\c]	Not name characters (of XML).
\d	\p{Nd}	Numeric digits.
\D	[^\d]	Not numeric digits.
\w	[&#x00;-ÿ- [\p{P}\p{Z}\p{S}\p{C}]]	All characters except punctuation, separator, symbol, and other characters.
\W	[^\w]	Punctuation, separator, symbol and other characters.

The following Unicode categories may be specified in a regular expression category escape by use of the indicated property designator.

Category	Property Designator	Character Class
Letters	L	All letters.
	Lu	Uppercase letters.
	Ll	Lowercase letters.
	Lt	Title case letters.
	Lm	Modifier letters.
	Lo	Other letters.
Marks	M	All marks.
	Mn	Non-spacing marks.
	Mc	Spacing combining marks.
	Me	Enclosing marks.
Numbers	N	All numbers.
	Nd	Decimal digit numbers.
	Nl	Letter numbers.
	No	Other numbers.
Punctuation	P	All punctuation.
	Pc	Connector punctuation.
	Pd	Dash punctuation.
	Ps	Open punctuation.
	Pe	Close punctuation.
	Pi	Initial quote punctuation.
	Pf	Final quote punctuation.
	Po	Other punctuation.
Separators	Z	All separators.
	Zs	Space separators.
	Zl	Line separators.
	Zp	Paragraph separators.
Symbols	S	All symbols.
	Sm	Math symbols.
	Sc	Currency symbols.
	Sk	Modifier symbols.
	So	Other symbols.
Other	C	All others.
	Cc	Control others.
	Cf	Format others.
	Co	Private use others.

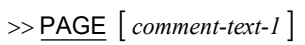
Category	Property Designator	Character Class
	Cn	Not assigned others.

Directives

IMP MARGIN-R

LISTING

PAGE



Program Structure

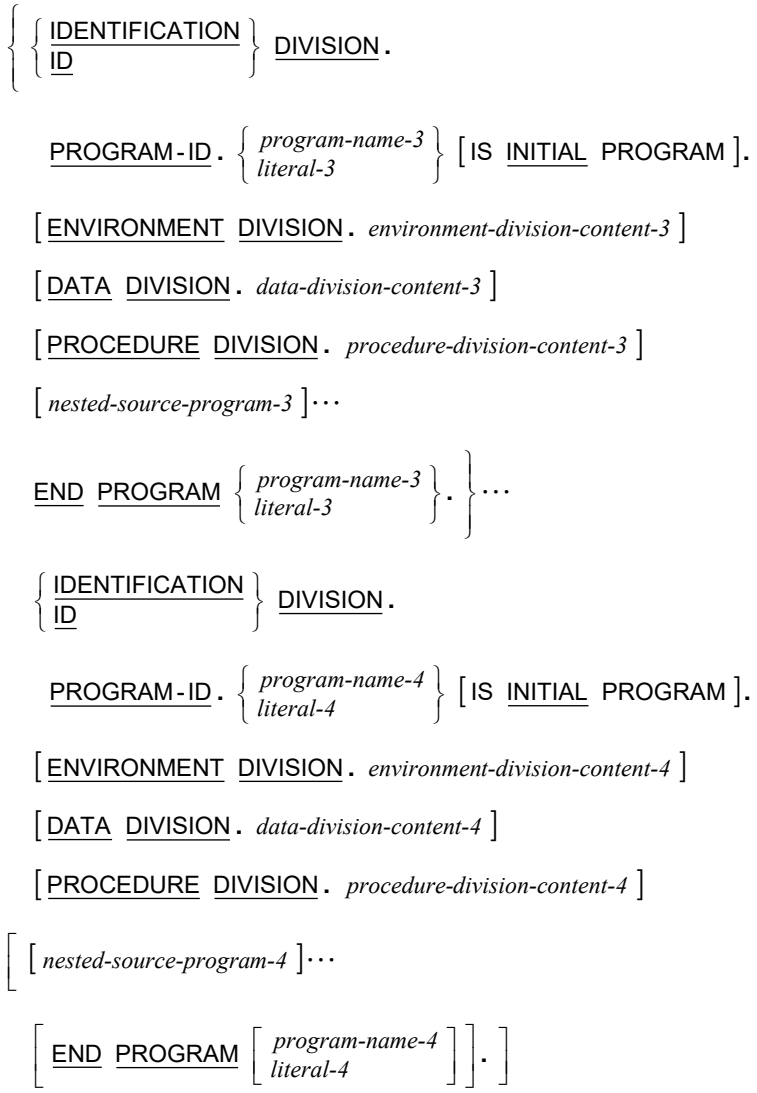
General Format for Nested Source Programs

```
{ IDENTIFICATION } DIVISION.
{ ID             }

 PROGRAM-ID. { program-name-1 } [ IS INITIAL PROGRAM ].
             { literal-1      }

[ ENVIRONMENT DIVISION. environment-division-content-1 ]

[ DATA DIVISION. data-division-content-1 ]

[ PROCEDURE DIVISION. procedure-division-content-1 ]

[ [ nested-source-program-1 ]... ]
 END PROGRAM [ program-name-1 ]. ]
             [ literal-1      ]
```

General Format for *nested-source-program*

```
{ IDENTIFICATION } DIVISION.
{ ID             }

 PROGRAM-ID. { program-name-2 } [ IS { COMMON  } PROGRAM ].
             { literal-2      }      { INITIAL }

[ ENVIRONMENT DIVISION. environment-division-content-2 ]

[ DATA DIVISION. data-division-content-2 ]

[ PROCEDURE DIVISION. procedure-division-content-2 ]

[ nested-source-program-2 ]...

 END PROGRAM [ program-name-2 ].
             [ literal-2      ]
```

General Format for a Sequence of Source Programs



COBOL Words

The reserved words are divided into the following alphabetical groups:

- Reserved Words (A - B) on page 72
- Reserved Words (C) on page 72
- Reserved Words (D) on page 73
- Reserved Words (E) on page 73
- Reserved Words (F - I) on page 74
- Reserved Words (J - N) on page 75

- Reserved Words (O - Q) on page 75
- Reserved Words (R) on page 76
- Reserved Words (S) on page 76
- Reserved Words (T - Z) on page 77

† *This word is not considered reserved if the RM/COBOL (74) 2.0 compatibility option is present in the Compile Command on page* 1. *In such cases, this word is treated as a user-defined word whenever it occurs in the source program. For further information, see* Chapter 6: Compiling*, in the* RM/COBOL User's Guide.

Reserved Words (A - B)

Reserved Words (A - B)

ACCEPT	ALPHANUMERIC-EDITED †	AT
ACCESS	ALSO †	AUTHOR
ADD	ALTER	
ADDRESS †	ALTERNATE	BEEP
ADVANCING	AND	BEFORE
AFTER	ANY †	BELL †
ALL	ARE	BINARY
ALPHABET †	AREA	BLANK
ALPHABETIC	AREAS	BLINK
ALPHABETIC-LOWER †	AS †	BLOCK
ALPHABETIC-UPPER †	ASCENDING †	BOTTOM †
ALPHANUMERIC †	ASSIGN	BY

Reserved Words (C)

Reserved Words (C)

CALL	COLUMN †	CONFIGURATION
CANCEL	COMMA	CONTAINS
CD †	COMMON †	CONTENT †
CENTURY-DATE †	COMMUNICATION †	CONTINUE †
CENTURY-DAY †	COMP	CONTROL †
CF †	COMP-1	CONTROLS †
CH †	COMP-3	CONVERT

Reserved Words (C)		
CHARACTER	COMP-4 †	CONVERTING †
CHARACTERS	COMP-5 †	COPY
CLASS †	COMP-6	CORR
CLOCK-UNITS †	COMPUTATIONAL	CORRESPONDING
CLOSE	COMPUTATIONAL-1	COUNT †
COBOL †	COMPUTATIONAL-3	COUNT-MAX †
CODE †	COMPUTATIONAL-4 †	COUNT-MIN †
CODE-SET	COMPUTATIONAL-5 †	CURRENCY
COL †	COMPUTATIONAL-6	CURSOR †
COLLATING	COMPUTE	

Reserved Words (D)

Reserved Words (D)		
DATA	DEBUG-LINE †	DEPENDING
DATA-POINTER †	DEBUG-NAME †	DESCENDING †
DATE	DEBUG-SUB-1 †	DESTINATION †
DATE-AND-TIME †	DEBUG-SUB-2 †	DETAIL †
DATE-COMPILED †	DEBUG-SUB-3 †	DISABLE †
DATE-WRITTEN	DEBUGGING †	DISPLAY
DAY	DECIMAL-POINT	DIVIDE
DAY-AND-TIME †	DECLARATIVES	DIVISION
DAY-OF-WEEK †	DEFAULT †	DOWN
DE †	DELETE	DUPLICATES
DEBUG-CONTENTS †	DELIMITED †	DYNAMIC
DEBUG-ITEM †	DELIMITER †	

Reserved Words (E)

Reserved Words (E)		
ECHO	END-MULTIPLY †	ENVIRONMENT
EGI †	END-OF-PAGE †	EOP †
ELSE	END-PERFORM †	EQUAL

EMI †	END-READ †	ERASE
ENABLE †	END-RECEIVE †	ERROR
END	END-RETURN †	ESCAPE †
END-ACCEPT †	END-REWRITE †	ESI †
END-ADD †	END-SEARCH †	EVALUATE †
END-CALL †	END-START †	EVERY †
END-COMPUTE †	END-STRING †	EXCEPTION
END-DELETE †	END-SUBTRACT †	EXCLUSIVE †
END-DIVIDE †	END-UNSTRING †	EXIT
END-EVALUATE †	END-WRITE †	EXTEND
END-IF †	ENTER †	EXTERNAL †

Reserved Words (F - I)

Reserved Words (F - I)

FALSE †	GOBACK †	IN
FD	GREATER	INDEX
FILE	GROUP †	INDEXED
FILE-CONTROL		INDICATE †
FILLER	HEADING †	INITIAL
FINAL †	HIGH	INITIAL-VALUE †
FIRST	HIGH-VALUE	INITIALIZE †
FIXED †	HIGH-VALUES	INITIATE †
FOOTING †	HIGHEST-VALUE	INPUT
FOR	HIGHLIGHT †	INPUT-OUTPUT
FROM		INSPECT
FUNCTION †	I-O	INSTALLATION
	I-O-CONTROL	INTO
GENERATE †	ID †	INVALID
GIVING	IDENTIFICATION	IS
GLOBAL †	IF	
GO	IMP †	

Reserved Words (J - N)

Reserved Words (J - N)		
JUST	LINE	MODE
JUSTIFIED	LINE-COUNTER †	MODULES
	LINES	MOVE
KEY	LINKAGE	MULTIPLY
	LOCK	
LABEL	LOW	NATIVE
LAST †	LOW-VALUE	NEGATIVE †
LEADING	LOW-VALUES	NEXT
LEFT	LOWEST-VALUE	NO
LENGTH †	LOWLIGHT †	NOT
LESS		NULL †
LIKE †	MAX-VALUE †	NULLS †
LIMIT †	MEMORY	NUMBER †
LIMITS †	MERGE †	NUMERIC
LINAGE †	MESSAGE †	NUMERIC-EDITED †
LINAGE-COUNTER †	MIN-VALUE †	

Reserved Words (O - Q)

Reserved Words (O - Q)		
OBJECT-COMPUTER	PACKED-DECIMAL †	PROCEDURE
OCCURS	PADDING †	PROCEDURE-NAME †
OF	PAGE	PROCEDURES †
OFF	PAGE-COUNTER †	PROCEED
OMITTED	PERFORM	PROGRAM
ON	PF †	PROGRAM-ID
OPEN	PH †	PROMPT
OPTIONAL †	PIC	PURGE †
OR	PICTURE	
ORDER †	PLUS †	QUEUE †
ORGANIZATION	POINTER †	QUOTE
OTHER †	POSITION	QUOTES

Reserved Words (O - Q)	
OUTPUT	POSITIVE †
OVERFLOW	PRINTING †

Reserved Words (R)

Reserved Words (R)		
RANDOM	REMAINDER	RETURN-CODE †
RD †	REMARKS †	RETURNING †
READ	REMOVAL †	REVERSE
RECEIVE †	RENAMES	REVERSE-VIDEO †
RECORD	REPLACE †	REVERSED †
RECORDING †	REPLACING	REWIND
RECORDS	REPORT †	REWRITE
REDEFINES	REPORTING †	RF †
REEL	REPORTS †	RH †
REFERENCE †	RERUN †	RIGHT
REFERENCES †	RESERVE	ROUNDED
RELATIVE	RESET †	RUN
RELEASE †	RETURN †	

Reserved Words (S)

Reserved Words (S)		
SAME	SEQUENTIAL	START
SCREEN †	SET	STATUS
SD †	SIGN	STOP
SEARCH †	SIZE	STRING †
SECTION	SORT †	SUB-QUEUE-1 †
SECURE †	SORT-MERGE †	SUB-QUEUE-2 †
SECURITY	SOURCE †	SUB-QUEUE-3 †
SEGMENT †	SOURCE-COMPUTER	SUBTRACT
SEGMENT-LIMIT †	SPACE	SUM †
SELECT	SPACES	SUPPRESS †

Reserved Words (S)		
SEND †	SPECIAL-NAMES	SYMBOLIC †
SENTENCE	STANDARD	SYNC
SEPARATE	STANDARD-1	SYNCHRONIZED
SEQUENCE	STANDARD-2 †	

Reserved Words (T - Z)

Reserved Words (T - Z)		
TAB	TOP †	VALUE
TABLE †	TRAILING	VALUES
TALLYING	TRUE †	VARIABLE †
TAPE †	TYPE †	VARYING
TERMINAL †		
TERMINATE †	UNIT	WHEN
TEST †	UNLOCK	WHEN-COMPILED †
TEXT †	UNSTRING †	WITH
THAN	UNTIL	WORDS
THEN †	UP	WORKING-STORAGE
THROUGH	UPDATE	WRITE
THRU	UPON †	
TIME	USAGE	ZERO
TIMES	USE	ZEROES
TO	USING	ZEROS

Unused Reserved Words

RM/COBOL reserves several words that do not currently appear in any format. These words are reserved because they are reserved words in ANSI COBOL within an optional module not supported by RM/COBOL or within another dialect of COBOL. The ANSI COBOL optional modules not supported by RM/COBOL include the Debug Module, the Intrinsic Function Module, and the Report Writer Module. Note that the Debug Module was stated to be obsolete in the 1985 ANSI COBOL Standard, which means it is to be removed from the next revision of ANSI COBOL.

The unused reserved words are as follows:

CF; CH; CODE; CONTROLS; DE; DEBUG-CONTENTS; DEBUG-ITEM; DEBUG-LINE; DEBUG-NAME; DEBUG-SUB-1; DEBUG-SUB-2; DEBUG-SUB-3; DETAIL; FINAL; FIXED; FUNCTION; GENERATE; GROUP; HEADING; INDICATE; INITIATE; LIMIT;

LIMITS; LINE-COUNTER; PAGE-COUNTER; PF; PH; PROCEDURES; RD; RECORDING; REFERENCES; REPORT; REPORTING; REPORTS; RESET; RF; RH; SUM; TERMINATE; TYPE; VARIABLE

Context-Sensitive Words

A context-sensitive word is a COBOL word that is reserved only in the context of the general formats in which it is specified. In other contexts, the word can be used as a user-defined word, for example, as a user-defined data-name.

Context-sensitive words and the contexts in which they are reserved are specified in the following table.

† *This word is not considered reserved if the RM/COBOL (74) 2.0 compatibility option is present in the Compile Command on page* 1. *In such cases, this word is treated as a user-defined word whenever it occurs in the source program. For further information, see* Chapter 6: Compiling*, in the* RM/COBOL User's Guide.

Context-Sensitive Word	Language Construct or Context
AUTO †	Screen description entry (for AUTO clause) Format 3 (Terminal I-O) ACCEPT statement (for AUTO phrase)
AUTO-SKIP †	Screen description entry (for AUTO-SKIP clause) Format 3 (Terminal I-O) ACCEPT statement (for AUTO-SKIP phrase)
AUTOMATIC †	LOCK MODE clause in file control entry
BACKGROUND †	Screen description entry (for BACKGROUND clause)
BACKGROUND-COLOR †	Screen description entry (for BACKGROUND-COLOR clause)
CARD-PUNCH	ASSIGN clause (*device-name*) in file control entry
CARD-READER	ASSIGN clause (*device-name*) in file control entry
CASE-INSENSITIVE †	LIKE relational operator in LIKE relation condition
CASE-SENSITIVE †	LIKE relational operator in LIKE relation condition
CASSETTE	ASSIGN clause (*device-name*) in file control entry
CONSOLE	ASSIGN clause (*device-name*) in file control entry Special-Names paragraph (for CONSOLE IS *mnemonic-name* and CONSOLE IS CRT clauses)
CRT †	Special-Names paragraph (for CONSOLE IS CRT and CRT STATUS clauses)
CYCLE †	Format 3 EXIT statement
DISC	ASSIGN (*device-name*) clause in file control entry
DISK	ASSIGN (*device-name*) clause in file control entry
END-COPY †	COPY statement

Context-Sensitive Word	Language Construct or Context
END-REPLACE †	REPLACE statement
EOL	ERASE clause in screen description entry ERASE phrase in ACCEPT and DISPLAY statements
EOS	ERASE clause in screen description entry ERASE phrase in ACCEPT and DISPLAY statements
FOREGROUND †	Screen description entry (for FOREGROUND clause)
FOREGROUND-COLOR †	Screen description entry (for FOREGROUND-COLOR clause)
FULL †	Screen description entry (for FULL clause)
IMP †	Compiler directive (for implementor-defined directive)
KEYBOARD	ASSIGN clause (*device-name*) in file control entry
LISTING	ASSIGN clause (*device-name*) in file control entry Compiler directive (for LISTING directive)
MAGNETIC-TAPE	ASSIGN clause (*device-name*) in file control entry
MANUAL †	LOCK MODE clause in file control entry
MARGIN-R †	IMP compiler directive (for implementor-defined MARGIN-R directive)
MULTIPLE †	LOCK MODE clause in file control entry I-O-CONTROL paragraph (for MULTIPLE FILE TAPE clause)
PARAGRAPH †	Format 4 EXIT statement PROCEDURE-NAME special register
PREVIOUS †	Format 1 READ statement
PRINT	ASSIGN clause (*device-name*) in file control entry
PRINTER	ASSIGN clause (*device-name*) in file control entry
PRINTER-1	ASSIGN clause (*device-name*) in file control entry
REQUIRED †	Screen description entry (for REQUIRED clause)
SORT-WORK	ASSIGN clause (*device-name*) in file control entry
TRIMMED †	LIKE relational operator in LIKE relation condition
UNDERLINE †	Screen description entry (for UNDERLINE clause)
WHILE †	START statement (for WHILE phrase)
YYYYDDD †	FROM DAY phrase in ACCEPT statement (Format 2)
YYYYMMDD †	FROM DATE phrase in ACCEPT statement (Format 2)

Nonreserved System-Names

Code-Name

EBCDIC

(Color-Integer) Color-Names

(0) BLACK
(1) BLUE
(2) GREEN
(3) CYAN
(4) RED
(5) MAGENTA
(6) BROWN
(7) WHITE

Computer-Names

user-defined-word-1

Delimiter-Names

BINARY-SEQUENTIAL, LINE-SEQUENTIAL

Device-Names

CARD-PUNCH, CARD-READER, CASSETTE, CONSOLE, DISC, DISK, KEYBOARD, LISTING, MAGNETIC-TAPE, PRINT, PRINTER, PRINTER-1, SORT-WORK

Feature-Names

C01, C02, C03, C04, C05, C06, C07, C08, C09, C10, C11, C12

Label-Names

FILE-ID
user-defined-word-2

Language-Names

user-defined-word-3

Low-Volume-I-O-Names

CONSOLE, SYSIN, SYSOUT

Rerun-Names

user-defined-word-4

Switch-Names

```
SWITCH-1, SWITCH-2, SWITCH-3, SWITCH-4, SWITCH-5, SWITCH-6, SWITCH-7,
SWITCH-8

UPSI-0, UPSI-1, UPSI-2, UPSI-3, UPSI-4, UPSI-5, UPSI-6, UPSI-7
```

RM/COBOL Language Examples

The examples in the following sections illustrate the RM/COBOL language syntax for the procedure division verbs. Some data division excerpts are shown to help understand statement syntax for the verb.

ACCEPT Statement Examples

ACCEPT Format 1

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  ACCEPT01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  ACCEPT Format 1 statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     SYSIN IS input-terminal.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 NEXT-ITEM              PIC X(10).
 01 continuation-response  PIC X(02).
 PROCEDURE DIVISION.
 0010.
     ACCEPT NEXT-ITEM FROM CONSOLE.
     ACCEPT continuation-response FROM input-terminal.

 END PROGRAM ACCEPT01.
```

ACCEPT Format 2

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  ACCEPT02.
*
*  Examples for RM/COBOL Language Reference Manual.
*  ACCEPT Format 2 statement.
*
```

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    SYSIN IS input-terminal.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 YEAR-DAY-VALUE       PIC 99/999.
01 YEAR-MONTH-DAY-VALUE PIC 99/99/99.
01 TIME-VALUE           PIC 99/99/99/99.
01 CENTURY-DATE-VALUE   PIC 9999/99/99.
01 CENTURY-DAY-VALUE    PIC 9999/999.
01 DATE-AND-TIME-VALUE  PIC 9999/99/99BB99/99/99/99.
01 COMPILATION-DATE     PIC 9999/99/99.
01 DUMMY                PIC X.
PROCEDURE DIVISION.
0010.
    ACCEPT YEAR-DAY-VALUE FROM DAY.
    ACCEPT YEAR-MONTH-DAY-VALUE FROM DATE.
    ACCEPT TIME-VALUE FROM TIME.
    ACCEPT CENTURY-DATE-VALUE FROM CENTURY-DATE.
    ACCEPT CENTURY-DATE-VALUE FROM DATE YYYYMMDD.
    ACCEPT CENTURY-DAY-VALUE FROM CENTURY-DAY.
    ACCEPT CENTURY-DAY-VALUE FROM DAY YYYYDDD.
    ACCEPT DATE-AND-TIME-VALUE FROM DATE-AND-TIME.
    ACCEPT COMPILATION-DATE FROM DATE-COMPILED.

    INSPECT TIME-VALUE REPLACING ALL "/" BY ":".
    INSPECT DATE-AND-TIME-VALUE REPLACING ALL "/" BY ":"
         AFTER INITIAL SPACE.

    DISPLAY "YEAR-DAY-VALUE = " YEAR-DAY-VALUE.
    DISPLAY "TIME-VALUE = " TIME-VALUE.
    DISPLAY "CENTURY-DAY-VALUE = " CENTURY-DAY-VALUE.
    DISPLAY "DATE-AND-TIME-VALUE = " DATE-AND-TIME-VALUE.
    DISPLAY "COMPILATION-DATE = " COMPILATION-DATE.

    ACCEPT DUMMY PROMPT.

END PROGRAM ACCEPT02.
```

ACCEPT Format 3

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  ACCEPT03.
*
*  Examples for RM/COBOL Language Reference Manual.
*  ACCEPT Format 3 statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 ANSWER-1               PIC X(4).
 01 ANSWER-2               PIC X(4).
 01 START-VALUE            PIC S9(4)V99.
 01 K                      PIC 9(2) BINARY.
 01 NEXT-N                 PIC 9(4).
 01 DATE-G.
```

```
   02 YEAR                 PIC 9(4).
   02 MONTH                PIC 9(2).
   02 YR-LN                PIC 9(2) BINARY.
   02 YR-POS               PIC 9(2) BINARY.
   02 MN-LN                PIC 9(2) BINARY.
   02 MN-POS               PIC 9(2) BINARY.
01 PASSWORD-VALUE          PIC X(10).
01 INVENTORY-COUNT         PIC 9(4).
01 FUNCTION-CODE           PIC 9(4).
01 command-string          PIC X(10).
01 command-line            PIC 9(02) BINARY.
01 command-column          PIC 9(02) BINARY.
01 command-cursor-offset   PIC 9(02) BINARY.
01 command-control-string  PIC X(50) VALUE "PROMPT, ECHO".
01 FIELD-G.
   02 FIELD-TABLE          OCCURS 10 INDEXED BY INX1.
      03 FIELD-DATA        PIC X(10).
      03 FIELD-LINE        PIC 9(2) BINARY.
      03 FIELD-COLUMN      PIC 9(2) BINARY.
      03 FIELD-CONTROL     PIC X(80).
01 DUMMY                   PIC X.
PROCEDURE DIVISION.
0010.
    ACCEPT ANSWER-1, ANSWER-2.

    ACCEPT START-VALUE LINE 1, POSITION K,
      PROMPT, ECHO, CONVERT.

    ACCEPT NEXT-N POSITION 0, PROMPT, ECHO.


    ACCEPT YEAR, LINE YR-LN, POSITION YR-POS;
      MONTH, LINE MN-LN, POSITION MN-POS.

    ACCEPT PASSWORD-VALUE POSITION 0 OFF.

    ACCEPT INVENTORY-COUNT;
    ON EXCEPTION FUNCTION-CODE
      PERFORM FUNCTION-KEY-PROCEDURE
    END-ACCEPT.

    ACCEPT command-string
      LINE command-line
      COLUMN command-column
      CURSOR command-cursor-offset
      CONTROL command-control-string.

    ACCEPT FIELD-DATA (INX1) LINE FIELD-LINE (INX1)
      COL FIELD-COLUMN (INX1) CONTROL FIELD-CONTROL (INX1).


    ACCEPT DUMMY PROMPT.

FUNCTION-KEY-PROCEDURE.
    EXIT.

END PROGRAM ACCEPT03.
```

ACCEPT Format 4

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  ACCEPT04.
*
*  Examples for RM/COBOL Language Reference Manual.
*  ACCEPT Format 4 statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 DUMMY                    PIC X.
 COMMUNICATION SECTION.
 CD COM-LINE-1 FOR INPUT
     SYMBOLIC QUEUE IS L1-SYMQ
     SYMBOLIC SUB-QUEUE-1 IS L1-SYM-SUBQ1
     SYMBOLIC SUB-QUEUE-2 IS L1-SYM-SUBQ2
     SYMBOLIC SUB-QUEUE-3 IS L1-SYM-SUBQ3
     MESSAGE DATE IS L1-MSG-DT
     MESSAGE TIME IS L1-MSG-TM
     SYMBOLIC SOURCE IS L1-SYM-SRC
     TEXT LENGTH IS L1-TXT-LENGTH
     END KEY IS L1-END-KEY
     STATUS KEY IS L1-STATUS-KEY
     MESSAGE COUNT IS L1-MSG-COUNT.

 PROCEDURE DIVISION.
 0010.
     ACCEPT COM-LINE-1 MESSAGE COUNT.


     ACCEPT DUMMY PROMPT.

 END PROGRAM ACCEPT04.
```

ACCEPT Format 5

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  ACCEPT05.
*
*  Examples for RM/COBOL Language Reference Manual.
*    ACCEPT Format 5
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 WS-INV-DT           PIC 9(8) VALUE 02031999.
 01 WS-INV-AMT          PIC S9(7) VALUE 0.
 78 EMP-NAME-SIZE       VALUE 30.
 78 EMP-LOC-SIZE        VALUE 15.
 01 WS-EMP-NAME         PIC X(EMP-NAME-SIZE) VALUE SPACES.
 01 WS-EMP-LOC          PIC X(EMP-LOC-SIZE) VALUE SPACES.
 01 EOB-COL             PIC 9(2) BINARY VALUE 10.
 01 EOB-LINE            PIC 9(2) BINARY VALUE 15.
 01 ESCAPE-MESSAGE      PIC X(20) VALUE "Escape key!".
 SCREEN SECTION.
 01 INVOICE-FORM.
    02 BLANK SCREEN.
```

```
   02 "Invoice date:  ".
   02 INVOICE-DATE PIC 99/99/9999 FROM WS-INV-DT
                TO WS-INV-DT.
   02 "Invoice amount:  " LINE.
   02 INVOICE-AMOUNT PIC 9(5).99CR USING WS-INV-AMT.
01 EMPLOYEE-RECORD.
   02 BLANK SCREEN.
   02 "Employee name:  ".
   02 ER-NAME        PIC X(EMP-NAME-SIZE) USING WS-EMP-NAME.
   02 "Employee loc:  " LINE.
   02 ER-LOC         PIC X(EMP-LOC-SIZE) USING WS-EMP-LOC.
01 EOB-SCREEN.
   02 ERASE.
   02 "Explanation of Benefits Screen".
   02 "Benefit amount: " LINE + 2 COL 10.
   02 EOB-AMOUNT     PIC 9(5).99DB USING WS-INV-AMT.

PROCEDURE DIVISION.
A.

    DISPLAY INVOICE-FORM LINE 10 COLUMN 5.
    ACCEPT INVOICE-FORM AT LINE 10 COLUMN 5.

    DISPLAY EMPLOYEE-RECORD AT LINE 9.
    ACCEPT EMPLOYEE-RECORD LINE 9
      ON ESCAPE
        DISPLAY ESCAPE-MESSAGE LINE 23
      END-ACCEPT.

    DISPLAY EOB-SCREEN AT COL EOB-COL LINE EOB-LINE.
    ACCEPT EOB-SCREEN AT COL EOB-COL LINE EOB-LINE.

END PROGRAM ACCEPT05.
```

Add Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  ADD01.
*
*  Examples for RM/COBOL Language Reference Manual.
*    ADD statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 SALARY              PIC 9(08)V99.
 01 JOHNS-PAY           PIC 9(08)V99.
 01 PAULS-PAY           PIC 9(08)V99.
 01 ALBERTS-PAY         PIC 9(08)V99.
 01 COMPANY-PAY         PIC 9(10)V99.
 01 ACCUM-REC.
    02 DAY-TOTALS       OCCURS 31 TIMES INDEXED BY DAYX.
       03 CATEGORY-A    PIC 9(06) BINARY.
       03 CATEGORY-B    PIC 9(06) BINARY.
       03 CATEGORY-C    PIC 9(06) BINARY.
       03 CATEGORY-D    PIC 9(06) BINARY.
```

```
   02 MONTH-TOTALS     OCCURS 12 TIMES INDEXED BY MONTHX.
      03 CATEGORY-A    PIC 9(06) BINARY.
      03 CATEGORY-B    PIC 9(06) BINARY.
      03 CATEGORY-C    PIC 9(06) BINARY.
      03 CATEGORY-D    PIC 9(06) BINARY.
01 TOTAL-RECORD        PACKED-DECIMAL.
   02 ENTERTAINMENT    PIC S9(06)V99.
   02 GAS-AUTOMOTIVE   PIC S9(06)V99.
   02 HOUSING          PIC S9(06)V99.
   02 MEDICAL          PIC S9(06)V99.
   02 RESTAURANT       PIC S9(06)V99.
   02 SUPERMARKET      PIC S9(06)V99.
   02 TRAVEL           PIC S9(06)V99.
01 SUB-TOTAL-RECORD    PACKED-DECIMAL.
   02 ENTERTAINMENT    PIC S9(06)V99.
   02 GAS-AUTOMOTIVE   PIC S9(06)V99.
   02 HOUSING          PIC S9(06)V99.
   02 MEDICAL          PIC S9(06)V99.
   02 RESTAURANT       PIC S9(06)V99.
   02 SUPERMARKET      PIC S9(06)V99.
   02 TRAVEL           PIC S9(06)V99.
PROCEDURE DIVISION.
A.
    ADD SALARY TO SALARY.  *>(doubles the value of SALARY)

    ADD JOHNS-PAY, PAULS-PAY, ALBERTS-PAY
      GIVING COMPANY-PAY
    ON SIZE ERROR
      PERFORM BANKRUPTCY-PROC
    END-ADD.

    ADD CORRESPONDING
      DAY-TOTALS(DAYX) TO MONTH-TOTALS(MONTHX).

    ADD CORR SUB-TOTAL-RECORD TO TOTAL-RECORD ROUNDED
    ON SIZE ERROR GO TO ERROR-ROUTINE
    NOT ON SIZE ERROR PERFORM AUDIT-ROUTINE
    END-ADD.

AUDIT-ROUTINE.
    EXIT.

ERROR-ROUTINE.
    EXIT.

BANKRUPTCY-PROC.
    EXIT.

END PROGRAM ADD01.
```

Alter Statement Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  ALTER01.
*
```

```
*  Examples for RM/COBOL Language Reference Manual.
*    ALTER statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 EMPLOYEE-RECORD.
    02 EMP-NAME           PIC X(10).
    02 EMP-SSN            PIC 9(9) PACKED-DECIMAL.
    02 EMP-SALARY         PIC S9(8)V99 BINARY.
 PROCEDURE DIVISION.
 A.
     PERFORM SET-INITIALIZE-IT.

 SWITCH-PARAGRAPH.
     GO TO INITIALIZE-IT.
 INITIALIZE-IT.
     INITIALIZE EMPLOYEE-RECORD.
     ALTER SWITCH-PARAGRAPH TO INITIALIZED.
 INITIALIZED.

 SET-INITIALIZE-IT.
     ALTER SWITCH-PARAGRAPH TO INITIALIZE-IT.

 END PROGRAM ALTER01.
```

CALL Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  CALL01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  CALL statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 SUBPRG1              PIC X(30).
 01 CHOICE-1             PIC X(02).
 01 TABLE1.
    02 CATEGORY          OCCURS 10 INDEXED BY INX1.
       03 CAT-DESC       PIC X(10).
       03 CAT-VALUE      PIC 9(8)V99.
 01 TABLE1-TOTAL         PIC 9(10)V99.
 01 SUB-NAME-GROUP.
    02 SUBTABLE-V.
       03               PIC X(30) VALUE "APP01".
       03               PIC X(01) VALUE "F".
       03               PIC X(30) VALUE "APP02".
       03               PIC X(01) VALUE "F".
       03               PIC X(30) VALUE "APP03".
       03               PIC X(01) VALUE "F".
       03               PIC X(30) VALUE "APP04".
       03               PIC X(01) VALUE "F".
    02 SUBTABLE         REDEFINES SUBTABLE-V
                        OCCURS 4 TIMES INDEXED BY IX.
       03 SUBNAME        PIC X(30).
```

```
      03 SUB-LOAD-FLAG  PIC X.
         88 SUB-LOADED  VALUE "T" FALSE "F".
01 FUNCTION-TYPE        PIC X.
01 ITEM-1               PIC X(10).
01 ITEM-2               PIC X(10).
01 STATUS-1             PIC X.
01 SCREEN-BUFFER        PIC X(1920).
01 SCREEN-LINE          PIC 9(02) BINARY.
01 SCREEN-COLUMN        PIC 9(02) BINARY.
01 SUB-UNLOADED-FLAG    PIC X.
   88 SUB-UNLOADED      VALUE "T" FALSE "F".
PROCEDURE DIVISION.
0010.
    IF CHOICE-1 = "01"  MOVE "APP01" TO SUBPRG1
    ELSE IF CHOICE-1 = "02" MOVE "APP02" TO SUBPRG1
    ELSE PERFORM 0020-RETRY-CHOICE GO TO 0010
    END-IF END-IF.

    CALL SUBPRG1.  *>Call "APP01" or "APP02" per choice.

    CALL "REORDER" USING TABLE1 GIVING TABLE1-TOTAL.

 RETRY-1.
    CALL SUBNAME OF SUBTABLE (IX) GIVING STATUS-1
      USING FUNCTION-TYPE, ITEM-1, ITEM-2,
    ON EXCEPTION PERFORM CANCEL-PARAGRAPH GO TO RETRY-1
    NOT ON EXCEPTION SET SUB-LOADED (IX) TO TRUE
    END-CALL.

    CALL "C$SCRD" USING
      SCREEN-BUFFER, OMITTED, SCREEN-LINE, SCREEN-COLUMN.

0020-RETRY-CHOICE.
    DISPLAY "Choice not recognized.  Reenter choice:  "
      WITH NO ADVANCING.
    ACCEPT CHOICE-1.

CANCEL-PARAGRAPH.
     SET SUB-UNLOADED TO FALSE.
     PERFORM VARYING IX FROM 1 BY 1 UNTIL IX > 4
       IF SUB-LOADED OF SUBTABLE (IX)
         CANCEL SUBNAME OF SUBTABLE (IX)
         SET SUB-LOADED OF SUBTABLE (IX) TO FALSE
         SET SUB-UNLOADED TO TRUE
       END-IF
     END-PERFORM.
     IF NOT SUB-UNLOADED
       DISPLAY "Insufficient memory."
       STOP RUN
     END-IF.

END PROGRAM CALL01.
IDENTIFICATION DIVISION.
PROGRAM-ID.  APP01.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
```

```
0010.
    EXIT PROGRAM.
END PROGRAM APP01.
IDENTIFICATION DIVISION.
PROGRAM-ID.  APP02.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
0010.
    EXIT PROGRAM.
END PROGRAM APP02.
IDENTIFICATION DIVISION.
PROGRAM-ID.  REORDER.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WS-TOTAL             PIC 9(10)V99.
LINKAGE SECTION.
01 T.
   02 CATEGORY          OCCURS 10 INDEXED BY INX1.
      03 CAT-DESC       PIC X(10).
      03 CAT-VALUE      PIC 9(8)V99.
01 R                    PIC 9(10)V99.
PROCEDURE DIVISION USING T GIVING R.
0010.
    MOVE ZERO TO WS-TOTAL.
    PERFORM VARYING INX1 FROM 1 BY 1 UNTIL
                  INX1 > COUNT-MAX OF CATEGORY
      ADD CAT-VALUE(INX1) TO WS-TOTAL
    END-PERFORM.
    MOVE WS-TOTAL TO R.
    EXIT PROGRAM.
END PROGRAM REORDER.
```

CALL Program Statement Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  CALL03.
*
*  Examples for RM/COBOL Language Reference Manual.
*  CALL PROGRAM statement.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
01 COMMON-DATA           PIC X(100).
01 CHAIN-NAME            PIC X(30).
01 ARGUMENT-AREA         PIC X(200).
01 EX-STATUS             PIC 9(03).
PROCEDURE DIVISION.
0010.
    CALL PROGRAM "MENU2" USING COMMON-DATA
    ON EXCEPTION
      DISPLAY "Chain to MENU2 failed."
      STOP RUN
    END-CALL.
```

```
0020.
    CALL PROGRAM CHAIN-NAME USING ARGUMENT-AREA
    ON EXCEPTION
       ACCEPT EX-STATUS FROM EXCEPTION STATUS
       PERFORM 0030-CHAIN-ERROR-STATUS
       STOP RUN
    END-CALL.

0030-CHAIN-ERROR-STATUS.
    DISPLAY "Chain to next program failed, status = "
      EX-STATUS.

END PROGRAM CALL03.
```

CANCEL Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  CANCEL01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  CANCEL statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 SUBPROGRAM-NAME-HOLDER  PIC X(30).
 01 SUB-NAME-GROUP.
    02 SUBTABLE-V.
       03                 PIC X(30) VALUE "APP01".
       03                 PIC X(01) VALUE "F".
       03                 PIC X(30) VALUE "APP02".
       03                 PIC X(01) VALUE "F".
       03                 PIC X(30) VALUE "APP03".
       03                 PIC X(01) VALUE "F".
       03                 PIC X(30) VALUE "APP04".
       03                 PIC X(01) VALUE "F".
    02 SUBTABLE           REDEFINES SUBTABLE-V
                          OCCURS 4 TIMES INDEXED BY IX.
       03 SUBNAME         PIC X(30).
       03 SUB-LOAD-FLAG   PIC X.
          88 SUB-LOADED   VALUE "T" FALSE "F".
 01 SUB-UNLOADED-FLAG     PIC X.
    88 SUB-UNLOADED       VALUE "T" FALSE "F".
 PROCEDURE DIVISION.
 0010.

     CANCEL "SUB01", "SUB02".

     CANCEL SUBPROGRAM-NAME-HOLDER.

 CANCEL-PARAGRAPH.
      SET SUB-UNLOADED TO FALSE.
      PERFORM VARYING IX FROM 1 BY 1 UNTIL IX > 4
        IF SUB-LOADED OF SUBTABLE (IX)
          CANCEL SUBNAME OF SUBTABLE (IX)
          SET SUB-LOADED OF SUBTABLE (IX) TO FALSE
```

```
          SET SUB-UNLOADED TO TRUE
        END-IF
      END-PERFORM.
      IF NOT SUB-UNLOADED
        DISPLAY "Insufficient memory."
        STOP RUN
      END-IF.

 END PROGRAM CANCEL01.
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SUB01.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE DIVISION.
 0010.
     EXIT PROGRAM.
 END PROGRAM SUB01.
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SUB02.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE DIVISION.
 0010.
     EXIT PROGRAM.
 END PROGRAM SUB02.
```

CLOSE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  CLOSE01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  CLOSE statement.
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT TRANSACTION-FILE  ASSIGN TO TAPE.
     SELECT LOG-FILE          ASSIGN TO DISK
                              FILE STATUS IS LOG-FILE-STATUS.
     SELECT INPUT-FILE        ASSIGN TO TAPE.
     SELECT TAPE-FILE-1       ASSIGN TO TAPE.
     SELECT PRINT-FILE        ASSIGN TO PRINTER.
     SELECT DATA-BASE         ASSIGN TO DISK
                              INDEXED ACCESS DYNAMIC
                              RECORD KEY IS DB-KEY
                              FILE STATUS IS DB-STATUS.

 DATA DIVISION.
 FILE SECTION.
 FD TRANSACTION-FILE.
 01 TR-RECORD             PIC X(80).

 FD LOG-FILE.
 01 LOG-RECORD            PIC X(80).
```

```
FD INPUT-FILE.
01 IN-RECORD             PIC X(80).

FD TAPE-FILE-1.
01 TF1-RECORD            PIC X(512).

FD PRINT-FILE.
01 PF-RECORD             PIC X(60).

FD DATA-BASE.
01 DB-RECORD.
   02 DB-DATA-1          PIC X(10).
   02 DB-KEY             PIC X(20).
   02 DB-DATA-2          PIC X(50).

WORKING-STORAGE SECTION.
01 LOG-FILE-STATUS       PIC X(02).
01 DB-STATUS             PIC X(02).
PROCEDURE DIVISION.
DECLARATIVES.
I-O-ERROR SECTION.
    USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
I-O-ERROR1.
    EXIT.
END DECLARATIVES.
MAIN-01 SECTION.
0010.
    CLOSE TRANSACTION-FILE.

    CLOSE LOG-FILE WITH LOCK, PRINT-FILE.

    OPEN I-O LOG-FILE.
    IF LOG-FILE-STATUS = "38"
      DISPLAY "Log file closed with lock."
      STOP RUN
    END-IF.

    CLOSE INPUT-FILE REEL FOR REMOVAL.

    CLOSE TAPE-FILE-1 WITH NO REWIND.

    CLOSE DATA-BASE WITH LOCK.

    OPEN I-O DATA-BASE.
    IF DB-STATUS = "38"
      DISPLAY "Data-base file closed with lock."
      STOP RUN
    END-IF.

END PROGRAM CLOSE01.
```

COMPUTE Statement Example

```
IDENTIFICATION DIVISION.
```

```
 PROGRAM-ID.  COMPUTE1.
*
*  Examples for RM/COBOL Language Reference Manual.
*    COMPUTE statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 WAGES                 PIC  9(6)V99.
 01 REGULAR-HOURS         PIC S9(4)V99.
 01 OVERTIME-HOURS        PIC S9(4)V99.
 01 TOTAL-HOURS           PIC S9(4)V99.
 01 SALARY                PIC S9(10)V99.
 01 TIME-REC.
    02 HRS                PIC 9(2).
    02 MIN                PIC 9(2).
    02 SEC                PIC 9(2)V9(2).
 01 SECONDS               PIC 9(5)V9(2).
 01 AVERAGE               PIC 9(5)V9(2).
 01 TOTAL-1               PIC S9(10)V9(4).
 01 COUNT-1               PIC S9(5).
 01 PAYMENT-RND           PIC S9(6)V9(2).
 01 PAYMENT-TRUNC         PIC S9(6)V9(4).
 01 INITIAL-PRINCIPAL     PIC S9(8)V9(2) VALUE 1000.00.
 01 INTEREST-APR          PIC S9(4)V9(4) VALUE 8.25.
 01 INTEREST-PER-PERIOD   PIC S9(4)V9(4).
 01 NUMBER-OF-PERIODS     PIC S9(4)      VALUE 36.
 01 DUMMY                 PIC X.
 PROCEDURE DIVISION.
 A.
     COMPUTE TOTAL-HOURS = REGULAR-HOURS + OVERTIME-HOURS.
     IF TOTAL-HOURS > 80
       PERFORM EXCEPTIONAL-HOURS-PROC.

     COMPUTE SALARY ROUNDED = WAGES * REGULAR-HOURS
       + WAGES * OVERTIME-HOURS * 1.5.

     COMPUTE SECONDS = (((HRS * 60) + MIN) * 60) + SEC
     ON SIZE ERROR
       DISPLAY "Time value out of range."
       STOP RUN
     END-COMPUTE.

     COMPUTE AVERAGE = TOTAL-1 / COUNT-1
     ON SIZE ERROR MOVE 0 TO AVERAGE END-COMPUTE.

     COMPUTE INTEREST-PER-PERIOD ROUNDED =
       INTEREST-APR / 1200.
     COMPUTE PAYMENT-RND ROUNDED PAYMENT-TRUNC =
       (INITIAL-PRINCIPAL * INTEREST-PER-PERIOD) /
       (1 - (1 + INTEREST-PER-PERIOD) **
       (- NUMBER-OF-PERIODS)).

     DISPLAY "PAYMENT-RND   = " PAYMENT-RND CONVERT.
     DISPLAY "PAYMENT-TRUNC = " PAYMENT-TRUNC CONVERT.
     ACCEPT DUMMY PROMPT "#".

 EXCEPTIONAL-HOURS-PROC.
```

```
    EXIT.

END PROGRAM COMPUTE1.
```

CONTINUE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  CONTINUE01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  CONTINUE statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 NORMAL-RESULT         PIC X.
 01 PART-DESCRIPTION      PIC X(30).
 01 EXCP-CODE             PIC 9(3).
 PROCEDURE DIVISION.
 0010.
     CONTINUE.

     IF NORMAL-RESULT = "Y"
       CONTINUE
     ELSE
       PERFORM EXCEPTION-CASE-ANALYSIS
     END-IF.

     ACCEPT PART-DESCRIPTION UPDATE ERASE EOL
       ON EXCEPTION EXCP-CODE CONTINUE END-ACCEPT.

     STOP RUN.

 EXCEPTION-CASE-ANALYSIS.
     EXIT.

 END PROGRAM CONTINUE01.
```

DELETE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  DELETE01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  DELETE statement (relative and indexed I-O).
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT INVENTORY-FILE    ASSIGN TO DISK
```

```
                              RELATIVE ACCESS RANDOM
                                RELATIVE KEY IS INV-KEY.

    SELECT DATA-BASE          ASSIGN TO DISK
                              INDEXED ACCESS DYNAMIC
                              RECORD KEY IS DB-KEY
                              FILE STATUS IS DB-STATUS.

    SELECT STATUS-FILE        ASSIGN TO DISK
                              RELATIVE ACCESS RANDOM
                                RELATIVE KEY IS SF-KEY.

DATA DIVISION.
FILE SECTION.
FD INVENTORY-FILE.
01 INVENTORY-RECORD      PIC X(80).

FD DATA-BASE.
01 DB-RECORD.
   02 DB-DATA-1          PIC X(10).
   02 DB-KEY             PIC X(20).
   02 DB-DATA-2          PIC X(50).

FD STATUS-FILE.
01 STATUS-RECORD         PIC X(1).

WORKING-STORAGE SECTION.
01 DB-STATUS             PIC X(02).
01 DB-DELETE-KEY         PIC X(20).
01 INV-KEY               PIC 9(5) BINARY.
01 SF-KEY                PIC 9(5) BINARY.
PROCEDURE DIVISION.
DECLARATIVES.
I-O-ERROR SECTION.
    USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
I-O-ERROR1.
    EXIT.
END DECLARATIVES.
MAIN-01 SECTION.
0010.

    DELETE INVENTORY-FILE RECORD; INVALID KEY
      PERFORM BAD-KEY-PROCEDURE END-DELETE.

    DELETE STATUS-FILE RECORD.

    MOVE DB-DELETE-KEY TO DB-KEY.
    DELETE DATA-BASE RECORD
    INVALID KEY PERFORM DB-INVALID-KEY-HANDLER
    NOT INVALID KEY PERFORM DB-SUCCESS-HANDLER
    END-DELETE.
BAD-KEY-PROCEDURE.
    EXIT.

DB-SUCCESS-HANDLER.
    EXIT.
```

```
DB-INVALID-KEY-HANDLER.
    EXIT.

END PROGRAM DELETE01.
```

DELETE FILE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  DELETE02.
*
*  Examples for RM/COBOL Language Reference Manual.
*  DELETE FILE statement.
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT TEMP-FILE-1       ASSIGN TO DISK.

     SELECT TEMP-FILE-2       ASSIGN TO DISK.

     SELECT OLD-TRANSACTION-FILE
                              ASSIGN TO DISK.

 DATA DIVISION.
 FILE SECTION.
 FD TEMP-FILE-1.
 01 TF1-RECORD            PIC X(80).

 FD TEMP-FILE-2.
 01 TF2-RECORD            PIC X(80).

 FD OLD-TRANSACTION-FILE.
 01 OTF-RECORD.
    02 DB-DATA-1          PIC X(10).
    02 DB-KEY             PIC X(20).
    02 DB-DATA-2          PIC X(50).

 WORKING-STORAGE SECTION.
 PROCEDURE DIVISION.
 DECLARATIVES.
 I-O-ERROR SECTION.
     USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
 I-O-ERROR1.
     EXIT.
 END DECLARATIVES.
 MAIN-01 SECTION.
 0010.

     DELETE FILE TEMP-FILE-1 TEMP-FILE-2.

     DELETE FILE OLD-TRANSACTION-FILE END-DELETE.

 END PROGRAM DELETE02.
```

DISABLE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  DISABLE1.
*
*  Examples for RM/COBOL Language Reference Manual.
*  DISABLE statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 COM-PASSWORD          PIC X(30).
 COMMUNICATION SECTION.
 CD INPUT-COM FOR INPUT
     SYMBOLIC QUEUE IS INPUT-SYMQ
     SYMBOLIC SUB-QUEUE-1 IS INPUT-SYM-SUBQ1
     SYMBOLIC SUB-QUEUE-2 IS INPUT-SYM-SUBQ2
     SYMBOLIC SUB-QUEUE-3 IS INPUT-SYM-SUBQ3
     MESSAGE DATE IS INPUT-MSG-DT
     MESSAGE TIME IS INPUT-MSG-TM
     SYMBOLIC SOURCE IS INPUT-SYM-SRC
     TEXT LENGTH IS INPUT-TXT-LENGTH
     END KEY IS INPUT-END-KEY
     STATUS KEY IS INPUT-STATUS-KEY
     MESSAGE COUNT IS INPUT-MSG-COUNT.

 CD COM-LINE-1 FOR OUTPUT
     DESTINATION COUNT IS L1-DEST-COUNT
     TEXT LENGTH IS L1-TEXT-LENGTH
     STATUS KEY IS L1-STATUS-KEY
     DESTINATION TABLE OCCURS 5 TIMES
       INDEXED BY L1IX1, L1IX2
     ERROR KEY IS L1-ERROR-KEY
     SYMBOLIC DESTINATION IS L1-SYM-DEST.

 PROCEDURE DIVISION.
 0010.

     DISABLE INPUT INPUT-COM.

     DISABLE OUTPUT COM-LINE-1 WITH KEY COM-PASSWORD.

 END PROGRAM DISABLE1.
```

DISPLAY Statement Examples

DISPLAY Format 1

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  DISPLAY1.
*
*  Examples for RM/COBOL Language Reference Manual.
*  DISPLAY Format 1 (DISPLAY ... UPON) statement.
```

```
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     SYSOUT IS SYSTEM-OUTPUT.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 PROMPT-STRING        PIC X(5) VALUE "HELLO".
 01 OPERATOR-MESSAGE     PIC X(70).
 01 DUMMY                PIC X.
 PROCEDURE DIVISION.
 0010.
     DISPLAY "[" PROMPT-STRING "] " UPON SYSTEM-OUTPUT
       WITH NO ADVANCING.

     DISPLAY OPERATOR-MESSAGE UPON CONSOLE.

     ACCEPT DUMMY PROMPT.

 END PROGRAM DISPLAY1.
```

DISPLAY Format 2

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  DISPLAY2.
*
*  Examples for RM/COBOL Language Reference Manual.
*  DISPLAY Format 2 (Terminal I-O) statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     SYSOUT IS SYSTEM-OUTPUT.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 FLT-LN                  PIC 9(2) BINARY VALUE 10.
 01 GATE-NUMBER             PIC 9(3).
 01 MENU-HEADER             PIC X(70).
 01 REPORT-LINE             PIC X(40).
 01 display-group.
    02 display-table        OCCURS 5 TIMES INDEXED BY IX.
       03 display-data      PIC X(80).
       03 display-line      PIC 9(2) BINARY.
       03 display-column    PIC 9(2) BINARY.
       03 display-size      PIC 9(2) BINARY.
       03 display-control   PIC X(80).
 01 DUMMY                   PIC X.
 PROCEDURE DIVISION.
 0010.
     DISPLAY "Flight arriving at gate:", LINE FLT-LN,
       POSITION 1, ERASE; GATE-NUMBER, HIGH, BLINK.

     DISPLAY "Enter job code: " LINE 12 COLUMN 5.
```

```
    DISPLAY MENU-HEADER LINE 1 ERASE HIGH.

    DISPLAY ZEROES SIZE 5.  *> displays "00000"

    DISPLAY QUOTE. *> displays """" (one quote character)

    DISPLAY REPORT-LINE CONTROL "HIGH, ERASE EOL".

    DISPLAY display-data (ix),
      LINE display-line (ix),
      COL  display-column (ix),
      SIZE display-size (ix),
      CONTROL display-control (ix).

    ACCEPT DUMMY PROMPT.

END PROGRAM DISPLAY2.
```

DISPLAY Format 3

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  DISPLAY3.
*
*  Examples for RM/COBOL Language Reference Manual.
*    DISPLAY Format 3
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 WS-INV-DT           PIC 9(8) VALUE 02031999.
 01 WS-INV-AMT          PIC S9(7) VALUE 0.
 78 EMP-NAME-SIZE       VALUE 30.
 78 EMP-LOC-SIZE        VALUE 15.
 01 WS-EMP-NAME         PIC X(EMP-NAME-SIZE) VALUE SPACES.
 01 WS-EMP-LOC          PIC X(EMP-LOC-SIZE) VALUE SPACES.
 01 EOB-COL             PIC 9(2) BINARY VALUE 10.
 01 EOB-LINE            PIC 9(2) BINARY VALUE 15.
 SCREEN SECTION.
 01 INVOICE-FORM.
    02 BLANK SCREEN.
    02 "Invoice date:  ".
    02 INVOICE-DATE PIC 99/99/9999 FROM WS-INV-DT
                TO WS-INV-DT.
    02 "Invoice amount:  " LINE.
    02 INVOICE-AMOUNT PIC 9(5).99CR USING WS-INV-AMT.
 01 EMPLOYEE-RECORD.
    02 BLANK SCREEN.
    02 "Employee name:  ".
    02 ER-NAME        PIC X(EMP-NAME-SIZE) USING WS-EMP-NAME.
    02 "Employee loc:  " LINE.
    02 ER-LOC         PIC X(EMP-LOC-SIZE) USING WS-EMP-LOC.
 01 EOB-SCREEN.
    02 ERASE.
    02 "Explanation of Benefits Screen".
    02 "Benefit amount: " LINE + 2 COL 10.
    02 EOB-AMOUNT     PIC 9(5).99DB USING WS-INV-AMT.
```

```
PROCEDURE DIVISION.
A.

    DISPLAY INVOICE-FORM LINE 10 COLUMN 5.
    ACCEPT INVOICE-FORM LINE 10 COLUMN 5.

    DISPLAY EMPLOYEE-RECORD AT LINE 9.
    ACCEPT EMPLOYEE-RECORD AT LINE 9.

    DISPLAY EOB-SCREEN AT COL EOB-COL LINE EOB-LINE.
    ACCEPT EOB-SCREEN AT COL EOB-COL LINE EOB-LINE.

END PROGRAM DISPLAY3.
```

DIVIDE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  DIVIDE01.
*
*  Examples for RM/COBOL Language Reference Manual.
*    DIVIDE statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 TOTAL-WORK-LOAD     PIC 9(08)V99.
 01 AVERAGE-WORK-LOAD   PIC 9(08)V99.
 01 DIVIDEND-1          PIC S9(08)V99.
 01 DIVISOR-1           PIC S9(08)V99.
 01 QUOTIENT-1          PIC S9(08)V99.
 01 REMAINDER-1         PIC S9(08)V99.
 01 SIZE-ERROR-FLAG     PIC X VALUE SPACE.
 PROCEDURE DIVISION.
 A.
     DIVIDE 10 INTO TOTAL-WORK-LOAD. *>  10 FTEs

     DIVIDE 6 INTO TOTAL-WORK-LOAD   *>   6 FTEs
       GIVING AVERAGE-WORK-LOAD.

     DIVIDE TOTAL-WORK-LOAD BY 2.5   *> 2.5 FTEs
       GIVING AVERAGE-WORK-LOAD
     ON SIZE ERROR PERFORM OVERFLOW-ROUTINE
     END-DIVIDE.

     DIVIDE DIVISOR-1 INTO DIVIDEND-1
       GIVING QUOTIENT-1 ROUNDED
       REMAINDER REMAINDER-1.

     DIVIDE DIVIDEND-1 BY DIVISOR-1
       GIVING QUOTIENT-1
       REMAINDER REMAINDER-1
     ON SIZE ERROR  MOVE "E" TO SIZE-ERROR-FLAG
     END-DIVIDE.

 OVERFLOW-ROUTINE.
     EXIT.
```

```
 END PROGRAM DIVIDE01.
```

ENABLE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  ENABLE1.
*
*  Examples for RM/COBOL Language Reference Manual.
*  ENABLE statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 COM-PASSWORD          PIC X(30).
 COMMUNICATION SECTION.
 CD COM-PORT FOR INPUT
     SYMBOLIC QUEUE IS COM-PORT-SYMQ
     SYMBOLIC SUB-QUEUE-1 IS COM-PORT-SYM-SUBQ1
     SYMBOLIC SUB-QUEUE-2 IS COM-PORT-SYM-SUBQ2
     SYMBOLIC SUB-QUEUE-3 IS COM-PORT-SYM-SUBQ3
     MESSAGE DATE IS COM-PORT-MSG-DT
     MESSAGE TIME IS COM-PORT-MSG-TM
     SYMBOLIC SOURCE IS COM-PORT-SYM-SRC
     TEXT LENGTH IS COM-PORT-TXT-LENGTH
     END KEY IS COM-PORT-END-KEY
     STATUS KEY IS COM-PORT-STATUS-KEY
     MESSAGE COUNT IS COM-PORT-MSG-COUNT.

 CD COM-LINE-1 FOR OUTPUT
     DESTINATION COUNT IS L1-DEST-COUNT
     TEXT LENGTH IS L1-TEXT-LENGTH
     STATUS KEY IS L1-STATUS-KEY
     DESTINATION TABLE OCCURS 5 TIMES
       INDEXED BY L1IX1, L1IX2
     ERROR KEY IS L1-ERROR-KEY
     SYMBOLIC DESTINATION IS L1-SYM-DEST.

 PROCEDURE DIVISION.
 0010.

     ENABLE INPUT TERMINAL COM-PORT.

     ENABLE OUTPUT COM-LINE-1 WITH KEY COM-PASSWORD.

 END PROGRAM ENABLE1.
```

ENTER Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  ENTER01.
*
*  Examples for RM/COBOL Language Reference Manual.
```

```
*  ENTER statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 ARGUMENT-GROUP.
    02 ARG1               PIC X(10).
    02 ARG2               PIC X(05).
 PROCEDURE DIVISION.
 0010.

     ENTER LINKAGE.
     CALL "SUBROUTINE" USING ARGUMENT-GROUP.
     ENTER COBOL.

     ENTER FORTRAN SUBROUTINE-1.

 END PROGRAM ENTER01.
```

EVALUATE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  EVALUAT1.
*
*  Examples for RM/COBOL Language Reference Manual.
*  EVALUATE statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 OPERATION-TYPE        PIC X.
 01 TYPE-UPDATE           PIC X VALUE "U".
 01 TYPE-DELETE           PIC X VALUE "D".
 01 TYPE-INSERT           PIC X VALUE "I".
 01 DAY-VALUE             PIC 9.
 01 LEVEL-VALUE           PIC X(8).
    88 L-DETAILED         VALUE "DETAILED".
    88 L-SUMMARY          VALUE "SUMMARY".
 01 UPDATE-TYPE           PIC X.
    88 ANNUALLY           VALUE "A".
    88 QUARTERLY          VALUE "Q".
    88 MONTHLY            VALUE "M".
 01 YEAR-END-FLAG         PIC X.
    88 YEAR-END           VALUE "T" FALSE "F".
 01 QUARTER-END-FLAG       PIC X.
    88 QUARTER-END         VALUE "T" FALSE "F".
 01 MONTH-END-FLAG        PIC X.
    88 MONTH-END          VALUE "T" FALSE "F".
 PROCEDURE DIVISION.
 0010.
     EVALUATE OPERATION-TYPE
```

```
           WHEN TYPE-UPDATE PERFORM UPDATE-IT
           WHEN TYPE-DELETE PERFORM DELETE-IT
           WHEN TYPE-INSERT PERFORM INSERT-IT
           WHEN OTHER PERFORM BAD-OPERATION-TYPE
           END-EVALUATE.

           EVALUATE DAY-VALUE ALSO LEVEL-VALUE
           WHEN 1 ALSO ANY        PERFORM MONDAY-PROCESSING
           WHEN 2 THRU 4 ALSO "SUMMARY"
             PERFORM MIDWEEK-PROCESSING
           WHEN 2 ALSO "DETAILED" PERFORM TUESDAY-PROCESSING
           WHEN 3 ALSO "DETAILED" PERFORM WEDNESDAY-PROCESSING
           WHEN 4 ALSO "DETAILED" PERFORM THURSDAY-PROCESSING
           WHEN 5 ALSO ANY        PERFORM FRIDAY-PROCESSING
           WHEN 6 ALSO ANY
           WHEN 7 ALSO ANY        PERFORM WEEKEND-PROCESSING
           WHEN OTHER             PERFORM BAD-DAY-OR-LEVEL
           END-EVALUATE.

           EVALUATE TRUE
           WHEN ANNUALLY AND YEAR-END
             PERFORM ANNUAL-UPDATE
           WHEN QUARTERLY AND QUARTER-END
             PERFORM QUARTER-UPDATE
           WHEN MONTHLY AND MONTH-END
             PERFORM MONTH-UPDATE
           END-EVALUATE.

       UPDATE-IT.
       DELETE-IT.
       INSERT-IT.
       BAD-OPERATION-TYPE.

       MIDWEEK-PROCESSING.
       MONDAY-PROCESSING.
       TUESDAY-PROCESSING.
       WEDNESDAY-PROCESSING.
       THURSDAY-PROCESSING.
       FRIDAY-PROCESSING.
       WEEKEND-PROCESSING.
       BAD-DAY-OR-LEVEL.

       ANNUAL-UPDATE.
       QUARTER-UPDATE.
       MONTH-UPDATE.

       END PROGRAM EVALUAT1.
```

EXIT Statement Example

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  EXIT01.
      *
      *  Examples for RM/COBOL Language Reference Manual.
      *  EXIT statement.
```

```
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 RECORD-TYPE           PIC X(4).
 01 MY-RECORD-TYPE        PIC X(4) VALUE "TRAN".
 01 EXIT-LOOP-FLAG        PIC X.
 01 EXIT-CYCLE-FLAG       PIC X.
 PROCEDURE DIVISION.
 PRIMARY SECTION.
 0010.
     PERFORM WEEKEND-PROC THRU WEEKEND-PROC-EXIT.

 WEEKEND-PROC.

 WEEKEND-PROC-CONT.

 WEEKEND-PROC-EXIT.
     EXIT.

 0020.
     IF RECORD-TYPE NOT = MY-RECORD-TYPE
     THEN
         MOVE 4096 TO RETURN-CODE
         EXIT PROGRAM
     END-IF.

     IF RECORD-TYPE = MY-RECORD-TYPE
         EXIT PARAGRAPH
     END-IF.

     PERFORM UNTIL RECORD-TYPE = MY-RECORD-TYPE
         PERFORM WEEKEND-PROC THRU WEEKEND-PROC-EXIT
         IF EXIT-LOOP-FLAG = "Y"
             EXIT PERFORM
         END-IF
         IF EXIT-CYCLE-FLAG = "Y"
             EXIT PERFORM CYCLE
         END-IF
         PERFORM 0010
         *> CONTINUE from EXIT PERFORM CYCLE statement
     END-PERFORM.
     *> CONTINUE from EXIT PERFORM statement

 0030.
     IF RECORD-TYPE = MY-RECORD-TYPE
         EXIT SECTION
     END-IF.

 END PROGRAM EXIT01.
```

GOBACK Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  GOBACK01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  GOBACK statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 RECORD-TYPE           PIC X(4).
 01 MY-RECORD-TYPE        PIC X(4) VALUE "TRAN".
 PROCEDURE DIVISION.
 0010.
     GOBACK.
 0020.
     IF RECORD-TYPE NOT = MY-RECORD-TYPE
     THEN
       MOVE 4096 TO RETURN-CODE
       GOBACK
     END-IF.

 END PROGRAM GOBACK01.
```

GO TO Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  GOTO01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  GOBACK statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 STATE-1-FLAG          PIC X(1).
    88 STATE-1-UP         VALUE "U".
    88 STATE-1-DOWN       VALUE "D".
 01 USER-PICK             PIC 9.
 PROCEDURE DIVISION.
 0010.
     IF STATE-1-UP
       ALTER STATE-1-SWITCH TO STATE-1-UP-PROC
     ELSE
       ALTER STATE-1-SWITCH TO STATE-1-DOWN-PROC.

 STATE-1-SWITCH.
     GO TO.
```

```
STATE-1-UP-PROC.

STATE-1-DOWN-PROC.

0020.
    GO TO STATE-1-EXIT-PROC.

STATE-1-EXIT-PROC.

0030.
    GO TO CHOICE-1, CHOICE-2, CHOICE-3
      DEPENDING ON USER-PICK.

CHOICE-1.
CHOICE-2.
CHOICE-3.

END PROGRAM GOTO01.
```

IF Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  IF01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  IF statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     SWITCH-1 IS PRINT-SWITCH
       ON STATUS IS PRINT-SWITCH-ON.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 CHAR-STR              PIC X(10).
 01 ALPHA-STR             PIC X(10).
 01 NUM                   PIC 9(10).
 01 OLD-NUM               PIC 9(10).
 01 ERROR-CNT             PIC 9(5) BINARY.
 01 UPPER-LIMIT           PIC 9(10) VALUE 4000000000.
 PROCEDURE DIVISION.
 0010.
     IF CHAR-STR IS ALPHABETIC
     THEN MOVE CHAR-STR TO ALPHA-STR;
     ELSE IF CHAR-STR IS NUMERIC
     THEN MOVE CHAR-STR TO NUM;
     ELSE NEXT SENTENCE.

 0020.
     IF NUM = OLD-NUM GO TO RE-SET.

 0030.
     IF ALPHA-STR NOT = "TEST"
       ADD 1 TO ERROR-CNT
```

```
      IF ERROR-CNT >= 20
        DISPLAY "Excessive errors."
        STOP RUN
      END-IF
    ELSE
      PERFORM TEST-PROCEDURE
    END-IF.

0040.
    IF NUM < UPPER-LIMIT, ADD 1 TO NUM.

0050.
    IF NUM IS LESS THAN UPPER-LIMIT
    THEN
      ADD 1 TO NUM
    ELSE
      PERFORM RE-SET
    END-IF.

0060.
    IF PRINT-SWITCH-ON PERFORM PRINT-ROUTINE.

RE-SET.
TEST-PROCEDURE.
PRINT-ROUTINE.

END PROGRAM IF01.
```

INITIALIZE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  INITLZ01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  INITIALIZE statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 EMPLOYEE-RECORD.
    02 EMP-NAME           PIC X(30).
    02 EMP-SALARY         PIC S9(8)V99.
    02 EMP-DEPARTMENT     PIC X(20) VALUE "CORPORATE".
    02 FILLER             PIC A(20).
 01 HR-RECORD.
    02 HR-DEPARTMENT      PIC X(20).
    02 HR-GROUP           PIC X(20).
    02 HR-SALARY-TOTAL    PIC S9(10)V99.

 PROCEDURE DIVISION.
 0010.
     INITIALIZE EMPLOYEE-RECORD HR-RECORD.
```

```
    INITIALIZE EMPLOYEE-RECORD
      REPLACING NUMERIC DATA BY ZERO
                ALPHANUMERIC DATA BY ALL "#".

    INITIALIZE HR-RECORD
      REPLACING NUMERIC DATA BY 100.00.

    INITIALIZE EMPLOYEE-RECORD HR-RECORD
      WITH FILLER
      ALL TO VALUE
      THEN REPLACING
        ALPHANUMERIC ALPHABETIC DATA BY ALL "#"
      THEN TO DEFAULT.

END PROGRAM INITLZ01.
```

INSPECT Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  INSPECT1.
*
*  Examples for RM/COBOL Language Reference Manual.
*  INSPECT statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 WORD-1                  PIC X(9).
 01 COUNT-1                 PIC 9(4).
 01 COUNT-2                 PIC 9(4).
 PROCEDURE DIVISION.
 0010.
     MOVE "LARGE" TO WORD-1.
     PERFORM EXAMPLE1.
     IF COUNT-1 = 1 AND COUNT-2 = 0
       DISPLAY "Example 1a passed."
     ELSE
       DISPLAY "Example 1a failed."
     END-IF.

     MOVE "ANALYST" TO WORD-1.
     PERFORM EXAMPLE1.
     IF COUNT-1 = 0 AND COUNT-2 = 1
       DISPLAY "Example 1b passed."
     ELSE
       DISPLAY "Example 1b failed."
     END-IF.

 0020.
     MOVE "CALLAR" TO WORD-1.
     PERFORM EXAMPLE2.
     IF COUNT-1 = 2 AND WORD-1 = "CALLER"
       DISPLAY "Example 2a passed."
```

```
    ELSE
      DISPLAY "Example 2a failed."
    END-IF.

    MOVE "SALAMI" TO WORD-1.
    PERFORM EXAMPLE2.
    IF COUNT-1 = 1 AND WORD-1 = "SALEMI"
      DISPLAY "Example 2b passed."
    ELSE
      DISPLAY "Example 2b failed."
    END-IF.

    MOVE "LATTER" TO WORD-1.
    PERFORM EXAMPLE2.
    IF COUNT-1 = 1 AND WORD-1 = "LETTER"
      DISPLAY "Example 2c passed."
    ELSE
      DISPLAY "Example 2c failed."
    END-IF.

0030.
    MOVE "ARXAX" TO WORD-1.
    PERFORM EXAMPLE3.
    IF WORD-1 = "GRXAX"
      DISPLAY "Example 3a passed."
    ELSE
      DISPLAY "Example 3a failed."
    END-IF.

    MOVE "HANDAX" TO WORD-1.
    PERFORM EXAMPLE3.
    IF WORD-1 = "HGNDGX"
      DISPLAY "Example 3b passed."
    ELSE
      DISPLAY "Example 3b failed."
    END-IF.

0040.
    MOVE "ADJECTIVE" TO WORD-1.
    PERFORM EXAMPLE4.
    IF COUNT-1 = 6 AND WORD-1 = "BDJECTIVE"
      DISPLAY "Example 4a passed."
    ELSE
      DISPLAY "Example 4a failed."
    END-IF.

    MOVE "JACK" TO WORD-1.
    PERFORM EXAMPLE4.
    IF COUNT-1 = 3 AND WORD-1 = "JBCK"
      DISPLAY "Example 4b passed."
    ELSE
      DISPLAY "Example 4b failed."
    END-IF.

    MOVE "JUJMAB" TO WORD-1.
    PERFORM EXAMPLE4.
    IF COUNT-1 = 5 AND WORD-1 = "JUJMBB"
```

```
          DISPLAY "Example 4c passed."
        ELSE
          DISPLAY "Example 4c failed."
        END-IF.

    0050.
        MOVE "RXXBQWY" TO WORD-1.
        PERFORM EXAMPLE5.
        IF WORD-1 = "RYYZQQY"
          DISPLAY "Example 5a passed."
        ELSE
          DISPLAY "Example 5a failed."
        END-IF.

        MOVE "YZACDWBR" TO WORD-1.
        PERFORM EXAMPLE5.
        IF WORD-1 = "YZACDWZR"
          DISPLAY "Example 5b passed."
        ELSE
          DISPLAY "Example 5b failed."
        END-IF.

        MOVE "RAWRXEB" TO WORD-1.
        PERFORM EXAMPLE5.
        IF WORD-1 = "RAQRYEZ"
          DISPLAY "Example 5c passed."
        ELSE
          DISPLAY "Example 5c failed."
        END-IF.

    0060.
        MOVE "12 XZABCD" TO WORD-1.
        PERFORM EXAMPLE6.
        IF WORD-1(1:9) = "BBBBBABCD"
          DISPLAY "Example 6a passed."
        ELSE
          DISPLAY "Example 6a failed."
        END-IF.

        MOVE "123456789" TO WORD-1.
        PERFORM EXAMPLE6.
        IF WORD-1(1:9) = "BBBBBBBBB"
          DISPLAY "Example 6b passed."
        ELSE
          DISPLAY "Example 6b failed."
        END-IF.

        MOVE "A23456789" TO WORD-1.
        PERFORM EXAMPLE6.
        IF WORD-1(1:9) = "A23456789"
          DISPLAY "Example 6c passed."
        ELSE
          DISPLAY "Example 6c failed."
        END-IF.

    0070.
        MOVE "name" TO WORD-1.
```

```
     PERFORM EXAMPLE7.
     IF WORD-1 = "NAME"
       DISPLAY "Example 7a passed."
     ELSE
       DISPLAY "Example 7a failed."
     END-IF.

     MOVE "Day Count" TO WORD-1.
     PERFORM EXAMPLE7.
     IF WORD-1 = "DAY COUNT"
       DISPLAY "Example 7b passed."
     ELSE
       DISPLAY "Example 7b failed."
     END-IF.

 0080.
     MOVE "name" TO WORD-1.
     PERFORM EXAMPLE8.
     IF WORD-1 = "name#####" AND COUNT-1 = 5
       DISPLAY "Example 8a passed."
     ELSE
       DISPLAY "Example 8a failed."
     END-IF.

     MOVE "address" TO WORD-1.
     PERFORM EXAMPLE8.
     IF WORD-1 = "address##" AND COUNT-1 = 2
       DISPLAY "Example 8b passed."
     ELSE
       DISPLAY "Example 8b failed."
     END-IF.

     ACCEPT WORD-1 PROMPT "#" SIZE 1.
     STOP RUN.

 EXAMPLE1.
*>------------------------------------------------------------
     MOVE ZERO TO COUNT-1, COUNT-2.
     INSPECT WORD-1 TALLYING
       COUNT-1 FOR LEADING "L" BEFORE INITIAL "A"
       COUNT-2 FOR LEADING "A" BEFORE INITIAL "L".

*> WORD-1 = "LARGE"   -> COUNT-1 = 1, COUNT-2 = 0
*> WORD-1 = "ANALYST" -> COUNT-1 = 0, COUNT-2 = 1
*>------------------------------------------------------------

 EXAMPLE2.
     MOVE ZERO TO COUNT-1.
     INSPECT WORD-1 TALLYING
       COUNT-1 FOR ALL "L" REPLACING
       ALL "A" BY "E" AFTER INITIAL "L".

*> WORD-1 = "CALLAR" -> COUNT-1 = 2, WORD-1 = "CALLER"
*> WORD-1 = "SALAMI" -> COUNT-1 = 1, WORD-1 = "SALEMI"
*> WORD-1 = "LATTER" -> COUNT-1 = 1, WORD-1 = "LETTER"
*>------------------------------------------------------------
```

```
 EXAMPLE3.
     INSPECT WORD-1 REPLACING
       ALL "A" BY "G" BEFORE INITIAL "X".

*> WORD-1 = "ARXAX"  -> WORD-1 = "GRXAX"
*> WORD-1 = "HANDAX" -> WORD-1 = "HGNDGX"
*>-------------------------------------------------------

 EXAMPLE4.
     MOVE ZERO TO COUNT-1.
     INSPECT WORD-1 TALLYING
       COUNT-1 FOR CHARACTERS AFTER INITIAL "J"
       REPLACING ALL "A" BY "B".
*>-------------------------------------------------------

*> WORD-1 = "ADJECTIVE" -> COUNT-1 = 6, WORD-1 = "BDJECTIVE"

     MOVE ZERO TO COUNT-2.
     INSPECT WORD-1 TALLYING COUNT-2 FOR ALL SPACE.
     SUBTRACT COUNT-2 FROM COUNT-1.
*>-------------------------------------------------------

 EXAMPLE5.
     INSPECT WORD-1 REPLACING ALL "X" BY "Y",
       "B" BY "Z", "W" BY "Q" AFTER INITIAL "R".

*> WORD-1 = "RXXBQWY"  -> WORD-1 = "RYYZQQY"
*> WORD-1 = "YZACDWBR" -> WORD-1 = "YZACDWZR"
*> WORD-1 = "RAWRXEB"  -> WORD-1 = "RAQRYEZ"
*>-------------------------------------------------------

 EXAMPLE6.
     INSPECT WORD-1 REPLACING CHARACTERS BY "B"
       BEFORE INITIAL "A".

*> WORD-1 = "12 XZABCD" -> WORD-1 = "BBBBBABCD"
*> WORD-1 = "123456789" -> WORD-1 = "BBBBBBBBB"
*> WORD-1 = "A23456789" -> WORD-1 = "A23456789"
*>-------------------------------------------------------

 EXAMPLE7.
     INSPECT WORD-1 CONVERTING
       "abcdefghijklmnopqrstuvwxyz" TO
       "ABCDEFGHIJKLMNOPQRSTUVWXYZ".

*> WORD-1 = "name"      -> WORD-1 = "NAME"
*> WORD-1 = "Day Total" -> WORD-1 = "DAY TOTAL"
*>-------------------------------------------------------

 EXAMPLE8.
     MOVE ZERO TO COUNT-1.
     INSPECT WORD-1 TALLYING COUNT-1 FOR TRAILING SPACES
       REPLACING TRAILING SPACES BY "#".

*> WORD-1 = "name     " -> WORD-1 = "name#####", COUNT-1 = 5
*> WORD-1 = "address  " -> WORD-1 = "address##", COUNT-1 = 2
*>-------------------------------------------------------
```

```
END PROGRAM INSPECT1.
```

MERGE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  MERGE01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  MERGE statement.
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT MERGE-FILE ASSIGN TO SORT-WORK.
     SELECT SORTED-FILE-1 ASSIGN TO DISK.
     SELECT SORTED-FILE-2 ASSIGN TO DISK.

 DATA DIVISION.
 FILE SECTION.
 SD MERGE-FILE.
 01 MERGE-RECORD.
    02 MERGE-KEY-1        PIC X(05).
    02 MERGE-KEY-2        PIC 9(05) BINARY.
    02 MERGE-DATA-1       PIC X(20).
 FD SORTED-FILE-1.
 01 SORTED-FILE-1-RECORD.
    02 SORTED-KEY-1       PIC X(05).
    02 SORTED-KEY-2       PIC 9(05) BINARY.
    02 SORTED-DATA-1      PIC X(20).
 FD SORTED-FILE-2.
 01 SORTED-FILE-2-RECORD.
    02 SORTED-KEY-1       PIC X(05).
    02 SORTED-KEY-2       PIC 9(05) BINARY.
    02 SORTED-DATA-1      PIC X(20).
 WORKING-STORAGE SECTION.
 01 EOF-FLAG              PIC X(01).
    88 EOF                VALUE "T" WHEN FALSE "F".

 PROCEDURE DIVISION.
 MAIN1.
     MERGE MERGE-FILE
       ON ASCENDING KEY MERGE-KEY-1
       ON DESCENDING KEY MERGE-KEY-2
       USING SORTED-FILE-1 SORTED-FILE-2
       OUTPUT PROCEDURE IS PUT-RECORDS.
     STOP RUN.

 PUT-RECORDS.
     SET EOF TO FALSE.
     PERFORM UNTIL EOF
       RETURN MERGE-FILE RECORD
       AT END SET EOF TO TRUE
       NOT AT END CALL "WRITE-RECORD" USING MERGE-RECORD
       END-RETURN
```

```
    END-PERFORM.

END PROGRAM MERGE01.
```

MOVE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  MOVE01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  MOVE statement.
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT POPULATION-FILE  ASSIGN TO DISK.

 DATA DIVISION.
 FILE SECTION.
 FD POPULATION-FILE.
 01 FILE-RECORD.
    02 PERSON             PIC X(30).
 WORKING-STORAGE SECTION.
 01 INCOME                PIC S9(10)v99.
 01 TOTAL-INCOME          PIC S9(10)v99.
 01 PAGE-COUNT            PIC 9(5) BINARY.
 01 LINE-NUM              PIC 9(5) BINARY.
 01 TITLE-HEADER          PIC X(50).
 01 ALABAMA.
    02 I-A                PIC 9(04) BINARY.
    02 PERSON             PIC X(30)
                          OCCURS 1000 TIMES.
 01 CROSS-CENSUS.
    02 PERSON             PIC X(30).
 01 NUM                   PIC S9(5)v9(4).
 01 NUM-ED                PIC $+(6).9(4).
 01 TG.
    02 G1                 OCCURS 5 TIMES INDEXED BY N.
       03 G2              OCCURS 5 TIMES INDEXED BY J.
          04 TABLE-ELT    PIC X(20)
                          OCCURS 5 TIMES INDEXED BY M.
 01 NEXT-ENTRY            PIC X(20).
 01 PREVIOUS-ENTRY        PIC X(20).
 01 DEFICIT               PIC S9(10)v99.
 01 SECTION-DIVIDER       PIC X(80).
 01 COUN-TER              PIC S9(8).
 PROCEDURE DIVISION.
 0010.
     MOVE INCOME TO TOTAL-INCOME.

     MOVE 1 TO PAGE-COUNT, LINE-NUM.

     MOVE "Marmack Industries" to TITLE-HEADER.

     MOVE PERSON IN FILE-RECORD TO
```

```
      PERSON OF ALABAMA (I-A OF ALABAMA),
      PERSON OF CROSS-CENSUS.

    MOVE NUM TO NUM-ED.

    MOVE TABLE-ELT (N, 1, M) TO NEXT-ENTRY
      PREVIOUS-ENTRY.

    MOVE -36.7 TO DEFICIT.

    MOVE QUOTES TO SECTION-DIVIDER.

    MOVE ZERO TO COUN-TER.

    MOVE ZEROES TO COUN-TER, NUM, NUM-ED.

END PROGRAM MOVE01.
```

MULTIPLY Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  MULTPLY1.
*
*  Examples for RM/COBOL Language Reference Manual.
*    MULTIPLY statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 INCOME              PIC 9(08)v99.
 01 PRINCIPAL           PIC S9(10)v99.
 01 INTEREST-RATE       PIC S9(04)v9(04).
 01 INTEREST            PIC S9(08)v9(02).
 01 INFLATION-RATE      PIC S9(04)v9(04).
 01 EXPENSES            PIC S9(10)v9(02).
 01 ECONOMY-RATING      PIC S9(05).
 PROCEDURE DIVISION.
 A.
     MULTIPLY 10 BY INCOME.  *> INCOME := (10 * INCOME)

     MULTIPLY PRINCIPAL BY INTEREST-RATE
       GIVING INTEREST ROUNDED.

     MULTIPLY INFLATION-RATE BY EXPENSES
     ON SIZE ERROR
       MOVE 0 TO ECONOMY-RATING
     END-MULTIPLY.

END PROGRAM MULTPLY1.
```

OPEN Statement Example

```
IDENTIFICATION DIVISION.
```

```
 PROGRAM-ID.  OPEN01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  OPEN statement.
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT TRANSACTION-FILE  ASSIGN TO TAPE.
     SELECT LOG-FILE          ASSIGN TO DISK
                              FILE STATUS IS LOG-FILE-STATUS.
     SELECT INPUT-FILE        ASSIGN TO TAPE.
     SELECT TAPE-FILE-1       ASSIGN TO TAPE.
     SELECT PRINT-FILE        ASSIGN TO PRINTER.
     SELECT DATA-BASE         ASSIGN TO DISK
                              INDEXED ACCESS DYNAMIC
                              RECORD KEY IS DB-KEY
                              FILE STATUS IS DB-STATUS.

 DATA DIVISION.
 FILE SECTION.
 FD TRANSACTION-FILE.
 01 TR-RECORD             PIC X(80).

 FD LOG-FILE.
 01 LOG-RECORD            PIC X(80).

 FD INPUT-FILE.
 01 IN-RECORD             PIC X(80).

 FD TAPE-FILE-1.
 01 TF1-RECORD            PIC X(512).

 FD PRINT-FILE.
 01 PF-RECORD             PIC X(60).

 FD DATA-BASE.
 01 DB-RECORD.
    02 DB-DATA-1          PIC X(10).
    02 DB-KEY             PIC X(20).
    02 DB-DATA-2          PIC X(50).

 WORKING-STORAGE SECTION.
 01 LOG-FILE-STATUS       PIC X(02).
 01 DB-STATUS             PIC X(02).
 PROCEDURE DIVISION.
 DECLARATIVES.
 I-O-ERROR SECTION.
     USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
 I-O-ERROR1.
     EXIT.
 END DECLARATIVES.
 MAIN-01 SECTION.
 0010.
     OPEN EXCLUSIVE INPUT TRANSACTION-FILE.

     OPEN EXCLUSIVE OUTPUT LOG-FILE WITH NO REWIND.
```

```
    OPEN I-O LOG-FILE.

    OPEN EXTEND INPUT-FILE.

    OPEN INPUT TAPE-FILE-1 REVERSED.

    OPEN I-O DATA-BASE WITH LOCK.

    OPEN INPUT DATA-BASE.

END PROGRAM OPEN01.
```

PERFORM Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  PERFORM1.
*
*  Examples for RM/COBOL Language Reference Manual.
*  PERFORM statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT INPUT-FILE ASSIGN TO DISK.

 DATA DIVISION.
 FILE SECTION.
 FD INPUT-FILE.
 01 INPUT-RECORD          PIC X(80).
 WORKING-STORAGE SECTION.
 01 ITEM-COUNT            PIC S9(5) BINARY.
 01 RECORD-COUNT          PIC S9(5) BINARY.
 01 EOF-FLAG              PIC X.
    88 EOF                VALUE "T" FALSE "F".
 01 G1.
    02 T1                 OCCURS 100 TIMES
                            INDEXED BY T1-IX.
       03 E1-FIELD        PIC X(5).
       03 E1-LINE         PIC 9(02) BINARY.
       03 E1-COL          PIC 9(02) BINARY.
 01 COUNT-1               PIC 9(04) BINARY.
 01 G2.
    02 T2                 OCCURS 5 TIMES
                            INDEXED BY IX1.
       03 T3              OCCURS 10 TIMES
                            INDEXED BY IX2.
          04 E2           PIC X.
 PROCEDURE DIVISION.
 0010.
     PERFORM INTIALIZATION-PROCEDURE.

     PERFORM GROUP1 THROUGH GROUP5.
```

```
           PERFORM
             DISPLAY "Ending run unit now"
             STOP RUN
           END-PERFORM.

       0020.
           PERFORM STEP-UP COUNT-1 TIMES.

           PERFORM 4 TIMES
             ADD ITEM-COUNT TO ITEM-COUNT
           END-PERFORM.

       0030.
           SET EOF TO FALSE.
           PERFORM UNTIL EOF
             READ INPUT-FILE
             AT END SET EOF TO TRUE
             NOT AT END ADD 1 TO RECORD-COUNT
             END-READ
           END-PERFORM.

           PERFORM ITEM-PROCEDURE
             WITH TEST AFTER UNTIL ITEM-COUNT = 0.

       0040.
           PERFORM VARYING T1-IX FROM 1 BY 1
                             UNTIL T1-IX > 100
             DISPLAY E1-FIELD(T1-IX)
               LINE  E1-LINE(T1-IX)
               COL   E1-COL(T1-IX)
           END-PERFORM.

           PERFORM TABLE-INITIALIZE
             VARYING IX1 FROM 1 BY 1 UNTIL IX1 > 5
             AFTER   IX2 FROM 1 BY 1 UNTIL IX2 > 10.

       INTIALIZATION-PROCEDURE.
       GROUP1.
       GROUP2.
       GROUP3.
       GROUP4.
       GROUP5.
       ITEM-PROCEDURE.
       STEP-UP.
       TABLE-INITIALIZE.

       END PROGRAM PERFORM1.
```

PURGE Statement Example

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  PURGE1.
      *
      *  Examples for RM/COBOL Language Reference Manual.
      *  PURGE statement.
```

```
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 COMMUNICATION SECTION.
 CD COM-LINE-1 FOR OUTPUT
     DESTINATION COUNT IS L1-DEST-COUNT
     TEXT LENGTH IS L1-TEXT-LENGTH
     STATUS KEY IS L1-STATUS-KEY
     DESTINATION TABLE OCCURS 5 TIMES
       INDEXED BY L1IX1, L1IX2
     ERROR KEY IS L1-ERROR-KEY
     SYMBOLIC DESTINATION IS L1-SYM-DEST.

 CD COM-LINE-2 FOR I-O
     SYMBOLIC TERMINAL IS COM-L2-TERMINAL-NAME
     MESSAGE DATE IS COM-L2-MSG-DT
     MESSAGE TIME IS COM-L2-MSG-TM
     TEXT LENGTH IS COM-L2-TXT-LENGTH
     END KEY IS COM-L2-END-KEY
     STATUS KEY IS COM-L2-STATUS-KEY.

 PROCEDURE DIVISION.
 0010.

     PURGE COM-LINE-1.

     PURGE COM-LINE-2.

 END PROGRAM PURGE1.
```

READ Statement Examples

READ Format 1

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  READ01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  READ statement (sequential access).
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT TRANSACTION-FILE  ASSIGN TO TAPE.
     SELECT LOG-FILE          ASSIGN TO DISK
                              FILE STATUS IS LOG-FILE-STATUS.
     SELECT INPUT-FILE        ASSIGN TO TAPE.
     SELECT TAPE-FILE-1       ASSIGN TO TAPE.
     SELECT INVENTORY-FILE    ASSIGN TO DISK
                              RELATIVE ACCESS DYNAMIC
                                RELATIVE KEY IS
                                  INVENTORY-KEY.
     SELECT DATA-BASE         ASSIGN TO DISK
```

```
                               INDEXED ACCESS DYNAMIC
                               RECORD KEY IS DB-KEY
                               FILE STATUS IS DB-STATUS.
DATA DIVISION.
FILE SECTION.
FD TRANSACTION-FILE.
01 TR-RECORD             PIC X(80).

FD LOG-FILE.
01 LOG-RECORD            PIC X(80).

FD INPUT-FILE.
01 IN-RECORD             PIC X(80).

FD TAPE-FILE-1.
01 TF1-RECORD            PIC X(512).

FD INVENTORY-FILE.
01 INVENTORY-RECORD      PIC X(80).

FD DATA-BASE.
01 DB-RECORD.
   02 DB-DATA-1          PIC X(10).
   02 DB-KEY             PIC X(20).
   02 DB-DATA-2          PIC X(50).

WORKING-STORAGE SECTION.
01 LOG-FILE-STATUS       PIC X(02).
01 DB-STATUS             PIC X(02).
01 INVENTORY-KEY         PIC 9(05) BINARY.
01 RECORD-SAVE           PIC X(80).
01 EOF-FLAG              PIC X.
   88 EOF                VALUE "T" FALSE "F".
PROCEDURE DIVISION.
DECLARATIVES.
I-O-ERROR SECTION.
    USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
I-O-ERROR1.
    EXIT.
END DECLARATIVES.
MAIN-01 SECTION.
0010.
    READ TRANSACTION-FILE RECORD.

    READ LOG-FILE NEXT RECORD INTO RECORD-SAVE
    AT END SET EOF TO TRUE
    NOT AT END PERFORM PROCESS-LOG-RECORD
    END-READ.

    READ INVENTORY-FILE PREVIOUS RECORD WITH LOCK
    AT END DISPLAY "Beginning-of-file reached."
    END-READ.

    READ DATA-BASE NEXT RECORD WITH NO LOCK
    AT END PERFORM EOF-PROCEDURE.
```

```
PROCESS-LOG-RECORD.
EOF-PROCEDURE.

END PROGRAM READ01.
```

READ Format 2

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  READ02.
*
*  Examples for RM/COBOL Language Reference Manual.
*  READ statement (random access).
*
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INVENTORY-FILE    ASSIGN TO DISK
                             RELATIVE ACCESS RANDOM
                               RELATIVE KEY IS
                                 INVENTORY-KEY.
    SELECT DATA-BASE         ASSIGN TO DISK
                             INDEXED ACCESS DYNAMIC
                             RECORD KEY IS DB-KEY
                             FILE STATUS IS DB-STATUS.
DATA DIVISION.
FILE SECTION.
FD INVENTORY-FILE.
01 INVENTORY-RECORD      PIC X(80).

FD DATA-BASE.
01 DB-RECORD.
   02 DB-DATA-1          PIC X(10).
   02 DB-KEY             PIC X(20).
   02 DB-DATA-2          PIC X(50).

WORKING-STORAGE SECTION.
01 INVENTORY-KEY         PIC 9(05) BINARY.
01 DB-STATUS             PIC X(02).
01 RECORD-WORK-AREA      PIC X(80).
PROCEDURE DIVISION.
DECLARATIVES.
I-O-ERROR SECTION.
    USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
I-O-ERROR1.
    EXIT.
END DECLARATIVES.
MAIN-01 SECTION.
0010.
    READ INVENTORY-FILE RECORD
    INVALID KEY PERFORM BAD-KEY-PROCEDURE
    END-READ.

    READ DATA-BASE WITH NO LOCK INTO RECORD-WORK-AREA
    INVALID KEY DISPLAY "Bad key"
    NOT INVALID KEY PERFORM PROCESS-WORK-AREA
    END-READ.
```

```
BAD-KEY-PROCEDURE.
PROCESS-WORK-AREA.

END PROGRAM READ02.
```

RECEIVE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  RECEIVE1.
*
*  Examples for RM/COBOL Language Reference Manual.
*  RECEIVE statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 MESSAGE-BUFFER          PIC X(1000).
 01 SEGMENT-BUFFER          PIC X(500).
 01 DEFAULT-SEGMENT         PIC X(500).
 COMMUNICATION SECTION.
 CD COM-PORT FOR INPUT
     SYMBOLIC QUEUE IS COM-PORT-SYMQ
     SYMBOLIC SUB-QUEUE-1 IS COM-PORT-SYM-SUBQ1
     SYMBOLIC SUB-QUEUE-2 IS COM-PORT-SYM-SUBQ2
     SYMBOLIC SUB-QUEUE-3 IS COM-PORT-SYM-SUBQ3
     MESSAGE DATE IS COM-PORT-MSG-DT
     MESSAGE TIME IS COM-PORT-MSG-TM
     SYMBOLIC SOURCE IS COM-PORT-SYM-SRC
     TEXT LENGTH IS COM-PORT-TXT-LENGTH
     END KEY IS COM-PORT-END-KEY
     STATUS KEY IS COM-PORT-STATUS-KEY
     MESSAGE COUNT IS COM-PORT-MSG-COUNT.

 CD COM-LINE-2 FOR I-O
     SYMBOLIC TERMINAL IS COM-L2-TERMINAL-NAME
     MESSAGE DATE IS COM-L2-MSG-DT
     MESSAGE TIME IS COM-L2-MSG-TM
     TEXT LENGTH IS COM-L2-TXT-LENGTH
     END KEY IS COM-L2-END-KEY
     STATUS KEY IS COM-L2-STATUS-KEY.

 PROCEDURE DIVISION.
 0010.

     RECEIVE COM-PORT MESSAGE INTO MESSAGE-BUFFER
     NO DATA PERFORM NO-MESSAGE-PROCEDURE
     WITH DATA PERFORM PROCESS-MESSAGE-PROCEDURE
     END-RECEIVE.

     RECEIVE COM-LINE-2 SEGMENT INTO SEGMENT-BUFFER
     NO DATA MOVE
       DEFAULT-SEGMENT TO SEGMENT-BUFFER
     END-RECEIVE.

 NO-MESSAGE-PROCEDURE.
```

```
 PROCESS-MESSAGE-PROCEDURE.

 END PROGRAM RECEIVE1.
```

RELEASE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  RELEASE1.
*
*  Examples for RM/COBOL Language Reference Manual.
*  RELEASE statement.
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT SORT-FILE ASSIGN TO SORT-WORK.
     SELECT SORTED-FILE-1 ASSIGN TO DISK.
     SELECT INPUT-FILE ASSIGN TO DISK.

 DATA DIVISION.
 FILE SECTION.
 SD SORT-FILE.
 01 SORT-RECORD.
    02 SORT-KEY-1         PIC X(05).
    02 SORT-KEY-2         PIC 9(05) BINARY.
    02 SORT-DATA-1        PIC X(20).
 FD SORTED-FILE-1.
 01 SORTED-FILE-1-RECORD.
    02 SORTED-KEY-1       PIC X(05).
    02 SORTED-KEY-2       PIC 9(05) BINARY.
    02 SORTED-DATA-1      PIC X(20).
 FD INPUT-FILE.
 01 INPUT-RECORD.
    02 INPUT-KEY-1        PIC X(05).
    02 INPUT-KEY-2        PIC 9(05) BINARY.
    02 INPUT-DATA-1       PIC X(20).
 WORKING-STORAGE SECTION.
 01 INPUT-EOF-FLAG        PIC X.
    88 INPUT-EOF          VALUE "T" FALSE "F".

 PROCEDURE DIVISION.
 MAIN1.
     SORT SORT-FILE
       ON ASCENDING KEY SORT-KEY-1
       ON DESCENDING KEY SORT-KEY-2
       INPUT PROCEDURE IS SORT-INPUT-PROCEDURE
       GIVING SORTED-FILE-1.
     STOP RUN.

 SORT-INPUT-PROCEDURE.
     SET INPUT-EOF TO FALSE.
     OPEN INPUT INPUT-FILE.
     PERFORM UNTIL INPUT-EOF
       READ INPUT-FILE AT END
         SET INPUT-EOF TO TRUE
```

```
      NOT AT END
        RELEASE SORT-RECORD FROM INPUT-RECORD
      END-READ
    END-PERFORM.
    CLOSE INPUT-FILE.
END PROGRAM RELEASE1.
```

RETURN Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  RETURN01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  RETURN statement.
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT SORT-FILE ASSIGN TO SORT-WORK.
     SELECT OUTPUT-FILE ASSIGN TO DISK.
     SELECT INPUT-FILE ASSIGN TO DISK.

 DATA DIVISION.
 FILE SECTION.
 SD SORT-FILE.
 01 SORT-RECORD.
    02 SORT-KEY-1          PIC X(05).
    02 SORT-KEY-2          PIC 9(05) BINARY.
    02 SORT-DATA-1         PIC X(20).
 FD OUTPUT-FILE.
 01 OUTPUT-RECORD.
    02 OUTPUT-KEY-1        PIC X(05).
    02 OUTPUT-KEY-2        PIC 9(05) BINARY.
    02 OUTPUT-DATA-1       PIC X(20).
 FD INPUT-FILE.
 01 INPUT-RECORD.
    02 INPUT-KEY-1         PIC X(05).
    02 INPUT-KEY-2         PIC 9(05) BINARY.
    02 INPUT-DATA-1        PIC X(20).
 WORKING-STORAGE SECTION.
 01 INPUT-EOF-FLAG         PIC X.
    88 INPUT-EOF           VALUE "T" FALSE "F".
 01 SORT-EOF-FLAG          PIC X.
    88 SORT-EOF            VALUE "T" FALSE "F".

 PROCEDURE DIVISION.
 MAIN1.
     SORT SORT-FILE
       ON ASCENDING KEY SORT-KEY-1
       ON DESCENDING KEY SORT-KEY-2
       INPUT PROCEDURE IS SORT-INPUT-PROCEDURE
       OUTPUT PROCEDURE IS SORT-MERGE-OUTPUT-PROCEDURE.
     STOP RUN.

 SORT-MERGE-OUTPUT-PROCEDURE.
```

```
    OPEN OUTPUT OUTPUT-FILE.
    SET SORT-EOF TO FALSE.
    PERFORM UNTIL SORT-EOF
      RETURN SORT-FILE RECORD INTO OUTPUT-RECORD
      AT END SET SORT-EOF TO TRUE
      NOT AT END
        WRITE OUTPUT-RECORD
      END-RETURN
    END-PERFORM.
    CLOSE OUTPUT-FILE.

SORT-INPUT-PROCEDURE.
    SET INPUT-EOF TO FALSE.
    OPEN INPUT INPUT-FILE.
    PERFORM UNTIL INPUT-EOF
      READ INPUT-FILE AT END
        SET INPUT-EOF TO TRUE
      NOT AT END
        RELEASE SORT-RECORD FROM INPUT-RECORD
      END-READ
    END-PERFORM.
    CLOSE INPUT-FILE.

END PROGRAM RETURN01.
```

REWRITE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  REWRITE01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  REWRITE statement.
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT LOG-FILE          ASSIGN TO DISK
                              FILE STATUS IS LOG-FILE-STATUS.

     SELECT INVENTORY-FILE    ASSIGN TO DISK
                              RELATIVE ACCESS RANDOM
                                RELATIVE KEY IS INVENTORY-KEY.

     SELECT DATA-BASE         ASSIGN TO DISK
                              INDEXED ACCESS DYNAMIC
                              RECORD KEY IS DB-KEY
                              FILE STATUS IS DB-STATUS.

 DATA DIVISION.
 FILE SECTION.
 FD LOG-FILE.
 01 LOG-RECORD            PIC X(80).

 FD INVENTORY-FILE.
 01 INVENTORY-RECORD      PIC X(80).
```

```
FD DATA-BASE.
01 DB-RECORD.
   02 DB-DATA-1          PIC X(10).
   02 DB-KEY             PIC X(20).
   02 DB-DATA-2          PIC X(50).

WORKING-STORAGE SECTION.
01 LOG-FILE-STATUS       PIC X(02).
01 INVENTORY-KEY         PIC 9(5) BINARY.
01 DB-STATUS             PIC X(02).
PROCEDURE DIVISION.
DECLARATIVES.
I-O-ERROR SECTION.
    USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
I-O-ERROR1.
    EXIT.
END DECLARATIVES.
MAIN-01 SECTION.
0010.
    REWRITE LOG-RECORD OF LOG-FILE.

    REWRITE LOG-RECORD FROM "END-OF-BATCH"
    END-REWRITE.

    REWRITE INVENTORY-RECORD
    INVALID KEY PERFORM INVALID-KEY-HANDLER
    END-REWRITE.

    REWRITE DB-RECORD OF DATA-BASE
    INVALID KEY
      REWRITE INVENTORY-RECORD END-REWRITE
    END-REWRITE.

INVALID-KEY-HANDLER.

END PROGRAM REWRITE01.
```

SEARCH Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SEARCH01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  SEARCH statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 OBJECT-COMPUTER. RMCOBOL
   PROGRAM COLLATING SEQUENCE IS CASE-INSENSITIVE.
 SPECIAL-NAMES.
     ALPHABET CASE-INSENSITIVE IS 1 THRU 32,
       SPACE ALSO "_", 34 THRU 65,
       "A" ALSO "a", "B" ALSO "b", "C" ALSO "c", "D" ALSO "d",
       "E" ALSO "e", "F" ALSO "f", "G" ALSO "g", "H" ALSO "h",
```

```
      "I" ALSO "i", "J" ALSO "j", "K" ALSO "k", "L" ALSO "l",
      "M" ALSO "m", "N" ALSO "n", "O" ALSO "o", "P" ALSO "p",
      "Q" ALSO "q", "R" ALSO "r", "S" ALSO "s", "T" ALSO "t",
      "U" ALSO "u", "V" ALSO "v", "W" ALSO "w", "X" ALSO "x",
      "Y" ALSO "y", "Z" ALSO "z", 92 THRU 95, 97, 124 THRU 128.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 STATE-GROUP.
   02 STATE-NAME-VALUES.
      03 PIC X(34)   VALUE "AK: Alaska          Juneau        ".
      03 PIC X(34)   VALUE "AL: Alabama         Montgomery    ".
      03 PIC X(34)   VALUE "AR: Arkansas        Little_Rock   ".
      03 PIC X(34)   VALUE "AZ: Arizona         Phoenix       ".
      03 PIC X(34)   VALUE "CA: California      Sacramento    ".
      03 PIC X(34)   VALUE "CN: Connecticut     Hartford      ".
      03 PIC X(34)   VALUE "CO: Colorado        Denver        ".
      03 PIC X(34)   VALUE "DE: Delaware        Dover         ".
      03 PIC X(34)   VALUE "FL: Florida         Tallahassee   ".
      03 PIC X(34)   VALUE "GA: Georgia         Atlanta       ".
      03 PIC X(34)   VALUE "HI: Hawaii          Honolulu      ".
      03 PIC X(34)   VALUE "IA: Iowa            Des_Moines    ".
      03 PIC X(34)   VALUE "ID: Idaho           Boise         ".
      03 PIC X(34)   VALUE "IL: Illinois        Springfield   ".
      03 PIC X(34)   VALUE "IN: Indiana         Indianapolis  ".
      03 PIC X(34)   VALUE "KS: Kansas          Topeka        ".
      03 PIC X(34)   VALUE "KY: Kentucky        Frankfort     ".
      03 PIC X(34)   VALUE "LA: Louisiana       Baton_Rouge   ".
      03 PIC X(34)   VALUE "MA: Massachusetts   Boston        ".
      03 PIC X(34)   VALUE "MD: Maryland        Annapolis     ".
      03 PIC X(34)   VALUE "ME: Maine           Augusta       ".
      03 PIC X(34)   VALUE "MI: Michigan        Lansing       ".
      03 PIC X(34)   VALUE "MN: Minnesota       St._Paul      ".
      03 PIC X(34)   VALUE "MO: Missouri        Jefferson_City".
      03 PIC X(34)   VALUE "MS: Mississippi     Jackson       ".
      03 PIC X(34)   VALUE "MT: Montana         Helena        ".
      03 PIC X(34)   VALUE "NC: North_Carolina  Raleigh       ".
      03 PIC X(34)   VALUE "ND: North_Dakota    Bismarck      ".
      03 PIC X(34)   VALUE "NE: Nebraska        Lincoln       ".
      03 PIC X(34)   VALUE "NH: New_Hampshire   Concord       ".
      03 PIC X(34)   VALUE "NJ: New_Jersey      Trenton       ".
      03 PIC X(34)   VALUE "NM: New_Mexico      Santa_Fe      ".
      03 PIC X(34)   VALUE "NV: Nevada          Carson_City   ".
      03 PIC X(34)   VALUE "NY: New_York        Albany        ".
      03 PIC X(34)   VALUE "OH: Ohio            Columbus      ".
      03 PIC X(34)   VALUE "OK: Oklahoma        Oklahoma_City ".
      03 PIC X(34)   VALUE "OR: Oregon          Salem         ".
      03 PIC X(34)   VALUE "PA: Pennsylvania    Harrisburg    ".
      03 PIC X(34)   VALUE "RI: Rhode_Island    Providence    ".
      03 PIC X(34)   VALUE "SC: South_Carolina  Columbia      ".
      03 PIC X(34)   VALUE "SD: South_Dakota    Pierre        ".
      03 PIC X(34)   VALUE "TN: Tennessee       Nashville     ".
      03 PIC X(34)   VALUE "TX: Texas           Austin        ".
      03 PIC X(34)   VALUE "UT: Utah            Salt_Lake_City".
      03 PIC X(34)   VALUE "VA: Virginia        Richmond      ".
      03 PIC X(34)   VALUE "VT: Vermont         Montpelier    ".
      03 PIC X(34)   VALUE "WA: Washington      Olympia       ".
      03 PIC X(34)   VALUE "WI: Wisconsin       Madison       ".
```

```
           03 PIC X(34)   VALUE "WV: West_Virginia   Charleston    ".
           03 PIC X(34)   VALUE "WY: Wyoming         Cheyenne     ".
       02 STATE-NAME-TABLE   REDEFINES STATE-NAME-VALUES
                            OCCURS 50 TIMES
                            ASCENDING KEY IS STATE-ABBREV
                            INDEXED BY IX1.
           03 STATE-ABBREV    PIC X(02).
           03                 PIC X(02).
           03 STATE-NAME      PIC X(14).
           03                 PIC X(02).
           03 STATE-CAPITAL   PIC X(14).
    01 CURR-ABBREV           PIC X(02).
    01 PREV-ABBREV           PIC X(02).
    01 INPUT-NAME            PIC X(14).
    01 CAPITAL-BUFFER        PIC X(20).
    01 STATE-BUFFER          PIC X(14).
    01 CAPITAL-COUNT         PIC 9(04) BINARY.
    01 STATE-COUNT           PIC 9(04) BINARY.
    01 DUMMY                 PIC X.
    PROCEDURE DIVISION.
    0010.
   * Verify OCCURS key in ascending order as required for SEARCH ALL.
        MOVE SPACES TO PREV-ABBREV.
        PERFORM VARYING IX1 FROM 1 BY 1 UNTIL IX1 > 50
          MOVE STATE-ABBREV(IX1) TO CURR-ABBREV
          IF CURR-ABBREV > PREV-ABBREV
            MOVE CURR-ABBREV TO PREV-ABBREV
          ELSE
            DISPLAY "State abbreviation out of order:  "
              CURR-ABBREV " < " PREV-ABBREV
            ACCEPT DUMMY PROMPT "#"
            STOP RUN
          END-IF
        END-PERFORM.

    0020.
   * Use serial search on unsorted STATE-NAME or STATE-CAPITAL
   *   and also on sorted STATE-ABBREV.
        ACCEPT INPUT-NAME TAB PROMPT.
        SET IX1 TO 1.
        SEARCH STATE-NAME-TABLE VARYING IX1
        AT END
          DISPLAY "The name """ INPUT-NAME
            """ is not in the state name table."
        WHEN STATE-NAME(IX1) = INPUT-NAME
          PERFORM SETUP-BUFFERS   *> Note: uses current IX1 setting.
          DISPLAY "The abbreviation for the state of """
            STATE-BUFFER(1:STATE-COUNT)
            """ is """ STATE-ABBREV(IX1) ""","
            "and the state capital is """ COL 5
            CAPITAL-BUFFER
        WHEN STATE-CAPITAL(IX1) = INPUT-NAME
          PERFORM SETUP-BUFFERS   *> Note: uses current IX1 setting.
          DISPLAY
            "The city """ CAPITAL-BUFFER(1:CAPITAL-COUNT)
            " is the state capital of """
            STATE-BUFFER(1:STATE-COUNT) """."
```

```
           WHEN STATE-ABBREV(IX1) = INPUT-NAME
             PERFORM SETUP-BUFFERS   *> Note: uses current IX1 setting.
             DISPLAY "The abbreviation """ STATE-ABBREV(IX1)
             """ stands for the state of """
             STATE-BUFFER(1:STATE-COUNT) """,","
             " and the state capital is """ COL 5 CAPITAL-BUFFER
           END-SEARCH.

       0030.
      * Use binary search on sorted STATE-ABBREV.
           ACCEPT CURR-ABBREV TAB PROMPT.
           SEARCH ALL STATE-NAME-TABLE
           AT END
             DISPLAY "The abbreviation """ CURR-ABBREV
             """ is not in the state name table."
           WHEN STATE-ABBREV(IX1) = CURR-ABBREV
             PERFORM SETUP-BUFFERS   *> Note: uses current IX1 setting.
             DISPLAY "The abbreviation """ STATE-ABBREV(IX1)
             """ stands for the state of """
             STATE-BUFFER(1:STATE-COUNT) """,","
             " and the state capital is """ COL 5 CAPITAL-BUFFER
           END-SEARCH.

           GO TO 0020.

       SETUP-BUFFERS.
           MOVE SPACES TO CAPITAL-BUFFER.
           STRING STATE-CAPITAL(IX1) DELIMITED BY SPACES,
                  ", " STATE-ABBREV(IX1) """." DELIMITED BY SIZE
             INTO CAPITAL-BUFFER.
           MOVE ZERO TO CAPITAL-COUNT.
           INSPECT CAPITAL-BUFFER TALLYING CAPITAL-COUNT
             FOR CHARACTERS BEFORE INITIAL "."
             REPLACING ALL "_" BY SPACE.

           MOVE STATE-NAME(IX1) TO STATE-BUFFER.
           MOVE ZERO TO STATE-COUNT.
           INSPECT STATE-BUFFER TALLYING STATE-COUNT
             FOR CHARACTERS BEFORE INITIAL SPACE
             REPLACING ALL "_" BY SPACE.

       END PROGRAM SEARCH01.
```

SEND Statement Example

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  SEND01.
      *
      *  Examples for RM/COBOL Language Reference Manual.
      *  SEND statement.
      *
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 MESSAGE-BUFFER          PIC X(1000).
       01 SEGMENT-BUFFER          PIC X(500).
```

```
COMMUNICATION SECTION.
CD COM-LINE-1 FOR OUTPUT
    DESTINATION COUNT IS L1-DEST-COUNT
    TEXT LENGTH IS L1-TEXT-LENGTH
    STATUS KEY IS L1-STATUS-KEY
    DESTINATION TABLE OCCURS 5 TIMES
      INDEXED BY L1IX1, L1IX2
    ERROR KEY IS L1-ERROR-KEY
    SYMBOLIC DESTINATION IS L1-SYM-DEST.

CD COM-LINE-2 FOR I-O
    SYMBOLIC TERMINAL IS COM-L2-TERMINAL-NAME
    MESSAGE DATE IS COM-L2-MSG-DT
    MESSAGE TIME IS COM-L2-MSG-TM
    TEXT LENGTH IS COM-L2-TXT-LENGTH
    END KEY IS COM-L2-END-KEY
    STATUS KEY IS COM-L2-STATUS-KEY.

PROCEDURE DIVISION.
0010.

    SEND COM-LINE-1 FROM "Enter your PIN: ".

    SEND COM-LINE-2 FROM SEGMENT-BUFFER WITH ESI
      AFTER ADVANCING 3 LINES.

END PROGRAM SEND01.
```

SET Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SET01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  SET statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     SWITCH-1 IS SUMMARY-SWITCH,
     SWITCH-2 IS DETAIL-SWITCH.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 G1.
    02 T1                 OCCURS 100 TIMES
                            INDEXED BY IX1, IX2, IX3, IX4.
       03 E1              PIC X(5).
 01 SUB1                  PIC 9(5) BINARY.
 01 EOF-FLAG              PIC X.
    88 EOF                VALUE "T" FALSE "F".
 01 COND-1-FLAG           PIC X.
    88 COND-1             VALUE "A" WHEN FALSE SPACE.
 01 P1                    POINTER.
 01 P2                    POINTER.
```

```
01 COUNT-1               PIC 9(5) BINARY.
LINKAGE SECTION.
01 BL-RECORD.
   02 BL-FIELD-1         PIC X(10).
   02 BL-FIELD-2         PIC X(20).
PROCEDURE DIVISION.
0010.

    SET IX1 IX2 TO IX3, IX3 IX4 TO SUB1.

0020.

    SET IX1 IX2 UP BY 1, IX3 IX4 DOWN BY 2.

0030.

    SET SUMMARY-SWITCH TO OFF, DETAIL-SWITCH TO ON.

0040.

    SET EOF TO TRUE, COND-1 TO FALSE.

0050.

    SET P1 TO P2.

    SET ADDRESS OF BL-RECORD TO P1.

    SET P1 TO ADDRESS OF G1.
    SET P2 TO NULL.

0060.

    SET P1 UP BY LENGTH OF T1(1).

    SET ADDRESS OF BL-RECORD DOWN BY COUNT-1.

END PROGRAM SET01.
```

SORT Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SORT01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  SORT statement.
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT SORT-FILE ASSIGN TO SORT-WORK.

 DATA DIVISION.
 FILE SECTION.
 SD SORT-FILE.
```

```
01 SORT-RECORD.
   02 SORT-KEY-1         PIC X(05).
   02 SORT-DATA-1        PIC X(20).
   02 SORT-KEY-2         PIC 9(05) BINARY.
WORKING-STORAGE SECTION.
01 EOF-FLAG              PIC X.
   88 EOF                VALUE "T" FALSE "F".

PROCEDURE DIVISION.
MAIN1.
    SORT SORT-FILE
      ON ASCENDING KEY SORT-KEY-1
      ON DESCENDING KEY SORT-KEY-2
      WITH DUPLICATES IN ORDER
      INPUT PROCEDURE IS GET-RECORDS
      OUTPUT PROCEDURE IS PUT-RECORDS.
    STOP RUN.

GET-RECORDS.
    PERFORM WITH TEST AFTER UNTIL EOF
      CALL "READ-RECORD" USING SORT-RECORD, EOF-FLAG
      IF NOT EOF
        RELEASE SORT-RECORD
      END-IF
    END-PERFORM.

PUT-RECORDS.
    SET EOF TO FALSE.
    PERFORM UNTIL EOF
      RETURN SORT-FILE RECORD
      AT END SET EOF TO TRUE
      NOT AT END
        CALL "WRITE-RECORD" USING SORT-RECORD
      END-RETURN
    END-PERFORM.

END PROGRAM SORT01.
```

START Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  START01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  START statement (relative and indexed I-O).
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT INVENTORY-FILE    ASSIGN TO DISK
                              RELATIVE ACCESS DYNAMIC
                                RELATIVE KEY IS INVENTORY-KEY.

     SELECT DATA-BASE         ASSIGN TO DISK
                              INDEXED ACCESS DYNAMIC
```

```
                              RECORD KEY IS DB-KEY
                              FILE STATUS IS DB-STATUS.

    SELECT STATUS-FILE        ASSIGN TO DISK
                              RELATIVE ACCESS DYNAMIC
                                RELATIVE KEY IS SF-KEY.

DATA DIVISION.
FILE SECTION.
FD INVENTORY-FILE.
01 INVENTORY-RECORD      PIC X(80).

FD DATA-BASE.
01 DB-RECORD.
   02 DB-DATA-1          PIC X(10).
   02 DB-KEY             PIC X(20).
   02 DB-DATA-2          PIC X(50).

FD STATUS-FILE.
01 STATUS-RECORD         PIC X(1).

WORKING-STORAGE SECTION.
01 DB-STATUS             PIC X(02).
01 DB-START-KEY          PIC X(20).
01 INVENTORY-KEY         PIC 9(5) BINARY.
01 SF-KEY                PIC 9(5) BINARY.
01 STATUS-START-KEY      PIC 9(5) BINARY.
PROCEDURE DIVISION.
DECLARATIVES.
I-O-ERROR SECTION.
    USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
I-O-ERROR1.
    EXIT.
END DECLARATIVES.
MAIN-01 SECTION.
0010.

    MOVE 10 TO INVENTORY-KEY.
    START INVENTORY-FILE; INVALID KEY
      DISPLAY "Key 10 not present in inventory file."
    NOT INVALID KEY
      DISPLAY "Key 10 present in inventory file."
    END-START.

    START STATUS-FILE KEY IS LAST SF-KEY.

    MOVE DB-START-KEY TO DB-KEY.
    START DATA-BASE KEY >= DB-KEY SIZE 10
    INVALID KEY PERFORM DB-INVALID-KEY-HANDLER
    NOT INVALID KEY PERFORM DB-SUCCESS-HANDLER
    END-START.

    *> set filter for finding all keys ending in
    *> "smith" (case insensitively)
    START DATA-BASE WHILE KEY LIKE ".*smith".

BAD-KEY-PROCEDURE.
```

```
    EXIT.

DB-SUCCESS-HANDLER.
    EXIT.

DB-INVALID-KEY-HANDLER.
    EXIT.

END PROGRAM START01.
```

STOP Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  STOP01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  STOP statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 STATUS-CODE            PIC 9(5) BINARY.
 PROCEDURE DIVISION.
 0010.
     STOP RUN.

 0020.
     STOP RUN 1.

 0030.
     STOP RUN STATUS-CODE.

 0040.
     STOP "End of Procedure.".

 END PROGRAM STOP01.
```

STRING Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  STRING01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  STRING statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
```

```
01 FIELD-1       PIC X(10) VALUE "Fred".
01 FIELD-2       PIC X(10) VALUE "T.".
01 FIELD-GROUP   PIC X(30).
01 MONTH-VALUE   PIC X(10) VALUE "March".
01 DAY-VALUE     PIC 9(02) VALUE 3.
01 YEAR-VALUE    PIC 9(04) VALUE 1999.
01 TITLE-RECORD  PIC X(70) VALUE SPACES.
01 COLUMN-CURSOR PIC 9(04) BINARY VALUE 5.
PROCEDURE DIVISION.
0010.
    STRING FIELD-1 DELIMITED BY SPACES
        ";" DELIMITED BY SIZE
        FIELD-2 DELIMITED BY "."
        ";" DELIMITED BY SIZE
      INTO FIELD-GROUP
    ON OVERFLOW
      DISPLAY "Overflow error."
      STOP RUN
    END-STRING.

0020.
    STRING MONTH-VALUE DELIMITED BY SPACES
        SPACE DAY-VALUE "," YEAR-VALUE
        DELIMITED BY SIZE
      INTO TITLE-RECORD
      WITH POINTER COLUMN-CURSOR.

    DISPLAY FIELD-GROUP.
    DISPLAY TITLE-RECORD.
    ACCEPT FIELD-GROUP PROMPT "#" SIZE 1.

END PROGRAM STRING01.
```

SUBTRACT Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SUBTRCT1.
*
*  Examples for RM/COBOL Language Reference Manual.
*  SUBTRACT statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 TAXES                 PIC S9(10)v99.
 01 INCOME                PIC S9(10)v99.
 01 TALLY-COUNTER         PIC S9(6).
 01 TALLY-1               PIC S9(6).
 01 INTEREST              PIC S9(6)v99.
 01 PENALTY               PIC S9(6)v99.
 01 PRINCIPAL             PIC S9(6)v99.
 01 DAILY-SALES.
    02 TOPS               PIC S9(5).
```

```
   02 SKIRTS             PIC S9(5).
   02 LINGERIE           PIC S9(5).
   02 SHOES              PIC S9(5).
01 INVENTORY-ON-HAND.
   02 TOPS               PIC S9(5).
   02 SKIRTS             PIC S9(5).
   02 LINGERIE           PIC S9(5).
   02 SHOES              PIC S9(5).
PROCEDURE DIVISION.
0010.
    SUBTRACT TAXES FROM INCOME.

    SUBTRACT 1 FROM TALLY-COUNTER GIVING TALLY-1.

    SUBTRACT 2.68, INTEREST, PENALTY
      FROM PRINCIPAL ROUNDED
    ON SIZE ERROR GO TO ERROR-HANDLER.

    SUBTRACT CORR DAILY-SALES FROM INVENTORY-ON-HAND.

ERROR-HANDLER.

END PROGRAM SUBTRCT1.
```

UNLOCK Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  UNLOCK01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  UNLOCK statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     C01 IS CHANNEL-1.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT INVENTORY-FILE    ASSIGN TO DISK
                              RELATIVE ACCESS RANDOM
                                RELATIVE KEY IS
                                  INVENTORY-KEY.

     SELECT DATA-BASE         ASSIGN TO DISK
                              INDEXED ACCESS DYNAMIC
                              RECORD KEY IS DB-KEY
                              FILE STATUS IS DB-STATUS.

 DATA DIVISION.
 FILE SECTION.
 FD INVENTORY-FILE.
 01 INVENTORY-RECORD      PIC X(80).

 FD DATA-BASE.
 01 DB-RECORD.
```

```
   02 DB-DATA-1          PIC X(10).
   02 DB-KEY             PIC X(20).
   02 DB-DATA-2          PIC X(50).

WORKING-STORAGE SECTION.
01 DB-STATUS             PIC X(02).
01 DB-DELETE-KEY         PIC X(20).
01 INVENTORY-KEY         PIC 9(5) BINARY.
01 NEW-INVENTORY-ITEM    PIC X(80).

PROCEDURE DIVISION.
DECLARATIVES.
I-O-ERROR SECTION.
    USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
I-O-ERROR1.
    EXIT.
END DECLARATIVES.
MAIN-01 SECTION.
0010.
    UNLOCK DATA-BASE RECORDS.

    UNLOCK INVENTORY-FILE.

END PROGRAM UNLOCK01.
```

UNSTRING Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  UNSTRNG1.
*
*  Examples for RM/COBOL Language Reference Manual.
*  UNSTRING statement.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 FIELD-COUNT   PIC S9(05) BINARY.
 01 FIELD-1       PIC  X(10).
 01 FIELD-2       PIC  X(10).
 01 FIELD-3       PIC  X(10).
 01 DELIM-1       PIC  X.
 01 DELIM-2       PIC  X.
 01 DELIM-3       PIC  X.
 LINKAGE SECTION.
 01 PARAMETER-1.
    02 PSIZE      PIC  9(04) BINARY (2).
    02 PSTRING.
       03 PCHAR   PIC  X OCCURS 0 TO 2048 TIMES
                    DEPENDING ON PSIZE.

 PROCEDURE DIVISION USING PARAMETER-1.
 0010.
     MOVE ZERO TO FIELD-COUNT.
```

```
       UNSTRING PSTRING DELIMITED BY ";" OR "."
         INTO FIELD-1 DELIMITER IN DELIM-1
              FIELD-2 DELIMITER IN DELIM-2
              FIELD-3 DELIMITER IN DELIM-3
         TALLYING IN FIELD-COUNT
       ON OVERFLOW
         DISPLAY "Too many fields in parameter."
         STOP RUN
       END-UNSTRING.

 END PROGRAM UNSTRNG1.
```

USE Statement Example

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  USE01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  USE statement.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 CONTINUE-FLAG     PIC X(02).
 PROCEDURE DIVISION.
 DECLARATIVES.
 I-O-ERROR SECTION.
     USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
 I-O-ERROR-ROUTINE.
     DISPLAY "Error for file in I-O open mode.".
     ACCEPT CONTINUE-FLAG POSITION 0 PROMPT.
     IF CONTINUE-FLAG = "NO" STOP RUN.
 END DECLARATIVES.

 END PROGRAM USE01.
```

WRITE Statement Examples

WRITE Format 1

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  WRITE01.
*
*  Examples for RM/COBOL Language Reference Manual.
*  WRITE statement (sequential I-O).
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     C01 IS CHANNEL-1.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
```

```
    SELECT TRANSACTION-FILE  ASSIGN TO TAPE.
    SELECT PRINT-FILE        ASSIGN TO PRINTER.
    SELECT REPORT-FILE       ASSIGN TO PRINTER.

DATA DIVISION.
FILE SECTION.
FD TRANSACTION-FILE.
01 TR-RECORD             PIC X(80).

FD PRINT-FILE.
01 PF-RECORD             PIC X(60).

FD REPORT-FILE          LINAGE IS 54 LINES
                        FOOTING AT 50
                        TOP 8 BOTTOM 4.
01 RF-RECORD            PIC X(60).

WORKING-STORAGE SECTION.
01 TITLE-LINE            PIC X(60).
01 DETAIL-LINE           PIC X(60).
01 LOG-FILE-STATUS       PIC X(02).
01 PAGE-COUNT            PIC 9(05) BINARY VALUE 0.

PROCEDURE DIVISION.
DECLARATIVES.
I-O-ERROR SECTION.
    USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
I-O-ERROR1.
    EXIT.
END DECLARATIVES.
MAIN-01 SECTION.
0010.
    WRITE TR-RECORD OF TRANSACTION-FILE.

    WRITE PF-RECORD FROM TITLE-LINE
      AFTER ADVANCING PAGE.

    WRITE PF-RECORD OF PRINT-FILE
      AFTER ADVANCING CHANNEL-1.

    WRITE RF-RECORD FROM DETAIL-LINE
      AFTER ADVANCING TO LINE 10
    AT END-OF-PAGE
      ADD 1 TO PAGE-COUNT
    END-WRITE.

END PROGRAM WRITE01.
```

WRITE Format 2

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  WRITE02.
*
*  Examples for RM/COBOL Language Reference Manual.
*  WRITE statement (relative & indexed I-O).
*
```

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    C01 IS CHANNEL-1.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INVENTORY-FILE    ASSIGN TO DISK
                             RELATIVE ACCESS RANDOM
                               RELATIVE KEY IS
                                 INVENTORY-KEY.

    SELECT DATA-BASE         ASSIGN TO DISK
                             INDEXED ACCESS DYNAMIC
                             RECORD KEY IS DB-KEY
                             FILE STATUS IS DB-STATUS.

DATA DIVISION.
FILE SECTION.
FD INVENTORY-FILE.
01 INVENTORY-RECORD      PIC X(80).

FD DATA-BASE.
01 DB-RECORD.
   02 DB-DATA-1          PIC X(10).
   02 DB-KEY             PIC X(20).
   02 DB-DATA-2          PIC X(50).

WORKING-STORAGE SECTION.
01 DB-STATUS             PIC X(02).
01 DB-DELETE-KEY         PIC X(20).
01 INVENTORY-KEY         PIC 9(5) BINARY.
01 NEW-INVENTORY-ITEM    PIC X(80).

PROCEDURE DIVISION.
DECLARATIVES.
I-O-ERROR SECTION.
    USE AFTER STANDARD EXCEPTION PROCEDURE ON I-O.
I-O-ERROR1.
    EXIT.
END DECLARATIVES.
MAIN-01 SECTION.
0010.
    WRITE DB-RECORD OF DATA-BASE
    INVALID KEY PERFORM BAD-KEY-PROCEDURE
    END-WRITE.

    MOVE 5 TO INVENTORY-KEY.
    WRITE INVENTORY-RECORD FROM NEW-INVENTORY-ITEM
    INVALID KEY DISPLAY "Key 5 not accepted."
    NOT INVALID KEY DISPLAY "Key 5 written."
    END-WRITE.

BAD-KEY-PROCEDURE.

END PROGRAM WRITE02.
```

Index

Special Characters and Symbols

- (minus) PICTURE symbol 60

$

$ data-pointer data 60
$ PICTURE symbol 60

*

* (asterisk) PICTURE symbol 60

,

, (comma) PICTURE symbol 60

.

. (period) PICTURE symbol 60

/

/ (slash) PICTURE symbol 60

+

+ (plus) PICTURE symbol 60

0

0 PICTURE symbol 60

6

66-level-description-entry 16

7

77-level-description-entry 14, 16
78-level-description-entry 16

8

88-level-description-entry 16

9

9 PICTURE symbol 60

A

A - B reserved words 72
A PICTURE symbol 60
Abbreviated combined relation condition 51
ACCEPT 24
ACCESS 12
ADD 26
ADDRESS 44, 55
ADVANCING 30, 43, 49
AFTER 36, 39, 43, 49, 69
ALL 36, 42, 48
ALPHABET 10
ALPHABETIC 36, 51
Alphabetic data 36, 57
ALPHABETIC-LOWER 51
ALPHABETIC-UPPER 51
Alphabet-name 10, 12, 15, 38, 45
ALPHANUMERIC 36
Alphanumeric data 36, 57
ALPHANUMERIC-EDITED 36
Alphanumeric-edited data 36, 57
ALSO 10, 34
ALTER 27
ALTERNATE 12
AND 42, 51, 57
ANY 34
ARE 10, 15, 16
AREA 10, 12
AREAS 12
Arithmetic statements 26, 29, 32, 38, 47
Arithmetic-expression 29, 34, 42, 51
AS 16
ASCENDING 16, 38, 45
ASSIGN 12
Asterisk PICTURE symbol 60
AT 15, 24, 30, 41, 42, 49
AUTHOR 9
AUTO 20, 24
AUTOMATIC 12
AUTO-SKIP 20, 24

B

B PICTURE symbol 60
BACKGROUND 20
BACKGROUND-COLOR 20
BEEP 20, 24, 30
BEFORE 24, 36, 39, 43, 49
BELL 20, 24, 30
BINARY 12, 16
BINARY-SEQUENTIAL 80
BLACK 80
BLANK 16, 20
BLINK 20, 24, 30
BLOCK 15, 24, 30
BLUE 80
BOTTOM 15
BROWN 80
BY 16, 19, 27, 32, 36, 38, 39, 44, 47, 48, 50

C

C reserved words 72
C01 80
C010 80
C011 80
C012 80
C02 80
C03 80
C04 80
C05 80
C06 80
C07 80
C08 80
C09 80
CALL 27, 28
CANCEL 28
CARD-PUNCH 80
CARD-READER 80
CASE-INSENSITIVE 46, 51
CASE-SENSITIVE 46, 51
CASSETTE 80
Category-name 36
CD 19
Cd-name 19, 24, 30, 33, 41, 43, 53, 55
CENTURY-DATE 24
CENTURY-DAY 24
CF 77
CH 77
CHARACTER 10, 12, 16, 20, 24
CHARACTERS 10, 15, 36
Character-string 16, 20
  PICTURE 57
CLASS 10
Class condition 51
Class-name 10, 51
CLOCK-UNITS 10
CLOSE 29
COBOL words 72, 73, 74, 75, 76, 77, 78, 80
COBOL, ENTER statement 33
CODE 77
Code-name 10, 80
CODE-SET 12, 15
COL 20, 24, 30, 69
COLLATING 10, 12, 38, 45
Color-name 20, 80
COLUMN 20, 24, 30, 69
Combined condition 51
COMMA 10
Comma PICTURE symbol 60
Commands
  Compile 1
  Debug 4
  Runtime 3
Comment-entry 9
Comment-text 69
COMMON 9, 70
COMMUNICATION 14
Communication statements 24, 30, 33, 41, 43
Communication-description-entry 14, 19
COMP 16
COMP-1 16
COMP-3 16
COMP-4 16
COMP-5 16
COMP-6 16
Compile commands 1
COMPUTATIONAL 16
COMPUTATIONAL-1 16
COMPUTATIONAL-3 16
COMPUTATIONAL-4 16
COMPUTATIONAL-5 16
COMPUTATIONAL-6 16
COMPUTE 29
Computer-name 10, 80
Concatenation expression 54, 56
Condition 34, 35, 39, 42, 51
Conditional-statement 54
Condition-name 10, 16, 42, 44, 51, 53, 55
Condition-name condition 51
CONFIGURATION 10
CONSOLE 10, 80
Constant-expression 16, 54, 57
Constant-name 16
CONTAINS 10, 15
CONTENT 27
Context-sensitive words 78
CONTINUE 29
CONTROL 24, 30
Control statements 27, 28, 29, 33, 34, 35, 39, 42, 46
CONTROLS 77
CONVERT 24, 30
CONVERTING 36

COPY 50
COPY and REPLACE statements 50
CORR 26, 38, 47
CORRESPONDING 26, 38, 47
COUNT 19, 24, 48, 55
COUNT-MAX 55
COUNT-MIN 55
CR (credit) PICTURE symbol 60
CRT 10
cs PICTURE symbol 60
CURRENCY 10
CURRENCY SIGN clause 60
Currency symbol PICTURE symbol 60
CURSOR 10, 24
CYAN 80
CYCLE 34

D

D reserved words 73
DATA 14, 15, 36, 41, 70, 71
Data Manipulation statements 36, 38, 44, 47, 48
Data-description-entry 16
Data-division 9, 14
Data-name 10, 12, 15, 16, 19, 23, 24, 38, 41, 42, 44, 45, 46, 53, 55, 57
DATA-POINTER 36
DATE 19, 24
DATE-AND-TIME 24
DATE-COMPILED 9, 24, 57
DATE-WRITTEN 9
DAY 24
DAY-AND-TIME 24
DAY-OF-WEEK 24
DB (debit) PICTURE symbol 60
DE 77
Debug commands 4
DEBUG-CONTENTS 77
DEBUGGING 10
DEBUG-ITEM 77
DEBUG-LINE 77
DEBUG-NAME 77
DEBUG-SUB-1 77
DEBUG-SUB-2 77
DEBUG-SUB-3 77
DECIMAL-POINT 10, 60
DECLARATIVES 23, 24
DEFAULT 36
DELETE 29
DELIMITED 47, 48
DELIMITER 12, 48
Delimiter-name 12, 80
DEPENDING 15, 16, 35
DESCENDING 16, 38, 45
DESTINATION 19
DETAIL 77
Device-name 12, 80
Directives 54, 69
DISABLE 30
DISC 80
DISK 80
DISPLAY 12, 16, 20, 30
DIVIDE 32
DIVISION 9, 10, 14, 23, 24, 70, 71
DOWN 44
DUPLICATES 12, 45
DYNAMIC 12

E

E reserved words 73
EBCDIC 80
ECHO 24
EGI 43
ELSE 35
EMI 43
ENABLE 33
END 10, 19, 23, 24, 41, 42, 50, 69, 70, 71
END-ACCEPT 24
END-ADD 26
END-CALL 27, 28
END-COMPUTE 29
END-COPY 50
END-DELETE 29
END-DIVIDE 32
END-EVALUATE 34
END-IF 35
END-MULTIPLY 38
END-OF-PAGE 49
END-PERFORM 39
End-program-header 9, 50
END-READ 41
END-RECEIVE 41
END-REPLACE 50
END-RETURN 42
END-REWRITE 42
END-SEARCH 42
END-START 46
END-STRING 47
END-SUBTRACT 47
END-UNSTRING 48
END-WRITE 49
ENTER 33
ENVIRONMENT 10, 70, 71
Environment-division 9, 10
EOL 20, 24, 30
EOP 49
EOS 20, 24, 30
EQUAL 42, 46, 51
ERASE 20, 24, 30
ERROR 19, 26, 29, 32, 38, 47, 49
ESCAPE 24

ESI 43
EVALUATE 34
EVERY 10
EXCEPTION 24, 27, 28, 49
EXCLUSIVE 12, 39, 57
EXIT 34
Expression 29, 34, 35, 39, 42, 51, 55, 63
  arithmetic 29, 34, 42, 51, 55
  conditional 34, 35, 39, 42, 51
  regular 63
EXTEND 39, 49
EXTERNAL 15, 16

F

F - I reserved words 74
FALSE 16, 34, 44
FD 15
Feature-name 10, 80
Figurative-constants 54, 56
FILE 10, 12, 14, 29
FILE-CONTROL 10
File-control-entry 10, 12
File-description-entry 14, 15
FILE-ID 80
File-name 10, 12, 15, 29, 38, 39, 41, 42, 45, 46, 48, 49, 53, 55
FILLER 16, 20, 36
FINAL 77
FIRST 36, 46
FIXED 77
FOOTING 15
FOR 10, 19, 29, 36
FOREGROUND 20
FOREGROUND-COLOR 20
FROM 15, 20, 24, 39, 42, 43, 47, 49
FULL 20
FUNCTION 77

G

General format for a sequence of source programs 70
General format for nested source programs 70
General format for *nested-source-program* 70
GENERATE 77
GIVING 23, 24, 26, 27, 32, 38, 45, 47
GLOBAL 15, 16, 49
GO 35
GOBACK 35
GREATER 46, 51
GREEN 80
GROUP 77

H

HEADING 77
HIGH 24, 30
HIGHEST-VALUE 55
HIGHLIGHT 20, 24, 30
HIGH-VALUE 56
HIGH-VALUES 56

I

ID 9, 70, 71
IDENTIFICATION 9, 70, 71
Identification-division 9
Identifier 20, 24, 26, 27, 28, 29, 30, 32, 33, 34, 35, 36, 38, 39, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 54, 55
IF 35
IMP 69
Imperative-statement 24, 26, 27, 28, 29, 32, 34, 38, 39, 41, 42, 46, 47, 49, 54
IN 10, 15, 44, 45, 48, 50, 53, 55
Index 60
INDEX 16
INDEXED 12, 16, 19
Index-name 16, 19, 39, 42, 44, 51, 53, 55
INDICATE 77
INITIAL 9, 19, 36, 70, 71
INITIALIZE 36
INITIAL-VALUE 55
INITIATE 77
INPUT 12, 19, 30, 33, 39, 45, 49
INPUT-OUTPUT 10, 12
INSPECT 36
INSTALLATION 9
Integer 10, 12, 15, 16, 19, 20, 24, 30, 39, 43, 44, 46, 49, 55, 57
INTO 32, 41, 42, 47, 48
INVALID 29, 41, 42, 46, 49
I-O 19, 30, 33, 39, 49
I-O statements 29, 39, 41, 42, 46, 48, 49
I-O-CONTROL 10
IS 9, 10, 12, 15, 16, 19, 20, 24, 30, 38, 41, 42, 45, 46, 51, 69, 70, 71

J

J - N reserved words 75
JUST 16, 20, 55
JUSTIFIED 16, 20, 55

K

KEY 12, 16, 19, 24, 29, 30, 33, 38, 41, 42, 45, 46, 49
KEYBOARD 80

L

LABEL 15

Label-name 15, 80
Language-name 33, 80
LAST 46
LEADING 10, 16, 20, 36
LEFT 16, 46, 51
Leftmost-character-position 55
LENGTH 19, 44, 55, 57
Length-1 55
LESS 46, 51
Level-number 16, 20
Library-name 50, 53
LIKE 51
LIKE condition 51, 54, 63
LIMIT 77
LIMITS 77
LINAGE 15
LINAGE-COUNTER 53, 55
LINE 12, 20, 24, 30, 43, 49
LINE-COUNTER 77
LINES 15, 43, 49
LINE-SEQUENTIAL 80
LINKAGE 14, 33
LISTING 69, 80
Literal 9, 10, 12, 15, 16, 20, 24, 26, 27, 28, 30, 32, 33, 34, 36, 38, 42, 43, 46, 47, 48, 49, 50, 51, 55, 56, 57, 70, 71
LOCK 12, 29, 39, 41
LOW 24, 30
Low Volume I-O statements 24, 30
LOWEST-VALUE 55
LOWLIGHT 20, 24, 30
LOW-VALUE 56
LOW-VALUES 56
Low-volume-I-O-name 10, 24, 30, 80

M

MAGENTA 80
MAGNETIC-TAPE 80
MANUAL 12
MARGIN-R 69
MAX-VALUE 55
MEMORY 10
MERGE 12, 38
MESSAGE 19, 24, 41
Minus PICTURE symbol 60
MIN-VALUE 55
Miscellaneous formats 54
Mnemonic-name 10, 24, 30, 43, 44, 49
MODE 10, 12, 24, 30
MODULES 10
MOVE 38
MULTIPLE 10, 12
MULTIPLY 38

N

NATIVE 10
Negated condition 51
NEGATIVE 51
nested-source-program 9, 70, 71
NEXT 24, 35, 41, 42, 49, 57
NO 12, 20, 24, 29, 30, 39, 41
NOT 24, 26, 27, 29, 32, 34, 38, 41, 42, 46, 47, 48, 49, 51, 57
NULL 44, 56
NULLS 44, 56
NUMBER 20, 24, 30
NUMERIC 10, 36, 51
Numeric data 36, 57
NUMERIC-EDITED 36
Numeric-edited data 36, 57

O

O - Q reserved words 75
OBJECT-COMPUTER 10
OCCURS 16, 19
OF 10, 15, 44, 50, 53, 55, 57, 69
OFF 10, 24, 44, 50, 69
OMITTED 15, 27, 28
ON 10, 12, 15, 16, 24, 26, 27, 28, 29, 32, 35, 38, 44, 45, 47, 48, 49, 69
OPEN 39
OPTIONAL 12
OR 46, 48, 51, 57
ORDER 45
ORGANIZATION 12
OTHER 34
OUTPUT 12, 19, 30, 33, 38, 39, 45, 49
OVERFLOW 27, 47, 48

P

P PICTURE symbol 60
PACKED-DECIMAL 16
PADDING 12
PAGE 43, 49, 69
PAGE-COUNTER 77
PARAGRAPH 34, 55
Paragraph-name 23, 24, 53
Pattern 63
  LIKE 63
PERFORM 39
Period PICTURE symbol 60
PF 77
PH 77
PIC 16, 20
PICTURE 16, 20, 57
PICTURE character-string 54, 57
PICTURE symbols 54, 60

PLUS 20
Plus PICTURE symbol 60
POINTER 16, 47, 48
POSITION 10, 24, 30
POSITIVE 51
PREVIOUS 41
PRINT 80
PRINTER 80
PRINTER-1 80
PRINTING 50
PROCEDURE 23, 24, 38, 45, 49, 55, 70, 71
Procedure-division 9, 23, 24
Procedure-name 27, 35, 38, 39, 45
PROCEDURE-NAME 55
PROCEDURES 77
PROCEED 27
PROGRAM 9, 10, 28, 34, 50, 70, 71
Program structure 70, 71
PROGRAM-ID 9, 55, 70, 71
Program-name 9, 50, 70, 71
PROMPT 24
Pseudo-text 50
PURGE 41

Q

Qualification 53
QUEUE 19
QUOTE 56
QUOTES 56

R

R reserved words 76
RANDOM 12
RD 77
READ 41
RECEIVE 41
RECORD 10, 12, 15, 29, 41, 42, 48, 69
Record-description-entry 14, 16
RECORDING 77
Record-name 42, 49
RECORDS 10, 12, 15, 48
RED 80
REDEFINES 16
REEL 10, 29
REFERENCE 27
Reference modification 54, 55
REFERENCES 77
Regular expression 63
Relation condition 51
Relational operator 16, 51
RELATIVE 12
RELEASE 42
REMAINDER 20, 32
REMARKS 9
REMOVAL 29
RENAMES 16
REPLACE 50
REPLACING 36, 43, 50
REPORT 77
REPORTING 77
REPORTS 77
REQUIRED 20
RERUN 10
Rerun-name 10, 80
RESERVE 12
Reserved words 72, 73, 74, 75, 76, 77
  A - B 72
  C 72
  context-sensitive 78
  D 73
  E 73
  F - I 74
  J - N 75
  O - Q 75
  R 76
  S 76
  T - Z 77
  unused reserved words 77
RESET 77
RETURN 42
RETURN-CODE 55
RETURNING 23, 24, 27
REVERSE 20, 24, 30
REVERSED 20, 24, 30, 39
REVERSE-VIDEO 20, 24, 30
REWIND 29, 39
REWRITE 42
RF 77
RH 77
RIGHT 16, 20, 46, 51, 55
rmcobol 1
ROUNDED 26, 29, 32, 38, 47
Routine-name 33
RUN 46
runcobol 3
Runtime commands 3

S

S PICTURE symbol 60
S reserved words 76
SAME 10, 16
SCREEN 14, 20
Screen-description-entry 14, 20
Screen-name 20, 24, 30, 53
SD 15
SEARCH 42
SECTION 10, 14, 23, 24, 34, 55
Section-name 23, 24, 53
SECURE 20, 24

SECURITY 9
SEGMENT 41
SEGMENT-LIMIT 10
Segment-number 10, 23, 24
SELECT 12
SEND 43
Sentence 54
  miscellaneous formats 54
  procedure division general formats 23, 24
  sentence 54
SENTENCE 24, 35, 42
SEPARATE 10, 16, 20
SEQUENCE 10, 12, 38, 45
SEQUENTIAL 12
SET 16, 44
SIGN 10, 16, 20
Sign condition 51
SIZE 10, 15, 24, 26, 29, 30, 32, 38, 46, 47, 57
Slash PICTURE symbol 60
SORT 10, 12, 45
SORT-MERGE 10, 12
Sort-Merge statements 38, 42, 45
Sort-merge-file-control-entry 12
Sort-merge-file-description-entry 14, 15
SORT-WORK 80
SOURCE 19
SOURCE-COMPUTER 10
Source-program 9
SPACE 56
SPACES 56
Special registers 54, 55
SPECIAL-NAMES 10
Split-key-name 12, 41, 46, 53
STANDARD 15, 49
STANDARD-1 10, 12
STANDARD-2 10
START 46, 57
Statements
  Arithmetic 26, 29, 32, 38, 47
  Communication 24, 30, 33, 41, 43
  Control 27, 28, 29, 33, 34, 35, 39, 42, 46
  COPY and REPLACE 50
  Data Manipulation 36, 38, 44, 47, 48
  I-O 29, 39, 41, 42, 46, 48, 49
  Low Volume I-O 24, 30
  Sort-Merge 38, 42, 45
Statement-sequence 54
STATUS 10, 12, 19, 24
STOP 46
STRING 47
SUB-QUEUE-1 19
SUB-QUEUE-2 19
SUB-QUEUE-3 19
Subscript 55
Subscripting 54, 55
SUBTRACT 47
SUM 77
SUPPRESS 50
Switch status condition 51
Switch-name 10, 80
SYMBOLIC 10, 19
Symbolic-character 10, 56
Symbols, PICTURE 60
SYNC 16
SYNCHRONIZED 16
SYSIN 80
SYSOUT 80
System-name 80

T

T - Z reserved words 77
TAB 24
TABLE 19
TALLYING 36, 48
TAPE 10, 12
TERMINAL 19, 30, 33
TERMINATE 77
TEST 39
TEXT 19
Text-name 50, 53
THAN 46, 51
THEN 35, 36, 54
THROUGH 10, 16, 34, 38, 39, 45
THRU 10, 16, 34, 38, 39, 45
TIME 19, 24
TIMES 16, 19, 39
TO 12, 15, 16, 20, 26, 27, 35, 36, 38, 42, 44, 46, 49, 51
TOP 15
TRAILING 10, 16, 20, 36
TRIMMED 46, 51
TRUE 34, 44
TYPE 77

U

UNDERLINE 20
Unicode property category escapes 68
UNIT 10, 24, 29, 30
UNLOCK 48
UNSTRING 48
UNTIL 39
Unused reserved words 77
UP 44
UPDATE 24
UPON 30
USAGE 16, 20
USE 23, 24, 49
USE statement 23, 24, 49
USING 20, 23, 24, 27, 28, 38, 45

V

V PICTURE symbol 60
VALUE 15, 16, 20, 36
VALUES 16
VARIABLE 77
VARYING 15, 39, 42

W

WHEN 16, 20, 34, 42
WHEN-COMPILED 55
WHILE 46
WHITE 80
WITH 10, 12, 15, 24, 29, 30, 33, 36, 39, 41, 43, 45, 46, 47, 48
Word 50
WORDS 10
Words reserved 72, 73, 74, 75, 76, 77
  A - B 72
  C 72
  D 73
  E 73
  F - I 74
  J - N 75
  O - Q 75
  R 76
  S 76
  T - Z 77
WORKING-STORAGE 14
WRITE 49

X

X PICTURE symbol 60

Y

YYYYDDD 24
YYYYMMDD 24

Z

Z PICTURE symbol 60
ZERO 16, 20, 51, 56
ZEROES 56
ZEROS 56