UNIVAC® 1108

MULTI-PROCESSOR SYSTEM

COBOL UNDER EXEC 8 REFERENCE CARD

REFERENCE CARD NOTATION

Complete details on UNIVAC 1108 COBOL are covered in *UNIVAC 1108 COBOL Under EXEC 8 Programmers Reference Manual, UP-7626* (current version).

> ***CAUTION TO COBOL UNDER EXEC II USER***
>
> **THERE ARE SLIGHT DIFFERENCES BETWEEN COBOL UNDER EXEC II AND COBOL UNDER EXEC 8. THESE DIFFERENCES ARE IN:**
>
> ***AREA:*** ***RESERVED WORDS, COBOL CONTROL CARD OPTIONS***
> ***FORMATS:*** ***FILE-CONTROL, RECORD DESCRIPTION, ADD, ALTER, CLOSE, DIVIDE, ENTER, INCLUDE, OPEN, SEEK, TABLE HANDLING***
>
> **SEE:**
> ***UNIVAC 1108 COBOL UNDER EXEC II** Programmers Reference Manual, UP-4048* **(current version) for details in these areas.**

CHARACTER SET IN COLLATING SEQUENCE

NAME	SYMBOL	FIELDATA CODE	80-COLUMN CARD CODE	SOURCE LANGUAGE USAGE
		00	7-8	
		01	12-5-8	
		02	11-5-8	
		03	12-7-8	
		04	11-7-8	
SPACE	ƀ, or blank	05	Blank	PUNCTUATION, EDITING
LETTERS	A thru Z	06 thru 37	A thru Z	WORDS
RIGHT PAREN	)	40	12-4-8	PUNCTUATION, GROUPING
MINUS/HYPHEN	–	41	11	WORDS, EDITING, ARITHMETIC
PLUS	+	42	12	EDITING, ARITHMETIC
LESS THAN	<	43	12-6-8	RELATION
EQUALS	=	44	3-8	RELATION
GREATER THAN	>	45	6-8	RELATION
		46	2-8	
DOLLAR	$	47	11-3-8	EDITING
ASTERISK	*	50	11-4-8	EDITING, ARITHMETIC
LEFT PAREN	(	51	0-4-8	PUNCTUATION, GROUPING
		52	0-5-8	
		53	5-8	
		54	12-0	
		55	11-0	
COMMA	,	56	0-3-8	PUNCTUATION, EDITING
		57	0-6-8	
NUMBERS	0 thru 9	60 thru 71	0 thru 9	WORDS, EDITING, ARITHMETIC
APOSTROPHE	'	72	4-8	PUNCTUATION, QUOTE
SEMICOLON	;	73	11-6-8	PUNCTUATION
SLASH	/	74	0-1	ARITHMETIC
PERIOD	.	75	12-3-8	PUNCTUATION, EDITING
		76	0-7-8	
		77	0-2-8	

NOTE: Any Fieldata character can be used as data. Source language uses only those named.

UP-7527 Rev. 2

RULES FOR EFFICIENCY

The following rules are optional, but following them reduces the running time of the object code.

1. When using MOVE, arithmetic, or conditional statements:
   - establish data fields that are multiples of six characters;
   - make the sending and receiving fields the same size;
   - align the decimal positions of sending and receiving fields;
   - avoid moving group items with mixed class and/or usage; i.e., alphanumeric and numeric;
   - when moving a literal to a computational field, make fields the same size.
   - avoid subscripting.
2. When subscripting cannot be avoided:
   - use fixed subscripts (numeric literal);
   - make subscripted items multiples of six characters;
   - define subscripts as COMP-1.
3. Avoid relationship tests where:
   - either value is subscripted;
   - operands are not congruent;
   - fields with different class and/or usage are compared;
   - point locations are not aligned in computational operands.
4. Define an item as COMP-1 if used primarily in arithmetic expressions.

GENERAL NOTES:

1. All arithmetic expressions are evaluated in double precision floating point mode.
2. Figurative constants allowed are:

```
ZERO [S ] = 0 or 0's     computational mode (code 60)
     [ES]                computational-1 mode (binary 0)

SPACE [S]     = blank or blanks (code 05)

QUOTE [S]     = quotation mark or marks (code 72)

HIGH VALUE [S]   = period (code 75)

LOW VALUE [S]    = spaces (code 05)

ALL 'any literal'   = a sequence of 'any literal'
```

3. The following fixed data-names (special registers) are implicit to each COBOL program:
   - TALLY (signed) PIC SH9(5) SYNC RIGHT
   - MONITOR-DATE current date in Fieldata, (YYDDD) PIC 9(5)

```
IDENTIFICATION DIVISION.

PROGRAM-ID. program-name.

[AUTHOR. author-name.]

[INSTALLATION. comment paragraph.]

[DATE-WRITTEN. comment paragraph.]

[DATE-COMPILED. comment paragraph.]

[SECURITY. comment paragraph.]

[REMARKS. comment paragraph.]
```

```
ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

Format 1:
[SOURCE-COMPUTER. COPY library-name.]

Format 2:
[SOURCE-COMPUTER. UNIVAC-1108.]

Format 1:
[OBJECT-COMPUTER. COPY library-name.]

Format 2:
[OBJECT-COMPUTER. UNIVAC-1108.                                                  ]
[     [WITH SUPERVISOR CONTROL]                                                 ]
[     [, MEMORY SIZE integer WORDS]                                             ]
[     [, [literal-1] hardware-name-1 [, [literal-2] hardware-name-2] . . . ] .  ]

Format 1:
[SPECIAL-NAMES. COPY library-name.]

Format 2:
[SPECIAL-NAMES. hardware-name-1 IS mnemonic-name-1
      [, hardware-name-2 IS mnemonic-name-2] . . . ] .

INPUT-OUTPUT SECTION.

Format 1:
[FILE-CONTROL. COPY library-name.]

Format 2:
[FILE-CONTROL. {SELECT [OPTIONAL] file-name-1 [RENAMING file-name-2]                            ]
[    ASSIGN TO {mnemonic-name} file-system-name                                                 ]
[              {hardware-name}                                                                  ]
[    [FOR MULTIPLE REEL]  [, RESERVE {integer} ALTERNATE [AREA ]]                               ]
[                                    {NO     }           [AREAS]]                               ]
[    [                   {SEQUENTIAL}]                                                          ]
[    [, ACCESS MODE IS   {RANDOM    }] [, ACTUAL KEY IS data-name-1]                            ]
[    [                   {MIXED     }]                                                          ]
[    [, SYMBOLIC KEY IS data-name-2].} . . .                                                    ]

Format 1:
[I-O-CONTROL. COPY library-name.]

Format 2:
[ I-O-CONTROL.                                                                  ]
[       [APPLY {DEMAND } ON file-name-1 [, file-name-2] . . . ] . . . .         ]
[       [      {STANDBY}                                      ]                 ]

Format 3:
[I-O-CONTROL.                                                                                 ]
[APPLY RERUN [ON file-name-1] EVERY {integer RECORD[S]} OF file-name-2.                       ]
[                                   {END OF REEL      }                                       ]
```

DATA DIVISION.

FILE SECTION.

Format 1:

```
[FD file-name COPY library-name.]
```

Format 2:

```
[FD file-name
 [; RECORDING MODE IS {BLANK [SIGN] | SIGN | XS3}]
 [; BLOCK CONTAINS [integer-1 {RECORD[S] | CHARACTER[S]}] [, integer-2 CONTROL WORD[S]]]
 [; FILE CONTAINS ABOUT integer-3 RECORD[S]]
 [; RECORD CONTAINS [integer-4 CHARACTER[S]] [, integer-5 CONTROL WORD[S]]]
 LABEL {RECORD IS | RECORDS ARE} {OMITTED | STANDARD | format definition}
 [; VALUE OF { {ID | IDENTIFICATION} IS {data-name-3 [, data-name-4] | literal-1}
               DATE-WRITTEN IS {data-name-5 | literal-2}
               LINES-PER-PAGE IS integer-6, LINES-AT-TOP
                  IS integer-7, LINES-AT-BOTTOM IS
                  integer-8 [, LINE-SPACING IS integer-9] }]
 DATA {RECORD IS | RECORDS ARE} data-name-1 [, data-name-2] . . . ]
```

Format 3:

```
[SD file-name COPY library-name.]
```

Format 4:

```
[SD file-name DATA {RECORD IS | RECORDS ARE} data-name-1 [, data-name-2] . . .
    FILE CONTAINS ABOUT integer RECORD[S]]
```

RECORD DESCRIPTION

Format 1:

```
[level-number {FILLER | data-name-1} [; REDEFINES data-name-2]
 [; {PIC | PICTURE} IS character-string]
 [; USAGE IS {COMP | COMP-1 | COMPUTATIONAL | COMPUTATIONAL-1 | DISPLAY | INDEX}]
 [; SIZE IS integer-1 {CHARACTERS | DIGITS}]
 [; POINT LOCATION IS {LEFT | RIGHT} integer-2 PLACE[S]]
 [; CLASS IS {ALPHABETIC | NUMERIC | ALPHANUMERIC | AN}] [;SIGNED]
 [; {ZERO SUPPRESS | CHECK PROTECT | FLOAT DOLLAR SIGN}] [[LEAVING integer-3 PLACE[S]] [BLANK WHEN ZERO]]
 [; OCCURS integer-4 TIME[S] [, {ASCENDING | DESCENDING} KEY IS data-name-1 [, data-name-2] . . .] . . .
    [INDEXED BY index-name-1 [, index-name-2] . . .]]
 [; {JUST | JUSTIFIED} RIGHT] [; {VALUE IS | VALUES ARE} literal-1]
 [; {SYNCHRONIZED | SYNC} {LEFT | RIGHT}] ]
```

Format 2:

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

Format 3: (For copying within the Data Division)

```
[level-number data-name-1 COPY data-name-2]
```

Format 4: (For copying from the Library)

```
[01 data-name-1 COPY data-name-2 FROM LIBRARY]
```

```
[COMMON-STORAGE SECTION.
 level-number data-name [descriptive clauses].]

[WORKING-STORAGE SECTION.
 level-number data-name [descriptive clauses].]

[CONSTANT SECTION.
 level-number data-name [descriptive clauses].]
```

PROCEDURE DIVISION.

```
[DECLARATIVES.
 section-name SECTION.
```

Format 1:

```
 USE AFTER STANDARD ERROR PROCEDURE ON {file-name-1 [, file-name-2] . . . | INPUT | OUTPUT | INPUT-OUTPUT | I-O}
```

Format 2:

```
 USE {BEFORE | AFTER} STANDARD [{BEGINNING | ENDING}] [REEL | FILE]
     LABEL PROCEDURE ON {file-name-1 [, file-name-2] . . . | INPUT | OUTPUT | INPUT-OUTPUT | I-O}
```

Format 3:

```
 USE FOR KEY CONVERSION ON {ALL | file-name-1 [, file-name-2] . . .}
```

Format 4:

```
 USE FOR ENTRY POINTS procedure-name-1 [, procedure-name-2] . . .
     [section-name-2 SECTION] . . .
 END DECLARATIVES.]
```

```
[section-name SECTION [priority-number] ]paragraph-name-1. sentence-1 [sentence-2] . . .
    [paragraph-name-2.]
        .
        .
        .
```

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

VERBS AND STATEMENTS (listed alphabetically)

```
ACCEPT identifier [FROM mnemonic-name]
```

Format 1:

```
ADD {literal-1 | identifier-1} [, {literal-2 | identifier-2}] . . . TO identifier-m
    [ROUNDED] [identifier-n [ROUNDED]] . . . [; ON SIZE ERROR imperative-statement]
```

Format 2:

```
ADD {literal-1 | identifier-1} {literal-2 | identifier-2} [{literal-3 | identifier-3}] . . . GIVING identifier-m [ROUNDED]
    [, identifier-n [ROUNDED]] . . . [; ON SIZE ERROR imperative-statement]
```

Format 3:

ADD {CORR | CORRESPONDING} *data-name-1* TO *data-name-2* [ROUNDED]

[; ON SIZE ERROR *imperative-statement*]

ALTER *procedure-name-1* TO [PROCEED TO] *procedure-name-2* [, *procedure-name-3*
TO [PROCEED TO] *procedure-name-4*] . . .

CLOSE *file-name-1* [REEL] [WITH {[NO]REWIND | LOCK}] [, *file-name-2*] . . .

COMPUTE *identifier-1* [ROUNDED] [, *identifier-2* [ROUNDED]] . . . {FROM | = | EQUALS}
{*identifier-n* | *literal* | *arithmetic-expression*} [; ON SIZE ERROR *imperative-statement*]

DISPLAY {*literal-1* | *identifier-1*} [, {*literal-2* | *identifier-2*}] . . . [UPON *mnemonic-name*]

Format 1:

DIVIDE {*identifier-1* | *literal-1*} INTO *identifier-2* [ROUNDED] [, *identifier-3* ROUNDED] . . .

[; ON SIZE ERROR *imperative-statement*]

Format 2:

DIVIDE {*identifier-1* | *literal-1*} INTO {*identifier-2* | *literal-2*} GIVING *identifier-3* [ROUNDED]

[, *identifier-4* [ROUNDED]] . . . [; ON SIZE ERROR *imperative-statement*]

Format 3:

DIVIDE *identifier-1* BY {*identifier-2* | *literal-1*} [; ON SIZE ERROR *imperative-statement*]

Format 4:

DIVIDE {*identifier-1* | *literal-1*} BY {*identifier-2* | *literal-2*} GIVING *identifier-3* [ROUNDED]

[, *identifier-4* [ROUNDED]] . . . [; ON SIZE ERROR *imperative-statement*]

ENTER [FORTRAN] *routine-name* SUBROUTINE [REFERENCING {*literal-1* | *identifier-1* | *file-name-1*}

[{*literal-2* | *identifier-2* | *file-name-2*}] . . .]

Format 1:

EXAMINE *identifier* TALLYING {ALL | LEADING | UNTIL FIRST} *literal-1* [REPLACING BY *literal-2*]

Format 2:

EXAMINE *identifier* REPLACING {ALL | LEADING | [UNTIL] FIRST} *literal-1* BY *literal-2*

EXECUTE *procedure-name-1* [THRU *procedure-name-2*]

EXIT.

Format 1:

GO TO [*procedure-name-1*]

Format 2:

GO TO *procedure-name-1* [*procedure-name-2*] [, *procedure-name-3*] . . . DEPENDING ON *identifier*.

IF *condition-1* [{AND | OR} *condition-2*] . . . [{THEN | ;}] {*statement-1* | NEXT SENTENCE} [{THEN | ;}]

[{ELSE | OTHERWISE} {*statement-2* | NEXT SENTENCE}]

Condition may be based on:

- a condition-name defined by an 88-level entry in the DATA DIVISION
- a class test:

  IF *identifier* IS [NOT] {NUMERIC | ALPHABETIC}
- a relational test:

  IF {*identifier-1* | *literal-1* | *arithmetic-expression-1*} {IS [NOT] GREATER THAN | IS [NOT] LESS THAN | IS [NOT] EQUAL TO | IS UNEQUAL TO | EQUALS | EXCEEDS | IS [NOT] = | IS [NOT] > | IS [NOT] <} {*identifier-2* | *literal-2* | *arithmetic-expression-2*}
- a sign test:

  IF {*identifier* | *arithmetic-expression*} IS[NOT] {POSITIVE | NEGATIVE | ZERO}

Format 1:

{INCLUDE | COPY} *procedure-name*

Format 2:

{INCLUDE | COPY} *paragraph-name* {IN | OF} *procedure-name*

Format 3:

{INCLUDE | COPY} *procedure-name* SECTION

MONITOR *identifier-1* [, *identifier-2*] . . .

Format 1:

MOVE {*identifier-1* | *literal*} TO *identifier-2* [, *identifier-3*] . . .

Format 2:

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

Format 1:

MULTIPLY {*identifier-1* | *literal-1*} BY *identifier-2* [ROUNDED] [, *identifier-3* [ROUNDED]] . . .

[; ON SIZE ERROR *imperative-statement*]

Format 2:

MULTIPLY {*identifier-1* | *literal-1*} BY {*identifier-2* | *literal-2*} GIVING *identifier-3* [ROUNDED]

[, *identifier-4* [ROUNDED]] . . . [; ON SIZE ERROR *imperative-statement*]

NOTE *character-string*

Format 1:

OPEN INPUT *file-name-1* [WITH [NO] REWIND | REVERSED] [, *file-name-2* [WITH [NO] REWIND | REVERSED]] . . .

Format 2:

OPEN OUTPUT *file-name-1* [WITH [NO] REWIND] [, *file-name-2* [WITH [NO] REWIND]] . . .

Format 3:

OPEN {INPUT-OUTPUT | I-O} *file-name-1* [, *file-name-2*] . . .

Format 1:

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

Format 2:

PERFORM *procedure-name-1* [THRU *procedure-name-2*] {*identifier* | *integer*} TIME[S]

Format 3:

PERFORM *procedure-name-1* [THRU *procedure-name-2*] UNTIL *condition-1*

Format 4:

```
                                                               { index-name-1 }
PERFORM procedure-name-1 [THRU procedure-name-2] VARYING       { identifier-1 }

         { index-name-2 }       { identifier-3 }
   FROM  { identifier-2 }   BY  { literal-2    }   UNTIL condition-1
         { literal-1    }

  [        { identifier-4 }         { index-name-4 }       { identifier-6 }
  [ AFTER  { index-name-3 }   FROM  { identifier-5 }   BY  { literal-4    }   UNTIL condition-2
  [                                 { literal-3    }

  [        { identifier-7 }         { index-name-6 }       { identifier-9 }                   ]
  [ AFTER  { index-name-5 }   FROM  { identifier-8 }   BY  { literal-6    }   UNTIL condition-3]]
                                    { literal-5    }
```

Format 1:

```
READ file-name RECORD [INTO identifier]  ; AT END imperative-statement
```

Format 2:

```
READ file-name RECORD [INTO identifier]  ; INVALID KEY imperative-statement

RELEASE record-name [FROM identifier]

RETURN file-name RECORD [INTO identifier] ; AT END imperative-statement
```

Format 1:

```
                           { index-name }
SEARCH table-name [VARYING { identifier } ][; AT END    imperative-statement-1]

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

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

Format 2:

```
SEARCH ALL  table-name [; AT END  imperative-statement-1]

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

SEEK file-name RECORD [WITH KEY CONVERSION]
```

Format 1:

```
     { index-name-1 } [  { index-name-2 } ]            { index-name-3 }
SET  { identifier-1 } [, { identifier-2 } ] . . .  TO  { identifier-3 }
                                                       { literal      }
```

Format 2:

```
                                            { UP BY   }  { index-name-3 }
SET index-name-1 [, index-name-2] . . .     { DOWN BY }  { identifier   }
                                                         { literal      }

                       { DESCENDING }
SORT file-name-1  ON   { ASCENDING  }  KEY data-name-1  [data-name2] . . .

   [      { DESCENDING }                                          ]
   [ ; ON { ASCENDING  }   KEY data-name-3 [data-name-4] . . .    ] . . .

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

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

     { literal }
STOP { RUN     }
```

Format 1:

```
          { literal-1    }  [   { literal-2    } ]
SUBTRACT  { identifier-1 }  [ , { identifier-2 } ] . . . FROM    identifier-m

     [ROUNDED] [, data-name-n [ROUNDED] ] . . .  [; ON SIZE ERROR  imperative-statement]
```

Format 2:

```
          { literal-1    }  [  { literal-2    } ]             { literal-m    }
SUBTRACT  { identifier-1 }  [ ,{ identifier-2 } ] . . .  FROM { identifier-m }

     GIVING  identifier-n  [ROUNDED] [, identifier-o   [ROUNDED] ] . . .

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

Format 3:

$\underline{\text{SUBTRACT}}\ \left\{\begin{array}{l}\underline{\text{CORR}}\\ \underline{\text{CORRESPONDING}}\end{array}\right\}\ \textit{data-name-1}\ \underline{\text{FROM}}\ \textit{data-name-2}\ [\underline{\text{ROUNDED}}]$

$[;\ \text{ON}\ \underline{\text{SIZE}}\ \underline{\text{ERROR}}\ \textit{imperative-statement}]$

Format 1:

$\underline{\text{WRITE}}\ \textit{record-name}\ [\underline{\text{FROM}}\ \textit{identifier-1}\,]$

$\left[\ \left\{\begin{array}{l}\underline{\text{AFTER}}\\ \underline{\text{BEFORE}}\end{array}\right\}\ \text{ADVANCING}\ \left\{\begin{array}{l}\textit{integer}\\ \textit{identifier-2}\end{array}\right\}\ \text{LINES}\right]$

Format 2:

$\underline{\text{WRITE}}\ \textit{record-name}\ [\underline{\text{FROM}}\ \textit{identifier}]\ [;\ \underline{\text{INVALID}}\ \text{KEY}\ \textit{imperative-statement}]$

COBOL CONTROL CARD OPTIONS

Blank – compiler assumes no options

The following options are under the control of the compiler:

A Accept results in spite of errors.

B Ignore check of sequence numbers (columns 1 through 6).

C List matched name of CORRESPONDING data-names.

D List data definitions (with qualifiers) .

E List detailed error diagnostics.

K List all parts incorporated by the COPY and INCLUDE verbs.

L List as if C, D, E, K, M, O, R, and S were present.

M List all procedure-names which are identical through the first five characters.

N Suppress listing.

O List octal output of final phase.

R List cross references (not sensitive to qualified names) .

S List source program.

T List on console printer (ACCEPT and DISPLAY verbs).

V Indicates subprogram rather than a main program. Prevents generation of starting address.

X Abort run if fatal error is detected.

The following options are applicable to a compilation:

I Insert; introduce source language into program file from control stream.

U Update; produce new cycle of source-language element.

W List correction deck prior to processor listing.

RESERVED WORDS

ABOUT
ACCEPT
ACCESS
ACTUAL
ADD
ADDRESS
ADVANCING
AFTER
ALL
ALPHABETIC
ALPHANUMERIC
ALTER
ALTERNATE
AN
AND
APPLY
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR
BEFORE
BEGINNING
BIT
BITS
BLANK
BLOCK
BLOCKS
BLOCK-COUNT
BY
CARD-PUNCH
CARD-PUNCH-EIGHTY
CARD-READER
CARD-READER-EIGHTY
CHARACTER
CHARACTERS
CHECK
CLASS
CLOCK-UNITS
CLOSE
CLUSTER-DUMPS
COBOL
COMMON-STORAGE
COMP
COMP-1
COMPUTATIONAL
COMPUTATIONAL-1
COMPUTATIONAL-2
COMPUTATIONAL-3
COMPUTE
CONFIGURATION
CONSOLE
CONSTANT
CONTAINS
CONVERSION
CONTROL
COPY
CORR
CORRESPONDING
DATA
DATE-COMPILED
DATE-WRITTEN
DECLARATIVES
DEFINE
DEMAND
DEPENDING
DESCENDING
DIGIT
DIGITS
DISPLAY
DISPLAY-1
DISPLAY-2
DISPLAY-3
DIVIDE
DIVIDED
DIVISION
DOLLAR
ELSE
END
ENDING
ENTER
ENTRY
ENVIRONMENT
EQUAL
EQUALS
ERROR
EVERY
EXAMINE
EXCEEDS
EXECUTE
EXIT
EXPONENTIATED
EXTENDED
FASTRAND*
FD
FILE
FILE-CONTROL
FILLER
FIRST
FLOAT
FOR
FORMAT
FORTRAN
FROM
GIVING
GO
GREATER
HASHED
HIGH-VALUE
HIGH-VALUES
ID
IDENTIFICATION
IF
IN
INCLUDE
INDEX
INDEXED
INPUT
INPUT-OUTPUT
INSTALLATION
INTERNAL
INTO
INVALID
IS
I-O
I-O-CONTROL
JUST
JUSTIFIED
KEY
LABEL
LEADING
LEAVING
LEFT
LESS

**Trademark of Sperry Rand Corporation*

LIBRARY
LINE
LINES
LINES-AT-BOTTOM
LINES-AT-TOP
LINES-PER-PAGE
LINE-SPACING
LOAD
LOCATION
LOCK
LOW-VALUE
LOW-VALUES
MASS-STORAGE
MEMORY
MINUS
MIXED
MODE
MODULES
MONITOR
MONITOR-DATE
MOVE
MULTIPLE
MULTIPLIED
MULTIPLY
NEGATIVE
NEXT
NO
NOT
NOTE
NUMERIC
OBJECT-COMPUTER
OBJECT-PROGRAM
OCCURS
OF
OFF
OMITTED
ON
OPEN
OPTIONAL
OR
OTHERWISE
OUTPUT
PERFORM
PIC
PICTURE
PLACE
PLACES
PLUS
POINT
POINTS
POSITION
POSITIVE
PREPARED
PRINTER
PRIORITY
PROCEDURE
PROCEED
PROGRAM-ID
QUOTE
QUOTES
RANDOM
RANGE
READ
RECORD
RECORD-COUNT
RECORDING
RECORDS
REDEFINES
REEL
REEL-NUMBER
REFERENCING
RELEASE
REMARKS
RENAMES
RENAMING
REPLACING
RERUN
RESERVE
RETURN
REVERSED
REWIND
RIGHT
ROUNDED
RUN
SAME
SD
SEARCH
SECTION
SECURITY
SEEK
SEGMENT-LIMIT
SELECT
SENTENCE
SENTINEL
SEQUENCED
SEQUENTIAL
SET
SIGN
SIGNED
SIZE
SORT
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STANDBY
STATUS
STOP
SUBROUTINE
SUBTRACT
SUPERVISOR
SUPPRESS
SYMBIONT
SYMBOLIC
SYNC
SYNCHRONIZED
TALLY
TALLYING
TAPE
THAN
THEN
THROUGH
THRU
TIME
TIMES
TO
TYPE
UNEQUAL
UNISERVO*
UNIVAC-1108
UNTIL
UPON
USAGE
USE
USING
VALUE
VALUES
VARYING
WHEN
WITH
WORD
WORDS
WORKING-STORAGE
WRITE
XS3
ZERO
ZEROES
ZEROS
/
**
*
>
=
<
+
−

*Trademark of Sperry Rand Corporation

PICTURE SYMBOLS

SYMBOL	FUNCTION	USED WITH	SPECIAL POSITION	NOTES
		DATA SYMBOLS		
A	An alphabetic character	X 9 B or 0	None	2, 3
X	An alphanumeric character	A 9 B or 0	None	3
9	A numeric character	Any other symbol	None	1, 2, 3, 4
		OPERATIONAL SYMBOLS		
S	Indicates signed data	P V 9 or H	Always leftmost except for H	1, 5
V	Indicates position of assumed decimal point within data item	Any symbol except A . or X	Must be within picture. Only one V is allowed.	1, 4, 5
P	Indicates position of an assumed decimal point. Is to the left or right of a data item. Each P represents one position	Any symbol except A . or X	Either first or last except for S CR DB + − or $	1, 5
		EDITING SYMBOLS		
H	COMPUTATIONAL-1 item	S P V or 9	Preceded only by S	1, 5
B	Insert space.	Any symbol except S H or more than one $ + or −	None	2, 4
0	Insert zero.	Any symbol except S or H	None	4
.	Insert point if following positions have not been blanked	Any symbol except A X P V S . or H	None	
Z	Zero suppression; replace leading zeros with blanks.	Any symbol except A S X H * or more than one $ or +	Preceded only by V . $, + − or P	4
*	Check protection, replace leading zeros with asterisks.	Any symbol except Z A X S H or more than one $ − or +	Preceded only by − + . V , $ or P	4
,	Insert comma unless preceding position has been blanked	Any except A X S or H	May not be adjacent to another comma	4
$	Insert dollar sign	Any except A X S + − or H	Leftmost	
or				4
$$$...$	Float dollar sign	Any except H A X Z * or more than one + − CR or DB	Leftmost	
+	Insert correct sign	Any except H, A X S − CR or DB	Rightmost or leftmost	4
or				
+++···+	Float correct sign	Any except A, X − S CR DB * Z or more than one $	Leading	4
−	Insert space if value is positive, minus sign if value is negative	Any except A X + S CR or DB	Rightmost or leftmost except for P	4
−−−...−	Float minus if value is negative	Any except A X + S CR DB * Z or more than one $	None	4
CR	Insert CR if value is negative; two spaces if positive	Any except A X + − S DB or H	Rightmost	4
DB	Insert DB if value is negative; two spaces if positive	Any except A X + − S CR or H	Rightmost	4

NOTES:

1. Pictures for numeric items may contain only S V P H and 9

2. Pictures for alphabetic items may contain only A and B

3. Pictures for nonedited alphanumeric items may contain only 9 A and X

4. Pictures for edited items may contain 9 V Z $ + − CR DB 0 B * and ,

5. S V P and H are not counted in the item size