

---

Liant Software Corporation

# RM/COBOL<sup>®</sup>

**Syntax Summary**

**First Edition**

**LIANT**

This document provides complete syntax for all RM/COBOL commands, divisions, entries, statements, and other general formats. Use this pamphlet in conjunction with the *RM/COBOL Language Reference Manual* and the *RM/COBOL User's Guide*.

The *RM/COBOL Syntax Summary* has been prepared for all implementations of RM/COBOL. Consult the *RM/COBOL User's Guide* for all appropriate operating system rules and conventions (such as command line invocation).

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopied, recorded, or otherwise, without prior written permission of Liant Software Corporation.

The information in this document is subject to change without prior notice. Liant Software Corporation assumes no responsibility for any errors that may appear in this document. Liant reserves the right to make improvements and/or changes in the products and programs described in this guide at any time without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted.

The software described in this document is furnished to the user under a license for a specific number of uses and may be copied (with inclusion of the copyright notice) only in accordance with the terms of such license.

Copyright © 1985-2005 by Liant Software Corporation. All rights reserved. Printed in U.S.A.

**Liant Software Corporation**

8911 N. Capital of Texas Highway  
Austin, TX 78759  
U.S.A.

Phone (512) 343-1010  
(800) 762-6265  
Fax (512) 343-9487

Website <http://www.liant.com/>

**Documentation Release History for the RM/COBOL Syntax Summary:**

<b>Edition Number</b>	<b>Document Part Number</b>	<b>Applies To Product Version</b>	<b>Publication Date</b>
1	401225	RM/COBOL version 9 and later	January 2005



# Contents

Compile Command.....	1
Runtime Command.....	3
Debug Commands .....	4
Source Program General Format .....	6
Identification Division General Format.....	6
Environment Division General Format .....	7
File Control Entry General Formats .....	10
Data Division General Format.....	12
file-description-entry .....	13
sort-merge-file-description-entry .....	13
record-description-entry .....	14
77-level-description-entry .....	14
data-description-entry.....	14
communication-description-entry.....	16
screen-description-entry .....	17
Procedure Division General Formats .....	21
General Formats for COBOL Statements.....	22
ACCEPT Statement.....	22
ADD Statement .....	24
ALTER Statement .....	25
CALL Statement .....	25
CALL PROGRAM Statement.....	26
CANCEL Statement.....	26
CLOSE Statement .....	26
COMPUTE Statement.....	27
CONTINUE Statement.....	27
DELETE Statement.....	27
DELETE FILE Statement .....	27
DISABLE Statement .....	27
DISPLAY Statement .....	28
DIVIDE Statement .....	29
ENABLE Statement .....	30
ENTER Statement.....	30
EVALUATE Statement.....	31
EXIT Statement.....	32
GOBACK Statement .....	32
GO TO Statement.....	32

IF Statement .....	33
INITIALIZE Statement .....	33
INSPECT Statement.....	33
MERGE Statement.....	34
MOVE Statement .....	35
MULTIPLY Statement.....	35
OPEN Statement .....	36
PERFORM Statement .....	36
PURGE Statement.....	37
READ Statement.....	37
RECEIVE Statement .....	38
RELEASE Statement .....	38
RETURN Statement.....	38
REWRITE Statement .....	39
SEARCH Statement .....	39
SEND Statement .....	40
SET Statement.....	40
SORT Statement.....	41
START Statement .....	42
STOP Statement .....	42
STRING Statement .....	43
SUBTRACT Statement .....	43
UNLOCK Statement .....	44
UNSTRING Statement.....	44
USE Statement .....	44
WRITE Statement .....	45
General Format for END PROGRAM Header.....	46
General Formats for COPY and REPLACE Statements .....	46
General Formats for Conditions .....	47
Relation Condition .....	47
LIKE Condition (Special Case of a Relation Condition) .....	47
Class Condition .....	47
Sign Condition.....	48
Condition-Name Condition .....	48
Switch-Status Condition.....	48
Negated Condition.....	48
Combined Condition .....	48
Abbreviated Combined Relation Condition .....	48
General Formats for Qualification.....	49
Miscellaneous Formats .....	50
Sentence .....	50
Statement Sequence.....	50
Subscripting.....	50
Reference Modification.....	50

Identifier .....	50
Special Registers .....	51
Figurative-Constants .....	52
Concatenation Expression .....	52
Constant-Expression.....	52
General Format for Nested Source Programs .....	53
General Format for <i>nested-source-program</i> .....	53
General Format for a Sequence of Source Programs.....	54
PICTURE Clause .....	55
PICTURE Character-Strings .....	55
PICTURE Symbols .....	56
Reserved Words .....	60
Context-Sensitive Words.....	65
Nonreserved System-Names.....	67
Code-Names .....	67
(Color-Integer) Color-Names .....	67
Computer-Names .....	67
Delimiter-Names .....	67
Device-Names .....	67
Feature-Names .....	68
Label-Names .....	68
Language-Names.....	68
Low-Volume-I-O-Names .....	68
Rerun-Names.....	68
Switch-Names .....	68

## List of Tables

Table 1: PICTURE Symbol Precedence.....	59
Table 2: Context-Sensitive Words.....	65





# 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.

Option	Description								
<b>A</b>	Direct the compiler to generate the allocation map in the listing.								
<b>B</b>	Define as binary sequential those sequential files not explicitly declared to be line sequential in their file control entries.								
<b>C</b>	Suppress the inclusion of copied text in the listing.								
<b>D</b>	Direct RM/COBOL to compile all source programs as if the WITH DEBUGGING MODE clause appeared in each compiled program.								
<b>E</b>	Suppress the inclusion of the source program component in the listing except for lines associated with diagnostic messages.								
<b>F</b> ={ <i>(keyword-list)</i> keyword}	Direct the compiler to flag occurrences of these language elements: <table data-bbox="747 1155 1104 1281" style="margin-left: 40px;"> <tr> <td>COM1</td> <td>INTERMEDIATE</td> </tr> <tr> <td>COM2</td> <td>OBSOLETE</td> </tr> <tr> <td>EXTENSION</td> <td>SEG1</td> </tr> <tr> <td>HIGH</td> <td>SEG2</td> </tr> </table> <p>If leading hyphens are used, the parentheses are optional.</p>	COM1	INTERMEDIATE	COM2	OBSOLETE	EXTENSION	SEG1	HIGH	SEG2
COM1	INTERMEDIATE								
COM2	OBSOLETE								
EXTENSION	SEG1								
HIGH	SEG2								
<b>G</b> = <i>path</i>	Designate a file to be used as the compiler configuration.								
<b>H</b> = <i>path</i>	Designate a file as a supplement to the compiler configuration.								
<b>K</b>	Suppress the banner message and the terminal error listing.								
<b>L</b> [= <i>path</i> ]	Direct the compiler to produce a listing file and optionally specify the directory in which to place the listing file.								
<b>M</b>	Direct 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.								
<b>N</b>	Suppress the generation of an object program.								

Option	Description
<b>O=</b> <i>path</i>	Specify the directory pathname where the object file will be placed.
<b>P</b>	Direct the compiler to write a copy of the listing to the printer.
<b>Q</b>	Direct the compiler to eliminate debugging information from generated object programs.
<b>R</b>	Direct the compiler to generate a sequential number in the first six columns of source records as they appear on the listing.
<b>S</b>	Direct 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).
<b>T</b>	Direct the compiler to write a copy of the listing to the standard output device.
<b>U[={B D P}]</b>	<p>Direct the compiler to assume an alternative usage for data items described as COMP or COMPUTATIONAL.</p> <p>The U Option specified alone or as U=B directs the compiler to assume BINARY usage for data items described as COMP or COMPUTATIONAL.</p> <p>The U=D Option directs the compiler to assume DISPLAY usage for items described as COMP or COMPUTATIONAL.</p> <p>The U=P Option directs the compiler to assume PACKED-DECIMAL usage for items described as COMP or COMPUTATIONAL.</p>
<b>V</b>	Define as line sequential those sequential files not explicitly declared to be binary sequential in their file control entries.
<b>W=</b> <i>n</i>	Specify the amount of memory (in kilobytes) that the compiler should use for its internal table storage. <i>n</i> can be a decimal number from 32 to 16384.
<b>X</b>	Direct the compiler to generate a cross reference map in the listing.
<b>Y=</b> <i>n</i>	Direct the compiler to output the symbol table and debug line table to the object program file. <i>n</i> can be 0 to 3.
<b>Z=</b> <i>version</i>	Indicate the version of the RM/COBOL runtime you want to use. <i>version</i> can be 7 through 11.
<b>2</b>	Direct the compiler to accept source programs created for the RM/COBOL 2. <i>n</i> compiler.
<b>7</b>	Specify 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.

Option	Description
<b>A</b> =[ <i>delim</i> ] [ <i>string</i> ] [ <i>delim</i> ]	Pass an argument to the main program. The delimiter characters are optional if string does not contain spaces.
<b>B</b> = <i>n</i>	Specify a maximum buffer size for use with the ACCEPT and DISPLAY statements.
<b>C</b> = <i>pathname</i>	Designate a file to be used as the runtime configuration file.
<b>D</b>	Invoke the RM/COBOL Interactive Debugger.
<b>F</b> =< <i>fill-char</i> >	Fill read-write memory with fill characters when loading programs.
<b>I</b>	Collect RM/COBOL program instrumentation data.
<b>K</b>	Suppress the banner message and the STOP RUN message.
<b>L</b> = <i>pathname</i>	Designate RM/COBOL non-COBOL subprogram libraries.
<b>M</b>	Direct that level 2 ANSI semantics are to be used for Format 1 ACCEPT and DISPLAY statements.
<b>S</b> = <i>n . . . n</i>	Set (or reset) the initial value of switches in the RM/COBOL program.
<b>T</b> = <i>n</i>	Specify the amount of memory ( <i>n</i> bytes) to be used for a sort operation.
<b>V</b>	Direct that a trace of support modules loaded by the RM/COBOL runtime system be displayed.
<b>X</b> = <i>pathname</i>	Designate a file as a supplement to the runtime configuration.

---

# Debug Commands

The Debug commands are as follows.

Command	Description
<b>A</b>	Set breakpoints and resume program execution from the current location. <b>A</b> [ <i>line</i> [ + <i>intra</i> <i>line</i> ] [ , [ <i>prog-name</i> ] [ , [ <i>count</i> ] ] ] ]
<b>B</b>	Display all currently set breakpoints or set breakpoints at specific procedural statements. <b>B</b> [ <i>line</i> [ + <i>intra</i> <i>line</i> ] [ , [ <i>prog-name</i> ] [ , [ <i>count</i> ] ] ] ]
<b>C</b>	Clear any breakpoints that have been set with the A or B Command. <b>C</b> [ <i>line</i> [ + <i>intra</i> <i>line</i> ] [ , [ <i>prog-name</i> ] ] ]
<b>D</b>	Display on the screen the value of a specified data item. <b>Identifier Format</b> <b>D</b> <i>name-1</i> [ { <b>IN</b>   <b>OF</b> } <i>name-2</i> ] ... [ <i>script</i> ] [ <i>refmod</i> ] [ , { <i>type</i>   { *   & } [ <i>type</i> ] ] [ # <i>alias</i> ] <b>Address-Size Format</b> <b>D</b> [ <i>base</i> : ] <i>address</i> [ + <i>occur-size</i> * <i>occur-num</i> ] ..., <i>size</i> , [ <i>type</i> ] [ # <i>alias</i> ] <b>Alias Format</b> <b>D</b> # <i>alias</i>
<b>E</b>	End Debug; the currently executing program runs until completion. <b>E</b>
<b>L</b>	Specify a line on the monitor screen at which command input echoes and Debug responses are to be displayed. <b>L</b> [ <i>line-display</i> ]
<b>M</b>	Change the value of a specified data item. <b>Identifier Format</b> <b>M</b> <i>name-1</i> [ { <b>IN</b>   <b>OF</b> } <i>name-2</i> ] ... [ <i>script</i> ] [ <i>refmod</i> ] [ , { <i>type</i>   { *   & } [ <i>type</i> ] ] [ # <i>alias</i> ] , <i>value</i> <b>Address-Size Format</b> <b>M</b> [ <i>base</i> : ] <i>address</i> [ + <i>occur-size</i> * <i>occur-num</i> ] ..., <i>size</i> , [ <i>type</i> ] [ # <i>alias</i> ] , <i>value</i> <b>Alias Format</b> <b>M</b> # <i>alias</i> , <i>value</i>
<b>Q</b>	Stop program execution. <b>Q</b>

Command	Description
<b>R</b>	Specify that program execution resume at the current location, or at another location specified in the command. <b>R</b> [ <i>statement-address</i> ]
<b>S</b>	Specify that program execution occur one step at a time. <b>S</b> [ <b>P</b>   <b>S</b> ] [ <i>count</i> ]
<b>T</b>	Monitor the value of a specified data item, and suspend execution whenever a change in that value occurs. That is, a data trap. <b>Identifier Format</b> <b>T</b> <i>name-1</i> [ { <b>IN</b>   <b>OF</b> } <i>name-2</i> ] ... [ <i>script</i> ] [ <i>refmod</i> ] [ , { <i>type</i>   { *   & } [ <i>type</i> ] } ] [ # <i>alias</i> ] <b>Address-Size Format</b> <b>T</b> [ <i>base</i> : ] <i>address</i> [ + <i>occur-size</i> * <i>occur-num</i> ] ..., <i>size</i> , [ <i>type</i> ] [ # <i>alias</i> ] <b>Alias Format</b> <b>T</b> # <i>alias</i>
<b>U</b>	Clear some or all currently active data traps. <b>Identifier Format</b> <b>U</b> <i>name-1</i> [ { <b>IN</b>   <b>OF</b> } <i>name-2</i> ] ... [ <i>script</i> ] [ <i>refmod</i> ] [ , { <i>type</i>   { *   & } [ <i>type</i> ] } ] <b>Address-Size Format</b> <b>U</b> [ <i>base</i> : ] <i>address</i> [ + <i>occur-size</i> * <i>occur-num</i> ] ..., <i>size</i> , [ <i>type</i> ] <b>Alias Format</b> <b>U</b> # <i>alias</i>

**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.

---

## Source Program General Format

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

---

## Identification Division General Format

{ IDENTIFICATION } DIVISION.  
{ ID }

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

```

(continued on next page)

(continued from previous page)

$$\left[ \text{ALPHABET } \textit{alphabet-name-1} \text{ IS } \left\{ \begin{array}{l} \text{STANDARD-1} \\ \text{STANDARD-2} \\ \text{NATIVE} \\ \textit{code-name-1} \\ \left\{ \textit{literal-1} \left[ \left\{ \begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right\} \textit{literal-2} \right] \right\} \dots \\ \left[ \text{ALSO } \textit{literal-3} \left[ \left\{ \begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right\} \textit{literal-4} \right] \right] \dots \dots \end{array} \right\} \dots \right]$$

$$\left[ \text{SYMBOLIC } \left[ \begin{array}{l} \text{CHARACTER} \\ \text{CHARACTERS} \end{array} \right] \left\{ \left\{ \textit{symbolic-character-1} \right\} \dots \left[ \text{IS ARE} \right] \right. \right. \\ \left. \left. \left\{ \textit{integer-1} \right\} \dots \right\} \dots \left[ \text{IN } \textit{alphabet-name-2} \right] \dots \right]$$

$$\left[ \text{CLASS } \textit{class-name-1} \text{ IS } \left\{ \textit{literal-5} \left[ \left\{ \begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right\} \textit{literal-6} \right] \right\} \dots \dots \right]$$

$$\left[ \text{CURRENCY SIGN IS } \textit{literal-7} \right]$$

$$\left[ \text{DECIMAL-POINT IS COMMA} \right]$$

$$\left[ \text{NUMERIC SIGN IS } \left\{ \begin{array}{l} \text{LEADING} \\ \text{TRAILING} \end{array} \right\} \left[ \text{SEPARATE CHARACTER} \right] \right]$$

$$\left[ \text{CONSOLE IS CRT} \right]$$

$$\left[ \text{CURSOR IS } \textit{data-name-1} \right]$$

$$\left[ \text{CRT STATUS IS } \textit{data-name-2} \right] . \left. \right] \left. \right] \left. \right]$$

$$\left[ \text{INPUT-OUTPUT SECTION.} \right.$$

$$\text{FILE-CONTROL.}$$

$$\left\{ \textit{file-control-entry-1} \right\} \dots$$

(continued on next page)



(continued from previous page)

[ I-O-CONTROL. [

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

[ [ END OF ] { REEL } ]

[ { UNIT } ] OF *file-name-2* ] ...

EVERY { *integer-1* RECORDS } ]

[ *integer-2* CLOCK-UNITS ]

[ *condition-name-1* ] ] ]

[ SAME [ RECORD ]

[ SORT ]

[ SORT-MERGE ] ] AREA FOR *file-name-3* { *file-name-4* } ... ] ...

[ MULTIPLE FILE TAPE CONTAINS

{ *file-name-5* [ POSITION IS *integer-3* ] } ... ] ... ] ] ] ]

## File Control Entry General Formats

### File Control Entry

SELECT [ [ NOT ] OPTIONAL ] *file-name-1*

ASSIGN TO { { *data-name-1* }  
 { *literal-1* } }  
 { DISPLAY  
INPUT  
OUTPUT  
INPUT - OUTPUT } [ *data-name-1* ]  
RANDOM } [ *literal-1* ]  
TAPE  
*device-name-1* }

[ RESERVE { *integer-1* }  
 { NO } ] [ ALTERNATE ] [ AREA  
AREAS ] ]

[ [ ORGANIZATION IS ] { { BINARY  
LINE } SEQUENTIAL }  
RELATIVE  
INDEXED } ] ]

[ PADDING CHARACTER IS { *data-name-2* }  
 { *literal-2* } ] ]

[ RECORD DELIMITER IS { STANDARD-1 }  
 { *delimiter-name-1* } ] ]

[ ACCESS MODE IS { { SEQUENTIAL  
RANDOM  
DYNAMIC } } [ RELATIVE KEY IS *data-name-3* ] ] ]

[ LOCK MODE IS  
 { { MANUAL  
AUTOMATIC } } [ WITH LOCK ON [ MULTIPLE ] { RECORD  
RECORDS } ] ] ]  
EXCLUSIVE ] ] ]

(continued on next page)

(continued from previous page)

[ CODE-SET IS *alphabet-name-1* ]  
 [ COLLATING SEQUENCE IS *alphabet-name-2* ]  
 [ RECORD KEY IS { *data-name-4*  
                   *split-key-name-1* = { *data-name-5* }... }  
           [ WITH DUPLICATES ] ]  
 [ ALTERNATE RECORD KEY IS { *data-name-6*  
                                   *split-key-name-2* = { *data-name-7* }... }  
           [ WITH DUPLICATES ] ]...  
 [ FILE STATUS IS *data-name-8* ] .

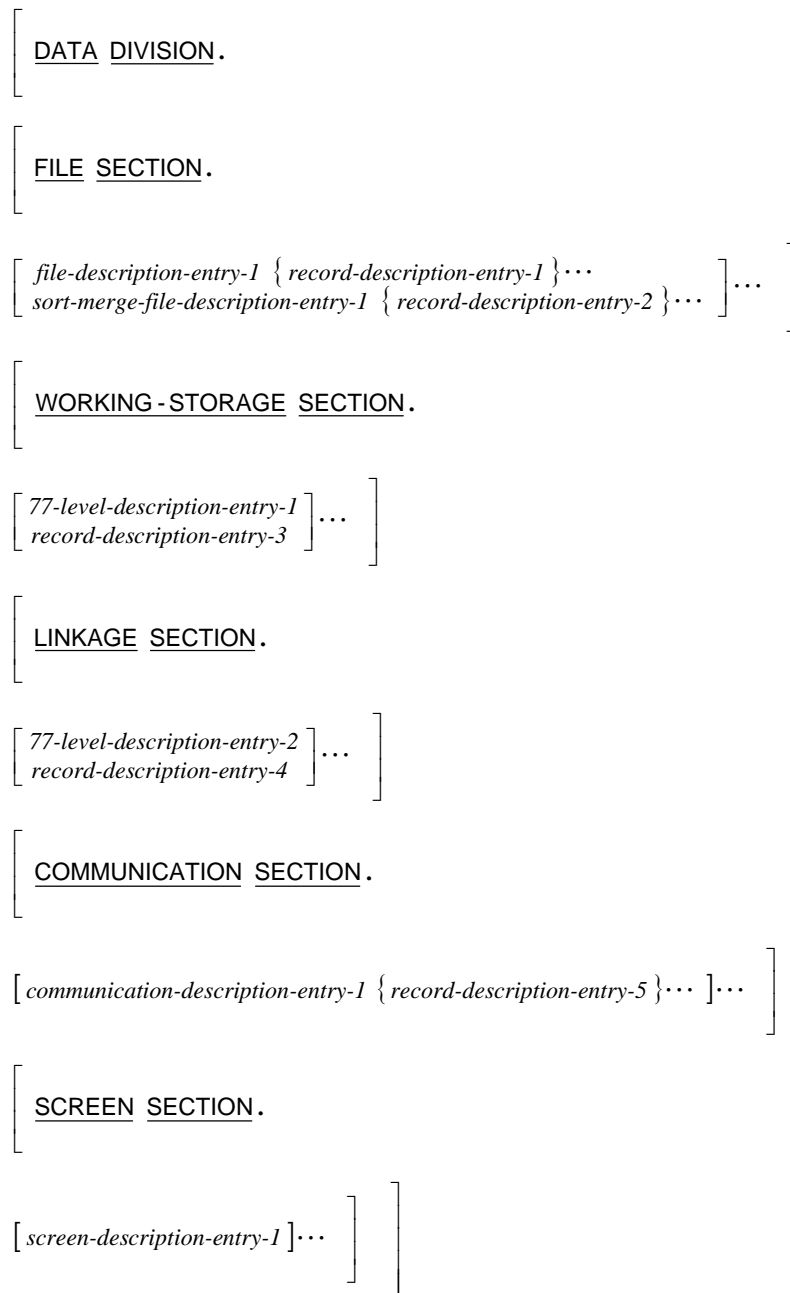
### Sort-Merge File Control Entry

SELECT *file-name-1*

ASSIGN TO { { *data-name-1* }  
                   { *literal-1* }  
           { SORT  
           { SORT-MERGE } [ *data-name-1* ]  
           { MERGE } [ *literal-1* ]  
           { *device-name-1* } } .

---

## Data Division General Format



## 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  
RECORDS ARE } { *data-name-3* } ... ] .

## record-description-entry

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

## 77-level-description-entry

data-description-entry-2

## data-description-entry

### 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 ]

[ USAGE IS ] { BINARY [ ( integer-3 ) ]  
                  COMPUTATIONAL  
                  COMP  
                  COMPUTATIONAL - 1  
                  COMP - 1  
                  COMPUTATIONAL - 3  
                  COMP - 3  
                  COMPUTATIONAL - 4 [ ( integer-3 ) ]  
                  COMP - 4 [ ( integer-3 ) ]  
                  COMPUTATIONAL - 5 [ ( integer-3 ) ]  
                  COMP - 5 [ ( integer-3 ) ]  
                  COMPUTATIONAL - 6  
                  COMP - 6  
                  DISPLAY  
                  INDEX  
                  PACKED - DECIMAL  
                  POINTER }

(continued on next page)

(continued from previous page)

$$\left[ \left[ \text{SIGN IS} \right] \left\{ \begin{array}{l} \text{LEADING} \\ \text{TRAILING} \end{array} \right\} \left[ \text{SEPARATE CHARACTER} \right] \right]$$

$$\left[ \text{OCCURS} \left\{ \begin{array}{l} \text{integer-2 TIMES} \\ \left[ \text{integer-1 TO} \right] \text{integer-2 TIMES DEPENDING ON } \text{data-name-3} \end{array} \right\} \right]$$

$$\left[ \left\{ \begin{array}{l} \text{ASCENDING} \\ \text{DESCENDING} \end{array} \right\} \text{KEY IS} \left\{ \text{data-name-4} \right\} \dots \right] \dots$$

$$\left[ \text{INDEXED BY} \left\{ \text{index-name-1} \right\} \dots \right]$$

$$\left[ \left\{ \begin{array}{l} \text{SYNCHRONIZED} \\ \text{SYNC} \end{array} \right\} \left[ \begin{array}{l} \text{LEFT} \\ \text{RIGHT} \end{array} \right] \right]$$

$$\left[ \left\{ \begin{array}{l} \text{JUSTIFIED} \\ \text{JUST} \end{array} \right\} \text{RIGHT} \right]$$

$$\left[ \text{BLANK WHEN ZERO} \right]$$

$$\left[ \text{VALUE IS } \text{literal-1} \right].$$

## Format 2: Data-Name Renames

66 *data-name-1*

$$\text{RENAMES } \text{data-name-2} \left[ \left\{ \begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right\} \text{data-name-3} \right].$$

## Format 3: Condition-Name Declaration

88 *condition-name-1*

$$\left\{ \begin{array}{l} \text{VALUE IS} \\ \text{VALUES ARE} \end{array} \right\} \left\{ \begin{array}{l} \text{literal-1} \left[ \left\{ \begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right\} \text{literal-2} \right] \\ \text{relational-operator } \text{literal-1} \end{array} \right\} \dots$$

$$\left[ \text{WHEN SET TO FALSE IS } \text{literal-3} \right].$$

## 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

```

CD cd-name-1 FOR [INITIAL] I-O
  [
    {
      [MESSAGE DATE IS data-name-1
      MESSAGE TIME IS data-name-2
      SYMBOLIC TERMINAL IS data-name-3
      TEXT LENGTH IS data-name-4
      END KEY IS data-name-5
      STATUS KEY IS data-name-6
      ]
    }
    {
      data-name-1 data-name-2 data-name-3 data-name-4
      data-name-5 data-name-6
    }
  ]
  .

```

### screen-description-entry

#### Format 1: Screen Group

```

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

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

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

  [ [USAGE IS ] DISPLAY ]

  [ [SIGN IS ] { LEADING
  TRAILING } [ SEPARATE CHARACTER ] ]

  [ AUTO
  AUTO-SKIP ]

  [ SECURE ]

  [ REQUIRED ]

  [ FULL ] .

  { screen-description-entry-1 } ...

```

## Format 2: Screen Literal

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

$\left[ \begin{array}{l} \underline{BELL} \\ \underline{BEEP} \end{array} \right]$

$\left[ \underline{BLANK} \left\{ \begin{array}{l} \underline{SCREEN} \\ \underline{LINE} \\ \underline{REMAINDER} \end{array} \right\} \right]$

$\left[ \underline{BLINK} \right]$

$\left[ \underline{ERASE} \left\{ \begin{array}{l} \underline{EOS} \\ \underline{EOL} \\ \underline{SCREEN} \end{array} \right\} \right]$

$\left[ \left[ \underline{NO} \right] \underline{HIGHLIGHT} \right]$   
 $\left[ \underline{LOWLIGHT} \right]$

$\left[ \begin{array}{l} \underline{REVERSE} \\ \underline{REVERSED} \\ \underline{REVERSE-VIDEO} \end{array} \right]$

$\left[ \underline{UNDERLINE} \right]$

$\left[ \begin{array}{l} \underline{BACKGROUND IS color-name-1} \\ \underline{BACKGROUND-COLOR IS integer-1} \end{array} \right]$

$\left[ \begin{array}{l} \underline{FOREGROUND IS color-name-2} \\ \underline{FOREGROUND-COLOR IS integer-2} \end{array} \right]$

$\left[ \underline{LINE} \left[ \begin{array}{l} \text{NUMBER IS } \left\{ \begin{array}{l} \left[ \underline{PLUS} \right] \\ + \end{array} \right\} integer-3 \\ identifier-1 \end{array} \right] \right]$

$\left[ \left\{ \begin{array}{l} \underline{COLUMN} \\ \underline{COL} \end{array} \right\} \left[ \begin{array}{l} \text{NUMBER IS } \left\{ \begin{array}{l} \left[ \underline{PLUS} \right] \\ + \end{array} \right\} integer-4 \\ identifier-2 \end{array} \right] \right]$

$\left[ \left[ \underline{VALUE IS} \right] literal-1 \right] .$



(continued from previous page)

[ { JUSTIFIED } RIGHT ]  
[ JUST ]

[ [ SIGN IS ] { LEADING } [ SEPARATE CHARACTER ] ]  
[ TRAILING ]

[ AUTO  
AUTO-SKIP ]

[ SECURE ]

[ REQUIRED ]

[ FULL ] .

# Procedure Division General Formats

## Format 1: Declaratives or Sections

$$\left[ \left[ \text{PROCEDURE DIVISION} \left\{ \left\{ \begin{array}{l} \text{USING } \{ data-name-1 \} \dots \\ \{ \text{GIVING} \\ \text{RETURNING} \} data-name-2 \end{array} \right\} \right\} \right] \right] .$$

$\left[ \text{DECLARATIVES} . \right.$   
 $\{ section-name-1 \text{ SECTION } [ segment-number-1 ] .$   
 $\quad \text{USE-statement-1} .$   
 $\left[ paragraph-name-1 . \right.$   
 $\quad [ sentence-1 ] \dots ] \dots \} \dots$   
 $\quad \text{END DECLARATIVES} . \left. \right]$   
 $\{ section-name-2 \text{ SECTION } [ segment-number-2 ] .$   
 $\left[ paragraph-name-2 . \right.$   
 $\quad [ sentence-2 ] \dots ] \dots \} \dots \left. \right]$

## Format 2: Paragraphs

$$\left[ \left[ \text{PROCEDURE DIVISION} \left\{ \left\{ \begin{array}{l} \text{USING } \{ data-name-1 \} \dots \\ \{ \text{GIVING} \\ \text{RETURNING} \} data-name-2 \end{array} \right\} \right\} \right] \right] .$$

$\{ paragraph-name-3 .$   
 $\quad [ sentence-3 ] \dots \} \dots \left. \right]$

---

# General Formats for COBOL Statements

The following sections describe the formats for COBOL statements.

## ACCEPT Statement

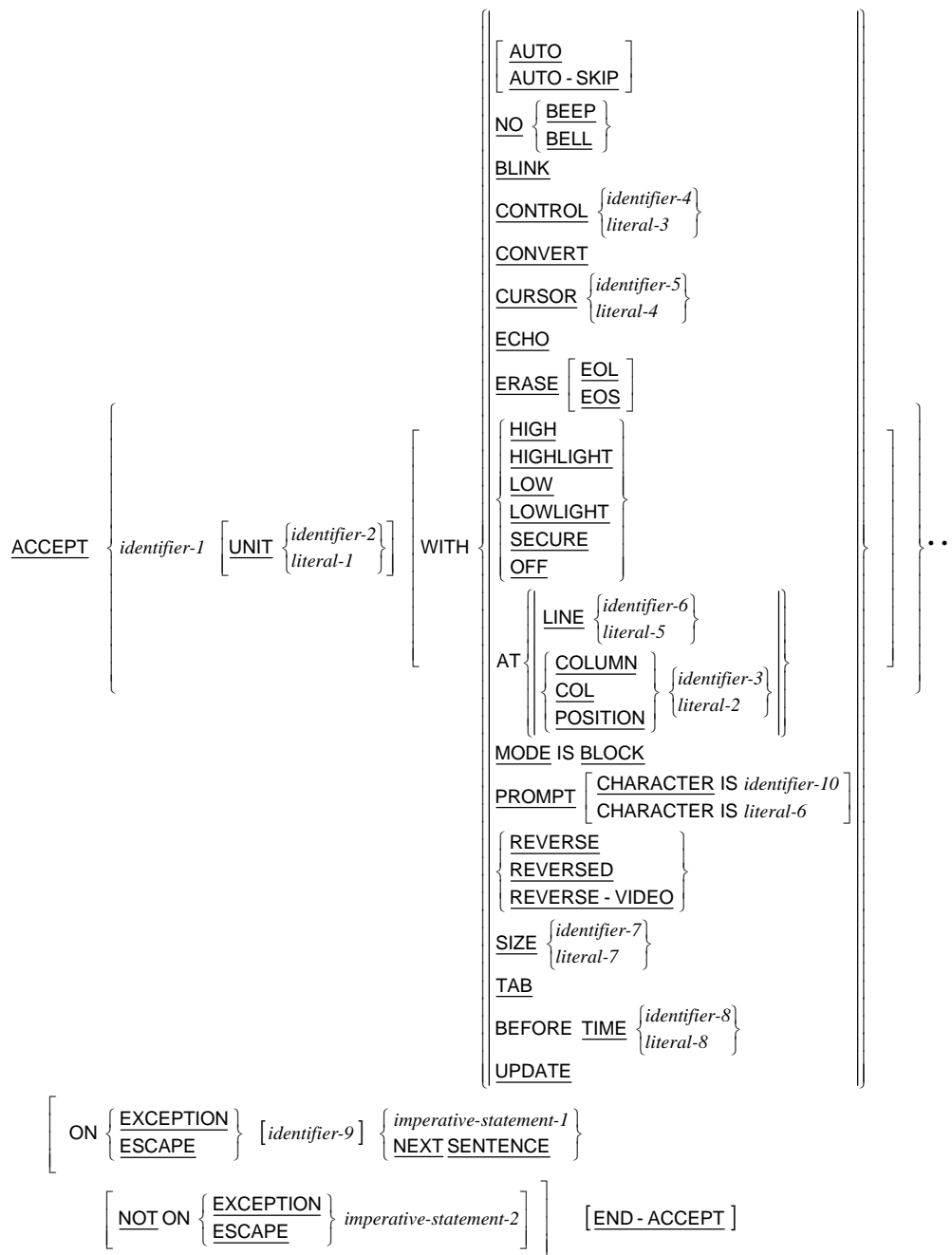
### Format 1: Accept From System-Name

ACCEPT *identifier-1* [ FROM { *mnemonic-name-3*  
*low-volume-I-O-name-1* } ] [ END - ACCEPT ]

### Format 2: Accept From Implicit Definition

ACCEPT *identifier-2* FROM { CENTURY - DATE  
CENTURY - DAY  
DATE [ YYYYMMDD ]  
DATE - AND - TIME  
DATE - COMPILED  
DAY [ YYYYDDD ]  
DAY - AND - TIME  
DAY - OF - WEEK  
ESCAPE KEY  
EXCEPTION STATUS  
TIME } [ END - ACCEPT ]

**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

ACCEPT *screen-name-1* AT  $\left\{ \begin{array}{l} \text{LINE NUMBER } \left\{ \begin{array}{l} \textit{identifier-1} \\ \textit{integer-1} \end{array} \right\} \\ \left\{ \begin{array}{l} \text{COLUMN} \\ \text{COL} \end{array} \right\} \text{NUMBER } \left\{ \begin{array}{l} \textit{identifier-2} \\ \textit{integer-2} \end{array} \right\} \end{array} \right\}$

[ ON { EXCEPTION } *imperative-statement-1* ]

[ NOT ON { EXCEPTION } *imperative-statement-2* ]

[END-ACCEPT]

### ADD Statement

#### Format 1: Add...To

ADD { *identifier-1* } ... TO { *identifier-2* [ROUNDED] } ...

[ ON SIZE ERROR *imperative-statement-1* ]

[ NOT ON SIZE ERROR *imperative-statement-2* ]

[END-ADD]

#### Format 2: Add...Giving

ADD { *identifier-1* } ... TO { *identifier-2* } ...

GIVING { *identifier-3* [ROUNDED] } ...

[ ON SIZE ERROR *imperative-statement-1* ]

[ NOT ON SIZE ERROR *imperative-statement-2* ]

[END-ADD]



### Format 3: Add Corresponding

ADD { CORRESPONDING  
CORR } *identifier-1* TO *identifier-2* [ ROUNDED ]  
 [ 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

CALL { *identifier-1*  
*literal-1* } {
   
 [ BY REFERENCE ] { *identifier-2*  
OMITTED } ...
   
USING { BY CONTENT { *identifier-2*  
*literal-2*  
OMITTED } ... } ...
   
 { *identifier-2*  
*literal-2*  
OMITTED } ...
   
 [ GIVING  
RETURNING ] *identifier-3*

[ ON OVERFLOW *imperative-statement-1* ]  
 [ END-CALL ]

### Format 2: Call...On Exception

$$\begin{array}{c}
 \text{CALL } \left\{ \begin{array}{l} \text{identifier-1} \\ \text{literal-1} \end{array} \right\} \left[ \begin{array}{l} \left[ \text{BY REFERENCE} \right] \left\{ \begin{array}{l} \text{identifier-2} \\ \text{OMITTED} \end{array} \right\} \dots \\ \text{USING } \left\{ \begin{array}{l} \text{BY CONTENT} \\ \left\{ \begin{array}{l} \text{identifier-2} \\ \text{literal-2} \\ \text{OMITTED} \end{array} \right\} \dots \end{array} \right\} \dots \\ \left\{ \begin{array}{l} \text{identifier-2} \\ \text{literal-2} \\ \text{OMITTED} \end{array} \right\} \dots \\ \left[ \text{GIVING} \right] \\ \left[ \text{RETURNING} \right] \text{identifier-3} \end{array} \right] \dots \\
 \left[ \text{ON EXCEPTION } \text{imperative-statement-1} \right] \\
 \left[ \text{NOT ON EXCEPTION } \text{imperative-statement-2} \right] \\
 \left[ \text{END-CALL} \right]
 \end{array}$$

### CALL PROGRAM Statement

$$\begin{array}{c}
 \text{CALL PROGRAM } \left\{ \begin{array}{l} \text{identifier-1} \\ \text{literal-1} \end{array} \right\} \left[ \text{USING } \left\{ \begin{array}{l} \text{identifier-2} \\ \text{literal-2} \\ \text{OMITTED} \end{array} \right\} \dots \right] \\
 \left[ \text{ON EXCEPTION } \text{imperative-statement-1} \right] \\
 \left[ \text{END-CALL} \right]
 \end{array}$$

### CANCEL Statement

$$\text{CANCEL } \left\{ \begin{array}{l} \text{identifier-1} \\ \text{literal-1} \end{array} \right\} \dots$$

### CLOSE Statement

$$\text{CLOSE } \left\{ \begin{array}{l} \text{file-name-1} \\ \left[ \begin{array}{l} \left\{ \text{REEL} \right\} \left[ \text{WITH NO REWIND} \right] \\ \left\{ \text{UNIT} \right\} \left[ \text{FOR REMOVAL} \right] \end{array} \right] \\ \text{WITH } \left\{ \begin{array}{l} \text{NO REWIND} \\ \text{LOCK} \end{array} \right\} \end{array} \right\} \dots$$

## 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

DISABLE

[	<u>INPUT</u> [ <u>TERMINAL</u> ]	]
	<u>I-O TERMINAL</u>	
	<u>OUTPUT</u>	
	<u>TERMINAL</u>	

*cd-name-1* [ WITH KEY { *identifier-1* } ]

## DISPLAY Statement

### Format 1: Display Upon System-Name

$$\underline{\text{DISPLAY}} \left\{ \begin{array}{l} \text{identifier-1} \\ \text{literal-1} \end{array} \right\} \dots \left[ \underline{\text{UPON}} \left\{ \begin{array}{l} \text{mnemonic-name-3} \\ \text{low-volume-I-O-name-1} \end{array} \right\} \right]$$

$$\left[ \text{WITH } \underline{\text{NO ADVANCING}} \right]$$

### Format 2: Display Terminal I-O

$$\underline{\text{DISPLAY}} \left\{ \begin{array}{l} \text{identifier-1} \\ \text{literal-1} \end{array} \right\} \left[ \underline{\text{UNIT}} \left\{ \begin{array}{l} \text{identifier-2} \\ \text{literal-2} \end{array} \right\} \right] \text{ WITH } \left\{ \begin{array}{l} \left\{ \begin{array}{l} \underline{\text{BEEP}} \\ \underline{\text{BELL}} \end{array} \right\} \\ \underline{\text{BLINK}} \\ \underline{\text{CONTROL}} \left\{ \begin{array}{l} \text{identifier-4} \\ \text{literal-4} \end{array} \right\} \\ \underline{\text{CONVERT}} \\ \underline{\text{ERASE}} \left[ \begin{array}{l} \underline{\text{EOL}} \\ \underline{\text{EOS}} \end{array} \right] \\ \left\{ \begin{array}{l} \underline{\text{HIGH}} \\ \underline{\text{HIGHLIGHT}} \\ \underline{\text{LOW}} \\ \underline{\text{LOWLIGHT}} \end{array} \right\} \\ \underline{\text{AT}} \left\{ \begin{array}{l} \underline{\text{LINE}} \left\{ \begin{array}{l} \text{identifier-5} \\ \text{literal-5} \end{array} \right\} \\ \left\{ \begin{array}{l} \underline{\text{COLUMN}} \\ \underline{\text{COL}} \\ \underline{\text{POSITION}} \end{array} \right\} \left\{ \begin{array}{l} \text{identifier-3} \\ \text{literal-3} \end{array} \right\} \end{array} \right\} \\ \underline{\text{MODE IS BLOCK}} \\ \left\{ \begin{array}{l} \underline{\text{REVERSE}} \\ \underline{\text{REVERSED}} \\ \underline{\text{REVERSE - VIDEO}} \end{array} \right\} \\ \underline{\text{SIZE}} \left\{ \begin{array}{l} \text{identifier-6} \\ \text{literal-6} \end{array} \right\} \end{array} \right\} \dots$$

### Format 3: Display Screen-Name

$$\underline{\text{DISPLAY}} \left\{ \text{screen-name-1} \right\} \left[ \underline{\text{AT}} \left\{ \begin{array}{l} \underline{\text{LINE NUMBER}} \left\{ \begin{array}{l} \text{identifier-1} \\ \text{integer-1} \end{array} \right\} \\ \left\{ \begin{array}{l} \underline{\text{COLUMN}} \\ \underline{\text{COL}} \end{array} \right\} \text{NUMBER} \left\{ \begin{array}{l} \text{identifier-2} \\ \text{integer-2} \end{array} \right\} \end{array} \right\} \right] \dots$$

## DIVIDE Statement

### Format 1: Divide...Into

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

### Format 2: Divide...Into...Giving

DIVIDE { *identifier-1*  
*literal-1* } INTO { *identifier-2*  
*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*  
*literal-2* } BY { *identifier-1*  
*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* }  
*literal-1* } INTO { *identifier-2* }  
*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* }  
*literal-2* } BY { *identifier-1* }  
*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 ]  
[ OUTPUT ]  
[ TERMINAL ] ] *cd-name-1* [ WITH KEY { *identifier-1* }  
*literal-1* } ]

#### ENTER Statement

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

## EVALUATE Statement

EVALUATE {
 identifier-1  
literal-1  
expression-1  
TRUE  
FALSE
} [
 ALSO {
 identifier-2  
literal-2  
expression-2  
TRUE  
FALSE
} ...

{
 {
 WHEN {
 ANY  
condition-1  
TRUE  
FALSE
}
 [
 NOT ] {
 {
 identifier-3  
literal-3  
arithmetic-expression-1
}
 [
 {
 THROUGH  
THRU
}
 {
 identifier-4  
literal-4  
arithmetic-expression-2
}
 ]
 ]
 }
 }
 }

[
 ALSO {
 ANY  
condition-2  
TRUE  
FALSE
}
 [
 NOT ] {
 {
 identifier-5  
literal-5  
arithmetic-expression-3
}
 [
 {
 THROUGH  
THRU
}
 {
 identifier-6  
literal-6  
arithmetic-expression-4
}
 ]
 ]
 }
 ]
 }
 ...

{
 imperative-statement-1
} ...

[ WHEN OTHER imperative-statement-2 ]

[ END-EVALUATE ]

## EXIT Statement

### Format 1: Exit Paragraph

EXIT

### Format 2: Exit Program

EXIT PROGRAM

### Format 3: Exit In-Line Perform

EXIT PERFORM [ CYCLE ]

### Format 4: Exit Paragraph or 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

INITIALIZE { *identifier-1* } ... [ WITH FILLER ]

[ { ALL  
*category-name* } TO VALUE ]

[ THEN REPLACING { *category-name* DATA BY { *identifier-2*  
*literal-1* } } ... ]

[ THEN TO DEFAULT ]

where category name is:

{  
ALPHABETIC  
ALPHANUMERIC  
ALPHANUMERIC-EDITED  
DATA-POINTER  
NUMERIC  
NUMERIC-EDITED  
}

## INSPECT Statement

### Format 1: Inspect...Tallying

INSPECT *identifier-1* TALLYING

{ *identifier-2* FOR { CHARACTERS [ { BEFORE  
AFTER } ] INITIAL { *identifier-4*  
*literal-2* } } ...  
{ ALL  
LEADING } { { *identifier-3*  
*literal-1* } } [ { BEFORE  
AFTER } ] INITIAL { *identifier-4*  
*literal-2* } } ... } ... }

## Format 2: Inspect...Replacing

INSPECT *identifier-1* REPLACING

$$\left\{ \begin{array}{l} \text{CHARACTERS BY } \left\{ \begin{array}{l} \text{identifier-5} \\ \text{literal-3} \end{array} \right\} \left[ \begin{array}{l} \text{BEFORE} \\ \text{AFTER} \end{array} \right] \text{INITIAL } \left\{ \begin{array}{l} \text{identifier-4} \\ \text{literal-2} \end{array} \right\} \dots \\ \left[ \begin{array}{l} \text{ALL} \\ \text{LEADING} \\ \text{FIRST} \end{array} \right] \left\{ \begin{array}{l} \text{identifier-3} \\ \text{literal-1} \end{array} \right\} \text{ BY } \left\{ \begin{array}{l} \text{identifier-5} \\ \text{literal-3} \end{array} \right\} \left[ \begin{array}{l} \text{BEFORE} \\ \text{AFTER} \end{array} \right] \text{INITIAL } \left\{ \begin{array}{l} \text{identifier-4} \\ \text{literal-2} \end{array} \right\} \dots \dots \end{array} \right\} \dots$$

## Format 3: Inspect...Tallying...Replacing

INSPECT *identifier-1* TALLYING

$$\left\{ \begin{array}{l} \text{identifier-2 FOR } \left[ \begin{array}{l} \text{CHARACTERS } \left[ \begin{array}{l} \text{BEFORE} \\ \text{AFTER} \end{array} \right] \text{INITIAL } \left\{ \begin{array}{l} \text{identifier-4} \\ \text{literal-2} \end{array} \right\} \dots \\ \left[ \begin{array}{l} \text{ALL} \\ \text{LEADING} \end{array} \right] \left\{ \begin{array}{l} \text{identifier-3} \\ \text{literal-1} \end{array} \right\} \left[ \begin{array}{l} \text{BEFORE} \\ \text{AFTER} \end{array} \right] \text{INITIAL } \left\{ \begin{array}{l} \text{identifier-4} \\ \text{literal-2} \end{array} \right\} \dots \dots \end{array} \right\} \dots \end{array} \right\} \dots$$

REPLACING

$$\left\{ \begin{array}{l} \text{CHARACTERS BY } \left\{ \begin{array}{l} \text{identifier-5} \\ \text{literal-3} \end{array} \right\} \left[ \begin{array}{l} \text{BEFORE} \\ \text{AFTER} \end{array} \right] \text{INITIAL } \left\{ \begin{array}{l} \text{identifier-4} \\ \text{literal-2} \end{array} \right\} \dots \\ \left[ \begin{array}{l} \text{ALL} \\ \text{LEADING} \\ \text{FIRST} \end{array} \right] \left\{ \begin{array}{l} \text{identifier-3} \\ \text{literal-1} \end{array} \right\} \text{ BY } \left\{ \begin{array}{l} \text{identifier-5} \\ \text{literal-3} \end{array} \right\} \left[ \begin{array}{l} \text{BEFORE} \\ \text{AFTER} \end{array} \right] \text{INITIAL } \left\{ \begin{array}{l} \text{identifier-4} \\ \text{literal-2} \end{array} \right\} \dots \dots \end{array} \right\} \dots$$

## Format 4: Inspect...Converting

INSPECT *identifier-1* CONVERTING

$$\left\{ \begin{array}{l} \text{identifier-6} \\ \text{literal-4} \end{array} \right\} \text{ TO } \left\{ \begin{array}{l} \text{identifier-7} \\ \text{literal-5} \end{array} \right\} \left[ \begin{array}{l} \text{BEFORE} \\ \text{AFTER} \end{array} \right] \text{INITIAL } \left\{ \begin{array}{l} \text{identifier-4} \\ \text{literal-2} \end{array} \right\} \dots$$

## MERGE Statement

MERGE *file-name-1*  $\left\{ \text{ON } \left[ \begin{array}{l} \text{ASCENDING} \\ \text{DESCENDING} \end{array} \right] \text{KEY } \left\{ \text{data-name-1} \right\} \dots \right\} \dots$

$\left[ \text{COLLATING SEQUENCE IS } \textit{alphabet-name-1} \right]$

USING *file-name-2*  $\left\{ \textit{file-name-3} \right\} \dots$

$\left[ \begin{array}{l} \text{OUTPUT PROCEDURE IS } \textit{procedure-name-1} \left[ \begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right] \textit{procedure-name-2} \\ \text{GIVING } \left\{ \textit{file-name-4} \right\} \dots \end{array} \right]$

## MOVE Statement

### Format 1: Move...To

MOVE { *identifier-1* }  
          { *literal-1* } } TO { *identifier-2* } ...

### Format 2: Move Corresponding

MOVE { CORRESPONDING  
          CORR } *identifier-1* TO { *identifier-2* } ...

## MULTIPLY Statement

### Format 1: Multiply...By

MULTIPLY { *identifier-1* }  
              { *literal-1* } } BY { *identifier-2* [ ROUNDED ] } ...

[ ON SIZE ERROR *imperative-statement-1* ]

[ NOT ON SIZE ERROR *imperative-statement-2* ]

[ END-MULTIPLY ]

### Format 2: Multiply...Giving

MULTIPLY { *identifier-1* }  
              { *literal-1* } } BY { *identifier-2* }  
  { *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 ]

$$\left. \begin{array}{l} \text{INPUT } \left\{ \text{file-name-1 [ WITH LOCK ] } \left[ \begin{array}{l} \text{REVERSED} \\ \text{WITH NO REWIND} \end{array} \right] \right\} \dots \\ \text{OUTPUT } \left\{ \text{file-name-2 [ WITH LOCK ] [ WITH NO REWIND ]} \right\} \dots \\ \text{I-O } \left\{ \text{file-name-3 [ WITH LOCK ]} \right\} \dots \\ \text{EXTEND } \left\{ \text{file-name-4 [ WITH LOCK ]} \right\} \dots \end{array} \right\} \dots$$

## PERFORM Statement

### Format 1: Perform (Once)

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

### Format 2: Perform...Times

PERFORM [ *procedure-name-1* [ { THROUGH / THRU } *procedure-name-2* ] ]  
{ *identifier-1* / *integer-1* } TIMES  
[ *imperative-statement-1* END-PERFORM ]

### Format 3: Perform...Until

PERFORM [ *procedure-name-1* [ { THROUGH / THRU } *procedure-name-2* ] ]  
[ WITH TEST { BEFORE / AFTER } ] UNTIL *condition-1*  
[ *imperative-statement-1* END-PERFORM ]

### Format 4: Perform...Varying

```

PERFORM [ procedure-name-1 [ { THROUGH }
      { THRU } procedure-name-2 ] ]
      [ WITH TEST { BEFORE }
      { AFTER } ]
      VARYING { identifier-2 }
      { index-name-1 } FROM { identifier-3 }
      { index-name-2 } BY { identifier-4 }
      { literal-1 }
      UNTIL condition-1
      [ AFTER { identifier-5 }
      { index-name-3 } FROM { identifier-6 }
      { index-name-4 } BY { identifier-7 }
      { literal-3 }
      UNTIL condition-2 ] ...
      [ imperative-statement-1 END-PERFORM ]

```

### 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)

SEARCH ALL *identifier-1*  
  
          [ AT END *imperative-statement-1* ]  
  
          WHEN { *data-name-1* { IS EQUAL TO } { *identifier-3*  
  { *literal-1*  
  { *arithmetic-expression-1* } } }  
                  { *condition-name-1* } }  
  
          [ AND { *data-name-2* { IS EQUAL TO } { *identifier-4*  
  { *literal-2*  
  { *arithmetic-expression-2* } } } } ... ]  
                  { *condition-name-2* } } ]  
  
          { *imperative-statement-2* }  
          { NEXT SENTENCE } }  
  
          [ END-SEARCH ]

## SEND Statement

### Format 1: Send (Simple)

SEND *cd-name-1* FROM { *identifier-1*  
*literal-1* }

### Format 2: Send (Advancing/Replacing)

SEND *cd-name-1* [ FROM { *identifier-1*  
*literal-1* } ] WITH { *identifier-2*  
ESI  
EMI  
EGI }

[ { BEFORE  
AFTER } ADVANCING { { *identifier-3*  
*integer-1* } [ LINE  
LINES ] } ] ]

[ { *mnemonic-name-2*  
PAGE } ] ]

[ REPLACING LINE ]

## SET Statement

### Format 1: Set Index

SET { { *index-name-1*  
*identifier-1* } ... TO { *index-name-2*  
*identifier-2*  
*integer-1* } } ...

### Format 2: Set Index Up/Down

SET { { *index-name-3* } ... { UP  
DOWN } BY { *identifier-3*  
*integer-2* } } ...

### Format 3: Set Switch On/Off

SET { { *mnemonic-name-1* } ... TO { ON  
OFF } } ...



#### Format 4: Set Condition-Name True/False

$$\underline{\text{SET}} \left\{ \left\{ \text{condition-name-1} \right\} \dots \underline{\text{TO}} \left\{ \begin{array}{l} \underline{\text{TRUE}} \\ \underline{\text{FALSE}} \end{array} \right\} \right\} \dots$$

#### Format 5: Set Pointer

$$\underline{\text{SET}} \left\{ \left\{ \begin{array}{l} \underline{\text{ADDRESS}} \left[ \begin{array}{l} \underline{\text{IN}} \\ \underline{\text{OF}} \end{array} \right] \text{data-name-1} \\ \text{identifier-4} \end{array} \right\} \dots \underline{\text{TO}} \left\{ \begin{array}{l} \underline{\text{ADDRESS}} \left[ \begin{array}{l} \underline{\text{IN}} \\ \underline{\text{OF}} \end{array} \right] \text{identifier-5} \\ \text{identifier-6} \\ \underline{\text{NULL}} \\ \underline{\text{NULLS}} \end{array} \right\} \right\} \dots$$

#### Format 6: Set Pointer Up/Down

$$\underline{\text{SET}} \left\{ \left\{ \begin{array}{l} \underline{\text{ADDRESS}} \left[ \begin{array}{l} \underline{\text{IN}} \\ \underline{\text{OF}} \end{array} \right] \text{data-name-1} \\ \text{identifier-4} \end{array} \right\} \dots \left\{ \begin{array}{l} \underline{\text{UP}} \\ \underline{\text{DOWN}} \end{array} \right\} \underline{\text{BY}} \left\{ \begin{array}{l} \text{identifier-7} \\ \text{integer-3} \\ \underline{\text{LENGTH}} \left[ \begin{array}{l} \underline{\text{IN}} \\ \underline{\text{OF}} \end{array} \right] \text{identifier-8} \end{array} \right\} \right\} \dots$$

### SORT Statement

$$\underline{\text{SORT}} \text{ file-name-1} \left\{ \underline{\text{ON}} \left\{ \begin{array}{l} \underline{\text{ASCENDING}} \\ \underline{\text{DESCENDING}} \end{array} \right\} \underline{\text{KEY}} \left\{ \text{data-name-1} \right\} \dots \right\} \dots$$

[ WITH DUPLICATES IN ORDER ]

[ COLLATING SEQUENCE IS *alphabet-name-1* ]

$$\left\{ \begin{array}{l} \underline{\text{INPUT PROCEDURE}} \text{ IS } \text{procedure-name-1} \left[ \left\{ \begin{array}{l} \underline{\text{THROUGH}} \\ \underline{\text{THRU}} \end{array} \right\} \text{procedure-name-2} \right] \\ \underline{\text{USING}} \left\{ \text{file-name-2} \right\} \dots \end{array} \right\}$$

$$\left\{ \begin{array}{l} \underline{\text{OUTPUT PROCEDURE}} \text{ IS } \text{procedure-name-3} \left[ \left\{ \begin{array}{l} \underline{\text{THROUGH}} \\ \underline{\text{THRU}} \end{array} \right\} \text{procedure-name-4} \right] \\ \underline{\text{GIVING}} \left\{ \text{file-name-3} \right\} \dots \end{array} \right\}$$

## START Statement

START *file-name-1* KEY { IS [NOT] LESS THAN  
IS [NOT] <  
IS EQUAL TO  
IS =  
IS [NOT] GREATER THAN  
IS [NOT] >  
IS GREATER THAN OR EQUAL TO  
IS >=  
IS LESS THAN OR EQUAL TO  
IS <=  
IS FIRST  
IS LAST } { *data-name-1*  
*split-key-name-1* }

[ WITH SIZE { *identifier-1*  
*integer-1* } ]

[ INVALID KEY *imperative-statement-1* ]

[ NOT INVALID KEY *imperative-statement-2* ]

[ END-START ]

## STOP Statement

STOP { RUN [ *identifier-1*  
*integer-1* ]  
{ *identifier-2*  
*literal-1* } }

## STRING Statement

STRING { { *identifier-1* }  
          { *literal-1* } } ... DELIMITED BY { { *identifier-2* }  
  { *literal-2* } }  
  { SIZE } } ...

INTO *identifier-3*

[ WITH POINTER *identifier-4* ]

[ ON OVERFLOW *imperative-statement-1* ]

[ NOT ON OVERFLOW *imperative-statement-2* ]

[ END-STRING ]

## SUBTRACT Statement

### Format 1: Subtract...From

SUBTRACT { { *identifier-1* }  
          { *literal-1* } } ... FROM { *identifier-3* [ ROUNDED ] } ...

[ ON SIZE ERROR *imperative-statement-1* ]

[ NOT ON SIZE ERROR *imperative-statement-2* ]

[ END-SUBTRACT ]

### Format 2: Subtract...Giving

SUBTRACT { { *identifier-1* }  
          { *literal-1* } } ... FROM { { *identifier-2* }  
  { *literal-2* } }

GIVING { *identifier-3* [ ROUNDED ] } ...

[ ON SIZE ERROR *imperative-statement-1* ]

[ NOT ON SIZE ERROR *imperative-statement-2* ]

[ END-SUBTRACT ]

### Format 3: Subtract Corresponding

SUBTRACT { CORRESPONDING  
CORR } *identifier-1* FROM *identifier-2* [ ROUNDED ]  
  
[ 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*  
*literal-1* } [ OR [ ALL ] { *identifier-3*  
*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

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

## WRITE Statement

### Format 1: Write Sequential File

```

WRITE record-name-1 [ FROM { identifier-1 }
                    { literal-1 } ]
    [ { BEFORE }
      { AFTER } } ADVANCING { { identifier-2 } [ LINE
                             { integer-1 } ] [ LINES
      TO LINE { identifier-3 } [ ON NEXT PAGE ]
              { integer-2 } }
      { mnemonic-name-2 }
      { PAGE } ]
    [ AT { END-OF-PAGE }
          { EOP } } imperative-statement-1 ]
    [ NOT AT { END-OF-PAGE }
             { EOP } } imperative-statement-2 ]
    [ END-WRITE ]
    
```

### Format 2: Write Relative and Indexed File

```

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

---

## General Format for END PROGRAM Header

$$\underline{\text{END PROGRAM}} \left[ \begin{array}{l} \textit{program-name-1} \\ \textit{literal-1} \end{array} \right].$$


---

## General Formats for COPY and REPLACE Statements

$$\underline{\text{COPY}} \left\{ \begin{array}{l} \textit{text-name-1} \\ \textit{literal-1} \end{array} \right\} \left[ \left[ \left\{ \begin{array}{l} \underline{\text{IN}} \\ \underline{\text{OF}} \end{array} \right\} \left\{ \begin{array}{l} \textit{library-name-1} \\ \textit{literal-2} \end{array} \right\} \right] \right] \left[ \underline{\text{SUPPRESS PRINTING}} \right]$$

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

### Format 1: Begin or Change Replacement

$$\underline{\text{REPLACE}} \left\{ \textit{== pseudo-text-1 ==} \underline{\text{BY}} \textit{== pseudo-text-2 ==} \right\} \dots$$

### Format 2: End Replacement

$$\underline{\text{REPLACE}} \underline{\text{OFF}}$$

# General Formats for Conditions

## Relation Condition

$$\left. \begin{array}{l} \{ identifier-1 \\ literal-1 \\ arithmetic-expression-1 \\ index-name-1 \} \end{array} \right\} relational-operator \left. \begin{array}{l} \{ identifier-2 \\ literal-2 \\ arithmetic-expression-2 \\ index-name-2 \} \end{array} \right\}$$

where the general format for the *relational-operator* is:

$$\left[ \begin{array}{l} IS [NOT] \underline{GREATER THAN} \\ IS [NOT] > \\ IS [NOT] \underline{LESS THAN} \\ IS [NOT] < \\ IS [NOT] \underline{EQUAL TO} \\ IS [NOT] = \\ IS \underline{GREATER THAN OR EQUAL TO} \\ IS >= \\ IS \underline{LESS THAN OR EQUAL TO} \\ IS <= \\ \\ IS [NOT] \underline{LIKE} \left[ \left[ \left[ \left[ \underline{TRIMMED} \left[ \begin{array}{l} \underline{RIGHT} \\ \underline{LEFT} \end{array} \right] \right] \right] \right] \right] \\ \left[ \left[ \underline{CASE - INSENSITIVE} \right] \right] \\ \left[ \left[ \underline{CASE - SENSITIVE} \right] \right] \end{array} \right] \end{array} \right]$$

## LIKE Condition (Special Case of a Relation Condition)

$$\left. \begin{array}{l} \{ identifier-1 \\ literal-1 \} \end{array} \right\} IS [NOT] \underline{LIKE} \left[ \left[ \left[ \left[ \underline{TRIMMED} \left[ \begin{array}{l} \underline{RIGHT} \\ \underline{LEFT} \end{array} \right] \right] \right] \right] \right] \left. \begin{array}{l} \{ identifier-2 \\ literal-2 \} \end{array} \right\}$$

## Class Condition

$$identifier-1 IS [NOT] \left. \begin{array}{l} \underline{NUMERIC} \\ \underline{ALPHABETIC} \\ \underline{ALPHABETIC - LOWER} \\ \underline{ALPHABETIC - UPPER} \\ class-name-1 \end{array} \right\}$$

## Sign Condition

*arithmetic-expression-1* IS [NOT] { POSITIVE  
NEGATIVE  
ZERO }

## 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 and Condition-Names

$$\left. \begin{array}{l} \{ data-name-1 \\ condition-name-1 \} \end{array} \right\} \left\{ \begin{array}{l} \left\{ \left\{ \begin{array}{l} \underline{IN} \\ \underline{OF} \end{array} \right\} data-name-2 \right\} \cdots \left[ \left\{ \begin{array}{l} \underline{IN} \\ \underline{OF} \end{array} \right\} \left\{ \begin{array}{l} file-name-1 \\ cd-name-1 \end{array} \right\} \right] \\ \left\{ \begin{array}{l} \underline{IN} \\ \underline{OF} \end{array} \right\} \left\{ \begin{array}{l} file-name-1 \\ cd-name-1 \end{array} \right\} \end{array} \right\}$$

### Format 2: Qualification for LINAGE-COUNTER

$$\underline{LINAGE - COUNTER} \left\{ \begin{array}{l} \underline{IN} \\ \underline{OF} \end{array} \right\} file-name-2$$

### Format 3: Qualification for Screen-Names

$$screen-name-1 \left\{ \left\{ \begin{array}{l} \underline{IN} \\ \underline{OF} \end{array} \right\} screen-name-2 \right\} \cdots$$

### Format 4: Qualification for Split-Key-Names

$$split-key-name-1 \left\{ \begin{array}{l} \underline{IN} \\ \underline{OF} \end{array} \right\} file-name-3$$

### Format 5: Qualification for Paragraph Names

$$paragraph-name-1 \left\{ \begin{array}{l} \underline{IN} \\ \underline{OF} \end{array} \right\} section-name-1$$

### Format 6: Qualification for Text-Names (COPY Statement)

$$text-name-1 \left\{ \begin{array}{l} \underline{IN} \\ \underline{OF} \end{array} \right\} library-name-1$$

---

## Miscellaneous Formats

### Sentence

*statement-sequence-1* .

### Statement Sequence

{ *imperative-statement-1* THEN } ... { *imperative-statement-2*  
*conditional-statement-1* }

### Subscripting

{ *data-name-1*  
*condition-name-1* } ( { *integer-1*  
{ *data-name-2*  
*index-name-1* } [ { + } *integer-2* ] } ... )

### Reference Modification

*data-name-1* ( *leftmost-character-position-1* : [ *length-1* ] )

### Identifier

*data-name-1* [ { IN } *data-name-2* ] ... [ { IN } { *file-name-1* }  
{ OF } { *cd-name-1* } ]  
[ ( { *subscript-1* } ... ) ] [ ( *leftmost-character-position-1* : [ *length-1* ] ) ]

## Special Registers

ADDRESS  $\left[ \begin{array}{c} \text{IN} \\ \text{OF} \end{array} \right] \text{identifier-1}$

COUNT  $\left[ \begin{array}{c} \text{IN} \\ \text{OF} \end{array} \right] \text{data-name-1}$

COUNT-MAX  $\left[ \begin{array}{c} \text{IN} \\ \text{OF} \end{array} \right] \text{data-name-1}$

COUNT-MIN  $\left[ \begin{array}{c} \text{IN} \\ \text{OF} \end{array} \right] \text{data-name-1}$

LENGTH  $\left[ \begin{array}{c} \text{IN} \\ \text{OF} \end{array} \right] \left\{ \begin{array}{l} \text{identifier-1} \\ \text{literal-1} \end{array} \right\}$

LINAGE-COUNTER  $\left[ \left\{ \begin{array}{c} \text{IN} \\ \text{OF} \end{array} \right\} \text{file-name-1} \right]$

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

[ NOT ]	<table style="border: none;"> <tr><td style="border: none;"><i>integer-1</i></td></tr> <tr><td style="border: none;"><u>NEXT</u></td></tr> <tr><td style="border: none;">{ <u>LENGTH</u> } OF { <i>data-name-4</i> }</td></tr> <tr><td style="border: none;">{ <u>SIZE</u> } OF { <i>literal-4</i> }</td></tr> <tr><td style="border: none;"><u>START OF</u> <i>data-name-6</i></td></tr> <tr><td style="border: none;"><u>DATE - COMPILED</u></td></tr> <tr><td style="border: none;">( <i>constant-expression-2</i> )</td></tr> </table>	<i>integer-1</i>	<u>NEXT</u>	{ <u>LENGTH</u> } OF { <i>data-name-4</i> }	{ <u>SIZE</u> } OF { <i>literal-4</i> }	<u>START OF</u> <i>data-name-6</i>	<u>DATE - COMPILED</u>	( <i>constant-expression-2</i> )	<table style="border: none;"> <tr><td style="border: none;">+</td></tr> <tr><td style="border: none;">-</td></tr> <tr><td style="border: none;">*</td></tr> <tr><td style="border: none;">/</td></tr> <tr><td style="border: none;">**</td></tr> <tr><td style="border: none;"><u>AND</u></td></tr> <tr><td style="border: none;"><u>OR</u></td></tr> <tr><td style="border: none;"><u>EXCLUSIVE OR</u></td></tr> </table>	+	-	*	/	**	<u>AND</u>	<u>OR</u>	<u>EXCLUSIVE OR</u>	[ NOT ]	<table style="border: none;"> <tr><td style="border: none;"><i>integer-2</i></td></tr> <tr><td style="border: none;"><u>NEXT</u></td></tr> <tr><td style="border: none;">{ <u>LENGTH</u> } OF { <i>data-name-5</i> }</td></tr> <tr><td style="border: none;">{ <u>SIZE</u> } OF { <i>literal-5</i> }</td></tr> <tr><td style="border: none;"><u>START OF</u> <i>data-name-7</i></td></tr> <tr><td style="border: none;"><u>DATE - COMPILED</u></td></tr> <tr><td style="border: none;">( <i>constant-expression-3</i> )</td></tr> </table>	<i>integer-2</i>	<u>NEXT</u>	{ <u>LENGTH</u> } OF { <i>data-name-5</i> }	{ <u>SIZE</u> } OF { <i>literal-5</i> }	<u>START OF</u> <i>data-name-7</i>	<u>DATE - COMPILED</u>	( <i>constant-expression-3</i> )	...
<i>integer-1</i>																											
<u>NEXT</u>																											
{ <u>LENGTH</u> } OF { <i>data-name-4</i> }																											
{ <u>SIZE</u> } OF { <i>literal-4</i> }																											
<u>START OF</u> <i>data-name-6</i>																											
<u>DATE - COMPILED</u>																											
( <i>constant-expression-2</i> )																											
+																											
-																											
*																											
/																											
**																											
<u>AND</u>																											
<u>OR</u>																											
<u>EXCLUSIVE OR</u>																											
<i>integer-2</i>																											
<u>NEXT</u>																											
{ <u>LENGTH</u> } OF { <i>data-name-5</i> }																											
{ <u>SIZE</u> } OF { <i>literal-5</i> }																											
<u>START OF</u> <i>data-name-7</i>																											
<u>DATE - COMPILED</u>																											
( <i>constant-expression-3</i> )																											

---

## General Format for Nested Source Programs

$$\left\{ \begin{array}{l} \text{IDENTIFICATION} \\ \text{ID} \end{array} \right\} \text{DIVISION.}$$

$$\text{PROGRAM-ID.} \left\{ \begin{array}{l} \textit{program-name-1} \\ \textit{literal-1} \end{array} \right\} [\text{IS } \underline{\text{INITIAL}} \text{ PROGRAM}].$$

$$[\underline{\text{ENVIRONMENT}} \text{ } \underline{\text{DIVISION}}. \textit{environment-division-content-1}]$$

$$[\underline{\text{DATA}} \text{ } \underline{\text{DIVISION}}. \textit{data-division-content-1}]$$

$$[\underline{\text{PROCEDURE}} \text{ } \underline{\text{DIVISION}}. \textit{procedure-division-content-1}]$$

$$[ \textit{nested-source-program-1} ] \cdots$$

$$\underline{\text{END}} \text{ } \underline{\text{PROGRAM}} \left[ \begin{array}{l} \textit{program-name-1} \\ \textit{literal-1} \end{array} \right].$$


---

## General Format for *nested-source-program*

$$\left\{ \begin{array}{l} \text{IDENTIFICATION} \\ \text{ID} \end{array} \right\} \text{DIVISION.}$$

$$\text{PROGRAM-ID.} \left\{ \begin{array}{l} \textit{program-name-2} \\ \textit{literal-2} \end{array} \right\} \left[ \text{IS } \left\{ \begin{array}{l} \underline{\text{COMMON}} \\ \underline{\text{INITIAL}} \end{array} \right\} \text{PROGRAM} \right].$$

$$[\underline{\text{ENVIRONMENT}} \text{ } \underline{\text{DIVISION}}. \textit{environment-division-content-2}]$$

$$[\underline{\text{DATA}} \text{ } \underline{\text{DIVISION}}. \textit{data-division-content-2}]$$

$$[\underline{\text{PROCEDURE}} \text{ } \underline{\text{DIVISION}}. \textit{procedure-division-content-2}]$$

$$[ \textit{nested-source-program-2} ] \cdots$$

$$\underline{\text{END}} \text{ } \underline{\text{PROGRAM}} \left[ \begin{array}{l} \textit{program-name-2} \\ \textit{literal-2} \end{array} \right].$$

## General Format for a Sequence of Source Programs

$$\left\{ \begin{array}{l} \text{IDENTIFICATION} \\ \text{ID} \end{array} \right\} \text{ DIVISION.}$$

$$\text{PROGRAM-ID.} \left\{ \begin{array}{l} \text{program-name-3} \\ \text{literal-3} \end{array} \right\} [\text{IS INITIAL PROGRAM}].$$

$$[\text{ENVIRONMENT DIVISION. } \textit{environment-division-content-3}]$$

$$[\text{DATA DIVISION. } \textit{data-division-content-3}]$$

$$[\text{PROCEDURE DIVISION. } \textit{procedure-division-content-3}]$$

$$[\textit{nested-source-program-3}] \dots$$

$$\text{END PROGRAM} \left\{ \begin{array}{l} \text{program-name-3} \\ \text{literal-3} \end{array} \right\} \cdot \dots$$

$$\left\{ \begin{array}{l} \text{IDENTIFICATION} \\ \text{ID} \end{array} \right\} \text{ DIVISION.}$$

$$\text{PROGRAM-ID.} \left\{ \begin{array}{l} \text{program-name-4} \\ \text{literal-4} \end{array} \right\} [\text{IS INITIAL PROGRAM}].$$

$$[\text{ENVIRONMENT DIVISION. } \textit{environment-division-content-4}]$$

$$[\text{DATA DIVISION. } \textit{data-division-content-4}]$$

$$[\text{PROCEDURE DIVISION. } \textit{procedure-division-content-4}]$$

$$[ [\textit{nested-source-program-4}] \dots ]$$

$$\left[ \text{END PROGRAM} \left[ \begin{array}{l} \text{program-name-4} \\ \text{literal-4} \end{array} \right] \right] \cdot \left[ \right]$$

# PICTURE Clause

The PICTURE clause describes a data item with a character-string composed of symbols.

## PICTURE Character-Strings

The five categories of data that can be described with a PICTURE clause<sup>1</sup> are defined as follows:

1. **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).
2. **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.
3. **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.
4. **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 applicable to the data description entry or when the data item is signed and the SEPARATE CHARACTER phrase is not specified in a SIGN clause applicable to the data description entry.
5. **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 (see Table 1 on page 59) 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.

<sup>1</sup> 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.

## PICTURE Symbols

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

- 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.
- 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.
- ,** 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 character period (“.”) will occur at the specified location in the data item.
  
- cs The currency symbol in a character-string is represented either by 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.

Table 1: PICTURE Symbol Precedence

Second Symbol	First Symbol	Non-floating Insertion Symbols								Floating Insertion Symbols						Other Symbols						
		B	0	/	,	.	{+}	{-}	{CR DB}	CS	{z*}	{z*}	{+}	{-}	CS	CS	9	A X	S	V	P	P
Non-floating Insertion Symbols	B	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X		X	
	0	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X		X		X
	/	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X		X		X
	,	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X		X		X
	.	X	X	X	X		X			X	X		X		X		X					
	{+}																					
	{-}	X	X	X	X	X				X	X	X			X	X	X			X	X	X
	{CR DB}	X	X	X	X	X				X	X	X			X	X	X			X	X	X
CS						X																
Floating Insertion Symbols	{z*}	X	X	X	X		X			X	X											
	{z*}	X	X	X	X	X	X			X	X	X								X		X
	{+}	X	X	X	X					X			X									
	{-}	X	X	X	X	X				X			X	X						X		
	CS	X	X	X	X		X								X							
	CS	X	X	X	X	X	X								X	X				X		
Other Symbols	9	X	X	X	X	X	X			X	X		X		X		X	X	X	X		X
	A X	X	X	X													X	X				
	S																					
	V	X	X	X	X		X			X	X		X		X		X		X		X	
	P	X	X	X	X		X			X	X		X		X		X		X		X	
	P						X			X										X	X	

---

## Reserved Words

The DERESERVE keyword of the COMPILER-OPTIONS configuration record, which is described in Chapter 10: *Configuration* of the *RM/COBOL User's Guide*, can be used to make a reserved word a user-defined word whenever it occurs in the source program, but then the language feature provided by the construct in which the word appears is not available for programs compiled with that particular configuration setting.

**A**

ACCEPT  
 ACCESS  
 ADD  
 ADDRESS<sup>2</sup>  
 ADVANCING  
 AFTER  
 ALL  
 ALPHABET<sup>2</sup>  
 ALPHABETIC  
 ALPHABETIC-LOWER<sup>2</sup>  
 ALPHABETIC-UPPER<sup>2</sup>  
 ALPHANUMERIC<sup>2</sup>  
 ALPHANUMERIC-EDITED<sup>2</sup>  
 ALSO<sup>2</sup>  
 ALTER  
 ALTERNATE  
 AND  
 ANY<sup>2</sup>  
 ARE  
 AREA  
 AREAS  
 ASCENDING<sup>2</sup>  
 ASSIGN  
 AT  
 AUTHOR

**B**

BEEP  
 BEFORE  
 BELL<sup>2</sup>  
 BINARY  
 BLANK  
 BLINK  
 BLOCK  
 BOTTOM<sup>2</sup>  
 BY

**C**

CALL  
 CANCEL  
 CD<sup>2</sup>  
 CENTURY-DATE<sup>2</sup>  
 CENTURY-DAY<sup>2</sup>  
 CF<sup>2</sup>  
 CH<sup>2</sup>  
 CHARACTER  
 CHARACTERS  
 CLASS<sup>2</sup>  
 CLOCK-UNITS<sup>2</sup>  
 CLOSE  
 COBOL<sup>2</sup>  
 CODE<sup>2</sup>  
 CODE-SET  
 COL<sup>2</sup>  
 COLLATING  
 COLUMN<sup>2</sup>  
 COMMA  
 COMMON<sup>2</sup>  
 COMMUNICATION<sup>2</sup>  
 COMP  
 COMP-1  
 COMP-3  
 COMP-4<sup>2</sup>  
 COMP-5<sup>2</sup>  
 COMP-6  
 COMPUTATIONAL  
 COMPUTATIONAL-1  
 COMPUTATIONAL-3  
 COMPUTATIONAL-4<sup>2</sup>  
 COMPUTATIONAL-5<sup>2</sup>  
 COMPUTATIONAL-6  
 COMPUTE  
 CONFIGURATION  
 CONTAINS  
 CONTENT<sup>2</sup>

<sup>2</sup> This word is not considered reserved if the RM/COBOL (74) 2.0 compatibility option is present in the Compile Command (see the *RM/COBOL User's Guide* for details on this option). In such cases, this word is treated as a user-defined word whenever it occurs in the source program.

CONTINUE<sup>2</sup>  
 CONTROL<sup>2</sup>  
 CONTROLS<sup>2</sup>  
 CONVERT  
 CONVERTING<sup>2</sup>  
 COPY  
 CORR  
 CORRESPONDING  
 COUNT<sup>2</sup>  
 COUNT-MAX<sup>2</sup>  
 COUNT-MIN<sup>2</sup>  
 CURRENCY  
 CURSOR<sup>2</sup>

**D**

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

**E**

ECHO  
 EGI<sup>2</sup>  
 ELSE  
 EMI<sup>2</sup>  
 ENABLE<sup>2</sup>  
 END  
 END-ACCEPT<sup>2</sup>  
 END-ADD<sup>2</sup>  
 END-CALL<sup>2</sup>  
 END-COMPUTE<sup>2</sup>  
 END-DELETE<sup>2</sup>  
 END-DIVIDE<sup>2</sup>  
 END-EVALUATE<sup>2</sup>  
 END-IF<sup>2</sup>  
 END-MULTIPLY<sup>2</sup>  
 END-OF-PAGE<sup>2</sup>  
 END-PERFORM<sup>2</sup>  
 END-READ<sup>2</sup>  
 END-RECEIVE<sup>2</sup>  
 END-RETURN<sup>2</sup>  
 END-REWRITE<sup>2</sup>  
 END-SEARCH<sup>2</sup>  
 END-START<sup>2</sup>  
 END-STRING<sup>2</sup>  
 END-SUBTRACT<sup>2</sup>  
 END-UNSTRING<sup>2</sup>  
 END-WRITE<sup>2</sup>  
 ENTER<sup>2</sup>  
 ENVIRONMENT  
 EOP<sup>2</sup>  
 EQUAL  
 ERASE  
 ERROR  
 ESCAPE<sup>2</sup>  
 ESI<sup>2</sup>  
 EVALUATE<sup>2</sup>  
 EVERY<sup>2</sup>  
 EXCEPTION  
 EXCLUSIVE<sup>2</sup>  
 EXIT  
 EXTEND  
 EXTERNAL<sup>2</sup>

**F**

FALSE<sup>2</sup>  
 FD  
 FILE  
 FILE-CONTROL  
 FILLER

<sup>2</sup> This word is not considered reserved if the RM/COBOL (74) 2.0 compatibility option is present in the Compile Command (see the *RM/COBOL User's Guide* for details on this option). In such cases, this word is treated as a user-defined word whenever it occurs in the source program.

FINAL<sup>2</sup>  
FIRST  
FIXED<sup>2</sup>  
FOOTING<sup>2</sup>  
FOR  
FROM  
FUNCTION<sup>2</sup>

**G**  
GENERATE<sup>2</sup>  
GIVING  
GLOBAL<sup>2</sup>  
GO  
GOBACK<sup>2</sup>  
GREATER  
GROUP<sup>2</sup>

**H**  
HEADING<sup>2</sup>  
HIGH  
HIGH-VALUE  
HIGH-VALUES  
HIGHLIGHT

**I**  
I-O  
I-O-CONTROL  
ID<sup>2</sup>  
IDENTIFICATION  
IF  
IN  
INDEX  
INDEXED  
INDICATE<sup>2</sup>  
INITIAL  
INITIALIZE<sup>2</sup>  
INITIATE<sup>2</sup>  
INPUT  
INPUT-OUTPUT  
INSPECT  
INSTALLATION  
INTO  
INVALID  
IS

**J**  
JUST  
JUSTIFIED

**K**  
KEY

**L**  
LABEL  
LAST<sup>2</sup>  
LEADING  
LEFT  
LENGTH<sup>2</sup>  
LESS  
LIKE<sup>2</sup>  
LIMIT<sup>2</sup>  
LIMITS<sup>2</sup>  
LINAGE<sup>2</sup>  
LINAGE-COUNTER<sup>2</sup>  
LINE  
LINE-COUNTER<sup>2</sup>  
LINES  
LINKAGE  
LOCK  
LOW  
LOWLIGHT<sup>2</sup>  
LOW-VALUE  
LOW-VALUES

**M**  
MEMORY  
MERGE<sup>2</sup>  
MESSAGE<sup>2</sup>  
MODE  
MODULES  
MOVE  
MULTIPLY

**N**  
NATIVE  
NEGATIVE<sup>2</sup>  
NEXT  
NO  
NOT  
NULL<sup>2</sup>  
NULLS<sup>2</sup>  
NUMBER<sup>2</sup>  
NUMERIC  
NUMERIC-EDITED<sup>2</sup>

<sup>2</sup> This word is not considered reserved if the RM/COBOL (74) 2.0 compatibility option is present in the Compile Command (see the *RM/COBOL User's Guide* for details on this option). In such cases, this word is treated as a user-defined word whenever it occurs in the source program.

**O**  
 OBJECT-COMPUTER  
 OCCURS  
 OF  
 OFF  
 OMITTED  
 ON  
 OPEN  
 OPTIONAL<sup>2</sup>  
 OR  
 ORDER<sup>2</sup>  
 ORGANIZATION  
 OTHER<sup>2</sup>  
 OUTPUT  
 OVERFLOW

**P**  
 PACKED-DECIMAL<sup>2</sup>  
 PADDING<sup>2</sup>  
 PAGE  
 PAGE-COUNTER<sup>2</sup>  
 PERFORM  
 PF<sup>2</sup>  
 PH<sup>2</sup>  
 PIC  
 PICTURE  
 PLUS<sup>2</sup>  
 POINTER<sup>2</sup>  
 POSITION  
 POSITIVE<sup>2</sup>  
 PRINTING<sup>2</sup>  
 PROCEDURE  
 PROCEDURES<sup>2</sup>  
 PROCEED  
 PROGRAM  
 PROGRAM-ID  
 PROMPT  
 PURGE<sup>2</sup>

**Q**  
 QUEUE<sup>2</sup>  
 QUOTE  
 QUOTES

**R**  
 RANDOM  
 RD<sup>2</sup>  
 READ  
 RECEIVE<sup>2</sup>  
 RECORD

RECORDING<sup>2</sup>  
 RECORDS  
 REDEFINES  
 REEL  
 REFERENCE<sup>2</sup>  
 REFERENCES<sup>2</sup>  
 RELATIVE  
 RELEASE<sup>2</sup>  
 REMAINDER  
 REMARKS<sup>2</sup>  
 REMOVAL<sup>2</sup>  
 RENAMES  
 REPLACE<sup>2</sup>  
 REPLACING  
 REPORT<sup>2</sup>  
 REPORTING<sup>2</sup>  
 REPORTS<sup>2</sup>  
 RERUN<sup>2</sup>  
 RESERVE  
 RESET<sup>2</sup>  
 RETURN<sup>2</sup>  
 RETURN-CODE<sup>2</sup>  
 RETURNING<sup>2</sup>  
 REVERSE  
 REVERSE-VIDEO<sup>2</sup>  
 REVERSED<sup>2</sup>  
 REWIND  
 REWRITE  
 RF<sup>2</sup>  
 RH<sup>2</sup>  
 RIGHT  
 ROUNDED  
 RUN

**S**  
 SAME  
 SCREEN<sup>2</sup>  
 SD<sup>2</sup>  
 SEARCH<sup>2</sup>  
 SECTION  
 SECURE<sup>2</sup>  
 SECURITY  
 SEGMENT<sup>2</sup>  
 SEGMENT-LIMIT<sup>2</sup>  
 SELECT  
 SEND<sup>2</sup>  
 SENTENCE  
 SEPARATE  
 SEQUENCE  
 SEQUENTIAL

<sup>2</sup> This word is not considered reserved if the RM/COBOL (74) 2.0 compatibility option is present in the Compile Command (see the *RM/COBOL User's Guide* for details on this option). In such cases, this word is treated as a user-defined word whenever it occurs in the source program.

SET  
SIGN  
SIZE  
SORT<sup>2</sup>  
SORT-MERGE<sup>2</sup>  
SOURCE<sup>2</sup>  
SOURCE-COMPUTER  
SPACE  
SPACES  
SPECIAL-NAMES  
STANDARD  
STANDARD-1  
STANDARD-2<sup>2</sup>  
START  
STATUS  
STOP  
STRING<sup>2</sup>  
SUB-QUEUE-1<sup>2</sup>  
SUB-QUEUE-2<sup>2</sup>  
SUB-QUEUE-3<sup>2</sup>  
SUBTRACT  
SUM<sup>2</sup>  
SUPPRESS<sup>2</sup>  
SYMBOLIC<sup>2</sup>  
SYNC  
SYNCHRONIZED

**T**  
TAB  
TABLE<sup>2</sup>  
TALLYING  
TAPE<sup>2</sup>  
TERMINAL<sup>2</sup>  
TERMINATE<sup>2</sup>  
TEST<sup>2</sup>  
TEXT<sup>2</sup>  
THAN  
THEN<sup>2</sup>  
THROUGH

THRU  
TIME  
TIMES  
TO  
TOP<sup>2</sup>  
TRAILING  
TRUE<sup>2</sup>  
TYPE<sup>2</sup>

**U**  
UNIT  
UNLOCK  
UNSTRING<sup>2</sup>  
UNTIL  
UP  
UPDATE  
UPON<sup>2</sup>  
USAGE  
USE  
USING

**V**  
VALUE  
VALUES  
VARIABLE<sup>2</sup>  
VARYING

**W**  
WHEN  
WHEN-COMPILED<sup>2</sup>  
WITH  
WORDS  
WORKING-STORAGE  
WRITE

**Z**  
ZERO  
ZEROES  
ZEROS

<sup>2</sup> This word is not considered reserved if the RM/COBOL (74) 2.0 compatibility option is present in the Compile Command (see the *RM/COBOL User's Guide* for details on this option). In such cases, this word is treated as a user-defined word whenever it occurs in the source program.



## Context-Sensitive Words

The words listed in Table 2 are context-sensitive words and are reserved in the specified language construct or context. If a context-sensitive word is used where the context-sensitive word is permitted in the general format, the word is treated as a keyword; otherwise, it is treated as a user-defined word.

**Table 2: Context-Sensitive Words**

Context-Sensitive Word	Language Construct or Context
AUTO <sup>3</sup>	screen description entry
AUTO-SKIP <sup>3</sup>	screen description entry ACCEPT statement
AUTOMATIC <sup>3</sup>	LOCK MODE clause
BACKGROUND <sup>3</sup>	screen description entry
BACKGROUND-COLOR <sup>3</sup>	screen description entry
CARD-PUNCH	ASSIGN clause in file control entry (device-name)
CARD-READER	ASSIGN clause in file control entry (device-name)
CASE-INSENSITIVE <sup>3</sup>	LIKE relational-operator
CASE-SENSITIVE <sup>3</sup>	LIKE relational-operator
CASSETTE	ASSIGN clause in file control entry (device-name)
CONSOLE	ASSIGN clause in file control entry (device-name) CONSOLE IS mnemonic-name clause in Special-Names paragraph (low-volume-I-O-name) CONSOLE IS CRT clause in Special-Names paragraph
CRT <sup>3</sup>	CONSOLE IS CRT clause in Special-Names paragraph CRT STATUS clause in Special-Names paragraph
CYCLE <sup>3</sup>	EXIT statement (Format 3)
DISC	ASSIGN clause in file control entry (device-name)
DISK	ASSIGN clause in file control entry (device-name)
EOL	ERASE clause in screen description entry and ERASE phrase in ACCEPT and DISPLAY statements
EOS	ERASE clause in screen description entry and ERASE phrase in ACCEPT and DISPLAY statements
BACKGROUND <sup>3</sup>	screen description entry
BACKGROUND-COLOR <sup>3</sup>	screen description entry
FULL <sup>3</sup>	screen description entry
<sup>3</sup> This word is not considered to be a context-sensitive word if the RM/COBOL (74) 2.0 compatibility option is present in the Compile Command (see the RM/COBOL User's Guide for details on this option). When that option is present, this word is treated as a user-defined word whenever it occurs in the source program.	

**Table 2: Context-Sensitive Words (Cont.)**

Context-Sensitive Word	Language Construct or Context
KEYBOARD	ASSIGN clause in file control entry (device-name)
LISTING	ASSIGN clause in file control entry (device-name)
MAGNETIC-TAPE	ASSIGN clause in file control entry (device-name)
MANUAL <sup>3</sup>	LOCK MODE clause
MULTIPLE <sup>3</sup>	LOCK MODE clause and I-O-CONTROL paragraph
PARAGRAPH <sup>3</sup>	EXIT statement (Format 4)
PREVIOUS <sup>3</sup>	READ statement (Format 1)
PRINT	ASSIGN clause in file control entry (device-name)
PRINTER	ASSIGN clause in file control entry (device-name)
PRINTER-1	ASSIGN clause in file control entry (device-name)
REQUIRED <sup>3</sup>	screen description entry
SORT-WORK	ASSIGN clause in file control entry (device-name)
TRIMMED <sup>3</sup>	LIKE relational-operator
UNDERLINE <sup>3</sup>	screen description entry
YYYYDDD <sup>3</sup>	FROM DAY phrase of ACCEPT statement (Format 2)
YYYYMMDD <sup>3</sup>	FROM DATE phrase of ACCEPT statement (Format 2)

The DERESERVE keyword of the COMPILER-OPTIONS configuration record, which is described in Chapter 10: *Configuration* of the *RM/COBOL User's Guide*, can be used to make a context-sensitive word a user-defined word whenever it occurs in the source program, but then the language feature provided by the construct in which the word appears is not available for programs compiled with that particular configuration setting.

---

# Nonreserved System-Names

## Code-Names

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 UPSI-0  
SWITCH-2 UPSI-1  
SWITCH-3 UPSI-2  
SWITCH-4 UPSI-3  
SWITCH-5 UPSI-4  
SWITCH-6 UPSI-5  
SWITCH-7 UPSI-6  
SWITCH-8 UPSI-7