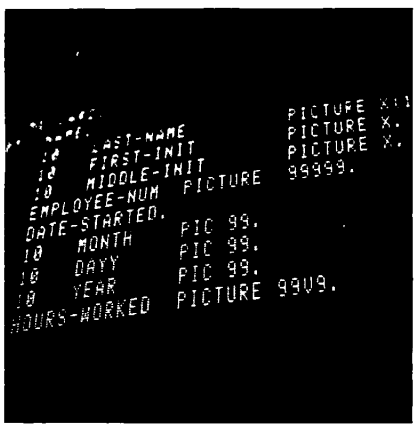
Prime Computer, Inc.

COBOL
Revision 18



FDR3339

The Programmer s Companion is a series of pocket-size, quick reference guides to Prime software products

Published by Prime Computer, Incorporated
500 Old Connecticut Path, Framingham, MA 01701

Produced by Prime Technical Publications Department, 500 Old Connecticut Path, Framingham, MA 01701

The information contained in the document reflects the software as of Revision 18 3 and is subject to change without notice Prime Computer, Incorporated assumes no responsibility for errors that may appear in this document

First Printing March 1982

Credits

Research and copy
Grace Na

Produced by
D Christine Benders

Designed by
William Agush

TABLE OF CONTENTS

FORMAT NOTATION

Abbreviation of PRIMOS commands

The minimum required abbreviation of PRIMOS commands is shown in rust colored letters

Keywords

Underlined and rust colored uppercase words are keywords They are required and should be entered literally whenever the statement or phrase in which they appear is used

Optional words

Uppercase words which are not underlined or rust colored are optional words They may be omitted or they may be included solely for improving readability of the program

Lowercase words

Words printed in lowercase letters in formats represent generic parts (such as data names) of which a valid representation must appear

< , - + -

These characters when appearing in formats although not underlined are required when such formats are used

[ ]

A statement phrase or word enclosed in square brackets is optional

{ }

Braces are used to enclose words phrases or statements which represent a choice of mutually exclusive options of which one must be chosen

Parentheses ( )

Parentheses where they appear are a required literal part of the command or statement syntax

...

The ellipsis indicates the immediately preceding unit may be repeated

COBOL PROGRAM STRUCTURE

```
 ID DIVISION. (or IDENTIFICATION DIVISION.)
 PROGRAM-ID. program-name.
[AUTHOR. [comment-entry] ... ]
[INSTALLATION. [comment-entry] ... ]
[DATE-WRITTEN. [comment-entry] ... ]
[DATE-COMPILED. [comment-entry] ... ]
[SECURITY. [comment-entry] ... ]
[REMARKS. [comment-entry] ... ]

 ENVIRONMENT DIVISION.
[CONFIGURATION SECTION.
[SOURCE-COMPUTER. entry.]
[OBJECT-COMPUTER. entry.]
[SPECIAL-NAMES. entry.]]
[INPUT-OUTPUT SECTION.
```

```
FILE-CONTROL. {entry} ...
[I-O-CONTROL. entry.]]

DATA DIVISION.

[ FILE SECTION.

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

[ WORKING-STORAGE SECTION.

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

continued

```
[ LINKAGE SECTION.

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

```
 PROCEDURE DIVISION [USING data-name-1 [, data-name-2] ... ].
[DECLARATIVES.
{section-name SECTION. USE sentence.
[paragraph-name. [sentence] ... ] ... } ...
 END DECLARATIVES.]
{section-name SECTION.
[paragraph-name. [sentence] ... ] ... } ...
```

IDENTIFICATION DIVISION

FORMAT:

ID DIVISION. (or IDENTIFICATION DIVISION.)

PROGRAM-ID. program-name. (no special characters in name)

[AUTHOR. [comment-entry] ...]

[INSTALLATION. [comment-entry] ...]

[DATE-WRITTEN. [comment-entry] ...]

[DATE-COMPILED. [comment-entry] ...]

[SECURITY. [comment-entry] ...]

[REMARKS. [comment-entry] ...]

ENVIRONMENT DIVISION

FORMAT:

```
ENVIRONMENT DIVISION.

[CONFIGURATION SECTION.

[SOURCE-COMPUTER. computer-name.]

[OBJECT-COMPUTER. computer-name.]

[SPECIAL-NAMES. [CONSOLE IS mnemonic-name]

        [, CURRENCY SIGN IS literal]

        [, DECIMAL-POINT IS COMMA]
```

```
      [, ASCII IS NATIVE]]. ]

[INPUT-OUTPUT SECTION.

 FILE-CONTROL.

      {file-control-entry.} . . .

[I-O-CONTROL.

      SAME AREA FOR file-name-1 , file-name-2 , ... ]]
```

► File-control Entry

FORMAT 1:

SELECT file-name

ASSIGN TO device

[; RESERVE integer-1 [AREA / AREAS]]

[; ORGANIZATION IS SEQUENTIAL]

[; ACCESS MODE IS SEQUENTIAL]

[; FILE STATUS IS data-name-1].

FORMAT 2:

```
SELECT file-name

    ASSIGN TO device

[; 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 data-name-2].
```

continued

FORMAT 3:

```
SELECT file-name

    ASSIGN TO device

[; RESERVE integer-1 [ AREA  ] ]
                     [ AREAS ]

    ; ORGANIZATION IS INDEXED

[; ACCESS MODE IS { SEQUENTIAL } ]
                  { RANDOM     }
                  { DYNAMIC    }

    ; RECORD KEY IS data-name-1
```

[; ALTERNATE RECORD KEY IS data-name-2 [WITH DUPLICATES]] ...

[; FILE STATUS IS data-name-3].

FORMAT 4:

SELECT file-name

ASSIGN TO device.

DATA DIVISION

FORMAT:

```
DATA DIVISION.

[FILE SECTION.

[FD file-name [UNCOMPRESSED]

    ; LABEL { RECORD IS   } { STANDARD }
           { RECORDS ARE } { OMITTED  }

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

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

[; VALUE OF FILE-ID IS literal-1]

[; OWNER IS literal-2]

[; DATA { RECORD IS / RECORDS ARE } data-name-1 [, data-name-2] ...]

[; CODE-SET IS ASCII].

[record-description-entry] ...] ...

[SD file-name

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

continued

```
[; DATA { RECORD IS  } data-name-1 [, data-name-2] . . . ] .
        { RECORDS ARE }
```

```
{record-description-entry } . . . ] . . . ]
```

```
[ WORKING-STORAGE SECTION.
  [ 77-level-description-entry   ]
  [ record-description-entry     ] . . . ]

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

▶ Record Description Entry

FORMAT 1:

```
level-number { data-name-1 } [; REDEFINES data-name-2]
             { FILLER      }

    [; OCCURS integer-1 TIMES

            [ { ASCENDING  } KEY IS data-name-3 [, data-name-4] . . .]
              { DESCENDING }

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

continued

```
[; { PICTURE } IS character-string]
   { PIC     }

[; [USAGE IS] { DISPLAY                } ]
              { COMPUTATIONAL          }
              { COMP                   }
              { INDEX                  }
              { COMPUTATIONAL-3        }
              { COMP-3                 }

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

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

[; { JUSTIFIED / JUST } RIGHT]

[; BLANK WHEN ZERO]

[; VALUE IS literal].

FORMAT 2:

66 data-name-1; **RENAMES** data-name-2 [{ THROUGH / THRU } data-name-3].

continued

FORMAT 3:

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

    [, literal-3 [ { THROUGH } literal-4]] . . . .
                   { THRU    }
```

FORMAT 4:

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

PROCEDURE DIVISION

FORMAT:

```
PROCEDURE DIVISION [USING data-name-1 [, data-name-2] . . . ] .

[DECLARATIVES.

{section-name SECTION. USE sentence.

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

END DECLARATIVES.]

{section-name SECTION.

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

► ACCEPT Statement

FORMAT 1:

ACCEPT **data-name** [FROM **mnemonic-name**]

FORMAT 2:

ACCEPT **data-name** **FROM** { DATE / DAY / TIME }

► ADD Statement

FORMAT 1:

[; ON SIZE ERROR imperative-statement]

FORMAT 2:

GIVING data-name-4 [ROUNDED] [; ON SIZE ERROR imperative-statement]

continued

FORMAT 3:

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

[ROUNDED] [; ON SIZE ERROR imperative-statement]
```

► ALTER Statement

FORMAT:

ALTER paragraph-name-1 TO [PROCEED TO] paragraph-name-2

► CALL Statement

FORMAT:

CALL literal-1 [USING data-name-1 [, data-name-2] . . .]

▶ CLOSE Statement

FORMAT:

CLOSE file-name-1 **[, file-name-2] ...**

▶ COMPUTE Statement

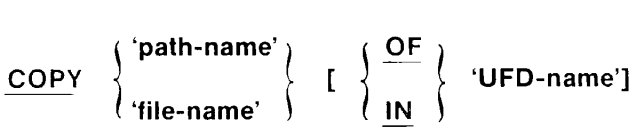
FORMAT:

COMPUTE **data-name-1** [ROUNDED] = { data-name-2 / numeric-literal / arithmetic-expression }

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

► COPY Statement

FORMAT:



► DELETE Statement

FORMAT:

DELETE file-name **RECORD [; INVALID KEY imperative-statement]**

► DISPLAY Statement

FORMAT:

```
DISPLAY { data-name          } [ , data-name          ]
        { literal            }   , literal            ] ...[UPON mnemonic-name]
        { figurative-constant }  [ , figurative-constant ]
```

► DIVIDE Statement

FORMAT 1:

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

[; ON SIZE ERROR imperative-statement]

FORMAT 2:

```
DIVIDE { data-name-1 } { INTO } { data-name-2 }  GIVING data-name-3 [ROUNDED]
       { literal-1   } { BY   } { literal-2   }
```

[; ON SIZE ERROR imperative-statement]

▶ ENTER Statement

FORMAT:

```
ENTER { COBOL     }
      { ASSEMBLER }
```

▶ EXHIBIT Statement

FORMAT:

```
EXHIBIT { literal              } ...
        { NAMED data-name      }
```

▶ EXIT Statement

FORMAT:

```
EXIT.
```

► EXIT PROGRAM Statement

FORMAT:

EXIT PROGRAM.

► GO TO Statement

FORMAT 1:

GO TO procedure-name.

continued

FORMAT 2:

GO TO procedure-name-1 [, **procedure-name-2**] ...

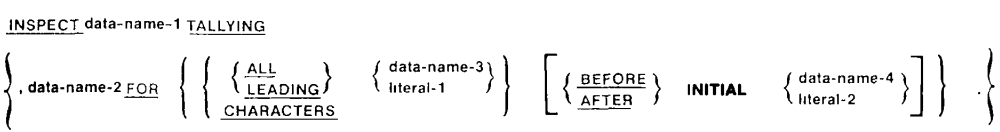
DEPENDING ON data-name.

▶ IF Statement

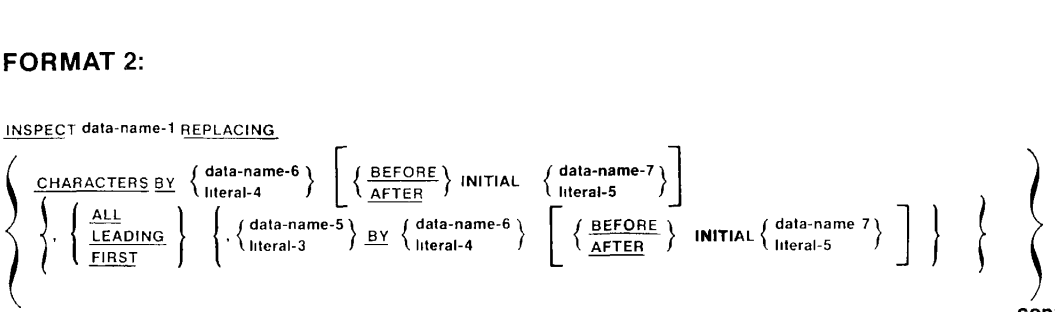
FORMAT:

► INSPECT Statement

FORMAT 1:

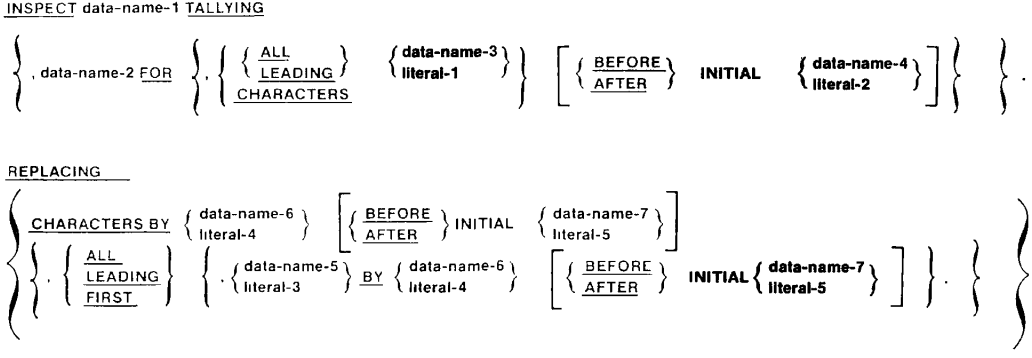


FORMAT 2:



continued

FORMAT 3:


INSPECT data-name-1 TALLYING
, data-name-2 FOR
ALL
LEADING
CHARACTERS
data-name-3
literal-1
BEFORE
AFTER
INITIAL
data-name-4
literal-2
REPLACING
CHARACTERS BY
data-name-6
literal-4
BEFORE
AFTER
INITIAL
data-name-7
literal-5
ALL
LEADING
FIRST
data-name-5
literal-3
BY
data-name-6
literal-4
BEFORE
AFTER
INITIAL
data-name-7
literal-5

► MOVE Statement

FORMAT 1:

```
MOVE { data-name-1 } TO data-name-2 [, data-name-3] . . .
     { literal     }
```

FORMAT 2:

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

▶ MULTIPLY Statement

FORMAT:

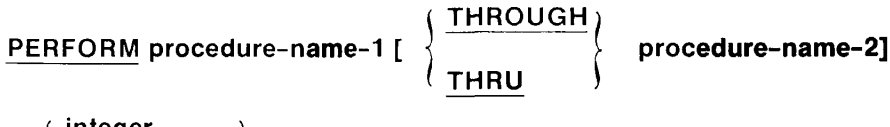
```
MULTIPLY { data-name-1       }
         { numeric-literal-1 }

BY { data-name-2 [GIVING data-name-3]  }
   { numeric-literal-2 GIVING data-name-3 }

[ROUNDED] [; ON SIZE ERROR imperative-statement]
```

► OPEN Statement

FORMAT:

► PERFORM Statement

FORMAT 1:



continued

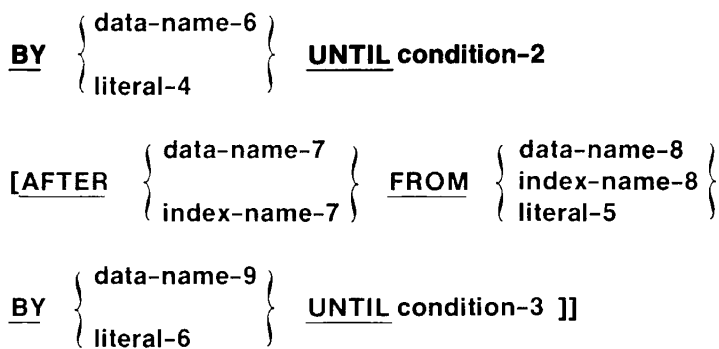
FORMAT 2:

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

[ VARYING { data-name-1  } FROM { data-name-2  } BY
          { index-name-1 }      { index-name-2 }
                                { literal-1    }

  { data-name-3 }
  { literal-2   } ] UNTIL condition-1
```

FORMAT 3:

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

VARYING { data-name-1  } FROM { data-name-2  }
        { index-name-1 }      { index-name-2 }
                              { literal-1    }

BY { data-name-3 } UNTIL condition-1
   { literal-2   }

[AFTER { data-name-4  } FROM { data-name-5  }
       { index-name-3 }     { index-name-4 }
                            { literal-3    }
```

continued


BY
data-name-6
literal-4
UNTIL condition-2
[AFTER
data-name-7
index-name-7
FROM
data-name-8
index-name-8
literal-5
BY
data-name-9
literal-6
UNTIL condition-3]]

► READ Statement

FORMAT 1:

READ file-name [NEXT] RECORD [INTO data-name-1]

[; AT END imperative-statement]

FORMAT 2:

READ file-name RECORD [INTO data-name-1] [; KEY IS data-name-2]

[; INVALID KEY imperative-statement]

▶ READY TRACE Statement

FORMAT:

READY TRACE.

▶ RESET TRACE Statement

FORMAT:

RESET TRACE.

► RELEASE Statement

FORMAT:

RELEASE **record-name [FROM data-name]**

► RETURN **Statement**

FORMAT:

RETURN file-name RECORD **[**INTO data-name**]**

; AT END imperative-statement

► REWRITE Statement

FORMAT:

REWRITE record-name [FROM data-**name]**

[; INVALID KEY imperative-statement]

▶ SEARCH Statement

FORMAT 1:

```
SEARCH identifier-1 [ VARYING { identifier-2   } ]
                              { index-name-1   }

[; AT END imperative-statement-1]

 ; WHEN condition-1 { imperative-statement-2 }
                   { NEXT SENTENCE          }

[; WHEN condition-2 { imperative-statement-3 } ]
                    { NEXT SENTENCE          }
```

continued

FORMAT 2:

```
SEARCH ALL identifier-1 [; AT END imperative-statement-1]
; WHEN { data-name-1      { IS EQUAL TO } { identifier-3 } }
     { condition-name-1    { IS =       } { literal-1    } }
[ AND { data-name-2      { IS EQUAL TO } { identifier-3 } } ]
      { condition-name-2  { IS =       } { literal-2    } }
   { imperative-statement-2 }
   { NEXT SENTENCE          }
```

► SET Statement

FORMAT 1:

```
SET  { index-name-1 [, index-name-2] ... }  TO  { index-name-3 }
     { data-name-1 [, data-name-2] ...    }      { data-name-3 }
                                                 { integer-1   }
```

FORMAT 2:

```
SET  index-name-4  [, index-name-5]  ...  { UP BY   }  { data-name-6 }
                                          { DOWN BY }  { integer-2   }
```

▶ SORT Statement

FORMAT:

```
SORT file-name-1 ON { ASCENDING  } KEY data-name-1 [, data-name-2] ...
                    { DESCENDING }

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

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

▶ START Statement

FORMAT:

```
START file-name [KEY IS [ { GREATER THAN     } ] data-name]
                          { NOT LESS THAN    }
                          { EQUAL TO         }
[; INVALID KEY imperative-statement]
```

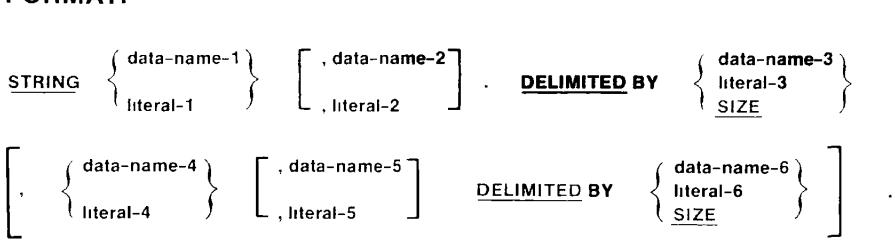
▶ STOP Statement

FORMAT:

```
STOP { RUN     }
     { literal }
```

▶ STRING Statement

FORMAT:



INTO data-name-7 [WITH POINTER data-name-8]

[; ON OVERFLOW imperative-statement]

► SUBTRACT Statement

FORMAT 1:

```
SUBTRACT { data-name-1 } [ , data-name-2 ] ...
         { literal-1   }   [ , literal-2  ]

FROM data-name-3 [ROUNDED]

[; ON SIZE ERROR imperative-statement]
```

continued

FORMAT 2:

```
SUBTRACT { data-name-1 } [ , data-name-2 ] ...
         { literal-1   }   [ , literal-2 ]

FROM { data-name-3 } GIVING data-name-4 [ROUNDED]
     { literal-3   }

[; ON SIZE ERROR imperative-statement]
```

FORMAT 3:

SUBTRACT { CORRESPONDING / CORR } **data-name-1**

FROM data-name-2 [ROUNDED]

[; ON SIZE ERROR imperative-statement]

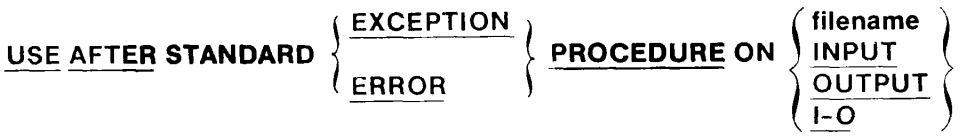
► UNSTRING Statement

FORMAT:

```
UNSTRING data-name-1

[ DELIMITED BY [ALL] { data-name-2 } [ , OR [ALL] { data-name-3 } ] ... ]
                     { literal-1   }               { literal-2   }

INTO data-name-4 [, DELIMITER IN data-name-5] [, COUNT IN data-name-6]

[, data-name-7 [, DELIMITER IN data-name-8] [, COUNT IN data-name-9] ]  ...

[WITH POINTER data-name-10] [TALLYING IN data-name-11]

[; ON OVERFLOW imperative-statement]
```

► USE Statement

FORMAT:



▶ WRITE Statement

FORMAT 1:

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

[ { AFTER  } ADVANCING { { data-name-2 } [ LINE  ] } ]
  { BEFORE }           { { integer      } [ LINES ] }
                       { PAGE           }
```

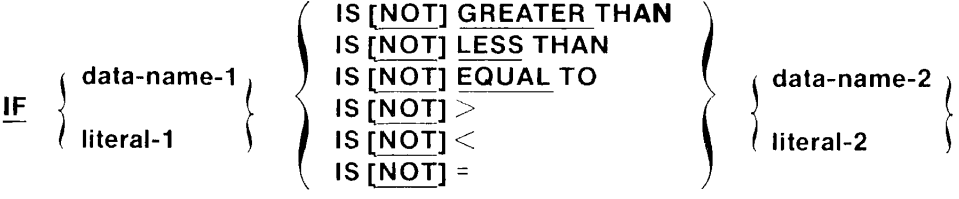
FORMAT 2:

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

[; INVALID KEY imperative-statement]
```

CONDITIONS

► Relation Condition

FORMAT:



▶ Class Condition

FORMAT:

IF data-name IS [NOT] { NUMERIC / ALPHABETIC }

▶ Condition-name Condition

FORMAT:

IF [NOT] condition-name

► Sign Condition

FORMAT:

IF data-name IS [NOT] { NEGATIVE / ZERO / POSITIVE }

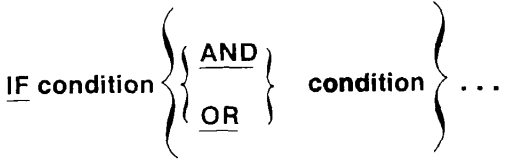
► Negated **Simple Condition**

FORMAT:

IF NOT **simple-condition**

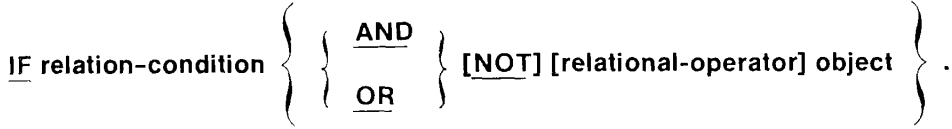
► **Combined Condition**

FORMAT:



► **Abbreviated Combined Relation Condition**

FORMAT:



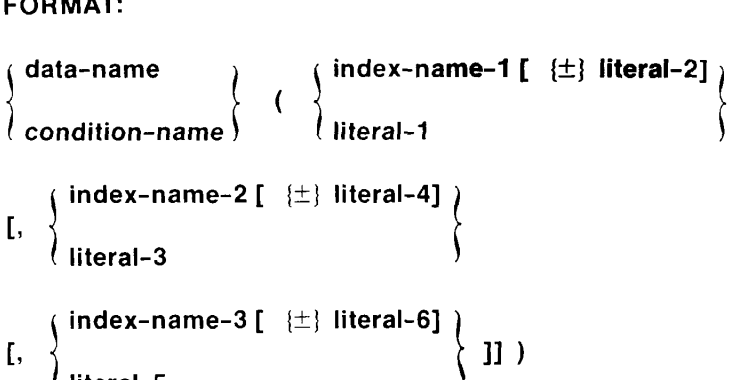
SUBSCRIPTING

FORMAT:

(subscript-1 [, subscript-2 [, subscript-3]])

INDEXING

FORMAT:



COBOL CHARACTER SET

CLASS		CHARACTER	MEANING	SPECIAL USAGE
numeric		0 1 9 figurative LOW-VALUE(S) constants ZERO/ZEROS/ZEROES	digit value (null) value (zero)	COBOL word formation figurative constant figurative constant
alphabetic		A B / space figurative SPACE(S) constants	letter blank value (blank)	COBOL word formation punctuation figurative constant
alpha-numeric		+	plus sign	sign symbol; arithmetic editing
		-	minus sign	sign symbol; arithmetic editing; symbol in being COBOL word formation
		*	asterisk	arithmetic operation; editing
		=	equal sign	arithmetic relation test; editing
		$	currency sign	editing
		,	comma	punctuation; editing
		;	semicolon	punctuation
		.	period	punctuation
		"	quotation mark	punctuation
		(	open parenthesis (opening delimiter of subscript)	punctuation
		)	close parenthesis (closing delimiter of subscript)	punctuation
		<	less than	relation test
		>	greater than	relation test
		figurative QUOTE(S) constants	value (quotation)	figurative constant
		figurative HIGH-VALUE(S) constants	value (highest)	figurative constant

Note

When the figurative constant LOW-VALUES is used with binary data, it is interpreted as numeric. In all other instances, it is interpreted as alphanumeric.

REFERENCE TABLES OF COBOL

Coding Format

1 6 The **sequence number area** may contain a six digit sequence number or blanks

7 The **special coding symbol area** may contain a blank or one of three symbols

- * indicates the current line is a comment line
- / indicates the next line will be printed at the top of a new page of the program listing
- indicates the current line is the continuation of a non numeric literal

8 11 All division section paragraph and procedure names must begin in the **A area** Level numbers may appear in the A area but are not required to

12 72 All other program elements must be confined to **B area**

73 80 The **identification area** is ignored by the compiler Frequently it is used to contain the program identification

Classes and Categories of Data

Level of Item	Class	Category
Elementary	Alphabetic	Alphabetic
	Numeric	Numeric Numeric Edited
	Alphanumeric	Alphanumeric Edited Alphanumeric
Nonelementary (Group)	Alphanumeric	Alphabetic Numeric Numeric Edited Alphanumeric Edited Alphanumeric

Data Representation and Usage

Usage Is	Machine Description
DISPLAY	EXTERNAL DECIMAL
COMPUTATIONAL	BINARY
INDEX	BINARY
COMPUTATIONAL-3	PACKED DECIMAL

Arithmetic Operators

Operator	Meaning
Binary:	
+	Addition
-	Subtraction
*	Multiplication
/	Division
Unary:	
+	The effect of multiplication by numeric literal +1
-	The effect of multiplication by numeric literal -1

Logical Operators

Operator	Meaning
AND	Logical conjunction, the truth value is "true" if both of the conjoined conditions are true, "false" if one or both of the conjoined conditions is false

continued

OR	Logical inclusive OR the truth value is 'true' if one or both of the included conditions is true, 'false' if both included conditions are false
NOT	Logical negation is the reversal of the truth value, i e , the truth value is 'true" if the condition is false and false if condition is true

Relational Operators

Operator	Meaning
=	is equal to
EQUAL	
<	is less than
LESS	
>	is greater than
GREATER	
NOT=	is not equal to
NOT EQUAL	
NOT<	is greater than or equal to
NOT LESS	
NOT>	is less than, or equal to
NOT GREATER	

Permissible Symbols in PICTURE Clause

Symbol	Meaning
A	Any alphabetic character or space
B	Space insertion character
P	Assumed decimal scaling position
S	Operational sign
V	Assumed decimal point
X	Any character
Z	Zero suppression and space replacement character
9	Any numeric character
/	Stroke insertion character
,	Comma insertion character
.	Decimal point
+	Plus sign insertion character
-	Minus sign insertion character
CR	Credit placement characters
DB	Debit placement characters

*	Zero suppression and asterisk replacement character
$	Currency sign insertion character

Sign Representation

SIGN Clause	Sign Representation
TRAILING	Embedded in rightmost byte
LEADING	Embedded in leftmost byte
TRAILING SEPARATE	Stored in separate rightmost byte
LEADING SEPARATE	Stored in separate leftmost byte

Results of Sign Control Symbols in Editing

Editing Symbol In Picture-String	RESULT	
	Data Item Positive Or Zero	Data Item Negative
+	+	-
-	space	-
CR	2 spaces	CR
DB	2 spaces	DB

Prime COBOL Reserved Words

ACCEPT
ACCESS
ADD
ADVANCING
AFTER
ALL
ALPHABETIC
ALTER
ALTERNATE
AND
ARE
AREA
AREAS
ASCII*
ASSEMBLER*
ASSIGN
AT
AUTHOR
BEFORE
BLANK
BLOCK
BY
CALL
CHARACTER
CHARACTERS
CLOSE
COBOL
CODE
CODE-SET
COMMA
COMP
COMP-3*
COMPUTATIONAL
COMPUTATIONAL-3*
COMPUTE
CONFIGURATION
CONSOLE*
CONTAINS

continued

COPY
CORR
CORRESPONDING
COUNT
CURRENCY

DATA
DATE
DATE COMPILED
DATE WRITTEN
DAY
DECIMAL POINT
DECLARATIVES
DELETE
DELIMITED
DELIMITER
DEPENDING
DISPLAY
DIVIDE
DIVISION
DOWN
DUPLICATES
DYNAMIC

ELSE
END
ENTER
ENVIRONMENT
EQUAL
ERROR
EVERY
EXCEPTION
EXHIBIT
EXIT
EXTEND

FD
FILE
FILE CONTROL
FILE ID
FILLER
FIRST
FOR
FROM

GIVING
GO
GREATER

HIGH VALUE
HIGH VALUES

I O
I—O CONTROL
ID
IDENTIFICATION
IF
IN
INDEX
INDEXED
INITIAL
INPUT
INPUT OUTPUT
INSPECT
INSTALLATION
INTO
INVALID
IS

JUST
JUSTIFIED

KEY

LABEL
LEADING
LEFT
LENGTH
LESS
LINE
LINES
LINKAGE
LOW VALUE
LOW VALUES

MODE
MOVE
MTS*
MULTIPLY

NAMED
NATIVE
NEGATIVE
NEXT
NOT
NUMBER
NUMERIC

OBJECT COMPUTER
OCCURS
OF
OFF
OFFLINE—PRINT*
OMITTED
ON
OPEN
OR
ORGANIZATION
OUTPUT
OVERFLOW
OWNER

PAGE
PERFORM
PFMS*

PIC
PICTURE
POINTER
POSITION
POSITIVE
PRINTER*
PROCEDURE
PROCEDURES
PROCEED
PROGRAM
PROGRAM ID
PUNCH*

QUOTE
QUOTES

RANDOM
READ
READER*
READY*
RECORD
RECORDS
REDEFINES
RELATIVE
RELEASE
REMARKS*
RENAMES
REPLACING
RESERVE
RETURN
REWRITE
RIGHT
ROUNDED
RUN

SAME
SEARCH
SECTION
SECURITY
SELECT
SENTENCE
SEPARATE
SEQUENTIAL
SET
SIGN
SIZE
SORT
SOURCE COMPUTER
SPACE
SPACES
SPECIAL NAMES
STANDARD
START
STATUS
STOP
STRING
SUBTRACT
SYNC
SYNCHRONIZED

TABLE
TALLYING
TAPE
TERMINAL
THAN
THROUGH
THRU
TIME
TIMES
TO
TRACE
TRAILING

UNCOMPRESSED*
UNSTRING
UNTIL
UP
UPON
USAGE
USE
USING

VALUE
VALUES
VARYING

WHEN
WITH
WORDS*
WORKING STORAGE
WRITE

ZERO
ZEROES
ZEROS

+
-
*
/
>
<

*Prime extension to ANSI standard

ASCII Character Set

ASCII Character	Hexadecimal	Octal	Punched Cards
NUL (Low value)	80	200	
(space)	A0	240	No Punch
! (exclamation)	A1	241	12 8 2
" (quote)	A2	242	7 8
# (number)	A3	243	8 3
$	A4	244	11 8 3
' (apostrophe)	A7	247	5 8
(	A8	250	12 5 8
)	A9	251	11 5 8
*	AA	252	11 4 8
+	AB	253	12 6 8
, (comma)	AC	254	0 3 8
- (minus)	AD	255	11
. (period)	AE	256	12 3 8
/ (virgule, slash, stroke)	AF	257	0 1
0 (zero)	B0	260	0
1	B1	261	1
2	B2	262	2
3	B3	263	3
4	B4	264	4
5	B5	265	5
6	B6	266	6
7	B7	267	7
8	B8	270	8
9	B9	271	9
: (colon)	BA	272	8 2
; (semicolon)	BB	273	11 6 8
<	BC	274	12 4 8
=	BD	275	6 8
>	BE	276	0 6 8
?	BF	277	0 7 8
@	C0	300	8 4
A	C1	301	12 1
B	C2	302	12 2
C	C3	303	12 3
D	C4	304	12 4
E	C5	305	12 5
F	C6	306	12 6
G	C7	307	12 7
H	C8	310	12 8
I	C9	311	12 9
J	CA	312	11 1
K	CB	313	11 2
L	CC	314	11 3
M	CD	315	11 4
N	CE	316	11 5
O	CF	317	11 6
P	D0	320	11 7
Q	D1	321	11 8

R	D9	329	11 9
S	E2	330	0 2
T	E3	331	0 3
U	E4	332	0 4
V	E5	333	0 5
W	E6	334	0 6
X	E7	335	0 7
Y	E8	336	0 8
Z	E9	337	0 9
a	81	201	
b	82	202	
c	83	203	
d	84	204	
e	85	205	
f	86	206	
g	87	207	
h	88	210	
i	89	211	
j	91	221	
k	92	222	
l	93	223	
m	94	224	
n	95	225	
o	96	226	
p	97	227	
q	98	230	
r	99	231	
s	A2	242	
t	A3	243	
u	A4	244	
v	A5	245	
w	A6	246	
x	A7	247	
y	A8	250	
z	A9	251	
0 (+zero)	C0	300	12 0
0 (-zero)	D0	320	11 0
DEL (high value)	FF	377	

Note

(Characters with no Punched Cards code are not supported for punched card entry)

File Status Code Definitions

File Organization	Status Code	Error Type	Interpretation
SEQUENTIAL	00	None	Successful completion of operation
	10	End of File	End of file reached on READ operation file pointer positioned past logical end of file
	30	Permanent I/O Error	Operation unsuccessful due to an I/O error such as data check parity error or transmission error
	34	Permanent I/O Error	Operation unsuccessful due to a boundary violation disk space full
	41	Forms Error	FORMS validation error on a READ
RELATIVE	00	None	Successful completion of operation
	10	End of File	End of file encountered during a READ
	21	Invalid Key	User has attempted to write beyond predefined boundaries of the file allocated by CREATK
	22	Invalid Key	Record already exists in data subfile user attempted to add a record with a non unique record number
	3	Invalid key	Record not found no record found with the specified key value
	24	Invalid Key	Boundary violation disk space full
	30	Permanent I/O Error	Permanent I/O error could be a parity error data check or transmission error
	90	Prime defined conditions are codes 90 99	Locked record attempt to access a record that is locked by another user or process
	91		Unlocked record REWRITE attempted without first locking the record with a READ
	94		MIDAS concurrency error in other user has deleted the record you were trying to access
	95		Relative record number error user supplied a record number larger than the number pre allocated by CREATK
	98		Attempt to do an indexed add to a direct access file can't add entries to a RELATIVE file even if it's opened for INDEXED access

	39		System error possibly serious Verify that error is not due to a START that encountered a locked record before calling your System Analyst
INDEXED	00	NONE	Successful completion of operation
	10	End of File	End of file reached on READ operation file pointer positioned past logical end of file
	22	Invalid Key	Attempt to perform a WRITE or REWRITE which would create a duplicate primary key entry Duplicate primary key values are illegal
	23	Invalid Key	Record not found on an unsuccessful key search there is no record in the file with this key value
	30	System I O	Operation unsuccessful due to an I O error such as a data check parity error or a transmission error
	90	Prime defined conditions are codes 90 99	Record already locked another user or process has already locked this record for update
	90		Record not locked a REWRITE operation was attempted without first locking the record via a READ operation
	91		Attempt to add a duplicate secondary key value to a secondary index subfile that does not permit duplicates
	93		The indexes referred to in the program do not match those defined during template creation
	94		MIDAS concurrency error another user has deleted the record you were trying to perform some operation on
	95		Bad record length supplied the program has incorrectly specified the record length (data size) of the MIDAS file
	99	System Error	System error could be serious trouble Verify that the program is not seriously flawed before you call your System Analyst

Permissible Input/Output Statements — OPEN Statements and Access Modes

File Organization	File Access Mode	Procedure Statement	Option			
			Input	Output	I-O	Extend
Sequential	SEQUENTIAL	READ	X		X	
		WRITE		X		X
		REWRITE			X	
Indexed Relative	SEQUENTIAL	READ	X		X	
		WRITE		X		
		REWRITE			X	
		START	X		X	
		DELETE			X	
	RANDOM	READ	X		X	
		WRITE		X	X	
		REWRITE			X	
		START				
		DELETE			X	
	DYNAMIC	READ	X		X	
		WRITE		X	X	
		REWRITE			X	
		START	X		X	
		DELETE			X	

Permissible MOVEs

SENDING DATA ITEM	RECEIVING DATA ITEM: ALPHABETIC	BINARY	ALPHANUMERIC EDITED	NUMERIC INTEGER / NUMERIC NONINTEGER	NUMERIC EDITED	ALPHANUMERIC
ALPHABETIC	X			X		X
BINARY		X		X	X	X(1)
ALPHANUMERIC EDITED	X			X(3)		X
NUMERIC INTEGER		X		X	X	X(2)
NUMERIC NONINTEGER				X	X	
NUMERIC EDITED				X(3)		X(3)
ALPHANUMERIC	X			X	X(4)	X

Notes

1. If receiving operand length L is less than or equal to 18 target Picture 9(L) is assumed. Otherwise the MOVE is disallowed.
2. The source is converted to DISPLAY form with separate trailing sign (blank for positive) then moved as a character string source subject to truncation or blank padding depending on receiving item's length.
3. The source is considered as a character string.
4. If source length L is less than or equal to 18 source Picture 9 (L) is assumed. Otherwise the MOVE is disallowed.

COMPILING

COBOL pathname [-option-1] [-option-n]

Invokes the COBOL compiler

Options: (• indicates Prime-supplied defaults)

Specify Input/Output Devices.

	pathname
-BINARY	NO
	• YES

Specifies binary (object) file. Default name file name BIN or B_filename

-INPUT pathname

Specifies the source file

	pathname
	• NO
-LISTING	YES
	TTY
	SPOOL

Specifies listing file. Default name filename LIST or L_filename

-SOURCE pathname

Same as **INPUT**

Enable Expanded Listings/Cross References:

• -NOEXPLIST	Suppresses expanded listing
-EXPLIST	Prints listing including assembler-like output
• -NOXREF	Suppresses cross reference listing
-XREF	Generates cross-reference listing at the end of listing file

Specify Addressing Mode:

• -64V	Generates code to run in 64V mode

RUN-TIME ASSIGNMENTS

The file assignments facility allows the user to override at run time filenames as stated in the VALUE OF FILE-ID clauses of FDs After invoking the run-time image the utility program CSIN will prompt for file assignments and wait for user input

```
ENTER FILE ASSIGNMENTS
>
```

The user may enter the name of the file as defined in the FILE ID of the FD, an equal sign = and then the pathname or tape address of the actual file to be associated with the ID The pathname can be a filename if the file resides in the current UFD Formats are

```
disk file-id = pathname
tape file id = $MTx, label type tape id, tape name
```

There should be one entry (preceded by prompt character >) for each FD if its FILE-ID is to be reassigned or overridden When no files remain to be entered the user inserts the single slash character to conclude the session Execution of the applications program will then begin using the files which were just assigned