Extended COBOL

OS/3

Summary

Environment: 90/25, 30, 30B, 40 Systems

SPERRY UNIVAC

UP-8055 Rev. 6

H

RELEASE
LEVEL: 7.0 Forward

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reprod or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of the Sperry Corporation.

FASTRAND, SPERRY UNIVAC, UNISCOPE, UNISERVO, and UNIVAC are registered trademarks of the Sperry Corporation. ESCORT, PAGEWRITER, PIXIE, and UNIS are additional trademarks of the Sperry Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

©1974, 1975, 1976, 1977, 1979 —
SPERRY CORPORATION

PRINTED IN U.S.A.

CONTENTS

The SPERRY UNIVAC Operating System/3 (OS/3) COBOL language is fully described in the OS/3 Extended COBOL supplementary reference, UP-8059 (current version).

SUMMARY NOTATION:

- Key words (that is, words that result in action by the compiler) are capitalized and underscored.
- Optional words (that is, words included for readability only) are capitalized, but not underscored.
- Brackets [ ] enclose words, phrases, or clauses that may be omitted if their functions are not required.
- Braces { } indicate a mandatory choice of various forms or functions.
- Ellipsis . . . indicates optional repetition of elements enclosed in the preceding pair of brackets or braces.
- Lowercase words represent generic terms that must be supplied by the user.
- Periods must be used where shown and must also appear at the end of each paragraph. Statements which do not contain periods on the reference card must be followed by a period when used at the end of a paragraph.

RULES AND SUGGESTIONS FOR EFFICIENCY:

1. Use legal abbreviations for reserved words to reduce compilation time, that is, PIC instead of PICTURE.
2. Use relational operators instead of relational clauses.
3. Avoid needless qualification and/or subscripting.
4. With ADD, SUBTRACT, IF, and MOVE:
   - use same size sending and receiving fields;
   - align decimal positions of sending and receiving fields.
5. Use indexing instead of subscripting whenever possible.

FIGURATIVE CONSTANTS:

ZERO$\begin{bmatrix} \text{S} \\ \text{ES} \end{bmatrix}$ = 0 or 0's	DISPLAY mode = code F0 (EBCDIC) or 30 (ASCII) COMPUTATIONAL mode = binary 0
QUOTE[S]	code 7D (EBCDIC) or 27 (ASCII); apostrophe is the generated character
HIGH-VALUE[S]	code FF (EBCDIC) or 7F (ASCII)
LOW-VALUE[S]	code 00 (lowest value in collating sequence)
ALL literal = a sequence of any nonnumeric literal or figurative constant	
SPACE[S] = blank character(s)	code 40 (EBCDIC) or 20 (ASCII)

IDENTIFICATION DIVISION

```
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

```
ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

                   { UNIVAC-9025. }
SOURCE-COMPUTER.   { UNIVAC-9030. }
                   { UNIVAC-9040. }
                   { UNIVAC-9025 } [
OBJECT-COMPUTER.   { UNIVAC-9030 } [ , MEMORY SIZE integer
                   { UNIVAC-9040 } [

        { CHARACTERS } ]
        { MODULES    } ] [, SEGMENT-LIMIT IS priority-number] .
        { WORDS      } ]

SPECIAL-NAMES.
     [CURRENCY SIGN IS literal]
     [; DECIMAL-POINT IS COMMA]
     [; SYSCOM IS mnemonic-name-1]
     [; SYSDATE IS mnemonic-name-2]
     [; SYSTIME IS mnemonic-name-3]
     [; SYSCONSOLE IS mnemonic-name-4]
     [; SYSCHAN-t IS mnemonic-name-5]
     [; SYSLST IS mnemonic-name-6]
     [; SYSERR[-m]                                                                    ]
     [     { ON STATUS IS condition-name-3 [, OFF STATUS IS condition-name-4] }       ] . . .
     [     { OFF STATUS IS condition-name-4 [, ON STATUS IS condition-name-3] }       ]
     [; SYSSWCH [-n]                                                     ]
     [                                                                   ]
     [    ( IS mnemonic-name-7 [, ON STATUS IS condition-name-5        ) ]
     [    (     [, OFF STATUS IS condition-name-6] ]                   ) ]
     [    (                                                            ) ]
     [    ( IS mnemonic-name-7 [, OFF STATUS IS condition-name-6       ) ]
     [    (     [, ON STATUS IS condition-name-5] ]                    ) ]
     [    (                                                            ) ] . . .
     [    ( ON STATUS IS condition-name-5                              ) ]
     [    (     [, OFF STATUS IS condition-name-6]                     ) ]
     [    ( OFF STATUS IS condition-name-6                             ) ]
     [    (     [, ON STATUS IS condition-name-5]                      ) ]
     [                                                                   ]
     [; SYSIN IS mnemonic-name-8]
     [; SYSIN-96 IS mnemonic-name-9]
     [; SYSIN-128 IS mnemonic-name-10]
     [; SYSLOG IS mnemonic-name-11] .

INPUT-OUTPUT SECTION
FILE-CONTROL  {SELECT [OPTIONAL] file-name
    ASSIGN TO [external-name] [integer-1]  implementor-name-1

         [OR implementor-name-2]  [ FOR MULTIPLE { REEL } ]
                                  [              { UNIT } ]

     [ ; RESERVE { integer-2 } ALTERNATE [ AREA  ] ]
     [           {    NO     }           [ AREAS ] ]

     [ ; { FILE-LIMIT IS   } { data-name-1 } THRU { data-name-2 } ]
     [   { FILE-LIMITS ARE } { literal-1   }      { literal-2   } ]
     [                                                            ]
     [    [ { data-name-3 } THRU { data-name-4 } ] . . .          ]
     [    [ { literal-3   }      { literal-4   } ]                ]
```

ENVIRONMENT DIVISION (CONT)

```
 [           {EXTENDED  }]
 [; ACCESS MODE IS {RANDOM    }][; PROCESSING MODE IS SEQUENTIAL]
 [           {SEQUENTIAL}]

 [                    {INDEXED   }]
 [; ORGANIZATION IS   {RELATIVE  }]
 [                    {SEQUENTIAL}]

 [; {ACTUAL KEY IS data-name-5  }]
 [  {RELATIVE KEY IS data-name-6}]
 [; SYMBOLIC KEY IS data-name-7]
 [; RECORD KEY IS data-name-8] . } . . .

I-O-CONTROL.

   [ RERUN ON external-name EVERY integer-1 RECORDS OF file-name-1
        [, file-name-2] . . . ] . . .

   [; SAME [RECORD] AREA FOR file-name-3 { , file-name-4 } . . . ] . . .
           [SORT  ]

   [[; MULTIPLE FILE TAPE CONTAINS file-name-5]
        [POSITION integer-2] [file-name-6[POSITION integer-3]] . . . ] . . .

   [; APPLY VERIFY ON file-name-8 [, file-name-n] . . . ] . . .

   [; APPLY BLOCK-COUNT ON {file-name-9 [file-name-10] . . .}] . . .
                           {          TAPES                  }

  †[; APPLY MASTER-INDEX ON file-name-11 [, file-name-12] . . .] . . .

   [; APPLY CYLINDER-INDEX AREA OF integer-5 INDICES ON file-name-13
        [, file-name-14] . . .] . . .

   [; APPLY CYLINDER-OVERFLOW AREA OF integer-6
        PERCENT ON file-name-15 [, file-name-16] . . .] . . .

  †[; APPLY EXTENDED-INSERTION AREA ON file-name-17
        [, file-name-18] . . . ] . . .

   [; APPLY FILE-PREPARATION ON file-name-19 [, file-name-20] . . .] . . .

   [ ; APPLY ASCII* [WITH BUFFER-OFFSET

        {FOR BLOCK-LENGTH-CHECK}] ON file-name-21 [, file-name-22] . . .].
        {OF integer CHARACTERS }
```

DATA DIVISION

```
DATA DIVISION.

FILE SECTION.

FD file-name

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

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

                                    {OMITTED                               }
   ; LABEL {RECORD IS  } {STANDARD                              }
           {RECORDS ARE} {data-name-1 [, data-name-2] . . .     }

   [                        {D}]
   [ ; RECORDING MODE* IS   {F}]
   [                        {U}]
   [                        {V}]
```

†*Accepted for OS/4 and OS/7 compatiblity only.*
**Extension to American National Standard COBOL (1968).*

DATA DIVISION (CONT)

```
[ ; VALUE OF { unqualified-data-name IS { data-name-3 } } ... ]
                                         { literal-1   }

[ ; DATA { RECORD IS    } data-name-4 [, data-name-5] ... ] .
         { RECORDS ARE  }

SD file-name

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

    [ ; RECORDING MODE* IS { D } ]
                           { F }
                           { V }

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

DATA DESCRIPTION

Format 1:

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

[ { OCCURS integer-2 TIMES [ { ASCENDING  } KEY IS data-name-2        } ]
  {                          { DESCENDING }                           }
  {                                                                   }
  {     [ data-name-3] ... ] ...                                      }
  {                                                                   }
  {   [INDEXED BY index-name-1 [, index-name-2] ...]                  }
  {                                                                   }
  { ; OCCURS [integer-1 TO] integer-2 TIMES DEPENDING ON data-name-1  }
  {                                                                   }
  {   [ { ASCENDING  } KEY IS data-name-2 [, data-name-3] ... ] ...   }
  {     { DESCENDING }                                                }
  {   [ INDEXED BY index-name-1 [, index-name-2] ... ]                }

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

[               { COMP                 } ]
[               { COMPUTATIONAL        } ]
[               { COMP-1*              } ]
[               { COMPUTATIONAL-1*     } ]
[               { COMP-2*              } ]
[               { COMPUTATIONAL-2*     } ]
[ ; [USAGE IS]  { COMP-3*              } ]
[               { COMPUTATIONAL-3*     } ]
[               { COMP-4*              } ]
[               { COMPUTATIONAL-4*     } ]
[               { DISPLAY              } ]
[               { INDEX                } ]

[; MAP* IS integer-3, CHARACTERS]

[ ; { SYNC         } [ LEFT  ] ] [ ; { JUST      } RIGHT ]
    { SYNCHRONIZED } [ RIGHT ]       { JUSTIFIED }

[; VALUE IS literal] [; BLANK WHEN ZERO]

[   { [SIGN IS*] { LEADING  } SEPARATE CHARACTER } ] .
[ ; {            { TRAILING }                    } ]
[   { [SIGN IS*] TRAILING                        } ]
```

Format 2:

```
66 unqualified-data-name-1; RENAMES data-name-2 [THRU data-name-3].
```

Format 3:

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

    [literal-3 [THRU literal-4]] ...
```

*Extension to American National Standard COBOL (1968).

DATA DIVISION (CONT)

```
⎡ WORKING-STORAGE SECTION.                  ⎤
⎢                                           ⎥
⎣ ⎡ 77-level-description-entry ⎤            ⎦
  ⎣ record-description-entry   ⎦

⎡ LINKAGE SECTION*                                              ⎤
⎣ [level-number data-name [descriptive clauses] ] . . .         ⎦
```

PROCEDURE DIVISION

```
PROCEDURE DIVISION. [USING* unqualified-data-name-1

          [unqualified-data-name-2] . . .].

[DECLARATIVES.

{section-name SECTION. declarative-sentence.

{paragraph-name. {sentence} . . .} . . .} . . .

END DECLARATIVES.]

{ [section-name SECTION. [priority-number] .]

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

VERBS AND STATEMENTS (listed alphabetically)

```
                      ⎡      ⎧ mnemonic-name ⎫ ⎤
ACCEPT identifier     ⎢ FROM ⎨ DATE*         ⎬ ⎥
                      ⎢      ⎪ DAY*          ⎪ ⎥
                      ⎣      ⎩ TIME*         ⎭ ⎦
```

Format 1:

```
ADD ⎧ identifier-1 ⎫ ⎡ , identifier-2 ⎤ . . . TO identifier-m [ROUNDED]
    ⎩ literal-1    ⎭ ⎣ , literal-2    ⎦

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

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

Format 2:

```
ADD ⎧ identifier-1 ⎫ , ⎧ identifier-2 ⎫ ⎡ , identifier-3 ⎤ . . .
    ⎩ literal-1    ⎭   ⎩ literal-2    ⎭ ⎣ , literal-3    ⎦

   GIVING  identifier-n [ROUNDED] [; ON SIZE ERROR imperative-statement]
```

Format 3:

```
ADD ⎧ CORR          ⎫  identifier-1 TO identifier-2
    ⎩ CORRESPONDING ⎭

   [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]

                  ⎡       ⎧ file-name      ⎫       ⎤
CALL*entry-name   ⎢ USING ⎨ identifier     ⎬ . . . ⎥
                  ⎢       ⎪ procedure-name ⎪       ⎥
                  ⎣       ⎩ sort-name      ⎭       ⎦

CLOSE file-name-1 ⎡ REEL ⎤ ⎡ WITH ⎧ LOCK      ⎫ ⎤
                  ⎣ UNIT ⎦ ⎣      ⎩ NO REWIND ⎭ ⎦

   ⎡ , file-name-2 ⎡ REEL ⎤ ⎡ WITH ⎧ LOCK      ⎫ ⎤ ⎤ . . .
   ⎣               ⎣ UNIT ⎦ ⎣      ⎩ NO REWIND ⎭ ⎦ ⎦

                                        ⎧ arithmetic-expression ⎫
COMPUTE identifier-1 [ROUNDED] =        ⎨ identifier-2          ⎬
                                        ⎩ literal               ⎭

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

**Extension to American National Standard COBOL (1968).*

PROCEDURE DIVISION (CONT)

Format 1:

$\underline{\text{COPY}}$ library-name

Format 2:

$\underline{\text{COPY}}$ library-name

$$\left[\underline{\text{REPLACING}}\ \text{word-1}\ \underline{\text{BY}}\left\{\begin{matrix}\text{identifier-1}\\ \text{literal-1}\\ \text{word-2}\end{matrix}\right\}\left[,\ \text{word-3}\ \underline{\text{BY}}\left\{\begin{matrix}\text{identifier-2}\\ \text{literal-2}\\ \text{word-4}\end{matrix}\right\}\right]\ldots\right]$$

$$\underline{\text{DISPLAY}}\left\{\begin{matrix}\text{identifier-1}\\ \text{literal-1}\end{matrix}\right\}\left[\begin{matrix},\ \text{identifier-2}\\ ,\ \text{literal-2}\end{matrix}\right]\ldots\ [\underline{\text{UPON}}\ \text{mnemonic-name}]$$

Format 1:

$$\underline{\text{DIVIDE}}\left\{\begin{matrix}\text{identifier-1}\\ \text{literal}\end{matrix}\right\}\ \underline{\text{INTO}}\ \text{identifier-2}\ [\underline{\text{ROUNDED}}]$$

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

Format 2:

$$\underline{\text{DIVIDE}}\left\{\begin{matrix}\text{identifier-1}\\ \text{literal-1}\end{matrix}\right\}\ \underline{\text{INTO}}\left\{\begin{matrix}\text{identifier-2}\\ \text{literal-2}\end{matrix}\right\}\ \underline{\text{GIVING}}\ \text{identifier-3}\ [\underline{\text{ROUNDED}}]$$

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

Format 3:

$$\underline{\text{DIVIDE}}\left\{\begin{matrix}\text{identifier-1}\\ \text{literal-1}\end{matrix}\right\}\ \underline{\text{BY}}\left\{\begin{matrix}\text{identifier-2}\\ \text{literal-2}\end{matrix}\right\}\ \underline{\text{GIVING}}\ \text{identifier-3}\ [\underline{\text{ROUNDED}}]$$

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

Format 4:

$$\underline{\text{DIVIDE}}\left\{\begin{matrix}\text{identifier-1}\\ \text{literal-1}\end{matrix}\right\}\ \underline{\text{INTO}}\left\{\begin{matrix}\text{identifier-2}\\ \text{literal-2}\end{matrix}\right\}\ \underline{\text{GIVING}}\ \text{identifier-3}\ [\underline{\text{ROUNDED}}]$$

$$\underline{\text{REMAINDER}}\ \text{identifier-4}\ [;\ \text{ON}\ \underline{\text{SIZE}}\ \underline{\text{ERROR}}\ \text{imperative-statement}]$$

Format 5:

$$\underline{\text{DIVIDE}}\left\{\begin{matrix}\text{identifier-1}\\ \text{literal-1}\end{matrix}\right\}\ \underline{\text{BY}}\left\{\begin{matrix}\text{identifier-2}\\ \text{literal-2}\end{matrix}\right\}\ \underline{\text{GIVING}}\ \text{identifier-3}\ [\underline{\text{ROUNDED}}]$$

$$\underline{\text{REMAINDER}}\ \text{identifier-4}\ [;\ \text{ON}\ \underline{\text{SIZE}}\ \underline{\text{ERROR}}\ \text{imperative-statement}]$$

Format 1:

$\underline{\text{ENTER}}\ \underline{\text{LINKAGE}}$.

$$\underline{\text{CALL}}^{*}\text{entry-name}\left[\underline{\text{USING}}\left\{\begin{matrix}\text{file-name}\\ \text{identifier}\\ \text{procedure-name}\\ \text{sort-name}\end{matrix}\right\}\ldots\right].$$

$\underline{\text{ENTER}}\ \underline{\text{COBOL}}$.

Format 2:

$\underline{\text{ENTER}}\ \underline{\text{LINKAGE}}$.

$$\underline{\text{ENTRY}}^{*}\ \text{entry-name}\ [\underline{\text{USING}}\ \{\text{unqualified-data-name}\}\ \ldots]\ .$$

$\underline{\text{ENTER}}\ \underline{\text{COBOL}}$.

**Extension to American National Standard COBOL (1968).*

PROCEDURE DIVISION (CONT)

Format 3:

```
ENTER LINKAGE.
{EXIT PROGRAM.}
{RETURN.      }
ENTER COBOL.
```

```
EXAMINE identifier

  {              {ALL        }                                        }
  { TALLYING     {LEADING    }  literal-1 [REPLACING BY literal-2]    }
  {              {UNTIL FIRST}                                        }
  {                                                                   }
  {              {ALL          }                                      }
  { REPLACING    {LEADING      }  literal-3 BY literal-4              }
  {              {[UNTIL] FIRST}                                      }
```

```
EXIT [PROGRAM] *
```

Format 1:

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

Format 2:

```
GO TO procedure-name-1 [, procedure-name-2] . . . , procedure-name-n
   DEPENDING ON identifier
```

Format 3:

```
GO TO MORE-LABELS*
```

```
IF condition; [THEN] * {NEXT SENTENCE}
                       {statement-1  }

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

condition may be any of the following:

- Relation condition

```
                          (  IS {[NOT] GREATER THAN} )
                          (     {[NOT] >           } )
     {arithmetic-    }    (  IS {[NOT] LESS THAN   } )  {arithmetic-    }
     {  expression-1 }    (     {[NOT] <           } )  {  expression-2 }
IF   {identifier-1   }    (  IS {[NOT] EQUAL TO    } )  {identifier-2   }
     {literal-1      }    (     {[NOT] =           } )  {literal-2      }
                          (      EQUALS*             )
                          (      UNEQUAL*            )
                          (      EXCEEDS*            )
```

- Class condition

```
IF identifier IS [NOT] {ALPHABETIC}
                       {NUMERIC   }
```

- Condition-name condition as defined by an 88-level entry in the Data Division

```
IF [NOT] condition-name
```

Switch-status condition

```
IF [NOT] condition-name
```

- Sign condition

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

Extension to American National Standard COBOL (1968).

PROCEDURE DIVISION (CONT)

```
INSERT* record-name [FROM identifier-1] [; INVALID KEY imperative-statement]
```

Format 1:

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

Format 2:

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

Format 1:

```
MULTIPLY  { identifier-1 }  BY identifier-2 [ROUNDED]
          { literal-1    }
   [; ON SIZE ERROR imperative-statement]
```

Format 2:

```
MULTIPLY  { identifier-1 }  BY { identifier-2 }  GIVING identifier-3 [ROUNDED]
          { literal-1    }     { literal-2    }
   [; ON SIZE ERROR imperative-statement]
```

```
NOTE character-string.
```

```
      { I-O { file-name } . . .                                          }
OPEN  { INPUT  { file-name [ REVERSED          ] } . . .                 } . . .
      {                    [ WITH NO REWIND    ]                         }
      { OUTPUT { file-name [WITH NO REWIND] } . . .                      }
```

Format 1:

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

Format 2:

```
PERFORM procedure-name-1 [THRU procedure-name-2] { identifier-1 }  TIMES
                                                 { integer-1    }
```

Format 3:

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

Format 4:

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

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

          BY { identifier-3 }  UNTIL condition-1
             { literal-3    }

   [ AFTER { identifier-4 }  FROM { identifier-5 }
   [       { index-name-4 }       { index-name-5 }
   [                              { literal-5    }

          BY { identifier-6 }  UNTIL condition-2
             { literal-6    }

   [ AFTER { identifier-7 }  FROM { identifier-8 }
   [       { index-name-7 }       { index-name-8 }
                                  { literal-8    }

          BY { identifier-9 }  UNTIL condition-3 ]]
             { literal-9    }
```

*Extension to American National Standard COBOL (1968).

PROCEDURE DIVISION (CONT)

```
                                                { AT END      }
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:

REWRITE* record-name [FROM identifier]

Format 2:

REWRITE record-name [FROM identifier] [; INVALID KEY imperative-statement ]

Format 1:

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

    [; AT END imperative-statement-1]

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

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

Format 2:

SEARCH ALL identifier-1 [; AT END imperative-statement-1]

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

SEEK file-name RECORD

Format 1:

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

Format 2:

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

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

    [       { ASCENDING  }                        ]
    [ ; ON  { DESCENDING }  KEY  {data-name-2} . . . ] . . .

    { 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:

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

    FROM identifier-m [ROUNDED]  [ , identifier-n [ROUNDED] ] . . .
    [; ON SIZE ERROR imperative-statement]
```

**Extension to American National Standard COBOL (1968).*

Format 2:

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

    FROM {identifier-m} GIVING identifier-n [ROUNDED]
         {literal-m   }

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

Format 3:

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

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

Format 1:

```
TRANSFORM* identifier-3 [, identifier-4] . . . CHARACTERS

    FROM {figurative-constant-1 } TO {figurative-constant-2 }
         {identifier-1          }    {identifier-2          }
         {nonnumeric-literal-1  }    {nonnumeric-literal-2  }
```

Format 2:

```
TRANSFORM identifier-3 [, identifier-4] . . . CHARACTERS

    FROM {ASCII TO EBCDIC}
         {EBCDIC TO ASCII}
```

Format 3:

```
TRANSFORM identifier-3 [, identifier-4] . . . CHARACTERS

    {BY} identifier-5
    {ON}
```

Format 1:

```
USE {AFTER } STANDARD [ BEGINNING ] [ FILE ]
    {BEFORE}          [ ENDING    ] [ REEL ]
                                    [ UNIT ]

    LABEL PROCEDURE ON {file-name-1 [, file-name-2] . . . }
                       {I-O                               }
                       {INPUT                             }
                       {OUTPUT                            }
```

Format 2:

```
USE AFTER STANDARD ERROR PROCEDURE ON

    {file-name-1 [, file-name-2] . . . }
    {I-O                               }
    {INPUT                             }
    {OUTPUT                            }
```

Format 3:*

```
USE FOR FORM-OVERFLOW ON file-name-1
```

Format 1:

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

    [ {AFTER } ADVANCING {identifier-2 LINES} ]
    [ {BEFORE}           {integer LINES     } ]
    [                    {mnemonic-name     } ]
```

**Extension to American National Standard COBOL (1968).*

Format 2:

```
WRITE record-name [FROM identifier-1] [;. INVALID KEY imperative-statement]
```

DEBUGGING AIDS

DEBUGGING AIDS

(An extension to 1968 American National Standard COBOL):

SYSLST must be specified on an LFD control card.

```
READY TRACE.*
RESET TRACE.*
```

```
         ( CHANGED       )   ( identifier-1          )
EXHIBIT* { CHANGED NAMED }   {                       }
         ( NAMED         )   ( nonnumeric-literal-1  )

    [ , { identifier-2         } ]
    [   { nonnumeric-literal-2 } ] . . .
```

where:

CHANGED
: Provides a columnar display of nonnumeric literals and identifier values that have changed.

CHANGED NAMED
: Provides a noncolumnar display of nonnumeric literals and identifier values that have changed.

NAMED
: Provides a noncolumnar display of specified identifier values and nonnumeric literals.

Debug* Packet Control Card

```
|1      |8
|*DEBUG |location
```

where:

location
: Is a section name or a paragraph name.

RESERVED WORDS

ACCEPT
ACCESS
ACTUAL
ADD
ADVANCING
AFTER
ALL
ALPHABETIC
ALTER
ALTERNATE
AND
APPLY*
ARE
AREA
AREAS
ASCENDING
ASCII*
ASSIGN
AT
AUTHOR
BEFORE
BEGINNING
BLANK
BLOCK
BLOCK-COUNT*
BLOCK-LENGTH-CHECK*
BUFFER-OFFSET*
BY
CALL*
CARD-PUNCH*
CARD-READER*
CARD-READER-51*
CARD-READER-66*
CHARACTER*
CHARACTERS
CHANGED*
CLOSE
COBOL
COMMA
COMP
COMP-1*
COMP-2*
COMP-3*
COMP-4*
COMPUTATIONAL
COMPUTATIONAL-1*
COMPUTATIONAL-2*
COMPUTATIONAL-3*
COMPUTATIONAL-4*
COMPUTE
CONFIGURATION
CONTAINS
COPY
CORR

Extension to American National Standard COBOL (1968).

RESERVED WORDS (CONT)

CORRESPONDING
CURRENCY
CYLINDER-INDEX*
CYLINDER-OVERFLOW*
DATA
DATE-COMPILED
DATE-WRITTEN
DECIMAL-POINT
DECLARATIVES
DEPENDING
DESCENDING
DIRECT*
DISC*
DISC-8411*
DISC-8414*
DISC-8415*
DISC-8416*
DISC-8418*
DISC-8430*
DISC-8433*
DISPLAY
DIVIDE
DIVISION
DOWN
EBCDIC*
ELSE
END
ENDING
ENTER
ENTRY*
ENVIRONMENT
EQUAL
EQUALS*
ERROR
EVERY
EXAMINE
EXCEEDS*
EXHIBIT*
EXIT
EXTENDED
EXTENDED-INSERTION*
FD
FILE
FILE-CONTROL
FILE-LIMIT
FILE-LIMITS
FILE-PREPARATION*
FILLER
FIRST
FOR
FORM-OVERFLOW*
FROM
GENERATE
GIVING
GO
GREATER
HIGH-VALUE
HIGH-VALUES
I-O
I-O-CONTROL
ID
IDENTIFICATION
IF
IN
INDEX
INDEXED
INDICES*
INITIATE
INPUT
INPUT-OUTPUT
INSERT*
INSTALLATION
INTO
INVALID
IS
JUST

JUSTIFIED
KEY
LABEL
LEADING
LEFT
LESS
LINE
LINES
LINKAGE*
LOCK
LOW-VALUE
LOW-VALUES
MAP*
MASTER-INDEX*
MEMORY
MODE
MODULES
MORE-LABELS*
MOVE
MULTIPLE
MULTIPLY
NAMED*
NEGATIVE
NEXT
NO
NOT
NOTE
NUMERIC
OBJECT-COMPUTER
OCCURS
OF
OFF
OMITTED
ON
OPEN
OPTIONAL
OR
ORGANIZATION*
OTHERWISE*
OUK-90-250*
OUK-90-300*
OUK-90-400*
OUK-90-600*
OUK-90-700*
OUTPUT
PERCENT *
PERFORM
PIC
PICTURE
POSITION
POSITIVE
PRINTER*
PROCEDURE
PROCEED
PROCESSING
PROGRAM*
PROGRAM-ID
QUOTE
QUOTES
RANDOM
READ
READY *
RECORD
RECORDING*
RECORDS
REDEFINES
REEL
RELATIVE*
RELEASE
REMAINDER
REMARKS
RENAMES
REPLACING
RERUN
RESERVE
RESET*

**Extension to American National Standard COBOL (1968).*

RESERVED WORDS (CONT)

RESTRICTED*
RETURN
REVERSED
REWIND
REWRITE*
RIGHT
ROUNDED
RUN
SAME
SD
SEARCH
SECTION
SECURITY
SEEK
SEGMENT-LIMIT
SELECT
SENTENCE
SEPARATE
SEQUENTIAL*
SET
SIGN
SIZE
SORT
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STATUS
STOP
SUBTRACT
SYMBOLIC*
SYNC
SYNCHRONIZED
SYSCHAN-1*
SYSCHAN-2*
SYSCHAN-3*
SYSCHAN-4*
SYSCHAN-5*
SYSCHAN-6*
SYSCHAN-7*
SYSCHAN-8*
SYSCHAN-9*
SYSCHAN-10*
SYSCHAN-11*
SYSCHAN-12*
SYSCHAN-13*
SYSCHAN-14*
SYSCHAN-15*
SYSCOM*
SYSCONSOLE*
SYSDATE*
SYSERR*
SYSERR-0*
SYSERR-1*
SYSERR-2*
SYSERR-3*
SYSERR-4*
SYSERR-5*
SYSERR-6*
SYSERR-7*
SYSERR-8*
SYSERR-9*
SYSERR-10*
SYSERR-11*
SYSERR-12*
SYSERR-13*
SYSERR-14*
SYSERR-15*
SYSERR-16*
SYSERR-17*
SYSERR-18*
SYSERR-19*
SYSERR-20*
SYSERR-21*
SYSERR-22*
SYSERR-23*
SYSERR-24*
SYSERR-25*
SYSERR-26*
SYSERR-27*
SYSERR-28*
SYSERR-29*
SYSERR-30*
SYSERR-31*
SYSIN*
SYSIN-96*
SYSIN-128*
SYSLOG
SYSLST*
SYSSWCH*
SYSSWCH-0*
SYSSWCH-1*
SYSSWCH-2*
SYSSWCH-3*
SYSSWCH-4*
SYSSWCH-5*
SYSSWCH-6*
SYSSWCH-7*
SYSTIME*
TALLY
TALLYING
TAPE
TAPE-6*
TAPES*
THAN
THEN*
THROUGH
THRU
TIME*
TIMES
TO
TRACE*
TRACKS*
TRAILING*
TRANSFORM*
UNEQUAL*
UNIT
UNIVAC-9000*
UNIVAC-9025*
UNIVAC-9030*
UNIVAC-9040*
UNIVAC-9060*
UNIVAC-9070*
UNIVAC-9200II*
UNIVAC-9300*
UNIVAC-9300II*
UNIVAC-9400*
UNIVAC-9480*
UNIVAC-9700*
UNTIL
UP
UPON
USAGE
USE
USING
VALUE
VALUES
VARYING
VERIFY*
WHEN
WITH
WORDS
WORKING-STORAGE
WRITE
ZERO
ZEROES
ZEROS

**Extension to American National Standard COBOL (1968).*

PARAM CARD OPTIONS

PARAM CARD	RESULT
// PARAM LST=A	Activates ambiguity mode of reference resolution. The definition search process is not terminated when the reference has been resolved, but is continued in an attempt to find and report duplicate definitions.
// PARAM LST=C	Produces cross-reference information for the Data Division and/or Procedure Division maps as specified. If the C option is used without the M and P options, both a Data Division and Procedure Division map listing will be produced with cross-reference information.
// PARAM LST=D	Produces Data Division alphabetized cross-reference listing.
// PARAM LST=E	Printer mismatch errors during compilation are ignored.
// PARAM LST=I	Suppress listing of lines from COPY library.
// PARAM LST=K	Suppresses source sequence number diagnostics.
// PARAM LST=L	Single-spaces all requested listings. If no listings were requested, a single-spaced diagnostic listing is produced.
// PARAM LST=M	Produces Data Division storage map listing.
// PARAM LST=N	Suppresses all output listings except the PARAM card listing.
// PARAM LST=O	Produces object code listing.
// PARAM LST=P	Produces Procedure Division storage map listing.
// PARAM LST=R	Allows quotation mark symbol in nonumeric literal bounded by apostrophes.
// PARAM LST=S	Produces source program listing.
// PARAM LST=T	Allows apostrophe symbol in nonnumeric literal bounded by quotation marks.
// PARAM LST=W	Suppresses precautionary diagnostic listing.
// PARAM LST=X	Produces Procedure Division alphabetized cross-reference listing.
// PARAM OUT=A	Produces ASCII sensitive object program.
// PARAM OUT=C	Conversion mode.
// PARAM OUT=E	Inhibits display of ISAM file status on console.
// PARAM OUT=K	All data items described as USAGE IS COMP or COMPUTATIONAL are treated as packed decimal (COMP-3 or COMPUTATIONAL-3).
// PARAM OUT=L	Suppresses generation of linker control information in the object module.
// PARAM OUT=M	Produces shared-code COBOL action program to be executed under the control of information management system (IMS/90).
// PARAM OUT=N	Suppresses object program module generation
// PARAM OUT=P	Disregards mismatched errors for all object program print files.
// PARAM OUT=R	Quote as figurative constant is generated as quotation marks; by default, quote is apostrophe.
// PARAM OUT=S	Disable object program SORT PARAM card processing.
// PARAM OUT=T	Suppresses compiler generation of a transfer address for the object program. The program cannot be executed unless it is called.
// PARAM OUT=V	Suppresses automatic page overflow in the object program.
// PARAM IN= program-name/ filename	Identifies the file containing source program input.
// PARAM LIN= filename	Identifies the file containing the COPY library.
// PARAM VER=vv/rr	Applies version and revision number to compiler output module.
// PARAM OBJ= filename	Identifies the file where the generated object mode is to be placed.

NOTE:

In the absence of PARAM cards, the compiler will produce a source program listing, a diagnostic report, an object program, assume jobstream input, and produce a version number of 00/00 for the object program.