HP COBOL II/XL Quick Reference Guide

HP 3000 MPE/iX Computer Systems

Edition 3

Manufacturing Part Number: 31500-90003
E0791

U.S.A. July 1991

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for direct, indirect, special, incidental or consequential damages in connection with the furnishing or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013. Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19 (c) (1,2).

Acknowledgments

UNIX is a registered trademark of The Open Group.

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304 U.S.A.

© Copyright 1991 by Hewlett-Packard Company

Printing History

New editions are complete revisions of the manual. The dates on the title page change only when a new edition is published.

The software code printed alongside the data indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition	November 1987	31500A.00.12
Second Edition	October 1988	31500A.01.06
Third Edition	July 1991	31500A.04.03

Preface

This is a quick reference guide for the HP COBOL II programming language on the MPE XL operating system. HP COBOL II is based on the ANSI COBOL'1974 and ANSI COBOL'1985 Standard X3.23-1985.

This guide is intended for programmers who have a working knowledge of COBOL. It summarizes HP COBOL II language elements and gives the syntax[REV BEG] for statements, commands, compiler directives, and COBOL functions.This manual is organized as follows:

Chapter 1	Preprocessor Commands and $CONTROL Options
Chapter 2	Program Format
Chapter 3	IDENTIFICATION DIVISION Statements
Chapter 4	ENVIRONMENT DIVISION Statements
Chapter 5	DATA DIVISION Statements
Chapter 6	PROCEDURE DIVISION Statements
Chapter 7	COBOL Reserved Word List
Appendix A	HP COBOL II/XL Compiler Commands
Appendix B	HP COBOL II/V Compiler Commands
Appendix C	MPE XL Run-Time Trap Handling
Appendix D	COBEDIT Program
Appendix E	COBOL Functions

What's New in This Release

The following lists major changes to this manual since the last edition:

* Addition of Appendix E, which describes the built-in COBOL functions recently defined by Addendum 1 of the ANSI COBOL'85 standard.

* Reorganization of the IDENTIFICATION, ENVIRONMENT, DATA, and PROCEDURE DIVISION statements (Chapters 3 through 6).

* Miscellaneous changes identified with change bars in the side margins.

Additional Documentation

More information on HP COBOL II/XL is in the following manuals:

* *HP COBOL II/XL Reference Manual* (31500-90001)
* *HP COBOL II/XL Programmer's Guide* (31500-90002)
* *HP COBOL II/XL Migration Guide* (31500-90004)

Acknowledgement

At the request of the American National Standards Institute (ANSI), the following acknowledgement is reproduced in its entirety:

> Any organization interested in reproducing the COBOL standard and specifications in whole or in part, using ideas from this document as the basis for an instruction manual or for any other purpose, is free to do so. However, all such organizations are requested to reproduce the following acknowledgement paragraphs in their entirety as part of the preface to any such publication (any organization using a short passage from this document, such as in a book review, is requested to mention "COBOL" in acknowledgement of the source, but need not quote the acknowledgement):
>
> COBOL is an industry language and is not the property of any company or group of companies, or of any organization or group of organizations.
>
> No warranty, expressed or implied, is made by any contributor or by the CODASYL Programming Language Committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.
>
> The authors and copyright holders of the copyrighted material used herein have specifically authorized the use of this material in whole or in part, in the COBOL specifications. Such authorization

extends to the reproduction and use of COBOL specifications in programming manuals or similar publications.

FLOW-MATIC (trademark of Sperry Rand Corporation), Programming for the Univac(R) I and II, Data Automation Systems copyrighted 1958, 1959, by Sperry Rand Corporation; IBM Commercial Translator Form No. F 28-8013, copyrighted 1959 by IBM, FACT, DSI 27A5260-2760, copyrighted 1960 by Minneapolis-Honeywell.

Conventions

Notation	Description
	Change bars in the margin show where substantial changes have been made to this manual since the last edition.
UPPERCASE and UNDERLINING	Within syntax statements, characters in uppercase must be entered in exactly the order shown. Uppercase words that are underlined are keywords that are always required when the clause or statement in which they appear is used in your program. Uppercase words that are *not* underlined are optional, and may be included or omitted. They have no effect on program execution and serve only to make source program listings more readable. The following example illustrates this: [FILE STATUS IS *stat-item*]. STATUS must be entered, FILE may be either included or omitted. See also "Underlining in dialog" on the following page.
italics	Within syntax statements, a word in italics represents a formal parameter, argument, or literal that you must replace with an actual value. In the following example, you must replace *filename* with the name of the file you want to release: RELEASE *filename*
punctuation	Within syntax statements, punctuation characters (other than brackets, braces, vertical parallel lines, and ellipses) must be entered exactly as shown.
{ }	Within syntax statements, when several elements within braces are stacked, you must select one. In the following equivalent examples, you select ON or OFF: SETMSG {ON } {OFF} SETMSG {ON }

{OFF}

{| |} Within syntax statements, bars in braces are choice indicators. One or more of the items within the choice indicators must be specified, but a single option may be specified only once.

[ ] Within syntax statements, brackets enclose optional elements. In the following example, brackets around ,TEMP indicate that the parameter and its delimiter are not required:

```
PURGE filename [,TEMP]
```

When several elements within brackets are stacked, you can select any one of the elements or none. In the following equivalent examples, you can select *devicename* or *deviceclass* or neither:

```
          [devicename ]
SHOWDEV [deviceclass ]
```

```
SHOWDEV [devicename  ]
        [deviceclass ]
```

Underlining in dialog When it is necessary to distinguish user input from computer output, the input is underlined. See also underlining on the previous page.

```
NEW NAME?  ALPHA
```

[ ] ... Brackets followed by a horizontal ellipsis indicate either that a previous bracketed element may be repeated zero or more times, or that elements have been omitted from the description.

```
[WITH DUPLICATES ] ...
```

The ellipsis shows that the preceding clause may be repeated indefinitely.

{ } ... Braces followed by a horizontal ellipses indicate either that the item within braces may be repeated one or more times, or that elements have been omitted from the description.

_ Within syntax statements, the space symbol _ shows a required blank. In the following example, you must separate *modifier* and *variable* with a blank:

```
SET [(modifier )]_(variable );
```

<, >, =, <=, >=, <> These symbols are used in conditional statements to represent the keywords LESS THAN, GREATER THAN, EQUAL

TO, LESS THAN OR EQUAL TO, GREATER THAN OR EQUAL TO, and NOT EQUAL TO, respectively. Although these symbols represent keywords, they are *not* underlined.

; The semicolon is used only to improve readability and is always optional.

, The comma is used only to improve readability, and is always optional.

. The period is a terminator or delimiter that is always required where shown; it must always be entered at the end of every division name, section name, paragraph name, and sentence.

^ The caret is occasionally used in examples to represent an implied decimal point in computer memory.

Shading Features that are part of the 1985 ANSI standard are shaded. They are accessible through the ANSI85 entry point.

LG200026_198 In some diagrams and tables, a number appears in the lower left corner. This number is for HP control purposes only and should not be interpreted as part of the diagram or table.

Chapter 1 Preprocessor Commands and $CONTROL Options

Table 1-1. Preprocessor Commands

Command	Purpose
$COMMENT	Writing comment lines.
$DEFINE $PREPROCESSOR	Defining and using macros.
$IF $SET	Conditionally compiling.
$INCLUDE $EDIT	File inserting, merging, and editing operations.
[REV BEG]$COPYRIGHT [REV END] $PAGE $TITLE[REV BEG] $VERSION [REV END] $CONTROL	Affecting compiler output (code and listing).

General Format

The preprocessor commands have the following format:

$*commandname* [*parameterlist*]

Parameters

commandname one of the command names shown in the list above.

parameterlist a list of parameters for a given preprocessor command. The specific parameters (if any) allowed for a given preprocessor command are listed later in this section where the command is described. A list of parameters in a command must be separated from the command by one or more spaces, and each parameter specified must be separated from any succeeding parameter by a comma optionally followed or preceded by one or more spaces.

$COMMENT

The $COMMENT command identifies comment text.

Syntax

$COMMENT [*comment-text*]

Parameters

comment-text a string containing anything you want to enter. *comment-text* requires no delimiters. It ends at the end of the line where the $COMMENT command is issued unless a continuation character is used. Use of COBOL comments, "*", is preferred.

$CONTROL

The $CONTROL command controls compilation and list options.

Syntax

$CONTROL *option* [, *optionlist*]

Parameters

optionlist one or more valid options, each separated from the preceding option by a comma and zero or more optional spaces.

option a valid option for the $CONTROL command.

These are the $CONTROL options:

ANSISORT
ANSISUB
BOUNDS
CALLINTRINSIC (1)
CHECKSYNTAX
CMCALL (1)
CODE
NOCODE
CROSSREF
NOCROSSREF
DEBUG
DIFF74
DIFF74=0BS
DIFF74=INC
DYNAMIC
ERRORS=*number*
INDEX16 (1)
INDEX32 (1)
LINES=*number*
LIST
NOLIST
LOCKING
LOCOFF
LOCON
MAP
NOMAP[REV BEG]
NLS=*options* (1) [REV END]
MIXED
NOMIXED
OPTFEATURES=*options* (1)
OPTIMIZE[=*number*] (1) [REV BEG]
POST85 (1) [REV END]
QUOTE=

RLFILE (1)
RLINIT (1)
SOURCE
NOSOURCE

STAT74
STDWARN[=*level*]
NOSTDWARN
SUBPROGRAM
SYMDEBUG
SYNC16
SYNC32
USLINIT
VALIDATE (1)
NOVALIDATE (1)
VERBS
NOVERBS
WARN
NOWARN

(1) This option is available only on HP COBOL II/XL.

The default $CONTROL options are shown below:

```
$CONTROL NOCODE, NOCROSSREF, ERRORS=100, LINES = 60, QUOTE=", LIST, LOCON, &
$ NOMAP, MIXED, SOURCE, NOSTDWARN, NOVERBS, WARN
```

NOTE For a description of other $CONTROL commands, refer to the *HP COBOL II/XL Reference Manual*.

$COPYRIGHT

The $COPYRIGHT command puts a copyright string into your object file.

Syntax

```
$COPYRIGHT [string  [, string ]...]
```

Parameter

string the data to be placed into the object file. The characters of *string* must be preceded and followed by a quotation mark. The total number of characters used in the strings is limited to 116. This includes any blanks appearing in strings, but does not include the quotation marks used to delimit the strings.

$DEFINE

The $DEFINE command defines a macro.

Syntax

```
$DEFINE macro-name =[string-text ]#
```

Parameters

macro-name the name of the macro being defined, and consists of an initial non-alphanumeric character (default is the percent sign, %), followed by an alphabetic character, followed by zero or more alphanumeric characters.

The length of the macro name may be any number of characters, but only the first fifteen are recognized by the preprocessor. Note that care must be taken to assure uniqueness of such names.

string-text can be any text you choose. However, because this text is sent to the compiler, it must be a valid COBOL statement or

sentence, with one exception. This exception is the use of formal parameters in the *string-text*.

$EDIT

The $EDIT preprocessor command can be used to bypass all records of the masterfile whose sequence fields contain a certain value, and renumber the numeric sequence fields of records in the newfile created by merging a textfile and a masterfile.

Syntax

$EDIT [*parameter=subparameter*] [, *parameter=subparameter*] [...]

Parameters

parameter either VOID, SEQNUM, NOSEQ, or INC.

subparameter either a sequence value, a sequence number, or an increment number. Which one is used depends on the parameter.

$IF

The $IF command interrogates any of the ten compilation switches.

Syntax

$IF [Xn= {ON }]
[{OFF}]

where X*n* is a compilation switch as described under the $SET command in the preceding paragraphs.

$INCLUDE

The $INCLUDE command allows you to specify an entire file to be sent, line by line, to the compiler as part of your source file.

Syntax

$INCLUDE *filename*

Parameter

filename the name of the file whose records are to be sent to the compiler.

$PAGE

The $PAGE command allows you to replace the first line of the title portion of the standard page heading in a listfile and to advance to the next logical page of the listfile.

Syntax

$PAGE [*string* [, *string*]...]

Parameter

string the data to be used in replacing the first line of the title. The characters of *string* must be preceded and followed by a quotation mark. The total number of characters used in the strings is limited to 97. This includes any blanks appearing in strings, but does not include the quotation marks used to delimit the strings.

$PREPROCESSOR

The $PREPROCESSOR command allows you to change the default characters used in macro definitions and names.

Syntax

$PREPROCESSOR *parameter=subparameter* [, *parameter=subparameter*]

Parameters

parameter one of the keywords shown below. Each may be used only once in a given $PREPROCESSOR command.

KEYCHAR specifies that the initial character of all macro names is to be *subparameter*.

PARMCHAR specifies that the initial character of all formal parameters in macro definitions is to be *subparameter*.

DELIMITER specifies that the delimiting character in a macro string-text is to be *subparameter*.

subparameter the character to be used in replacing the currently used initial character or delimiter.

$SET

The $SET command can be used to turn the ten compilation switches on or off.

Syntax

```
$SET [Xn={ON } [, Xr={ON }]...]
     [   {OFF} [     {OFF}]   ]
```

where X*n* and X*r* are compilation switches. The ten software switches are of the form, Xn, where n is an integer in the range 0 through 9.

$TITLE

The $TITLE command can be used to replace the first or second line of a title in the listfile.

Syntax

$TITLE [(*n*)] [*string* [, *string*]...]

Parameters

n specifies which line of the title is to be replaced. Thus, *n* can be either 1 or 2, and must be preceded and followed by a space. The default is 1.

string has the same format, restrictions, and use as in the $PAGE command.

$VERSION

The $VERSION command puts a version string into your object file.

Syntax

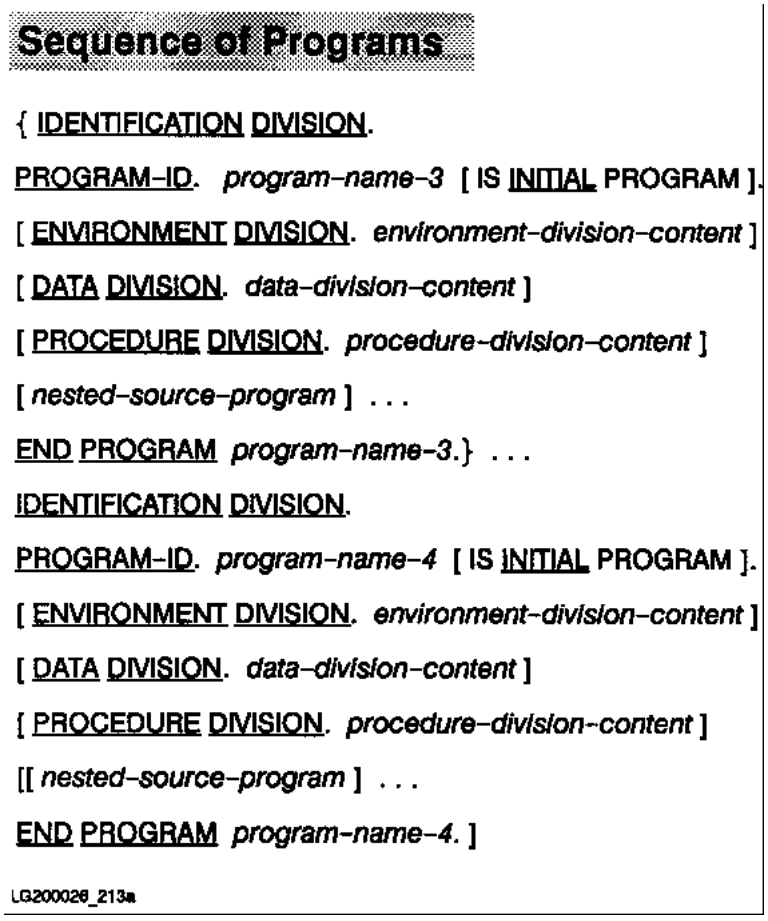
$VERSION [*string* [, *string*]...]

Parameter

string the data to be placed into the object file. The characters

of *string* must be preceded and followed by a quotation mark (") or an apostrophe ('). The total number of characters used in the strings is limited to 255. This includes any blanks appearing in strings, but does not include the quotation marks used to delimit the strings.

Chapter 2 Program Format



Nested Source Programs

IDENTIFICATION DIVISION.

PROGRAM-ID. *program-name-1* [IS INITIAL PROGRAM].

[ENVIRONMENTAL DIVISION. *environment-division-content*]

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

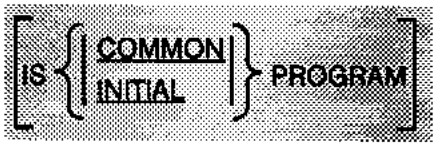
[PROCEDURE DIVISION. *procedure-division-content*]

[[*nested-source-program*] ...

END PROGRAM *program-name-1*.]

Where nested-source-program is:

IDENTIFICATION DIVISION.

PROGRAM-ID. *program-name-2* 

.

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

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

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

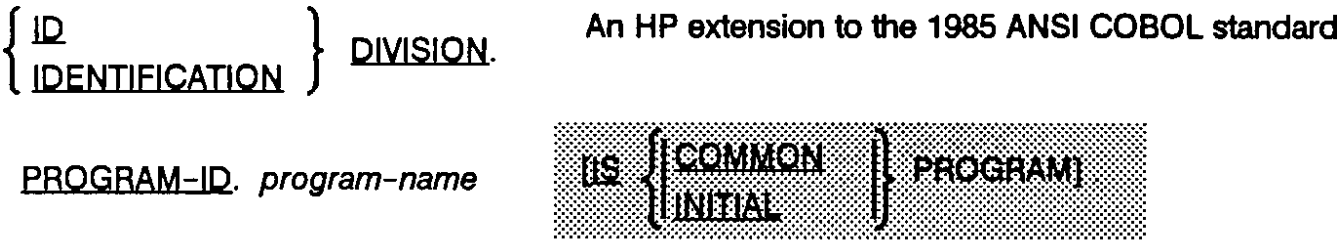
[*nested-source-program*] ...

END PROGRAM *program-name-2* .

LG200026_212a

Chapter 3 IDENTIFICATION DIVISION

The IDENTIFICATION DIVISION has the following format:



Chapter 4 ENVIRONMENT DIVISION

GENERAL FORMAT FOR ENVIRONMENT DIVISION

[ENVIRONMENT DIVISION.

[CONFIGURATION SECTION.

[SOURCE-COMPUTER. [*source-computer-entry*]]

[OBJECT-COMPUTER. [*object-computer-entry*]]

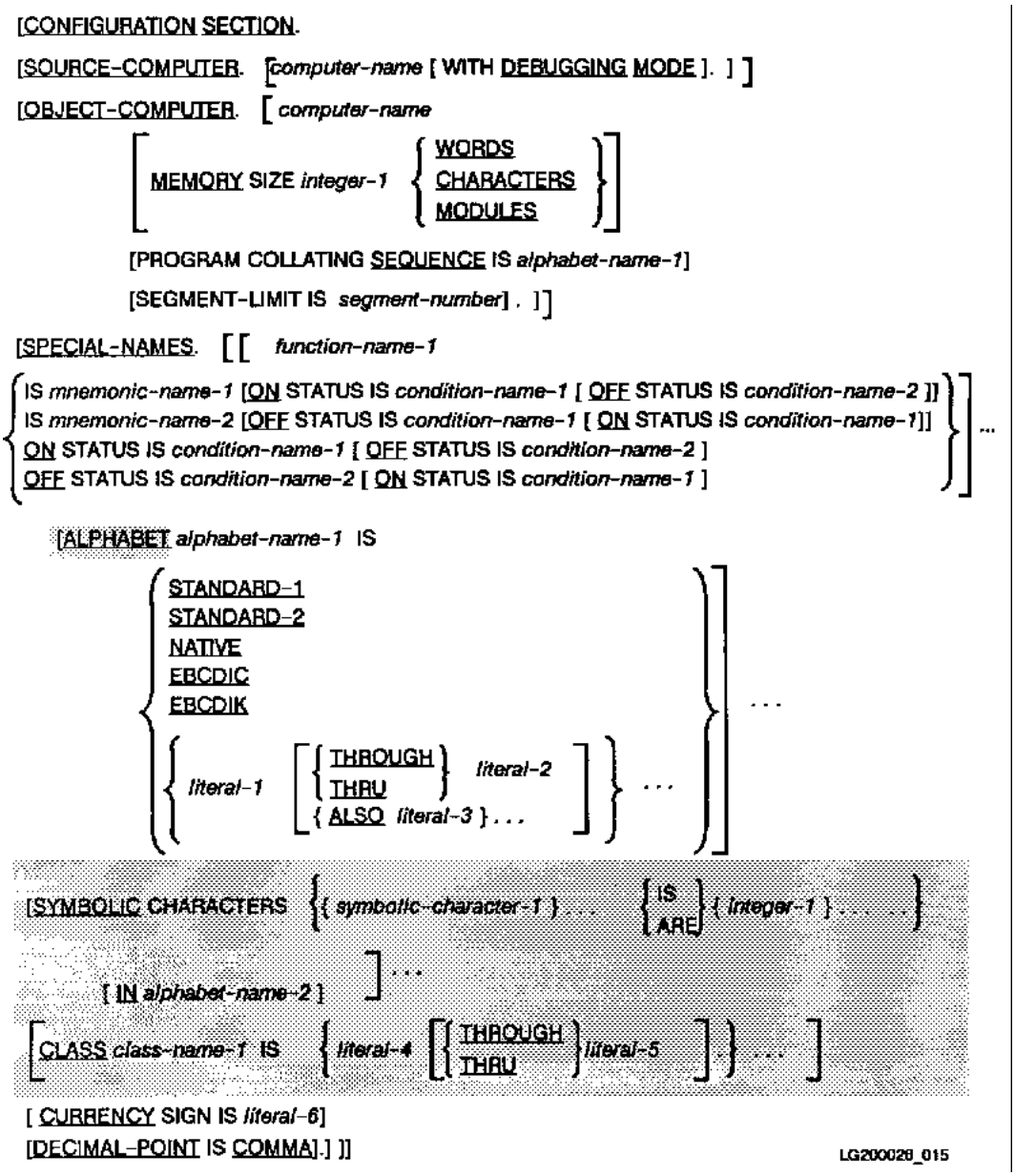
[SPECIAL-NAMES. [*special-names-entry*]]]

[INPUT-OUTPUT SECTION.

FILE-CONTROL. { file-control-entry } ...

[I-O-CONTROL. [*input-output-control-entry*]]]]

CONFIGURATION SECTION



SOURCE-COMPUTER Paragraph

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

OBJECT-COMPUTER Paragraph

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

SPECIAL-NAMES Paragraph

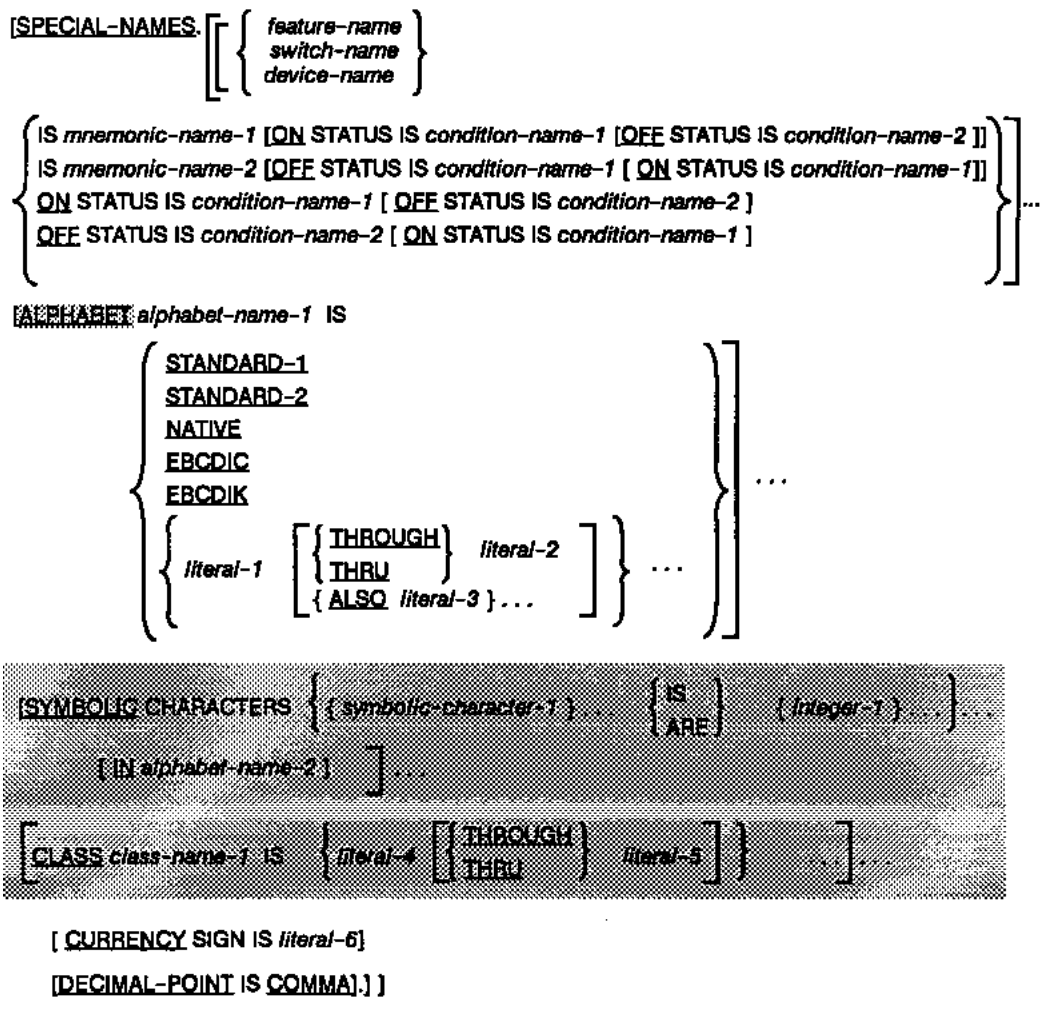


Table 4-1. HP COBOL II Feature, Switch, and Device Names

Feature Name	Function
CONDITION-CODE	Refers to condition codes returned by operating system intrinsics when they have been called through the CALL statement.
NO SPACE CONTROL	When included in the ADVANCING clause of the WRITE statement, this prevents the line printer from advancing vertically or horizontally.
TOP	When included in the ADVANCING clause of the WRITE statement, the mnemonic name assigned to TOP causes the line printer to perform a page eject.
C01 through C16	Used in the ADVANCING clause of the WRITE statement for sequential files. Each directs the line printer to skip to a particular channel (1 through 16) on the carriage control tape. Refer to the *HP COBOL II/XL Reference Manual* for details.

Table 4-1. HP COBOL II Feature, Switch, and Device Names (cont.)

Switch Name	**Function**
SW0 through SW15	Refer to software switches associated with condition names. (Software switches are described in the next section of this chapter.)
Device Name	**Function**
SYSIN	Refers to the operating system standard input device. In an interactive session, this is your terminal. In a batch job, it is either the card reader or operator's console.
SYSOUT	Refers to the operating system standard output device. In an interactive session, this is your terminal. In a batch job, it is the line printer[REV BEG]
CONSOLE	Refers to the computer operator's console (not your terminal).[REV END]

INPUT-OUTPUT SECTION

INPUT-OUTPUT SECTION Format

```
[INPUT-OUTPUT SECTION.

FILE-CONTROL.
    { file-control-entry } ...

[I-O-CONTROL.

    [[SAME [RECORD | SORT | SORT-MERGE] AREA FOR file-name-3 { file-name-4 } ...] ...

    [MULTIPLE FILE TAPE CONTAINS { file-name-5 [POSITION integer-3 ]} ... ] ... . ]]]
```

File Control Clauses

File Control Format

Format 1 – For Sequential Files

```
SELECT [OPTIONAL] file-name-1
        ASSIGN {[TO file-info-1] [USING data-name-1]}

        [RESERVE integer-1 [AREA | AREAS]]

        [[ORGANIZATION IS] SEQUENTIAL]
        [ACCESS MODE IS SEQUENTIAL]
        [FILE STATUS IS stat-item].
```

Format 2 – For Relative Files

```
SELECT [OPTIONAL] file-name-1
        ASSIGN {[TO file-info-1] [USING data-name-1]}

        [RESERVE integer-1 [AREA | AREAS]]

        [ORGANIZATION IS] RELATIVE

        [ACCESS MODE IS { SEQUENTIAL [RELATIVE KEY IS data-name-1] |
                          { RANDOM | DYNAMIC } RELATIVE KEY IS data-name-1 }]

        [FILE STATUS IS stat-item].
```

Format 3 - For Random-Access Files

```
SELECT [OPTIONAL] file-name
    ASSIGN      {[TO file-info-1] [USING data-name-1]}
    [ RESERVE integer-1   [ AREA ] ]
                          [ AREAS ]
    ACCESS MODE IS RANDOM
    ACTUAL KEY IS data-name-1
    [FILE STATUS IS stat-item].
```

Format 4 - For Indexed Files

```
SELECT [OPTIONAL] file-name-1
    ASSIGN      {[TO file-info-1] [USING data-name-1]}
    [ RESERVE integer-1   [ AREA ] ]
                          [ AREAS ]
    [ORGANIZATION IS] INDEXED
    [ ACCESS MODE IS  { SEQUENTIAL } ]
                      { RANDOM     }
                      { DYNAMIC    }
    RECORD KEY IS data-name-1 [WITH DUPLICATES]
    [ALTERNATE RECORD KEY IS data-name-2 [WITH DUPLICATES]] . . .
    [FILE STATUS IS stat-item].
```

Format 5 - For Sort-Merge Files

```
SELECT file-name-1
    ASSIGN  {[TO file-info-1] [USING data-name-1 ]}
```

Table 4-2. ANSI COBOL'85 File Status Codes

	SEQUENTIAL	RANDOM ACCESS or RELATIVE	INDEXED
SUCCESSFUL	00-Successful. No more information available. 04-READ length of record doesn't match file. 05-OPEN. Optional file not present, created. 07-File is not a tape as the OPEN/ CLOSE phrase implies.	00-Successful. No more information available. 04-READ length of record doesn't match file. 05-OPEN. Optional file not present, created.	00-Successful. No more information available. 02-READ current key = next key value -WRITE or REWRITE creates duplicate key for alternate key in which duplicates are allowed. 04-READ length of record doesn't match file. 05-OPEN. Optional file not present, created.
AT END	10-EOF or optional file not present on READ.	10-EOF or optional file not present on READ. 14-Record number too big for relative key data item on READ.	10-EOF or optional file not present on READ.
INVALID KEY		22-WRITE a duplicate key. (1) 23-Record does not exist. -START OR READ on missing optional file. 24-WRITE beyond file boundary. -Sequential WRITE record number too big for relative key data item.	21-Sequence error. 22-WRITE OR REWRITE a duplicate key. 23-Record does not exist. -START OR READ on missing optional file. 24-WRITE beyond file boundary.

Table 4-2. ANSI COBOL'85 File Status Codes (cont.)

	SEQUENTIAL	RANDOM ACCESS or RELATIVE	INDEXED
PERMANENT ERROR	30-No more information available.[REV BEG] 31-OPEN, SORT, or MERGE of dynamic file failed due to file attribute conflict.[REV END] 34-Boundary violation. 35-Nonoptional file not present for OPEN. 37-EXTEND or OUTPUT on unwritable file. -I-O for file that does not support it. -INPUT on invalid device for input. 38-OPEN on file closed with LOCK. 39-OPEN unsuccessful due to fixed file attribute conflict.	30-No more information available.[REV BEG] 31-OPEN, SORT, or MERGE of dynamic file failed due to file attribute conflict.[REV END] 35-Nonoptional file not present for OPEN. 37-EXTEND or OUTPUT on unwritable file. -I-O for file that does not support it. -INPUT on invalid device for input. 38-OPEN on file closed with LOCK. 39-OPEN unsuccessful due to fixed file attribute conflict.	30-No more information available.[REV BEG] 31-OPEN, SORT, or MERGE of dynamic file failed due to file name attribute conflict.[REV END] 35-Nonoptional file not present for OPEN. 37-EXTEND or OUTPUT on unwritable file. -I-O for file that does not support it. -INPUT on invalid device for input. 38-OPEN on file closed with LOCK. 39-OPEN unsuccessful due to fixed file attribute conflict.

Table 4-2. ANSI COBOL'85 File Status Codes (cont.)

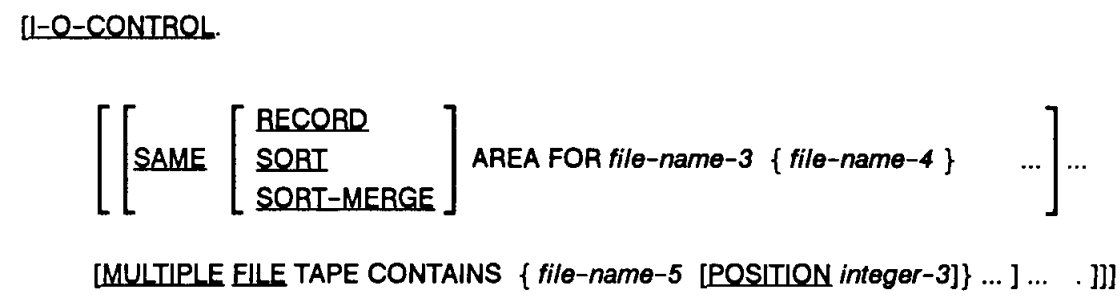
	SEQUENTIAL	RANDOM ACCESS or RELATIVE	INDEXED
LOGICAL ERROR	41-OPEN on file that is already open. 42-CLOSE for file not open. 43-No READ before REWRITE. 44-Boundary violation. -Record too big or too small. -Rewrite record not same size. 46-READ after AT END or after unsuccessful READ. 47-READ on file not open for input. 48-WRITE on file not open for output. 49-REWRITE on file not open for I-O.	41-OPEN on file that is already open. 42-CLOSE for file not open. 43-No READ before REWRITE/DELETE. 44-Boundary violation. -Record too big or too small. 46-READ after AT END or after unsuccessful READ or START. 47-READ or START on file not open for input or I-O. 48-WRITE on file not open for output or I-O. 49-REWRITE or DELETE on file not open for I-O.	41-OPEN on file that is already open. 42-CLOSE for file not open. 43-No READ before REWRITE/DELETE. 44-Boundary violation. -Record too big or too small. 46-READ after AT END or after unsuccessful READ or START. 47-READ or START on file not open for input or I-O. 48-WRITE on file not open for output or I-O. 49-REWRITE/DELETE on file not open for I-O.

(1) Does not apply to random files.

Table 4-3. ANSI COBOL'74 File Status Codes

	SEQUENTIAL	RANDOM ACCESS or RELATIVE	INDEXED
SUCCESSFUL	00-Successful. No more information available.	00-Successful. No more information available.	00-Successful. No more information available. 02-READ current key = next key value. -WRITE or REWRITE creates duplicate key for alternate key in which duplicates are allowed.
AT END	10-EOF or optional file not present.	10-EOF or optional file not present.	10-EOF or optional file not present.
INVALID KEY		22-WRITE a duplicate key. 23-Record does not exist. -START OR READ on missing optional file. 24-WRITE beyond file boundary. -Sequential WRITE record number too big for relative key data item.	21-Sequence error. 22-WRITE OR REWRITE a duplicate key. 23-Record does not exist -START OR READ on missing optional file. 24-WRITE beyond file boundary.
PERMANENT ERROR	30-No more information available. 34-Boundary violation.	30-No more information available.	30-No more information available.

I-O-CONTROL Paragraph



Chapter 5 DATA DIVISION

GENERAL FORMAT FOR DATA DIVISION

[DATA DIVISION.

[FILE SECTION.

```
[ file-description-entry  { record-description-entry } . . .                    ]  . . . .  ]
[ sort-merge-file-description-entry  { record-description-entry } . . .  ]
```

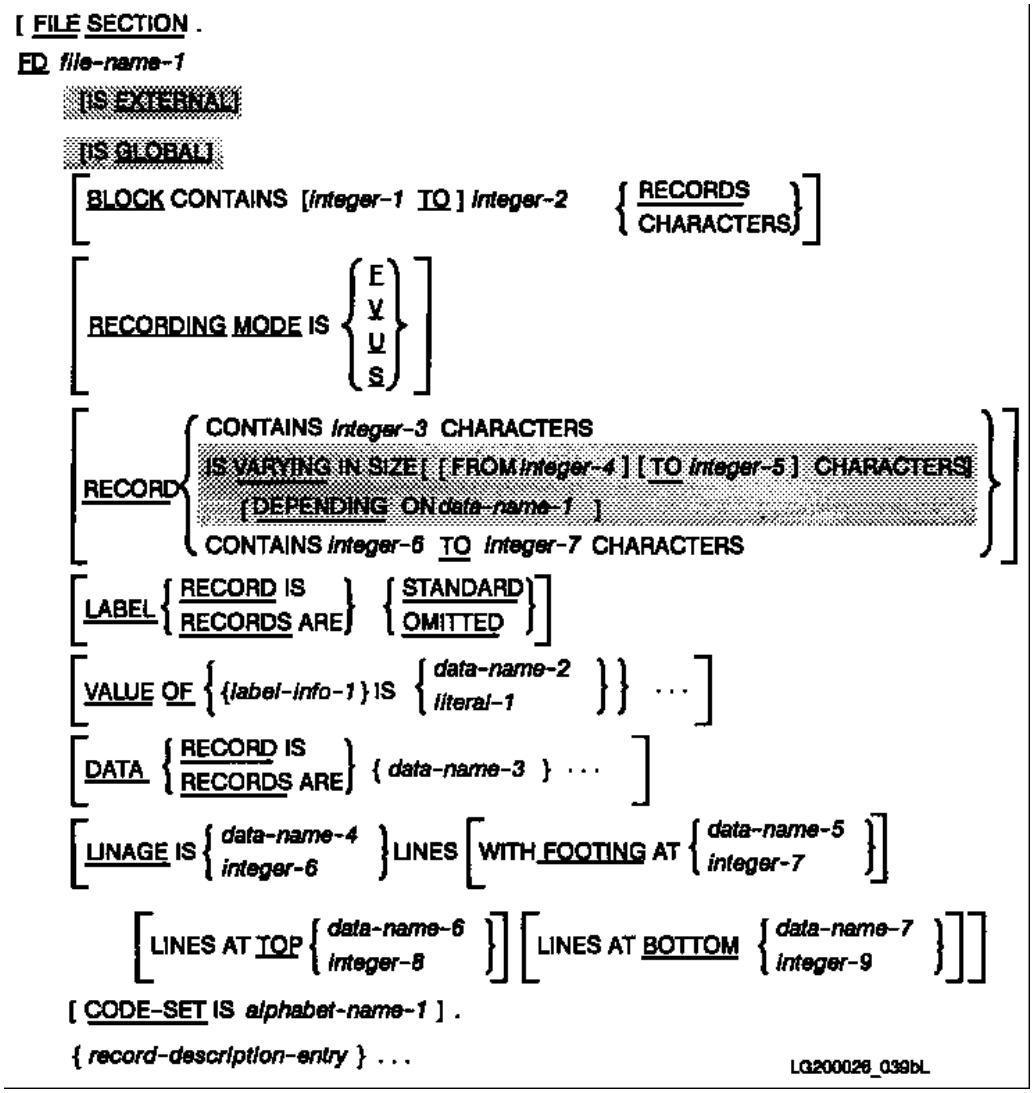
[WORKING-STORAGE SECTION.

```
[ 77-level-description-entry  ]  . . .  ]
[ record-description-entry    ]
```

[LINKAGE SECTION.

```
[ 77-level-description-entry  ]  . . .  ]]
[ record-description-entry    ]
```

FILE SECTION



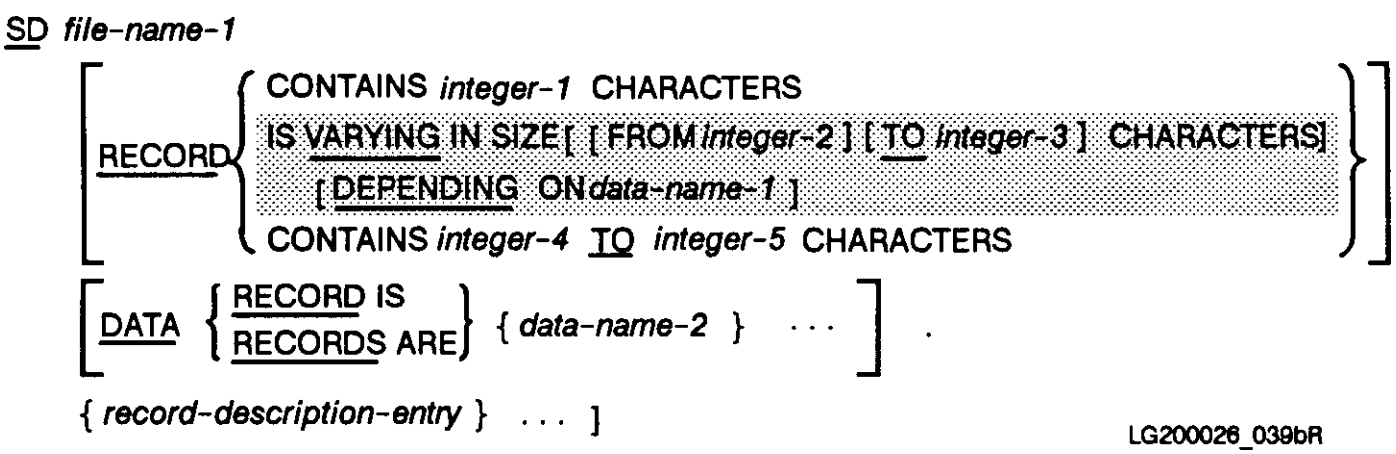
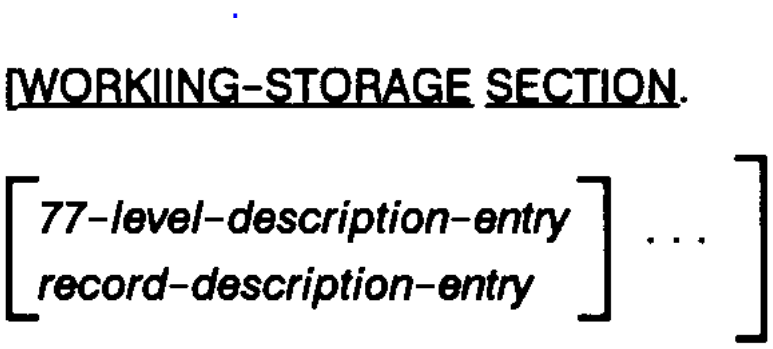


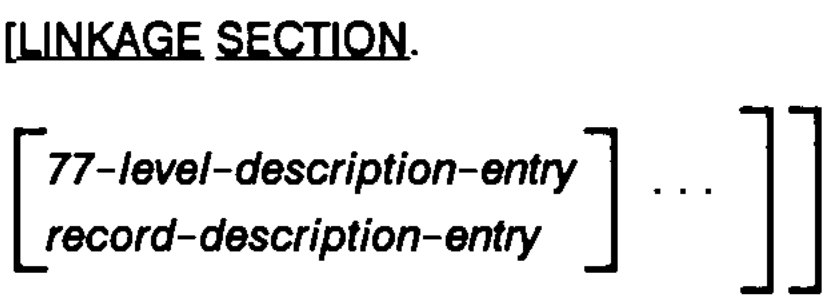
Table 5-1. Values of the LABEL INFO and DATA NAME Parameters in the VALUE OF Clause

label-info-n	Meaning	data-name-n or literal-n
VOL	Volume identification.	Any combination of one to six characters from the set A through Z, and 0 through 9.
LABELS	ANSI standard or IBM format.	ANS or IBM.
SEQ	Relative position of file on a magnetic tape.	0 to 9999, NEXT, or ADDF.
EXDATE	Date when file may be written over. Until that time, the file is protected.	Date, in the form month/day/year. The default is 00/00/00.

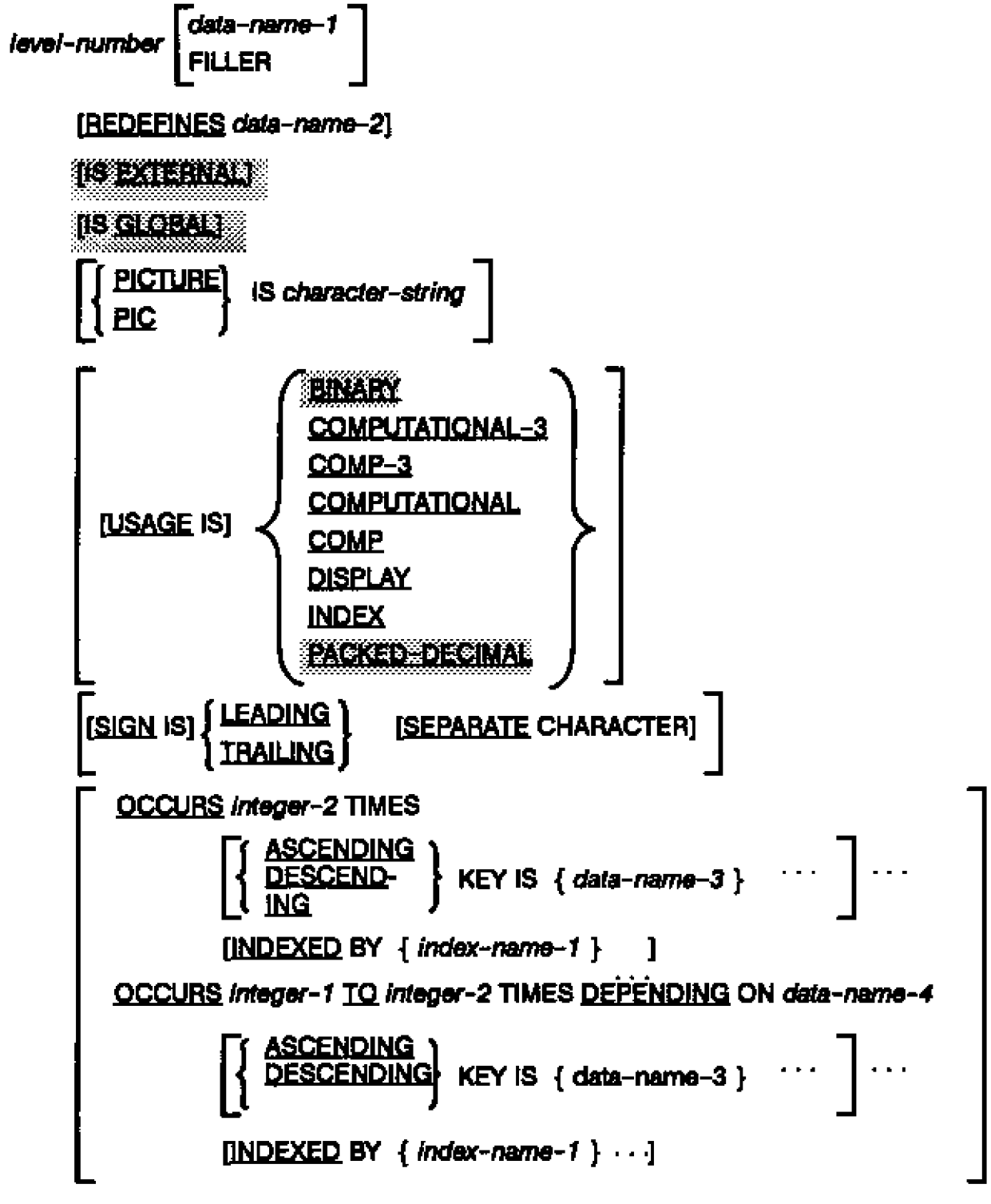
WORKING-STORAGE SECTION

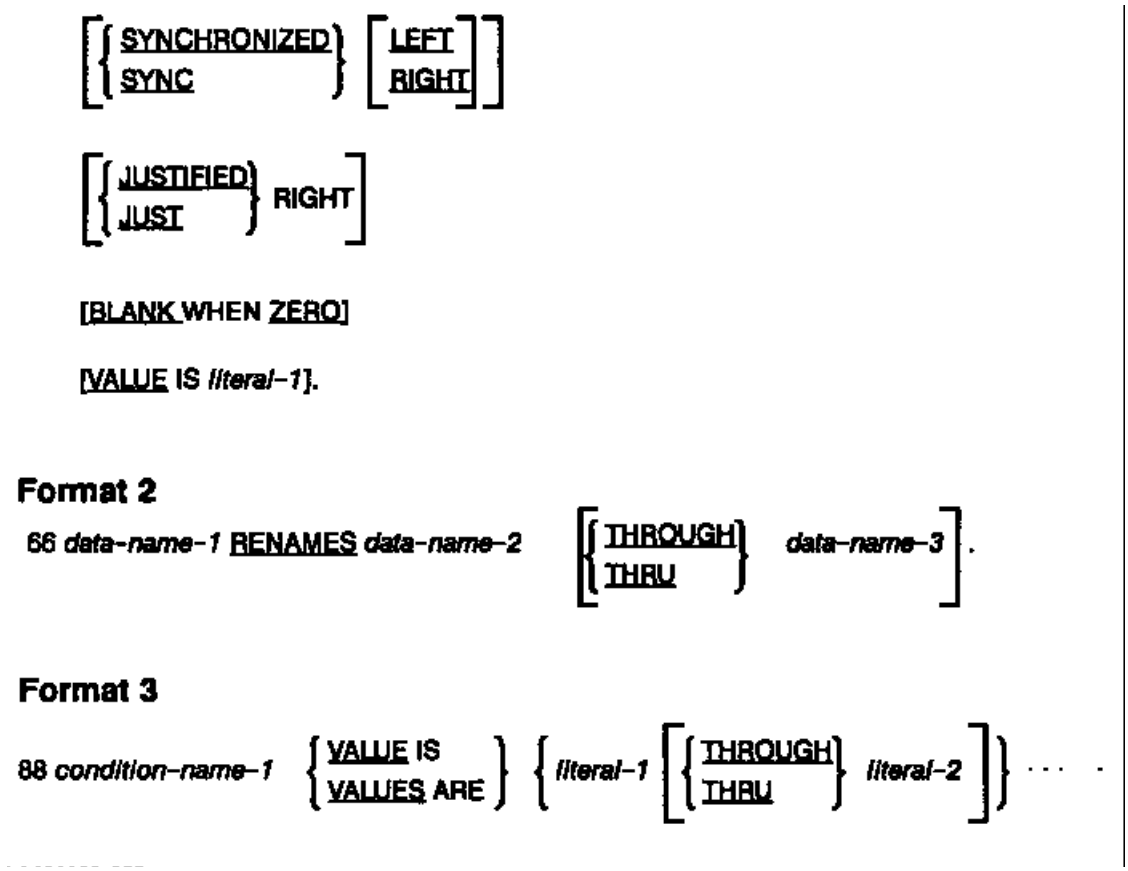


LINKAGE SECTION



Data Description Format
Format 1





The table below summarizes the type of editing permitted for each category.

Table 5-2. Allowable Types of Editing For Categories of Data Items

Category	Type of Editing
Alphabetic	Simple insertion B only.
Numeric	None.
Alphanumeric	None.
Alphanumeric-Edited	Simple insertion (0), (,), (B), and (/).
Numeric-Edited	All.

The table below shows the units digit, with the sign of its associated number represented in ASCII code.

Table 5-3. Overpunch Characters for Rightmost Digit in ASCII Coded Decimal Numbers

Units Digit	Internal Representation (ASCII)		
	Positive	Negative	No Sign
0	{	}	0
1	A	J	1
2	B	K	2
3	C	L	3
4	D	M	4
5	E	N	5
6	F	O	6
7	G	P	7
8	H	Q	8
9	I	R	9

The table below shows the number of bytes used depending on the size of a data item.

Table 5-4. Number of Bytes Used to Contain a BINARY Data Item

PICTURE	Number of Bytes
S9 to S9(4)	2
S9(5) to S9(9)	4
S9(10) to S9(18)	8

The table below shows the bit configuations used to represent signs in packed-decimal fields.

Table 5-5. COMPUTATIONAL-3 or PACKED-DECIMAL Sign Configuration

Sign	Bit Configuration	Hexadecimal Value
+	1100	C
-	1101	D
Unsigned	1111	F

Chapter 6 PROCEDURE DIVISION

The PROCEDURE DIVISION has the following format:

Format 1

```
[ PROCEDURE DIVISION  [ USING { data-name-1 } ... ] .

{ paragraph-name.
      [ sentence ] ... } ... ]
```

Format 2

```
[ PROCEDURE DIVISION  [ USING { data-name-1 } ... ] .

[DECLARATIVES.

{ section-name SECTION  [segment-number].
        USE statement
[paragraph-name .
        [ sentence ]  ...  ] ...  } ...

  END DECLARATIVES . ]

  { section-name  SECTION [segment-number].
  [ paragraph-name .
  [ sentence ]  ... ]  ...  }  ...  ]
```

Imperative Statements and Sentences

The following table lists verbs used in forming imperative statements.

Table 6-1. Imperative Verbs

ACCEPT(1) ON INPUT ERROR	EXCLUSIVE	RELEASE
ADD(2) ON SIZE ERROR	EXAMINE	REWRITE(3) INVALID KEY
ALTER	EXIT	SET
CALL(4) ON OVERFLOWON EXCEPTION	GO TO	SORT
CANCEL	GOBACK	START (3)
CLOSE	INITIALIZE	STOP
COMPUTE(2)	INSPECT	STRING (5) ON OVERFLOW
CONTINUE	MERGE	SUBTRACT (2)
DELETE (3)	MOVE	TERMINATE
DISPLAY	MULTIPLY (2)	UN-EXCLUSIVE
DIVIDE (2)	OPEN	UNSTRING (5)
ENTER	PERFORM	WRITE (6) INVALID KEY END-OF-PAGE
EVALUATE	READ (7) AT END INVALID KEY	

(1) Without the optional and NOT ON INPUT ERROR phrase.

(2) Without the optional and NOT ON SIZE ERROR phrases.

(3) Without the optional and NOT INVALID KEY phrases.

(4) Without the optional , , and NOT ON EXCEPTION phrases.

(5) Without the optional and NOT ON OVERFLOW phrases.

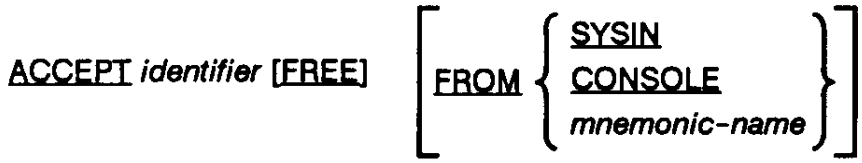
(6) Without the optional , NOT INVALID KEY, , and NOT AT END-OF-PAGE phrases.

(7) Without the optional , NOT AT END, , and NOT INVALID KEY phrases.

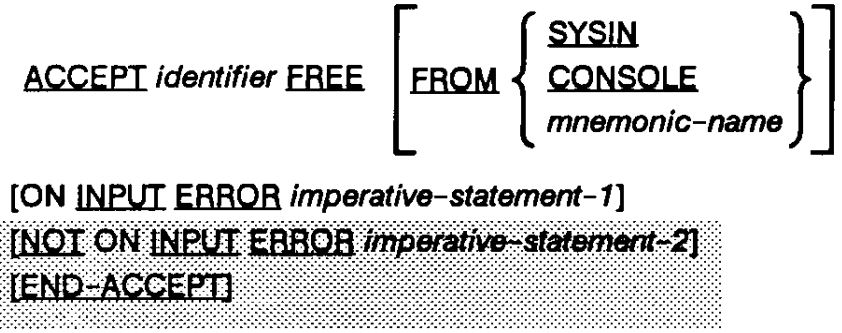
ACCEPT Statement

GENERAL FORMAT FOR VERBS

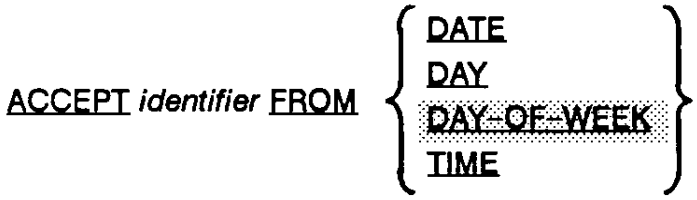
Format 1



Format 2



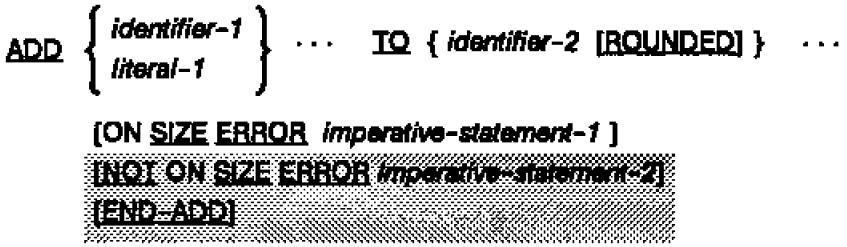
Format 3



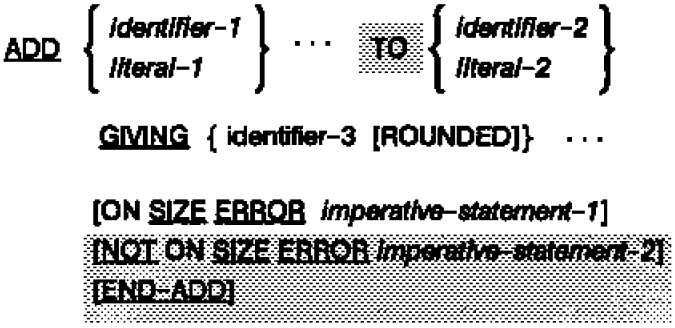
LG200026_090

ALTER Statement

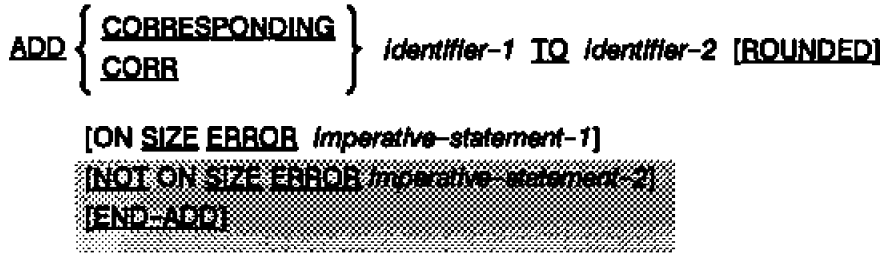
Format 1



Format 2



Format 3



ALTER Statement

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

CALL Statement Format (ANSI COBOL '85)

Format 1

[ON OVERFLOW *imperative-statement-1*

[END-CALL]]

Format 2

[ON EXCEPTION *imperative-statement-1*]
[NOT ON EXCEPTION *imperative-statement-2*]
[END-CALL]

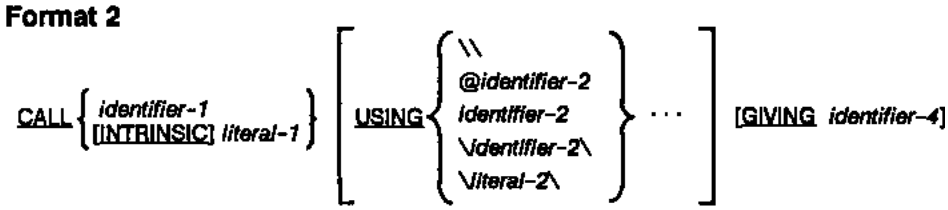
CALL Statement Format (An HP Extension to the 1985 ANSI COBOL Standard)

Format 1

[ON OVERFLOW *imperative-statement-1*

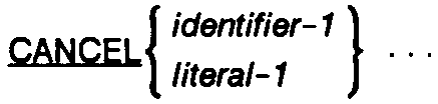
[END-CALL]

Format 2

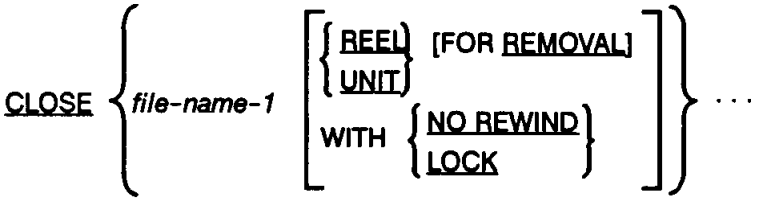


[ON EXCEPTION *imperative-statement-1*]
[NOT ON EXCEPTION *imperative-statement-2*]
[END-CALL]

CANCEL Statement



CLOSE Statement



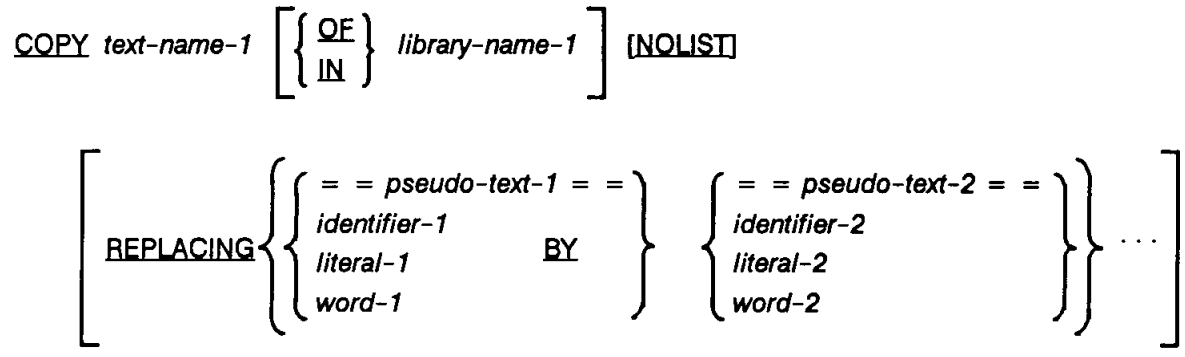
COMPUTE Statement

```
COMPUTE { identifier-1 [ROUNDED] } ... = arithmetic-expression

    [ON SIZE ERROR imperative-statement-1]
    [NOT ON SIZE ERROR imperative-statement-2]
    [END-COMPUTE]
```

CONTINUE Statement

```
CONTINUE
```

COPY Statement



DELETE Statement

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

DISPLAY Statement

```
DISPLAY { identifier-1 } ...  [ UPON { SYSOUT      } ]  [WITH NO ADVANCING]
        { literal-1    }              { CONSOLE     }
                                      { mnemonic-name }
```

DIVIDE Statement

Format 1

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

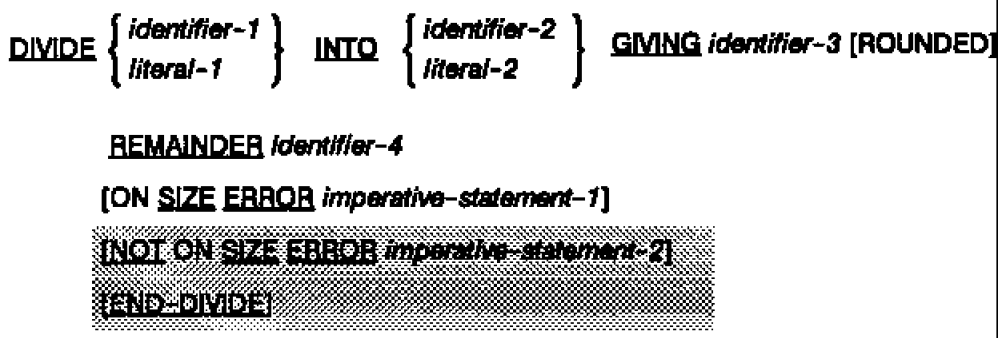
Format 2

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

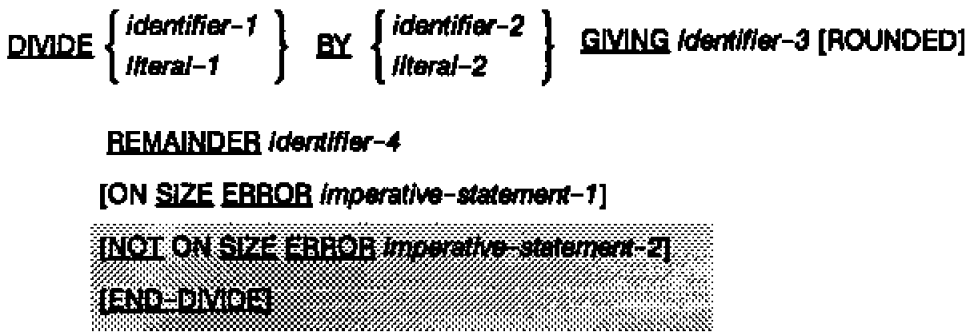
Format 3

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

Format 4



Format 5



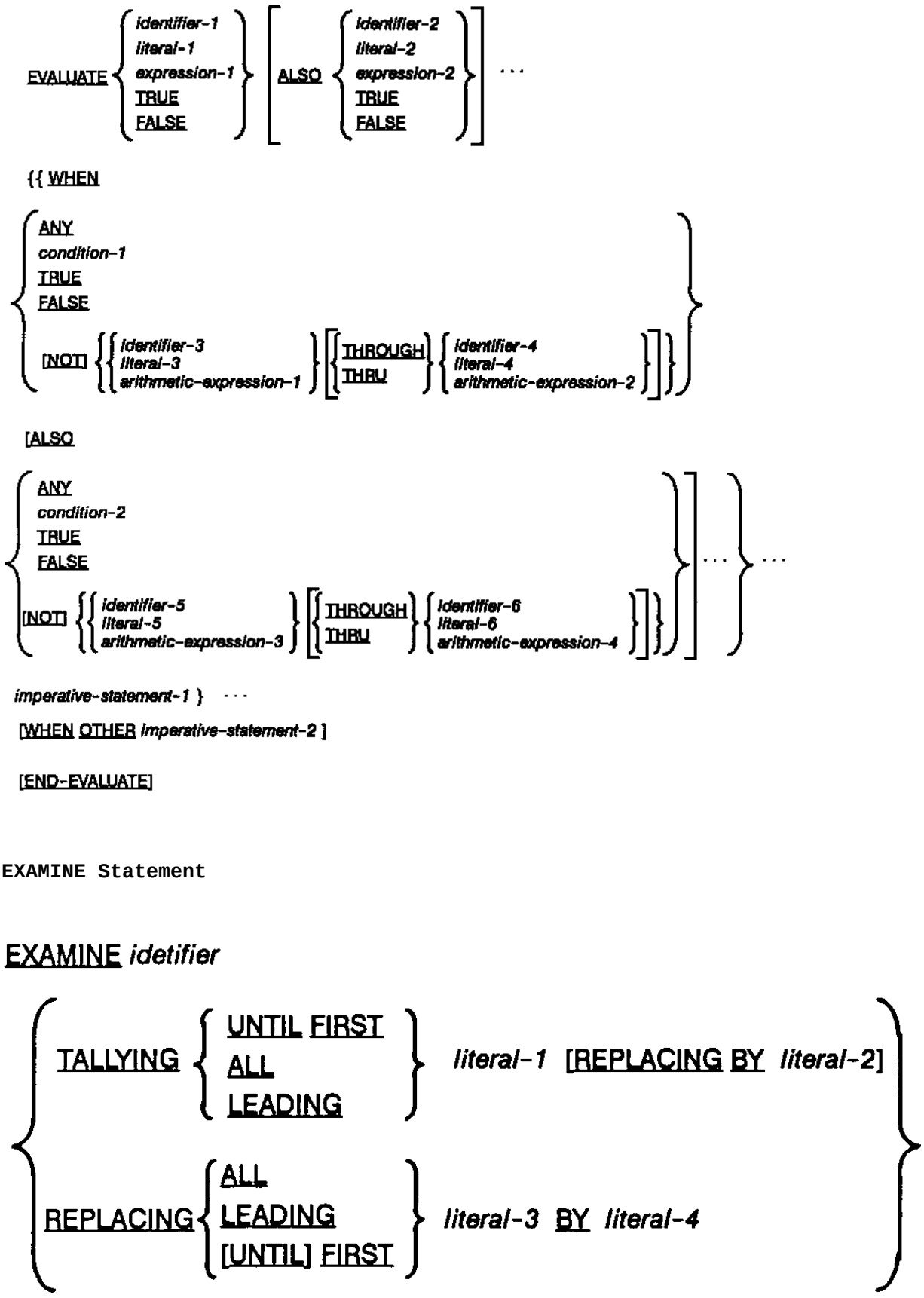
ENTER Statement

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

ENTRY Statement

```
ENTRY  literal-1  [USING  {data-name-1 }...]
```

EVALUATE Statement



EXAMINE Statement

EXCLUSIVE Statement

```
EXCLUSIVE  file-name  [CONDITIONALLY ]
```

EXIT Statement

```
paragraph-name.
    EXIT.
paragraph/section-name.
```

EXIT PROGRAM Statement

```
EXIT PROGRAM
```

GOBACK Statement

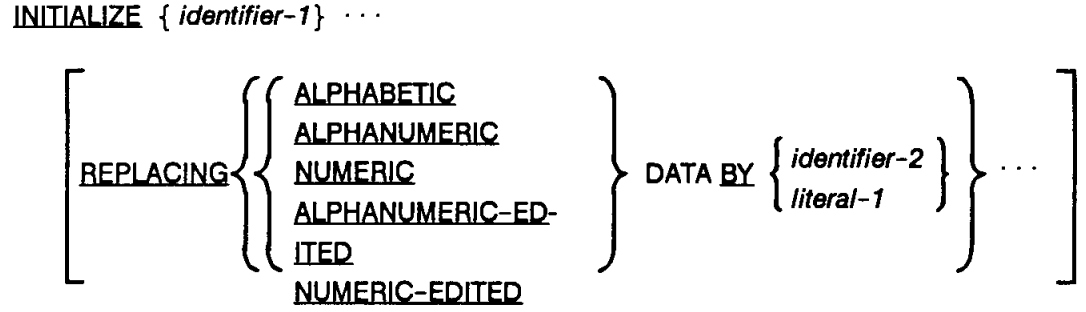
```
GOBACK
```

GO TO Statement

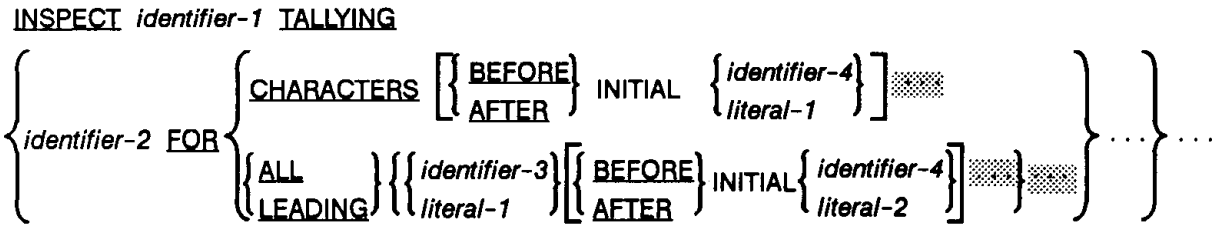
```
GO  TO [procedure-name-1 ]
GO  TO {procedure-name-1 }...DEPENDING  ON identifier-1
```

IF Statement

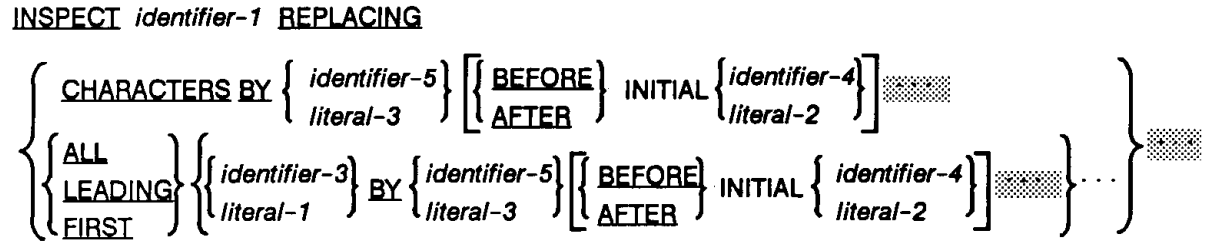
INITIALIZE Statement



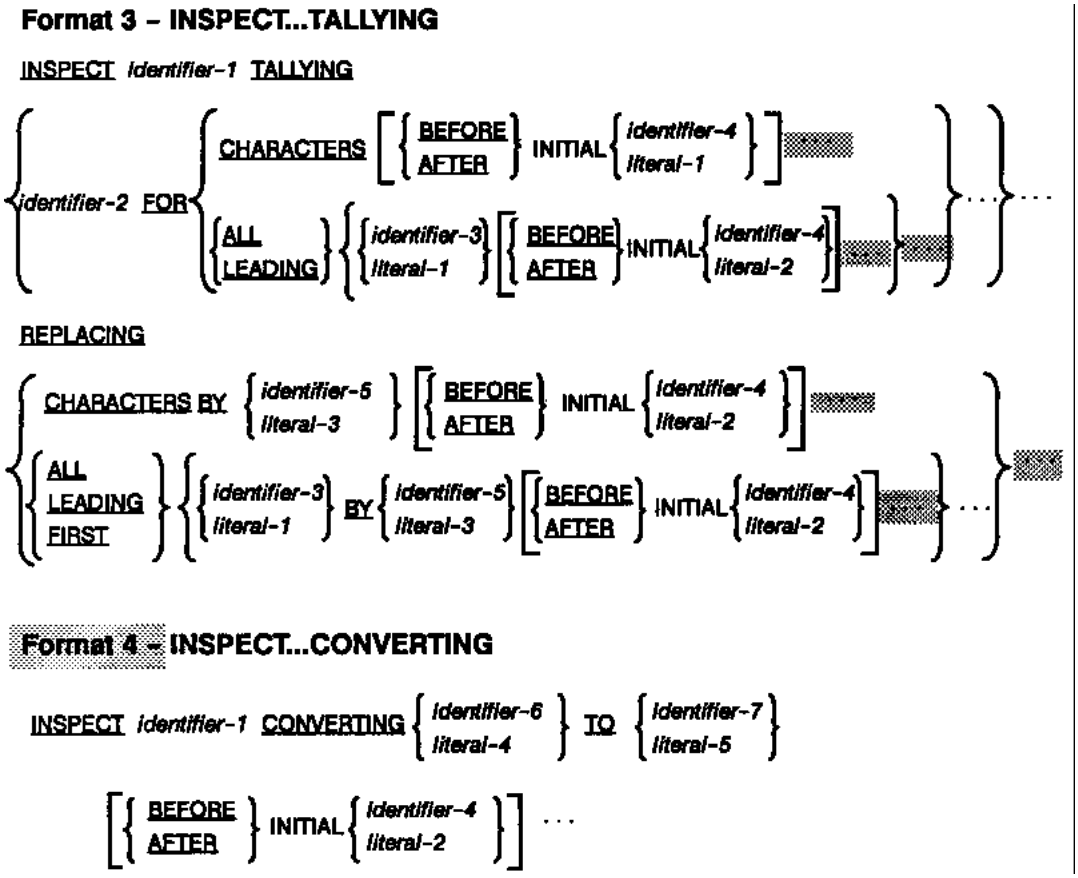
Format 1 – INSPECT...TALLYING



Format 2 – INSPECT...REPLACING

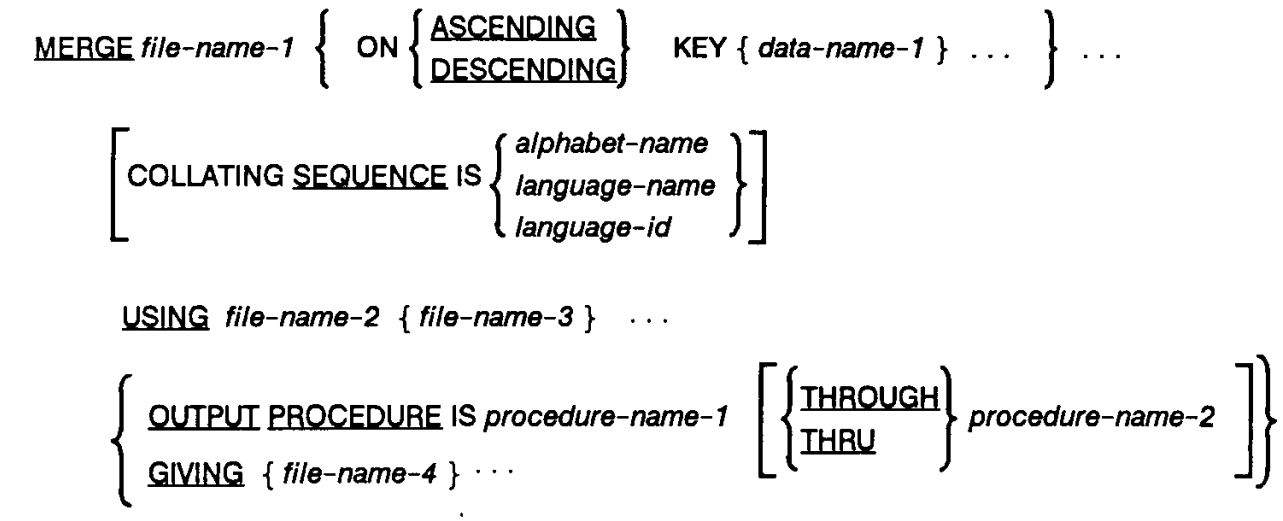


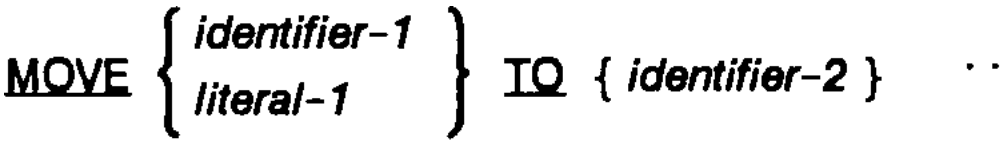
Format 3 – INSPECT...TALLYING



Format 4 – INSPECT...CONVERTING

MERGE Statement Format





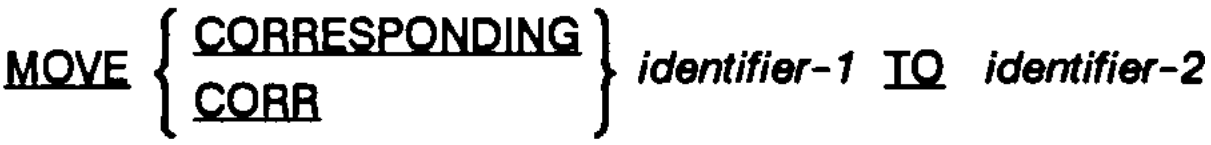


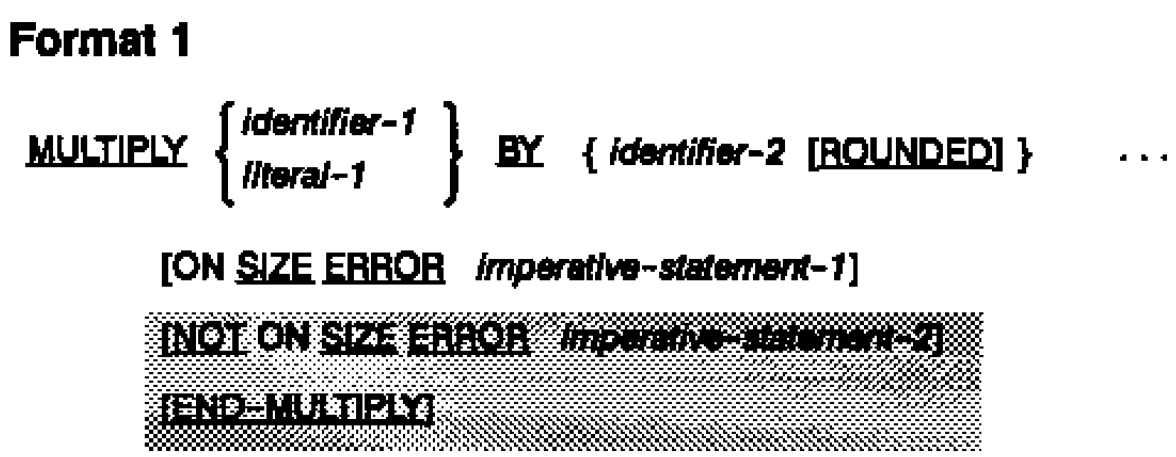
Table 6-2. Permissible Moves

Source Field \ Receiving Field	Group	Alphabetic	Alphanumeric	External Decimal (DISPLAY)	Binary (COMP)	Numeric Edited	Alphanumeric Edited	Packed Decimal (COMP-3)
Group	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]
Alphabetic	Y	Y	Y	Δ	Δ	Δ	Y	Δ
Alphanumeric	Y	Y	Y	Y[4]	Y[4]	Y[4]	Y	Y[4]
External Decimal (DISPLAY)	Y[1]	Δ	Y[2]	Y	Y	Y	Y[2]	Y
Binary (COMP)	Y[1]	Δ	Y[2]	Y	Y	Y	Y[2]	Y
Numeric Edited	Y	Δ	Y	Y[5]	Y[5]	Y[5]	Y	Y[5]
Alphanumeric Edited	Y	Y	Y	Δ	Δ	Δ	Y	Δ
Zero; (numeric or alphanumeric)	Y	Δ	Y	Y[3]	Y[3]	Y[3]	Y	Y[3]
Spaces	Y	Y	Y	Δ	Δ	Δ	Y	Δ
High-Value, Low-Value, Quotes	Y	Δ	Y	Δ	Δ	Δ	Y	Δ
All Literal	Y	Y	Y	Y[6]	Y[6]	Y[6]	Y	Y[6]
Numeric Literal	Y[2]	Δ	Y[2]	Y	Y	Y	Y[2]	Y
Nonnumeric Literal	Y	Y	Y	Y[6]	Y[6]	Y[6]	Y	Y[6]
Packed Decimal (COMP-3)	Y[1]	Δ	Y[2]	Y	Y	Y	Y[2]	Y

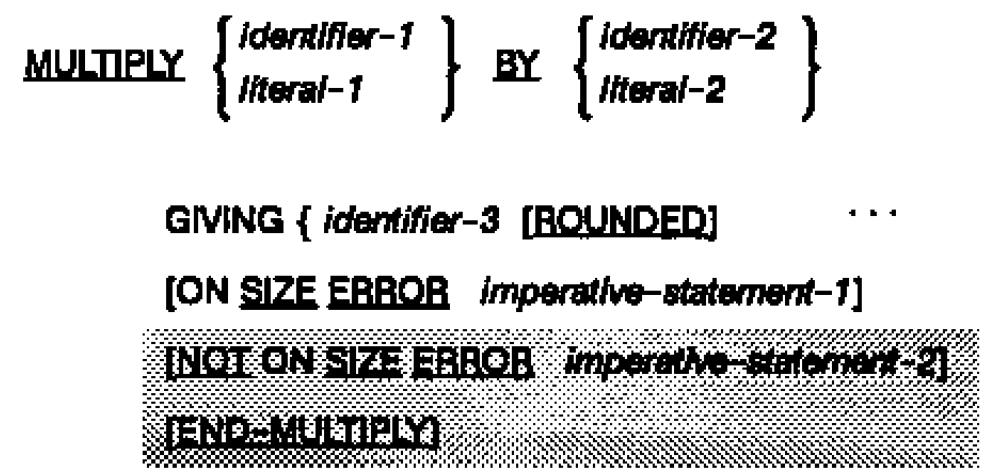
Y = permissible; Δ = prohibited
Y[1] = move without conversion
Y[2] = permissible only if the decimal point is to the right of the least significant digit.
Y[3] = a numeric move
Y[4] = the move is treated as an External Decimal (integer) field. (The characters must be numeric.)
Y[5] = the literal must consist only of numeric characters and is treated as an External Decimal (integer) field.
Y[6] = de-edited move

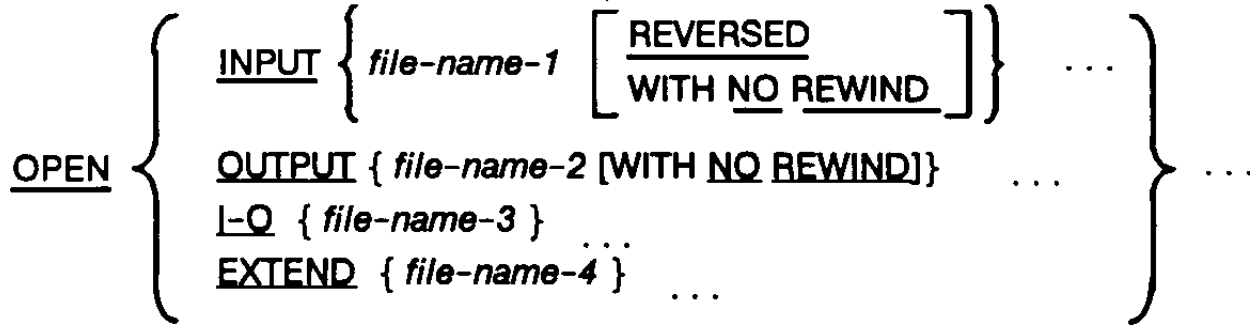
LG200028 210c

Format 1

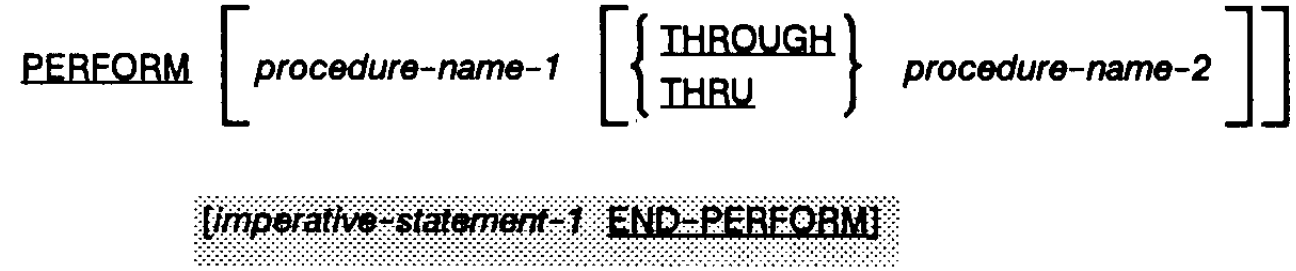


Format 2

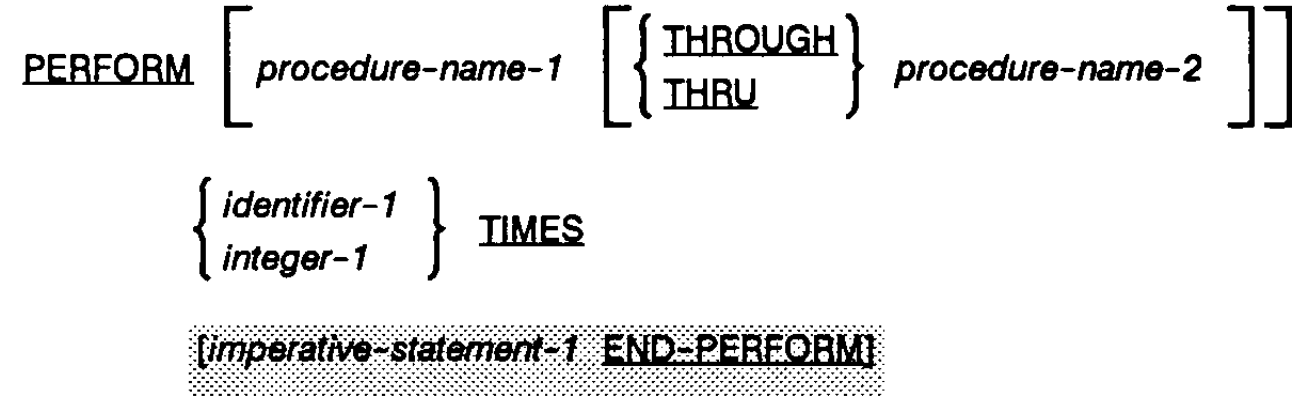




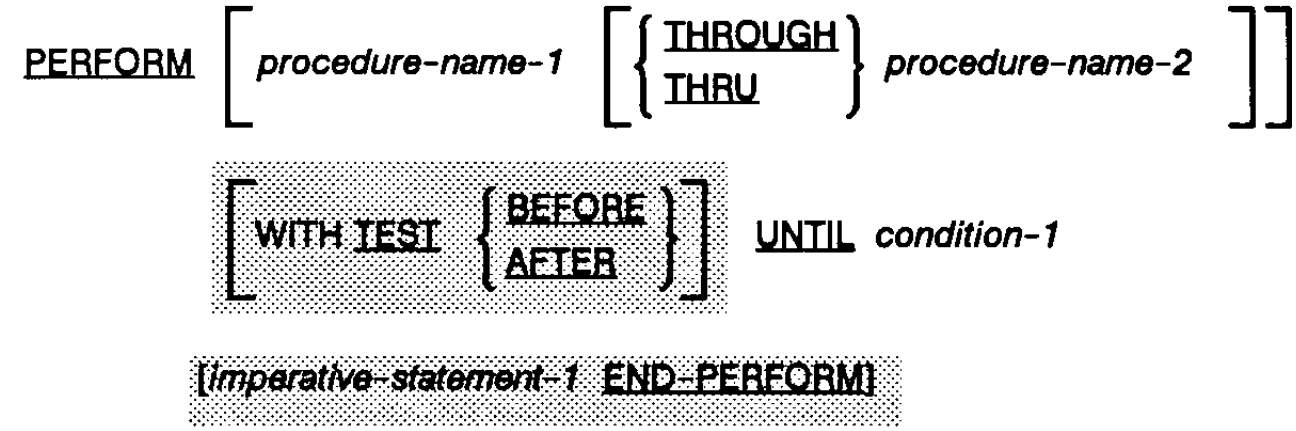
Format 1



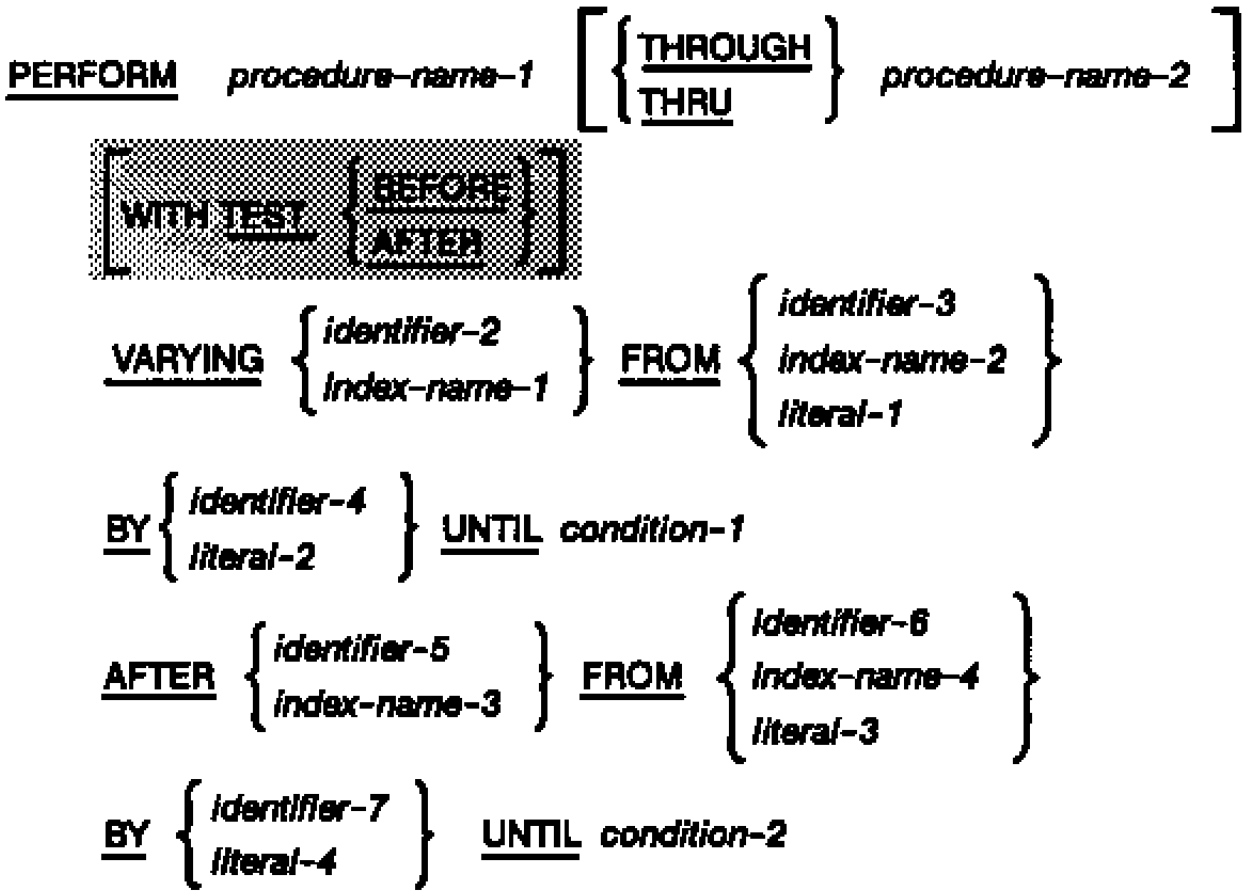
Format 2



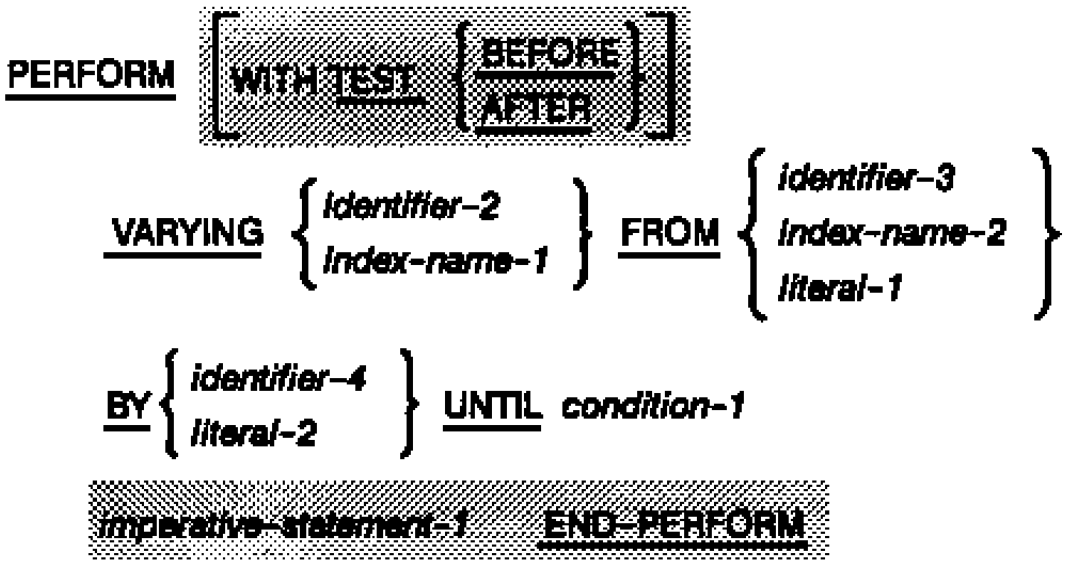
Format 3



Format 4a. (Out-of-line PERFORM . . . VARYING)



Format 4b. (In-line PERFORM . . . VARYING)



READ Statement

Format 1 – Sequential, Relative, Random, and Indexed Files

READ *file-name-1* [NEXT] RECORD [INTO *identifier-1*]

[AT END *imperative-statement-1*]
[NOT AT END *imperative-statement-2*]
[END-READ]

Format 2 – Relative and Random Files

READ *file-name-1* RECORD [INTO *identifier-1*]

[INVALID KEY *imperative-statement-1*]
[NOT INVALID KEY *imperative-statement-2*]
[END-READ]

Format 3 – Indexed Files

READ *file-name-1* RECORD [INTO *identifier-1*]

[KEY IS *data-name-1*]
[INVALID KEY *imperative-statement-1*]
[NOT INVALID KEY *imperative-statement-2*]
[END-READ]

RELEASE Statement

RELEASE *record-name-1* [FROM *identifier-1*]

REPLACE Statement

Format 1

REPLACE == *pseudo-text-1* == BY == *pseudo-text-2* == . . .

Format 2

REPLACE OFF

RETURN *file-name-1* RECORD [INTO *identifier-1*]

AT END *imperative-statement-1*
[NOT AT END *imperative-statement-2*]
[END-RETURN]

Format 1 - Sequential Files

REWRITE *record-name-1* [FROM *identifier-1*]

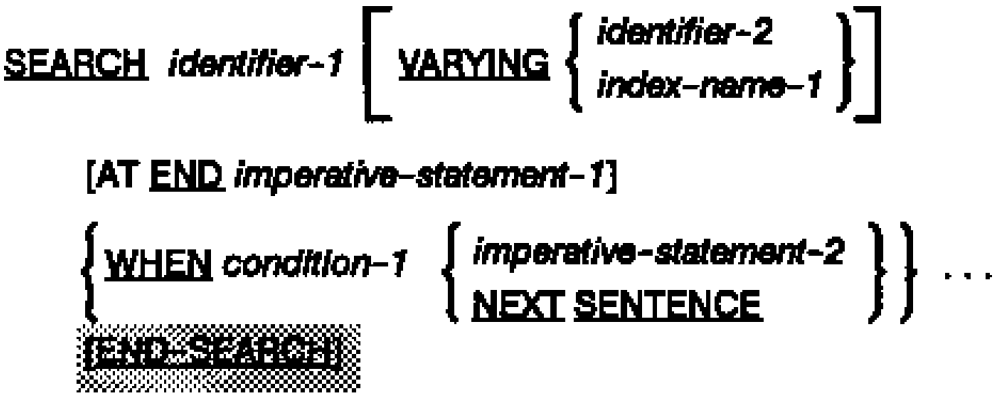
[END-REWRITE]

Format 2 - Relative, Random, and Indexed Files

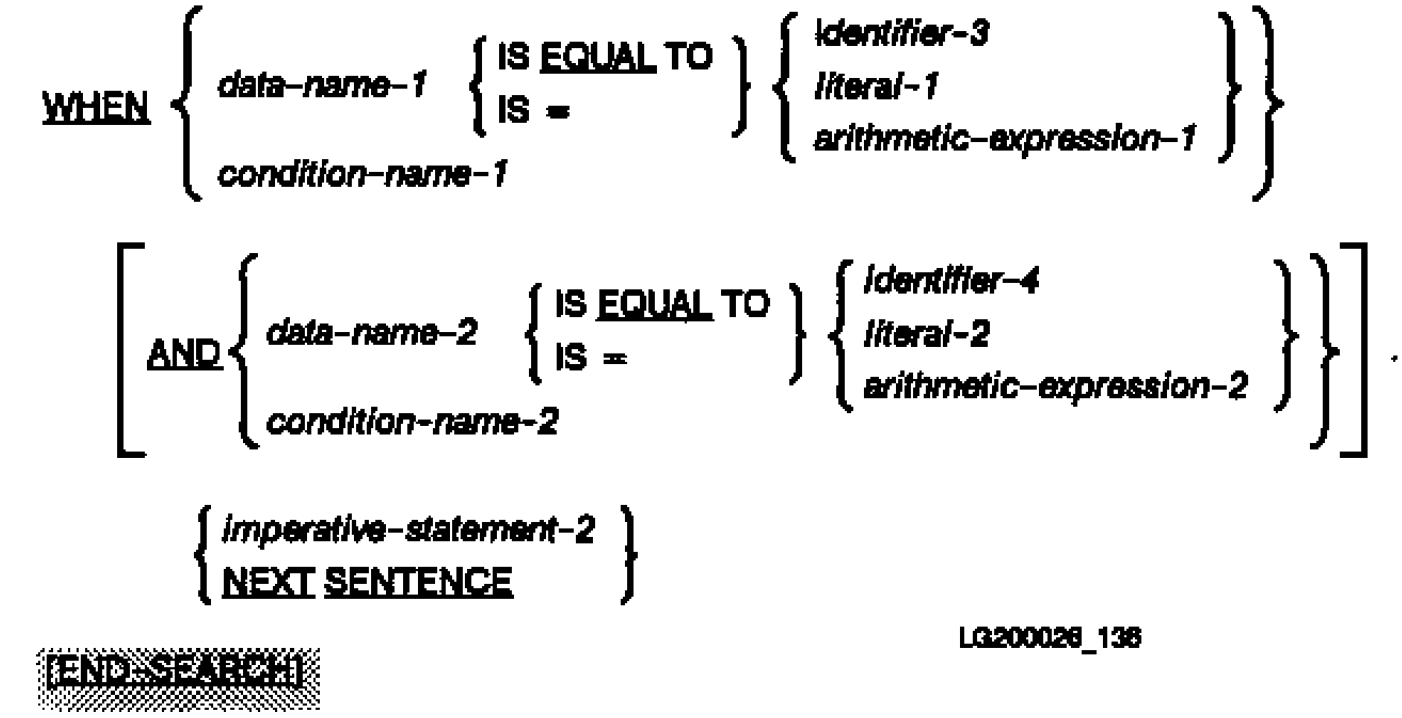
REWRITE *record-name-1* [FROM *identifier-1*]

[INVALID KEY *imperative-statement-1*]
[NOT INVALID KEY *imperative-statement-2*]
[END-REWRITE]

Format 1



Format 2



LG200026_136

SEEK Statement

SEEK *file-name* RECORD

SET Statement

Format 1

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

Format 2

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

Format 3

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

Format 4

```
SET { condition-name-1 } ... TO TRUE
```

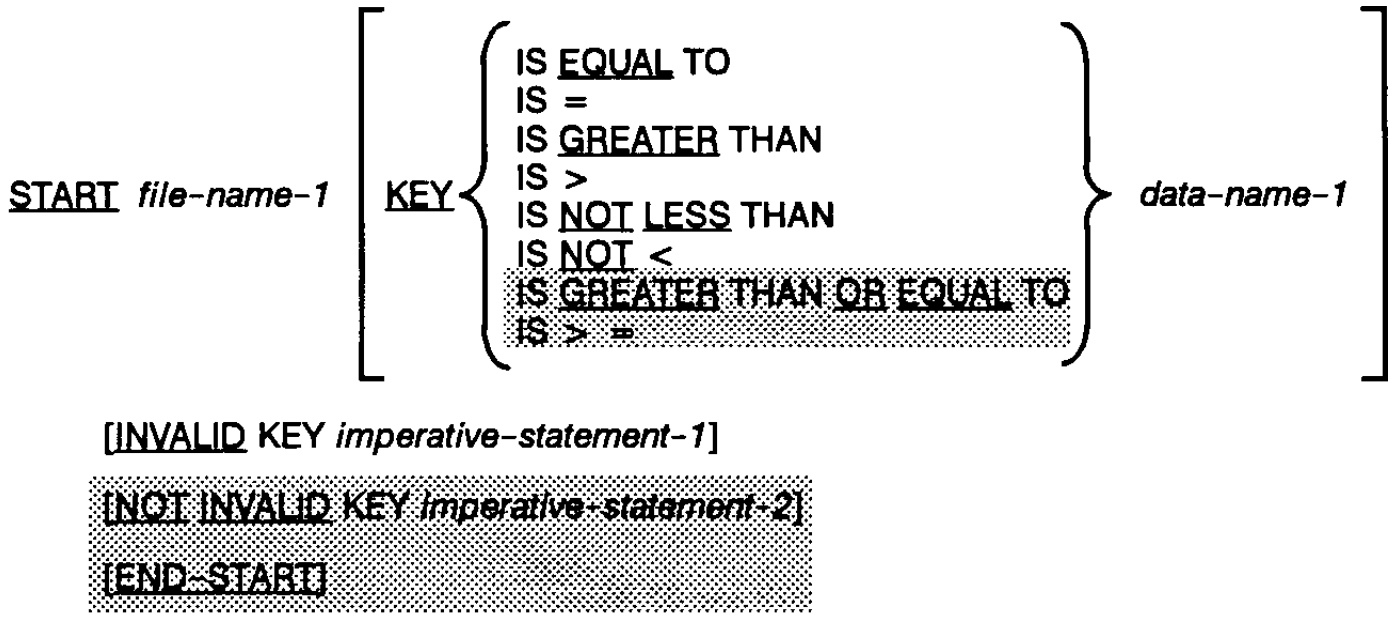
SORT Statement

```
SORT file-name-1 { ON { ASCENDING  } KEY { data-name-1 } ... } ...
                     { DESCENDING }
```

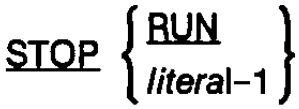
[WITH DUPLICATES IN ORDER]

```
[ COLLATING SEQUENCE IS { alphabet-name  } ]
                        { language-name }
                        { language-id   }

{ INPUT PROCEDURE IS procedure-name-1 [ { THROUGH } procedure-name-2 ] }
                                      { THRU    }
{ USING { file-name-2 } ...                                           }

{ OUTPUT PROCEDURE IS procedure-name-3 [ { THROUGH } procedure-name-4 ] }
                                       { THRU    }
{ GIVING { file-name-3 } ...                                           }
```

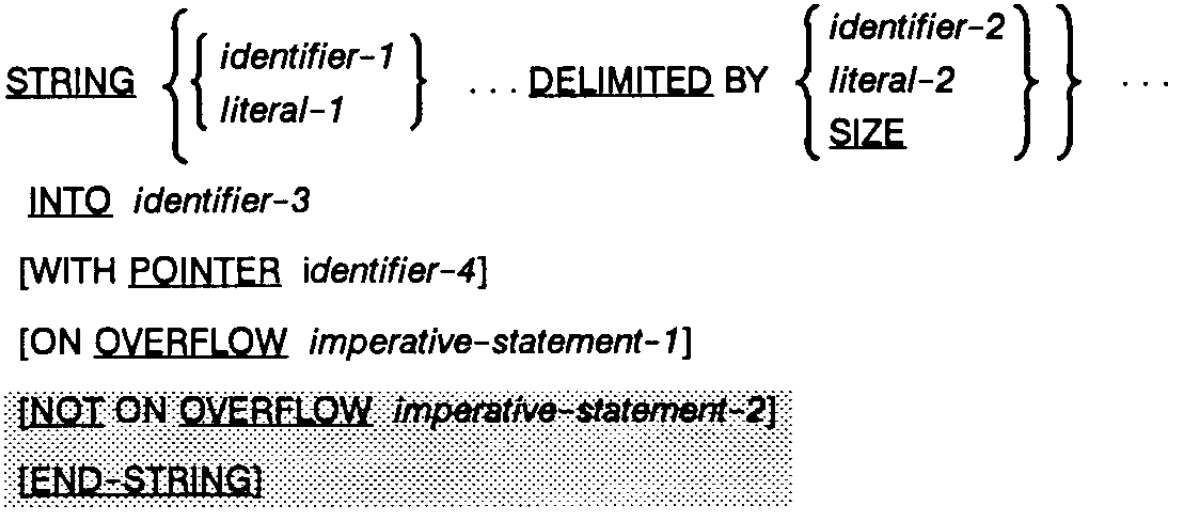
START Statement



STOP Statement



STRING Statement



SUBTRACT Statement

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

    [ON SIZE ERROR imperative-statement-1]
    [NOT ON SIZE ERROR imperative-statement-2]
    [END-SUBTRACT]
```

Format 2

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

    [ON SIZE ERROR imperative-statement-1]
    [NOT ON SIZE ERROR imperative-statement-2]
    [END-SUBTRACT]
```

UN-EXCLUSIVE Statement

```
UN-EXCLUSIVE  file-name-1
```

UNSTRING Statement

```
UNSTRING identifier-1
    [ [DELIIMITED BY [ALL] { identifier-2 } [ OR [ALL] { identifier-3 } ] ... ]
                           { literal-1    }             { literal-2    }
    INTO { identifier-4 [DELIMITER IN identifier-5] [COUNT IN identifier-6]} ...
    [WITH POINTER identifier-7]
    [TALLYING IN identifier-8]
    [ON OVERFLOW imperative-statement-1]
    [NOT ON OVERFLOW imperative-statement-2]
    [END-UNSTRING]
```

USE Statement

Format 1 – Error Handling Procedures

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

Format 2 – User Label Procedures

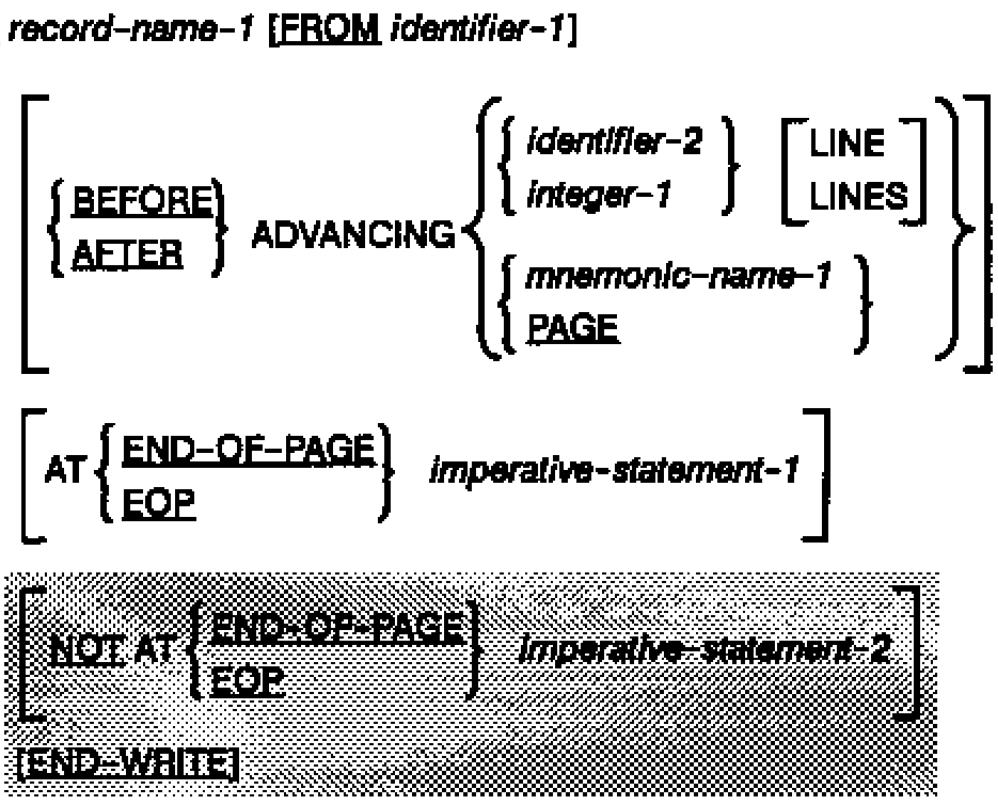
```
USE AFTER STANDARD BEGINNING [FILE]
    LABEL PROCEDURE ON { file-name-1 [ file-name-2] ... }
                       { INPUT                      }
                       { OUTPUT                     }
                       { I-O                        }
                       { EXTEND                     }
```

Format 3 – Debugging

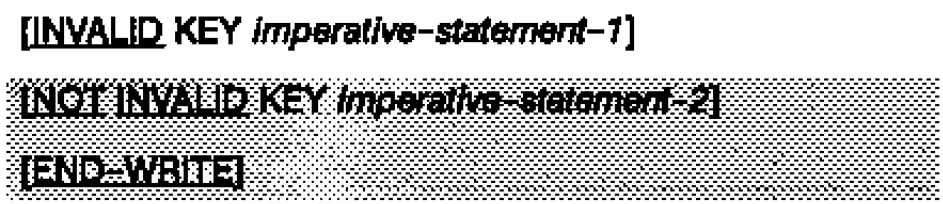
```
USE FOR DEBUGGING ON
    { { procedure-name-1 } ... }
    { ALL PROCEDURES           } .
```

LG200028_147a

Format 1 – Sequential Files

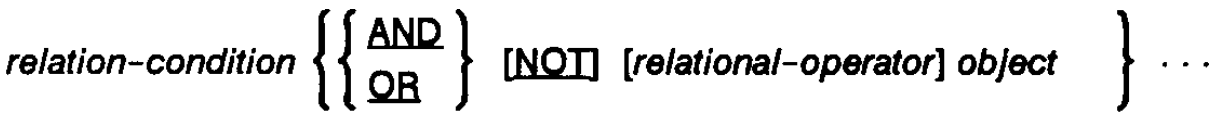


Format 2 – Relative, Indexed, or Random-Access Files

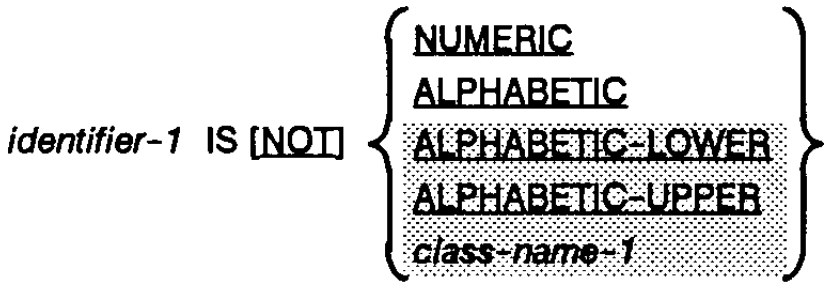


General Format for Conditions

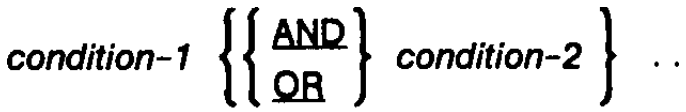
Abbreviated Combined Relation Condition Format



Class Condition Format



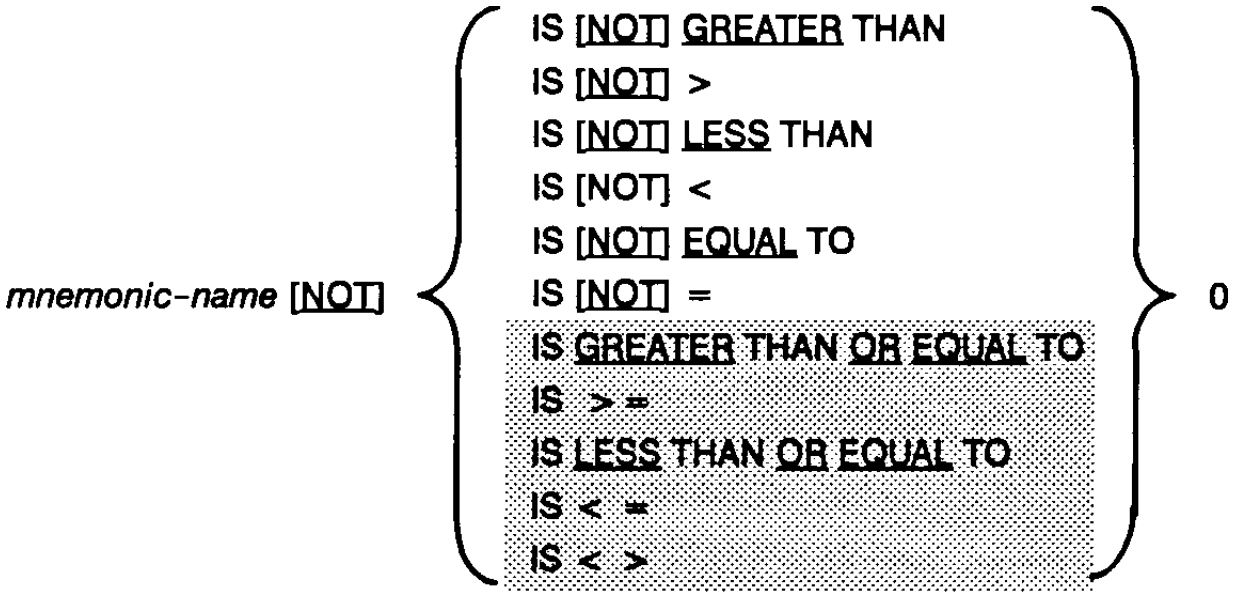
Combined Condition Format



Condition-Name Condition Format

condition-name-1

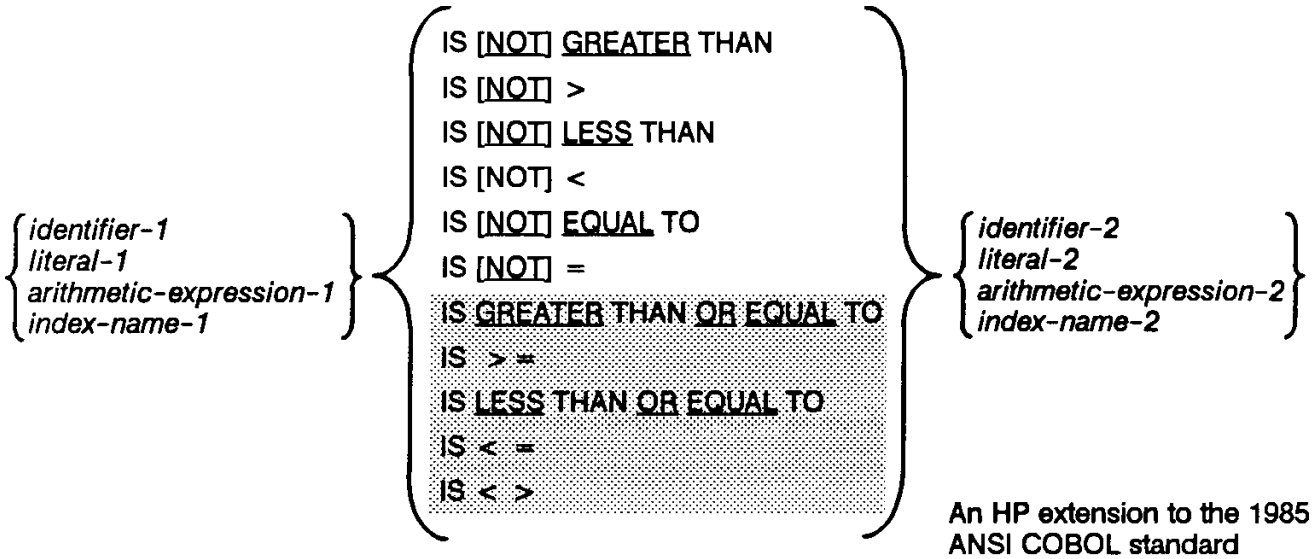
Intrinsic Relation Condition Format



Negated Condition Format

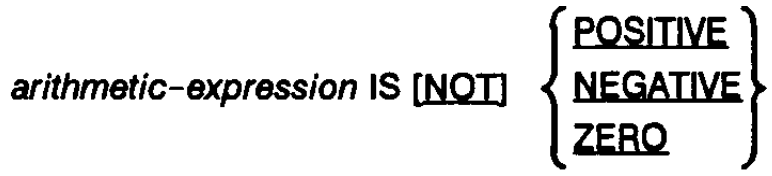
NOT *condition-name-1*

Relation Condition Format



An HP extension to the 1985 ANSI COBOL standard

Sign Condition Format



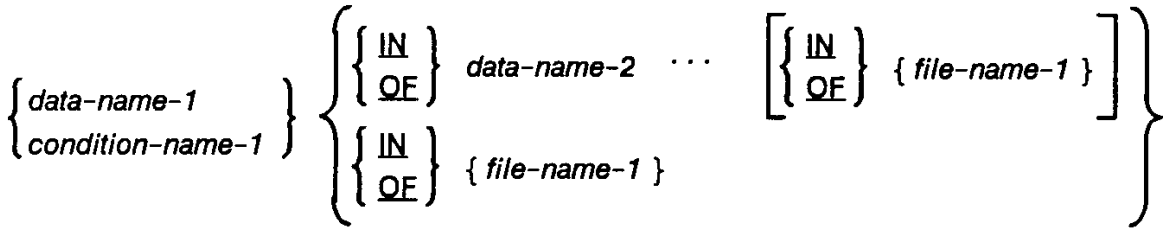
Switch-Status Condition Format

condition-name-1

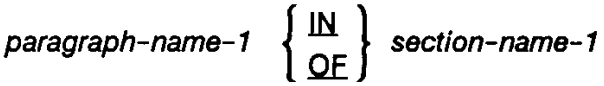
Miscellaneous Formats

Qualification

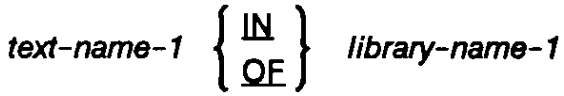
Format 1



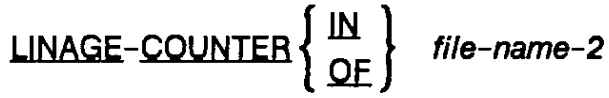
Format 2



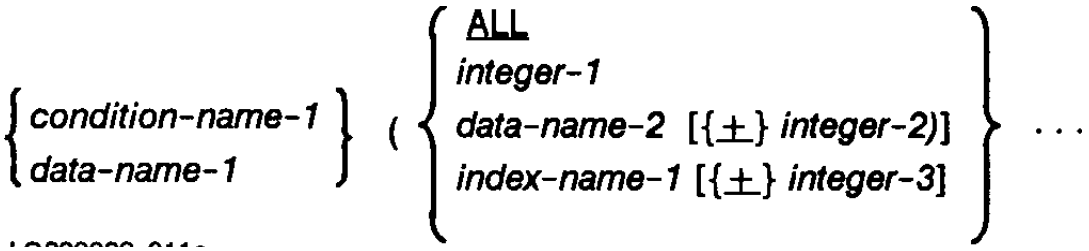
Format 3



Format 4



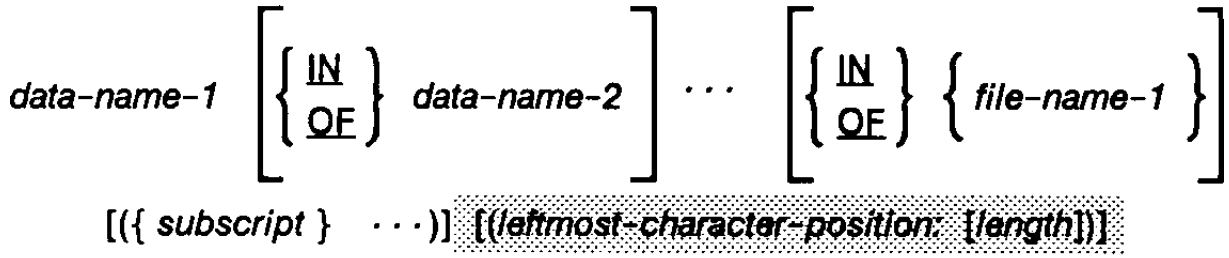
Subscripting/Indexing



Reference Modification

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

Identifier



Special Register Words

Table 6-2. Special Register Words

Word	Contents
LINAGE-COUNTER	An unsigned number used to keep track of the number of lines written to each page of a printed report.
DEBUG-ITEM	A data item used in support of the COBOL DEBUG facility.
TALLY	A 5-digit unsigned integer typically used to store information produced by the EXAMINE statement in the PROCEDURE DIVISION.
CURRENT-DATE	An 8-digit alphanumeric item used only as the sending field in a MOVE or DISPLAY statement in the PROCEDURE DIVISION.
WHEN-COMPILED	An 18-character alphanumeric item that represents the date and time that the program is compiled.[REV BEG]
RETURN-CODE	A predefined numeric data name in the PROCEDURE DIVISION of a subprogram, RETURN-CODE is used to pass a value back to the calling program.[REV END]
TIME-OF-DAY	A six-character numeric item accessed only as the transmitting field of a MOVE or DISPLAY statement in the PROCEDURE DIVISION to access the time of day.

Figurative Constant Words

Table 6-3. Figurative Constant Words

Word	Constant Value
ALL *literal*	The character string denoted by the variable *literal*.
HIGH-VALUE HIGH-VALUES	One or more occurrences of the character with the highest possible value in the program collating sequence.
LOW-VALUE LOW-VALUES	One or more occurrences of the character with the lowest possible value in the program collating sequence.
QUOTE QUOTES	One or more quotation marks. This constant is used to code the quotation mark as a literal in statements such as MOVE QUOTES.
SPACE SPACES	One or more spaces.
ZERO ZEROS ZEROES	One or more occurrences of the digit zero.
[ALL] symbolic-character	User-defined figurative constants which are defined using the SYMBOLIC CHARACTERS clause of the ENVIRONMENT DIVISION.

Appendix A HP COBOL II/XL Compiler Commands

Command Files

These are the MPX XL command files you can use to compile, link, and execute HP COBOL II/XL programs. The syntax follows.

Table A-1. Command Files

Command	Description
COB85XL	Invokes the COBOL compiler using the 1985 ANSI standard entry point and creates an object file.
COB85XLK	Invokes the COBOL compiler using the 1985 ANSI standard entry point, links the object file, and creates a program file.
COB85XLG	Invokes the COBOL compiler using the 1985 ANSI standard entry point, and creates and runs a program file in $NEWPASS.
COB74XL	Invokes the COBOL compiler using the 1974 ANSI standard entry point and creates an object file.
COB74XLK	Invokes the COBOL compiler using the 1974 ANSI standard entry point, links the object file, and creates a program file.
COB74XLG	Invokes the COBOL compiler using the 1974 ANSI standard entry point, and creates and runs a program file in $NEWPASS.

Syntax

```
COB85XL [textfile ][,[objectfile ][,[listfile ][,[masterfile ] [,newfile ]]]]
        [;INFO="info "][;WKSP=workspacename ] [;XDB=xdbfile ]

COB85XLK [textfile ][,[progfile ][,[listfile ][,[masterfile ] [,newfile ]]]]
         [;INFO="info "][;WKSP=workspacename ] [;XDB=xdbfile ]

COB85XLG [textfile ][,[listfile ][,[masterfile ][,newfile ]]]
         [;INFO="info "][;WKSP=workspacename ] [;XDB=xdbfile ]

COB74XL [textfile ][,[objectfile ][,[listfile ][,[masterfile ] [,newfile ]]]]
        [;INFO="info "][;WKSP=workspacename ] [;XDB=xdbfile ]

COB74XLK [textfile ][,[progfile ][,[listfile ][,[masterfile ] [,newfile ]]]]
         [;INFO="info "][;WKSP=workspacename ] [;XDB=xdbfile ]

COB74XLG [textfile ][,[listfile ][,[masterfile ][,newfile ]]]
         [;INFO="info "][;WKSP=workspacename ] [;XDB=xdbfile ]
```

Parameters

textfile MPE or TSAM file containing your source program. This file can be compiled. The default is $STDIN.

objectfile Relocatable object code file. This file can be linked. The default is $NEWPASS or $OLDPASS. The object file code can be NMOBJ or NMRL. The compiler will take the appropriate actions for existing files.

progfile Executable program file. This file can be executed. The default is $NEWPASS.

listfile File on which your source code will be listed. The default is $STDLIST.

masterfile MPE or TSAM file to be merged with *textfile* to produce a composite source program. If *masterfile* is omitted, the entire source is from *textfile*.

newfile MPE file into which the merged *textfile* and *masterfile* is written. For details, refer to the *HP COBOL II/XL Reference Manual*. If *newfile* is omitted, no new file is written.

info A string whose value is a command list of the form:

"$*compiler_command* [$*compiler_command*]..."

where no *compiler_command* contains the character $.

If the number of commands is long enough, you can use an ampersand (&) to continue the *info* string. The length limit for a compiler command is the same as the length limit for a source program line.

In the listing file, the string "INFO=" appears where the sequence numbers normally appear.

workspacename Work space in which HP TOOLSET/XL can manage versions of the source program.[REV BEG]

xdbfile MPE XL file into which a listing of the source code is written. *xdbfile* is used to view the source code in the HP Symbolic Debugger/XL.[REV END]

Compiling Your Program With the RUN Command

The MPE XL RUN command runs the HP COBOL II compiler, which compiles your source program. You can invoke the HP COBOL II compiler and compile your HP COBOL II program with either the RUN command or a command file.

Syntax

```
RUN {COBOL  }.PUB.SYS [,{ANSI85}];PARM=parm;INFO=info
    {COBOLII}         [ {ANSI74}]
```

Appendix B COBOL II/XL Compiler Commands

Any of the commands or UDCs that are summarized below can be used to invoke the HP COBOL II/V compiler. The COBOLII commands invoke the ANSI COBOL'74-compatible COBOLII compiler through the ANSI COBOL'74 entry point. The COBOLIIX system-wide UDCs invoke the extended COBOLII compiler through the ANSI COBOL'85 entry point. This provides access to the ANSI COBOL'85 feature set.

Syntax of UDCs

```
COBOLIIX [textfile ][,[uslfile [,[listfile ][,[masterfile ]
        [,[newfile ]]]]]][,info ]

COBOLIIXPREP [textfile ][,[progfile [,[listfile ][,[masterfile ]
           [,[newfile ]]]]]][,info ]

COBOLIIXGO [textfile ][,[listfile ][,[masterfile ] [,[newfile ]]]][,info ]
```

Syntax of Commands

```
COBOLII [textfile ][,[uslfile [,[listfile ][,[masterfile ] [,[newfile ]]]]]]
       [;INFO=info ][;WKSP=workspacename ]

COBOLIIPREP [textfile ][,[progfile [,[listfile ][,[masterfile ]
           [,[newfile ]]]]]][;INFO=info ][;WKSP=workspacename ]

COBOLIIGO [textfile ][,[listfile ][,[masterfile ][,[newfile ]]]]
         [;INFO=info ][WKSP=workspacename ]
```

Appendix C MPE/XL Run-Time Trap Handling

The HP COBOL II/XL compiler handles run-time traps for cases of bounds checking, divide by zero, invalid GOTO statements, and alignment errors.

The run-time environment is set using the MPE XL SETVAR command with the variable COBRUNTIME. The syntax for this command is:

SETVAR COBRUNTIME "*string* "
[REV BEG]

where *string* is a string of nine uppercase or lowercase[REV END] characters representing run-time options A, C, D, I, M, N, or blank as shown in the following table.

Table C-1. Run-Time Error Handling Options

Option	Meaning
A or blank	Print the error message and abort (the default).
C	Print the error message and continue.
D	Print the error message and enter debug mode.
I	Ignore the error (continue without printing an error message).
M	Print the error message, change the illegal digit to some legal digit, and continue. This option is only valid for illegal decimal or ASCII digit errors. (See character position 1 in the next table.) When used for other errors, M is treated as a blank.
N	Change the illegal digit to a legal digit and continue without printing an error message.[REV BEG] This option is only valid for illegal decimal or ASCII digit errors in positions 1, 7, and 8. See the description of character positions 1, 7, and 8 in the next table for details. When used in other positions,[REV END] N is treated as a blank.

Each character position in the above COBRUNTIME string represents a specific trap that you can request, as shown in the following table:

Table C-2. Character Position in Specific Traps

Character Position	Trap Type
1	Illegal ASCII or decimal digit.
2	Range error (OCCURS DEPENDING ON identifier, subscript, index, or reference modification out of bounds).

3	No SIZE ERROR phrase.
4	Invalid GO TO.
5	Address Alignment.
6	Paragraph stack overflow (recursive PERFORMs or too many PERFORMs with a common exit point).[REV BEG]
7	Leading blanks in a numeric field. If this position contains I, leading blanks in a numeric field are ignored. If this position contains N, leading blanks are changed to zeros. If this position contains a value other than N or I, the action entered in character position 1 is used.
8	Unsigned number in signed numeric field or signed number in unsigned numeric field. If this position contains I, the invalid sign is ignored. If this position contains N, the invalid sign is corrected. If this position contains a value other than N or I, the action entered in character position 1 is used.[REV END]

Table C-2. Character Position in Specific Traps (cont.)

Character Position	Trap Type
[REV BEG]	
9	Only affects a NUMERIC class condition with a PACKED-DECIMAL identifier. If this field contains the character I, then the following conditions do *not* make a NUMERIC test false: * A signed value in an unsigned PACKED-DECIMAL field. * An unsigned value in a signed PACKED-DECIMAL field. * Any invalid sign nibble (half-byte). If this field contains anything other than I, the above conditions make the NUMERIC class condition false.[REV END]

Appendix D COBEDIT Program

The COBEDIT program develops and maintains COPY libraries. This program resides in the PUB group of the SYS account. It allows you to create, modify, and list a COBOL COPY library file.

To enter the COBEDIT program, issue the following command:

```
COBEDIT
```

Table D-1. COBEDIT Commands

Command	Meaning
BUILD	Build a COPYLIB file.
COPY	Copy modules into the library as in the BUILD command.
EDIT	Create or edit a module to add to a COPYLIB file.
EXIT	Leave the COBEDIT program.
HELP	List all COBEDIT commands.
KEEP	Add a module to the currently active COPYLIB file.
LIBRARY	Activate an already existing COPYLIB file.
LIST	List text-names or one or more modules of the currently active COPYLIB file.
PURGE	Purge a module of the currently active library or purge the library itself.
SHOW	Show the name of the current library, its key file and the latest module to be accessed.

BUILD

The BUILD command allows you to build a new KSAM file to be used as a library file.

Syntax

BUILD [*file-name*] [,*maxrecs*]

Parameters

file-name any name you wish to give your new library file, subject to the naming conventions for any MPE file. The *file-name* may be from one to eight alphanumeric characters, the first of which must be alphabetic.

maxrecs if specified, must be greater than 0. It specifies the maximum number of records that may be placed in the file being built. If no value is specified for *maxrecs*, the default is 2500.

COPY

The COPY command allows you to copy additional modules into a library that was created previously using the BUILD command. To use COPY, the library must be the current library or it must be activated by using the LIBRARY command.

Syntax

COPY

EDIT

The EDIT command calls the EDIT/3000 subsystem, and optionally allows you to name a module from the currently active library to be edited.

Syntax

EDIT [*text-name*]

Parameters

text-name the name of a module in the currently active library.

EXIT

The EXIT command is used to exit the COBEDIT program.

Syntax

E[XIT]

HELP

The HELP command lists and gives a brief description of all commands available in the COBEDIT program.

Syntax

HELP

KEEP

The KEEP command allows you to add a module to the currently active library or replace an already existing module.

Syntax

KEEP [*text-name*]

Parameters

text-name is the name to be used for the module being kept.

LIBRARY

The LIBRARY command allows you to select the library that you wish to access. When you issue this command, the currently active library is closed and the specified library is opened and made available.

Syntax

LIBRARY *library-name*

Parameters

library-name is the name of the library file you want to access.

LIST

The LIST command allows you to list information about your currently active library.

Syntax

```
LIST  [text-name ]
        [ALL       ]
```

Parameters

text-name is the name of a module in the currently active

library.

ALL — indicates that all modules in the library are to be listed, beginning with the first module on the file, and proceeding to the last.

PURGE

The PURGE command allows you to purge either a single module from your currently active library or the entire library.

Syntax

```
PURGE  {text-name }
       {ALL      }
```

Parameters

text-name — is the name of a module to be purged from the currently active library. This is the module to be purged.

ALL — indicates that you want the entire library, including its key file, to be purged.

SHOW

The SHOW command is used to find out the name of the currently active library, its key file, and the name of the module that was most recently accessed by COBEDIT.

If no library is open, the message No library is open occurs.

Syntax

```
SHOW
```

Appendix E COBOL Functions

The following tables lists and briefly describes each COBOL function. The syntax for each function follows.

Table E-1. Date Functions

Function	Type	Value Returned
CURRENT-DATE	Alphanumeric	Current date and time and difference from Greenwich Mean Time.
DATE-OF-INTEGER	Integer	Standard date equivalent (YYYYMMDD) of integer date.
DAY-OF-INTEGER	Integer	Julian date equivalent (YYYYDDD) of integer date.
INTEGER-OF-DATE	Integer	Integer date equivalent of standard date (YYYYMMDDD).
INTEGER-OF-DAY	Integer	Integer date equivalent of Julian date (YYYYDDD).
WHEN-COMPILED	Alphanumeric	Date and time program was compiled.

Table E-2. String Functions

Function	Type	Value Returned
CHAR	Alphanumeric	The character in a specified position of the program collating sequence.
LENGTH	Integer	Length, in character positions, of the parameter.
LOWER-CASE	Alphanumeric	The same parameter with all uppercase letters replaced by lowercase letters.
NUMVAL	Numeric	Numeric value of a simple numeric string.
NUMVAL-C	Numeric	Numeric value of a numeric string with optional commas and currency sign.
ORD	Integer	Ordinal position of the parameter in collating sequence.
REVERSE	Alphanumeric	Same parameter with characters in reverse order.
UPPER-CASE	Alphanumeric	Same parameter with all lowercase letters replaced by uppercase letters.

Table E-3. General Functions

Function	Type	Value Returned
MAX	Depends on parameters.	Maximum value of all parameters.
MIN	Depends on parameters.	Minimum value of all parameters.
ORD-MAX	Integer	Ordinal position of maximum parameter.
ORD-MIN	Integer	Ordinal position of minimum parameter.

Table E-4. Arithmetic Functions

Function	Type	Value Returned
INTEGER	Integer	The greatest integer not greater than the given numeric value.
INTEGER-PART	Integer	Integer part of the given numeric value.
LOG	Numeric	Natural logarithm of a numeric value.
LOG10	Numeric	Logarithm to base 10 of a numeric value.
MOD	Integer	Modulo of two integer parameters.
RANDOM	Numeric	Pseudo-random number.
REM	Numeric	Remainder after division.
SQRT	Numeric	Square root of a numeric value.
SUM	Integer or Numeric	Sum of parameters.

Table E-5. Financial and Statistical Functions

Function	Type	Value Returned
ANNUITY	Numeric	Ratio of an annuity paid for a specified number of periods at a specified interest rate to an initial investment of one.
FACTORIAL	Integer	Factorial of an integer value.
MEAN	Numeric	Arithmetic mean of parameters.
MEDIAN	Numeric	Median of parameters.
MIDRANGE	Numeric	Mean of smallest and largest parameters.

PRESENT-VALUE	Numeric	Present value of a series of future period-end amounts at a given discount rate.
RANGE	Integer or Numeric	Value of largest parameter minus value of smallest parameter.
STANDARD-DEVIATION	Numeric	Standard deviation of parameters.
VARIANCE	Numeric	Variance of parameters.

Table E-6. Trigonometric Functions

Function	Type	Value Returned
COS	Numeric	Cosine of an angle in radians.
SIN	Numeric	Sine of an angle in radians.
TAN	Numeric	Tangent of an angle in radians.
ACOS	Numeric	Arccosine, in radians, of a numeric value.
ASIN	Numeric	Arcsine, in radians, of a numeric value.
ATAN	Numeric	Arctangent, in radians, of a numeric value.

The $CONTROL POST85 Option

You must specify $CONTROL POST85 in any program that calls a COBOL function. $CONTROL POST85 enables the COBOL functions and makes the word FUNCTION a reserved word. If you have used the word FUNCTION as an identifier, you must change it to another word before you can call any COBOL functions. Otherwise, the compiler gives an error message.

ANSI85 Entry Point

You must use the ANSI85 entry point of the HP COBOL II/XL compiler to call any COBOL functions.

ACOS

The ACOS function returns the arccosine of the parameter. The function type is numeric.

Syntax

FUNCTION ACOS (*parameter-1*)

Parameters

parameter-1 Must be class numeric and must be between -1 and 1, inclusive.

ANNUITY

The ANNUITY function (annuity immediate) returns a numeric value that is the ratio of an annuity paid at the end of each period for the number of periods specified by *parameter-2* to an initial investment of one. Interest is earned at the rate specified by *parameter-1* and is applied at the end of the period before the payment. The function type is numeric.

Syntax

FUNCTION ANNUITY (*parameter-1 parameter-2*)

Parameters

parameter-1 Must be class numeric and must be greater than or equal to zero.

parameter-2 Must be a positive integer.

ASIN

The ASIN function returns the arcsine of the parameter. The function type is numeric.

Syntax

FUNCTION ASIN (*parameter-1*)

Parameters

parameter-1 Must be class numeric and must be between -1 and 1, inclusive.

ATAN

The ATAN function returns the arctangent of the parameter. The function type is numeric.

Syntax

FUNCTION ATAN (*parameter-1*)

Parameters

parameter-1 Must be class numeric.

CHAR

The CHAR function returns a one-character alphanumeric value that is a character in the program collating sequence having the ordinal position equal to the value of *parameter-1*. The function type is alphanumeric.

Syntax

FUNCTION CHAR (*parameter-1*)

Parameters

parameter-1 Must be an integer. Must be greater than zero and less than or equal to the number of positions in the collating sequence.

COS

The COS function returns the cosine of an angle. The function type is numeric.

Syntax

FUNCTION COS (*parameter-1*)

Parameters

parameter-1 The size of an angle in radians. Must be class numeric.

CURRENT-DATE

The CURRENT-DATE function returns the calendar date, time of day, and the difference between the local time and Universal Coordinated Time (UTC), or Greenwich Mean Time. To get the correct time differential, you need to set the environment variable TZ to your local time zone. The function type is alphanumeric.

Syntax

FUNCTION CURRENT-DATE

DATE-OF-INTEGER

The DATE-OF-INTEGER function converts a date in the Gregorian calendar from integer date form to standard date form (YYYYMMDD). The function type is integer.

Syntax

FUNCTION DATE-OF-INTEGER (*parameter-1*)

Parameters

parameter-1 A positive integer that represents a number of days succeeding December 31, 1600 in the Gregorian calendar.

DAY-OF-INTEGER

The DAY-OF-INTEGER function converts a date in the Gregorian calendar from integer date form to Julian date form (YYYYDDD). The function type is integer.

Syntax

FUNCTION DAY-OF-INTEGER (*parameter-1*)

Parameters

parameter-1 A positive integer that represents a number of days succeeding December 31, 1600 in the Gregorian calendar.

FACTORIAL

The FACTORIAL function returns an integer that is the factorial of *parameter-1*. The function type is integer.

Syntax

FUNCTION FACTORIAL (*parameter-1*)

Parameters

parameter-1 Must be an integer greater than or equal to zero. (The largest value *parameter-1* can be is 20 in order for the result to fit in 18 digits.)

INTEGER

The INTEGER function returns the greatest integer value that is less than or equal to the argument. The function type is integer.

Syntax

FUNCTION INTEGER (*parameter-1*)

Parameters

parameter-1 Must be class numeric.

INTEGER-OF-DATE

The INTEGER-OF-DATE function converts a date in the Gregorian calendar from standard date form (YYYYMMDD) to integer date form. The function type is integer.

Syntax

FUNCTION INTEGER-OF-DATE (*parameter-1*)

Parameters

parameter-1 Must be an integer of the form YYYYMMDD, whose value is determined as follows:

(YYYY * 10000) + (MM * 100) + DD

where YYYY represents the year in the Gregorian calendar and must be an integer greater than 1600. MM represents a month and must be a positive integer less than thirteen. DD represents a day and must be a positive integer less than 32 ; DD must be valid for the specified month and year combination.

INTEGER-OF-DAY

The INTEGER-OF-DAY function converts a date in the Gregorian calendar from Julian date form (YYYYDDD) to integer date form. The function type is integer.

Syntax

FUNCTION INTEGER-OF-DAY (*parameter-1*)

Parameters

parameter-1 Must be an integer of the form YYYYDDD, whose value is obtained as follows:

(YYYY * 1000) + DDD

where YYYY represents the year in the Gregorian calendar and must be an integer greater than 1600. DDD represents the day of the year and must be a positive integer less than 367. DDD must be valid for the year specified.

INTEGER-PART

The INTEGER-PART function returns an integer that is the integer portion of *parameter-1* (*parameter-1* is truncated). The function type is integer.

Syntax

FUNCTION INTEGER-PART (*parameter-1*)

Parameters

parameter-1 Must be class numeric.

LENGTH

The LENGTH function returns an integer equal to the length of the argument in character positions (bytes). To conform to ANSI standard COBOL, you can use the LENGTH function instead of the .LEN. pseudo-intrinsic. The function type is integer.

Syntax

FUNCTION LENGTH (*parameter-1*)

Parameters

parameter-1 A nonnumeric literal or a data item of any class or category.

If *parameter-1* or any data item subordinate to *parameter-1* is described with the DEPENDING phrase of the OCCURS clause, the contents of the data item referenced by the data-name specified in the DEPENDING phrase are used at the time the LENGTH function is evaluated.