WANG

VS Cobol

Quick Reference Guide

VS Cobol Quick Reference Guide

5th Edition — May 1984

800-6200-05

WANG LABORATORIES, INC., ONE INDUSTRIAL AVENUE, LOWELL, MA 01851 • TEL. (617) 459-5000, TWX 710-343-6769, Telex 94-7421

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

The staff of Wang Laboratories, Inc., has taken due care in preparing this manual. However, nothing contained herein modifies or alters in any way the standard terms and conditions of the Wang purchase, lease, or license agreement by which the product was acquired, nor increases in any way Wang's liability to the customer. In no event shall Wang or its subsidiaries be liable for incidental or consequential damages in connection with or arising from the use of the product, the accompanying manual, or any related materials.

SOFTWARE NOTICE

All Wang Program Products (software) are licensed to customers in accordance with the terms and conditions of the Wang Standard Software License. No title or ownership of Wang software is transferred, and any use of the software beyond the terms of the aforesaid license, without the written authorization of Wang, is prohibited.

WARNING

This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device, pursuant to Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

INTRODUCTION

The VS COBOL Quick Reference is a guide intended for experienced COBOL programmers. It provides the syntax and program information you need to use the Wang VS COBOL compiler 3.8.4 or greater.

VS COBOL Language Syntax contains the general formats for the four program divisions. It also gives the formats for specific paragraphs and entries.

VS COBOL Statement Formats give the statement syntax used by the VS COBOL compiler. General formats for identifier, qualification, and conditions are also given. A glossary of terms is appended.

The guide also includes a list of VS COBOL Reserved Words, Hexadecimal To Decimal Conversion, Powers of 2 and 16, Field Attribute Characters (FACs), and Translation Table.

You need DMS/TX software and VS operating system 6.10 or greater to use DMS/TX statements. You also need VS operating system 6.20 or greater and any VS system, except VS 50 or VS 80, to use Relative Files.

For more information regarding syntax and general rules, consult the *VS COBOL Reference Manual* (800-1201).

TABLE OF CONTENTS

VS COBOL Language Syntax

VS COBOL Statement Formats

VS COBOL LANGUAGE SYNTAX

Language Conventions

The following conventions are used in this section:

Capitalized or uppercase words are reserved words and have preassigned meanings in COBOL. This does not apply to words in quotation marks.

Underlined reserved words are key words and must be used when that portion of the format is used. Optional reserved words are not underlined.

The characters "+", "−", "<", ">", "=", or equivalent reserved words, although not underlined, are required within the chosen formats.

Words printed in lowercase letters represent information to be supplied by the programmer.

Brackets "[ ]" indicate an optional portion of the format.

Braces " { } " indicate that one of the options within the braces must be selected.

When choice indicators, [| |], enclose a portion of a general format, one or more of the unique options contained within the choice indicators must be specified, but a single option may be specified only once.

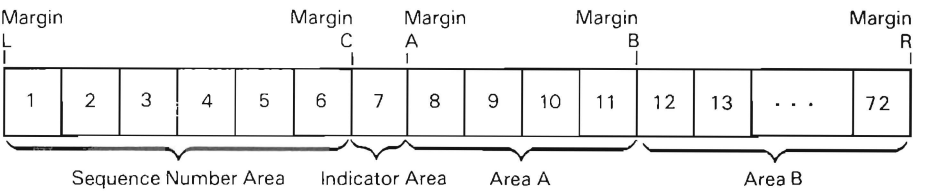
Options are stacked vertically within the brackets or braces. If one option within the brackets or braces contains only reserved words that are not underlined, that option is the default option.

An ellipsis ". . ." indicates a unit may be repeated. A unit is either a single lowercase word or a group of lowercase and reserved words enclosed in brackets or braces.

Entries shaded in **light gray** are treated as comments by the VS COBOL compiler and serve documentation purposes only.

Entries shaded in **dark gray** are VS extensions to ANSI COBOL.

The COBOL source-program reference format, which defines the permissible locations of COBOL code on a line of text, is:



General Format for the 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] . . . ]
```

General Format for the Environment Division

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. WANG-VS [ WITH DEBUGGING MODE ].

OBJECT-COMPUTER.  WANG-VS [ MEMORY SIZE integer { WORDS      } ]
                          { CHARACTERS }
                          { MODULES    }
                  [ PROGRAM COLLATING SEQUENCE IS alphabet-name ].

SPECIAL-NAMES.
    [ SWITCH-n [ IS mnemonic-name ]
        { ON STATUS IS condition-name-1 [ OFF STATUS IS condition-name-2 ] }
        { OFF STATUS IS condition-name-3 [ ON STATUS IS condition-name-4 ] } ] . . .
    [ alphabet-name IS { STANDARD-1 } ] . . .
                       { NATIVE     }
    [ CURRENCY SIGN IS literal-1 ]
    [ DECIMAL-POINT IS COMMA ].

FIGURATIVE-CONSTANTS.
    [ { data-name-1 IS "hexadecimal-value" } . . . ]
INPUT-OUTPUT SECTION.
FILE-CONTROL.  { file-control-entry } . . .
I-O CONTROL.
    [ RERUN [ ON file-name-1 ] EVERY integer-1 RECORDS OF file-name-2 ] . . .
```

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

File Control Entry:

Consecutive File Organization

```
SELECT file-name
    ASSIGN TO "parameter-reference-name" [ "DISK"      ]  [ NODISPLAY ]
                                         [ "DISPLAY"   ]
                                         [ "PRINTER"   ]
                                         [ "TAPE"      ]
    [ ORGANIZATION IS SEQUENTIAL ]
    [ ACCESS MODE IS { SEQUENTIAL } [ RELATIVE KEY IS data-name-1 ] ]
                     { RANDOM     }
                     { DYNAMIC    }
    [ FILE STATUS IS data-name-2 ]
    [ CURSOR POSITION IS data-name-3 ]
    [ BUFFER SIZE IS integer-1 BLOCKS ]
    [ PFKEY IS data-name-4 ].
```

Indexed File Organization

```
SELECT file-name
    ASSIGN TO "parameter-reference-name" [ "DISK" ] [ NODISPLAY ]
        ORGANIZATION IS INDEXED
        [ ACCESS MODE IS { SEQUENTIAL } ] RECORD KEY IS data-name-1
                         { RANDOM     }
                         { DYNAMIC    }
        [ ALTERNATE RECORD KEY  integer-1  IS data-name-2
                                           [ WITH DUPLICATES ]
                                [ [ integer-2  IS ] data-name-3
                                    [ WITH DUPLICATES ] ] . . . ]
        [ FILE STATUS IS data-name-4 ]
        [ RESERVE integer-3 [ AREA  ] ]
                            [ AREAS ]   .
```

Relative File Organization

```
SELECT file-name

    ASSIGN TO "parameter-reference-name" [ "DISK" ] [ NODISPLAY ]
[
    RESERVE integer-1   [ AREA  ]
                        [ AREAS ]
]
    ORGANIZATION IS RELATIVE

[
    ACCESS MODE IS { SEQUENTIAL  [ RELATIVE KEY IS data-name-1 ] }
                   { RANDOM      RELATIVE KEY IS data-name-1     }
                   { DYNAMIC                                     }
]
    [ FILE STATUS IS data-name-2 ]
    [ BUFFER SIZE IS integer-2 BLOCKS ]
```

Sort–Merge File Organization

```
SELECT file-name

    ASSIGN TO "parameter-reference-name"   [ "DISK" ]   [ NODISPLAY ]
                                           [ "TAPE" ]

    [ BUFFER SIZE IS integer BLOCKS ]
```

General Format for Data Division

```
DATA DIVISION.
[ FILE SECTION.
[ 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 Description Entry:

Format 1

```
FD file-name
[
    BLOCK CONTAINS [ integer-1 TO ] integer-2   { RECORDS    }
                                                { CHARACTERS }
]
```

```
[ RECORD CONTAINS [ integer-3 TO ] integer-4 [ COMPRESSED ] CHARACTERS ]

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

[ VALUE OF { FILENAME IS      { data-name-1 }                  } ]
           {                  { literal-1   }                  }
           { LIBRARY IS       { data-name-2 }                  }
           {                  { literal-2   }                  }
           { VOLUME IS        { data-name-3 }                  }
           {                  { literal-3   }                  }
           { SPACE IS           data-name-4                    }
           { POSITION IS        data-name-5                    }
           { INDEX AREA IS      data-name-6                    }
           { DATA AREA IS       data-name-7                    }
           { RECOVERY-BLOCKS IS { data-name-8 }                }
           {                    { literal-4   }                }
           { RECOVERY-STATUS IS   data-name-9                  }
           { DATABASE-NAME IS     data-name-10 }               ]

[ DATA { RECORD IS  } data-name-11 [ data-name-12 ] . . . ]
       { RECORDS ARE}

[ CODE-SET IS alphabet-name ] .
```

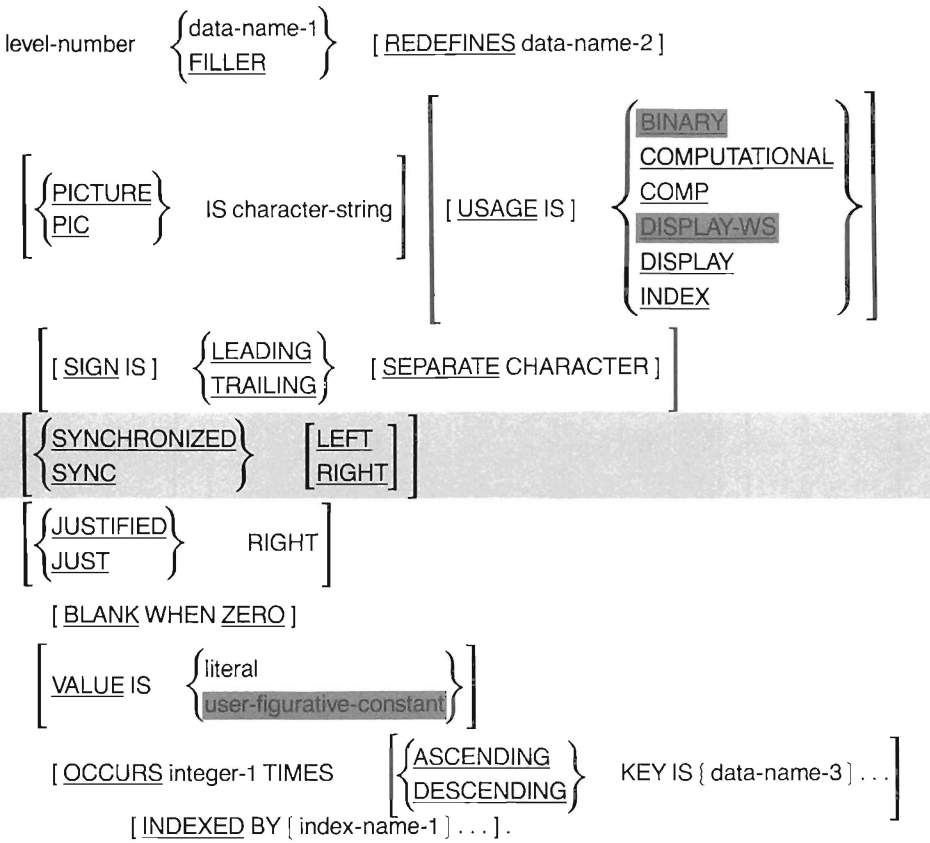
Format 2

SD file name

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

[ RECORD CONTAINS [ integer-2 TO ] integer-3 [ COMPRESSED ] CHARACTERS ]

[ VALUE OF { VOLUME IS { data-name-1 } } ]
           {           { literal-1   } }
           { SPACE IS   data-name-2    }

[ DATA { RECORD IS  } data-name-3 [ data-name-4 ] . . . ] .
       { RECORDS ARE}
```

Data and 77-Level Description Entries:

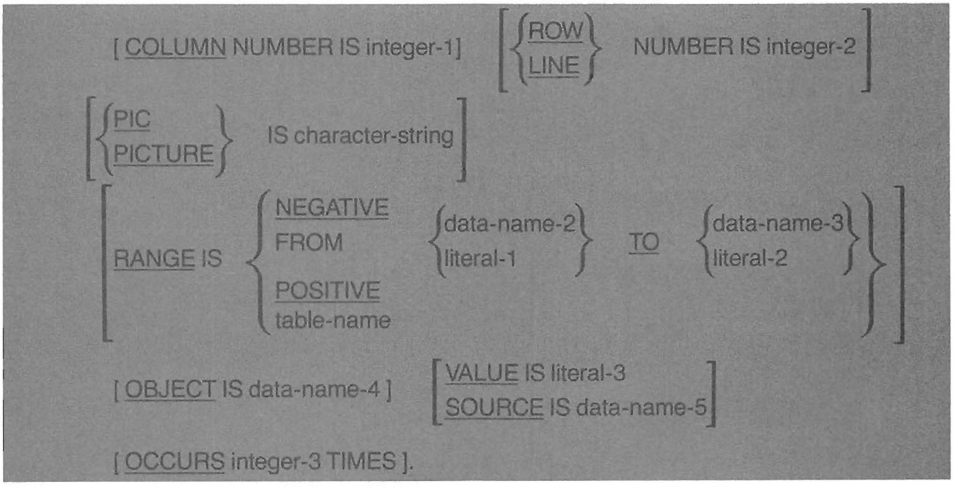
Format 1



Format 2

Record Description Entry for Workstation Screen:

01 record-name [USAGE IS] DISPLAY-WS.



General Format for Procedure Division

PROCEDURE DIVISION [USING data-name-1 [data-name-2] . . .] .
procedure division body.

Procedure Division Body:

Format 1

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

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

Format 2

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

Declarative Sentence:

USE STATEMENT

Format 1

```
USE AFTER STANDARD { EXCEPTION } PROCEDURE ON
                   { ERROR     }

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

Format 2

```
USE FOR DEBUGGING ON { procedure-name-1      [ procedure-name-2 ] . . . } .
                     { ALL PROCEDURES                                    }
```

Format 3

```
USE AFTER DEADLOCK.
```

VS COBOL STATEMENT FORMATS

ACCEPT STATEMENT

Format 1

```
ACCEPT identifier-1 [ identifier-2 ] . . .
```

Format 2

```
ACCEPT identifier FROM { DATE }
                       { DAY  }
                       { TIME }
```

ADD STATEMENT

Format 1

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

Format 2

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

Format 3

```
ADD   {CORRESPONDING}  identifier-1 TO identifier-2 [ ROUNDED ]
      {CORR         }
    [ ON SIZE ERROR imperative-statement ]
```

ALTER STATEMENT

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

CALL STATEMENT

```
CALL literal-1 [ USING identifier-1 [ identifier-2 ] ... ]
```

CLOSE STATEMENT

Consecutive File Organization

```
CLOSE file-name-1 [ {REEL  } [WITH NO REWIND]
                  {UNIT  } [FOR REMOVAL     ]
                  WITH     {NO REWIND}
                           {LOCK     }
                  file-name-2 [ {REEL  } [WITH NO REWIND] ]
                              {UNIT  } [FOR REMOVAL     ]
                              WITH     {NO REWIND}
                                       {LOCK     }
                  ]
```

Indexed and Relative File Organization

```
CLOSE { file-name-1 [ WITH LOCK ] } ...
```

COMPUTE STATEMENT

```
COMPUTE identifier-1 [ ROUNDED ] = arithmetic-expression
    [ ON SIZE ERROR imperative-statement ]
```

COPY STATEMENT

```
COPY { file-name  }
     { literal-1  }
   [ { IN } { library-name } [ { OF     } { volume-name } ] ]
   [ { OF } { literal-2    }   { IN     } { literal-3   }   ]
                               { ON     }
```

DELETE STATEMENT

```
DELETE file-name RECORD [ INVALID KEY imperative-statement ]
```

DISPLAY STATEMENT

```
DISPLAY { identifier-1 } [ identifier-2 ] ...
        { literal-1    }   [ literal-2   ]
```

DISPLAY AND READ STATEMENT

```
DISPLAY AND READ [ALTERED] record-name ON file-name
  [ [ONLY] { PFKEY  } { identifier-1 } [ identifier-2 ] ...
           { PFKEYS } { integer-1    }   [ integer-2   ]
    [ ON { PFKEY  } { identifier-3 } [ identifier-4 ] ... imperative-statement-1 ] ]
         { PFKEYS } { integer-3    }   [ integer-4   ]
  [ NO-MOD imperative-statement-2 ]
```

DIVIDE STATEMENT

Format 1

```
DIVIDE { identifier-1 } INTO identifier-2 [ ROUNDED ]
       { literal-1    }
  [ ON SIZE ERROR imperative-statement ]
```

Format 2

```
DIVIDE { identifier-1 } INTO { identifier-2 }
       { literal-1    }     { literal-2    }
  GIVING identifier-3 [ ROUNDED ]
  [ ON SIZE ERROR imperative-statement ]
```

Format 3

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

    [ ON SIZE ERROR imperative-statement ]
```

Format 4

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

    REMAINDER identifier-4  [ ON SIZE ERROR imperative-statement ]
```

Format 5

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

    REMAINDER identifier-4  [ ON SIZE ERROR imperative-statement ]
```

ENTER STATEMENT

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

EXIT STATEMENT

```
EXIT.
```

EXIT PROGRAM STATEMENT

```
EXIT PROGRAM.
```

FREE STATEMENT

```
FREE ALL  [ ON ERROR imperative-statement ]
```

GO TO STATEMENT

Format 1

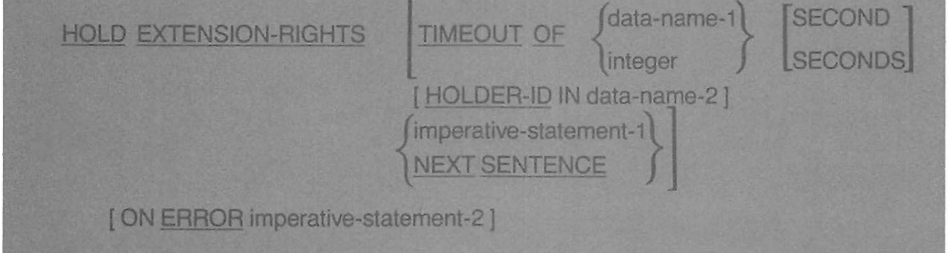
GO TO [procedure-name-1]

Format 2

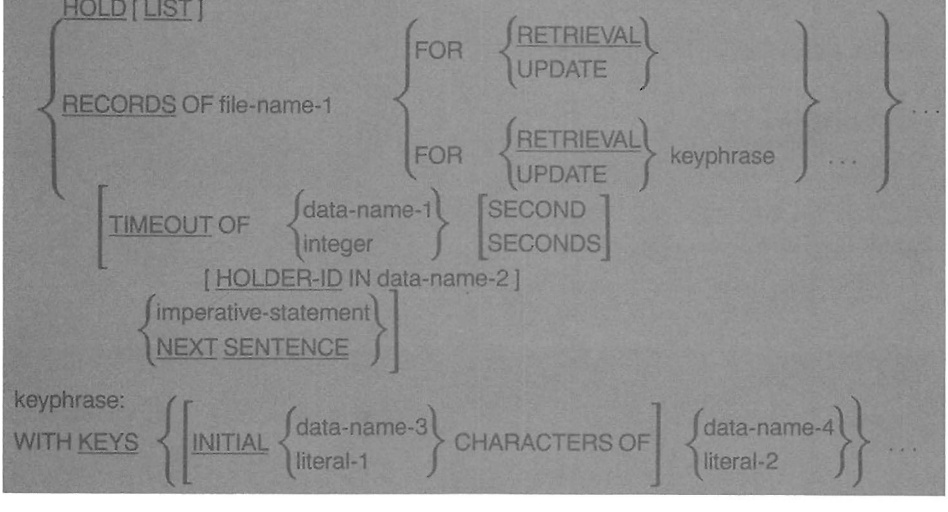
GO TO procedure-name-1 [procedure-name-2] . . . procedure-name-n
DEPENDING ON identifier

HOLD STATEMENT

Format 1



Format 2



OPTION:

IF STATEMENT

Format 1

```
IF condition THEN  { statement-1      }
                   { NEXT SENTENCE    }  .
```

Format 2

```
IF condition THEN  { statement-1      }
                   { NEXT SENTENCE    }  .
        ELSE       { statement-2      }
                   { NEXT SENTENCE    }
```

Format 3

```
IF FAC OF display-item { IS =                  }
                       { IS EQUAL TO           }  data-name
                       { IS NOT EQUAL TO       }
                       { IS NOT =              }

     THEN   { statement-1      }
            { NEXT SENTENCE    }
  [  ELSE   { statement-2      }  ]
            { NEXT SENTENCE    }
```

Format 4

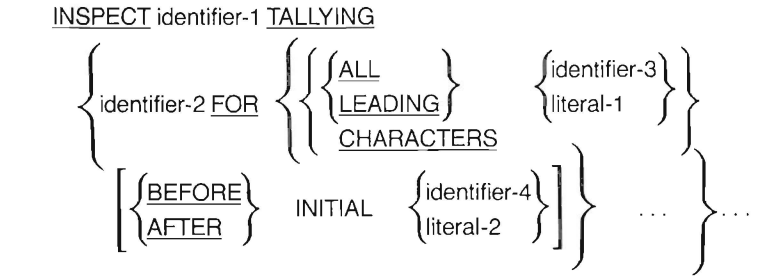
```
IF FAC OF display-item ALTERED THEN  { statement-1      }
                                     { NEXT SENTENCE    }
                                [ ELSE { statement-2      } ]
                                       { NEXT SENTENCE    }
```

Format 5

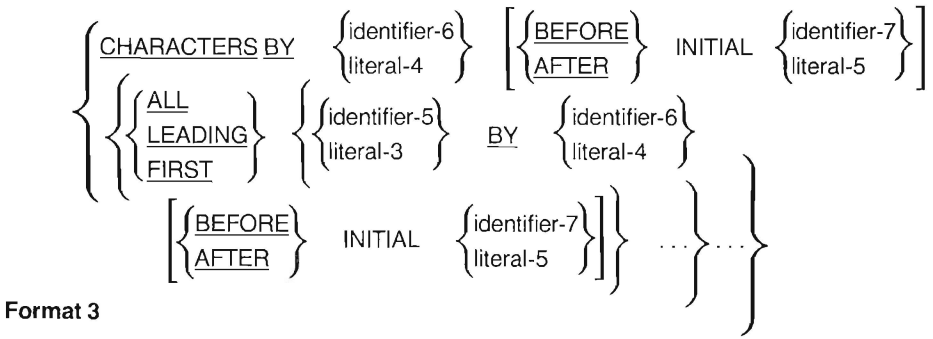
```
IF figurative-constant { IN }  { identifier          }  IS [ NOT ]  { ON  }
                       { OF }  { FAC OF display-item }              { OFF }

     THEN   { statement-1      }
            { NEXT SENTENCE    }
  [  ELSE   { statement-2      }  ]
            { NEXT SENTENCE    }
```

INSPECT STATEMENT

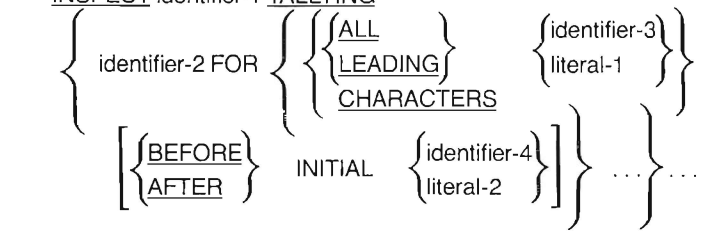
Format 1

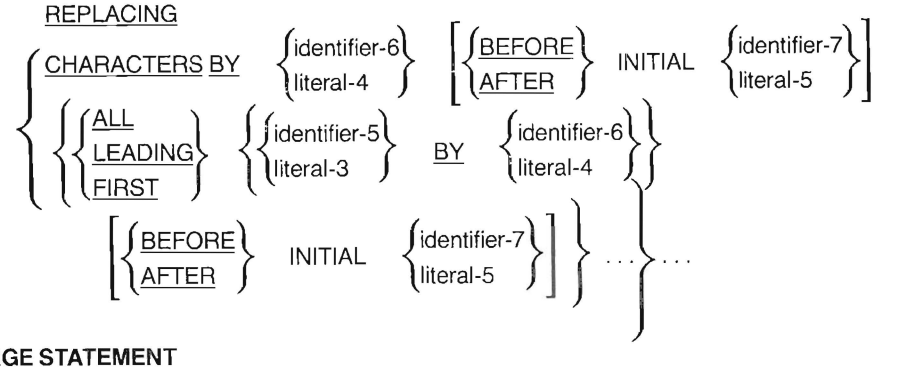


Format 2

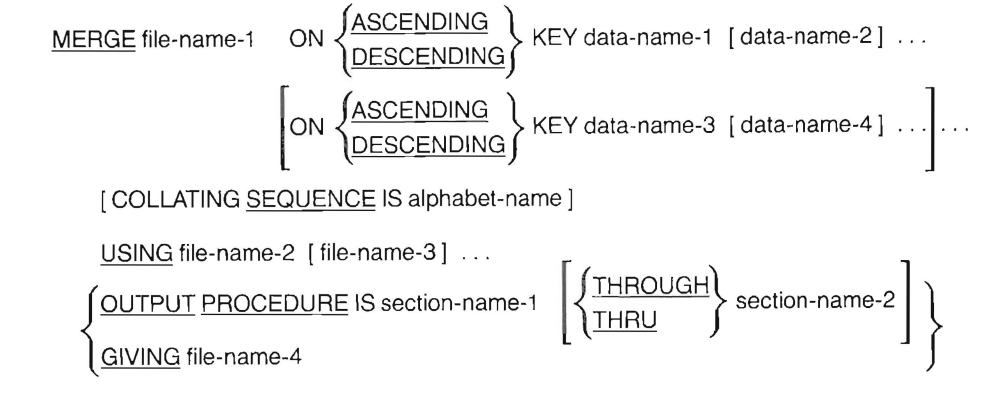


Format 3



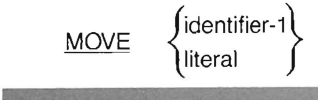


MERGE STATEMENT



MOVE STATEMENT

Format 1



Format 2

```
MOVE WITH CONVERSION identifier-1 TO identifier-2
    [ ON ERROR imperative-statement ]
```

Format 3

```
MOVE figurative-constant TO FAC OF data-name
```

Format 4

```
MOVE FAC OF data-name-1 TO data-name-2
```

Format 5

```
MOVE data-name TO ORDER-AREA OF record-name
```

Format 6

```
MOVE ORDER-AREA OF record-name TO data-name
```

Format 7

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

MULTIPLY STATEMENT

Format 1

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

Format 2

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

OPEN STATEMENT

Consecutive File Organization

```
OPEN { INPUT    { file-name-1 } ... }
     { OUTPUT   { file-name-2 } ... }
     { I-O      { file-name-3 } ... }  ...
     { SHARED   { file-name-4 } ... }
     { EXTEND   { file-name-5 } ... }
```

Indexed File Organization

```
OPEN { INPUT    { file-name-1 } ... }
     { OUTPUT   { file-name-2 } ... }
     { I-O      { file-name-3 } ... }  ...
     { SHARED   { file-name-4 } ... }
```

Relative File Organization

```
OPEN { INPUT    { file-name-1 } ... }
     { OUTPUT   { file-name-2 } ... }
     { I-O      { file-name-3 } ... }  ...
     { EXTEND   { file-name-4 } ... }
```

PERFORM STATEMENT

Format 1

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

Format 2

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

{ identifier-1 } TIMES
{ integer-1    }
```

Format 3

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

    UNTIL condition-1
```

Format 4

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

    VARYING { identifier-2 } FROM { identifier-3 }
            { index-name-1 }      { index-name-2 }
                                  { literal-1    }

    BY { identifier-4 } UNTIL condition-1
       { literal-2    }
```

READ STATEMENT

Consecutive File Organization

Format 1

```
READ file-name [ NEXT ] RECORD [ WITH HOLD ] [ INTO identifier ]
[ AT END imperative-statement ]
```

Format 2

```
READ file-name [ NEXT ] RECORD [ WITH HOLD | MODIFIABLE | ALTERED ] [ INTO identifier ]

[ { INVALID KEY | AT END } imperative-statement ]
```

Indexed File Organization

Format 1

```
READ file-name [ NEXT ] RECORD [ WITH HOLD ] [ INTO identifier ]

[ TIMEOUT OF { data-name-1 | integer } [ SECOND | SECONDS ]
    [ HOLDER-ID IN data-name-2 ]
    { imperative-statement | NEXT SENTENCE } ]

[ AT END imperative-statement ]
```

Format 2

```
READ file-name RECORD [ WITH HOLD ] [ INTO identifier ]

    [ KEY IS data-name-3 ]

[ TIMEOUT OF { data-name-1 | integer } [ SECOND | SECONDS ]
    [ HOLDER-ID IN data-name-2 ]
    { imperative-statement | NEXT SENTENCE }
[ INVALID KEY imperative-statement ] ]
```

Relative File Organization

Format 1

```
READ file-name [ NEXT ] RECORD [ WITH HOLD ] [ INTO identifier ]
    [ AT END imperative-statement ]
```

Format 2

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

READY TRACE STATEMENT

```
READY TRACE
```

RELEASE STATEMENT

```
RELEASE record-name [ FROM identifier ]
```

RESET TRACE STATEMENT

```
RESET TRACE
```

RETURN STATEMENT

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

REWRITE STATEMENT

Consecutive File Organization

Format 1

```
REWRITE record-name [ FROM identifier ]
```

Format 2

```
REWRITE record-name [ FROM identifier ] AFTER
    { ALARM
    { SETTING CURSOR COLUMN { identifier-1 } { ROW  } { identifier-2 }
    {                       { integer-1    } { LINE } { integer-2    }
    { ROLL DOWN
    { ROLL UP
    { ERASE PROTECT
    { ERASE MODIFY
```

Indexed and Relative File Organization

REWRITE record-name [FROM identifier]

[INVALID KEY imperative-statement]

ROLLBACK STATEMENT

ROLLBACK [ON ERROR 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          }
```

Format 2

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

WHEN { data-name-1      { IS EQUAL TO } { identifier-3   }
     {                  { IS =        } { literal-1      }
     { condition-name-1                 { arithmetic-expression-1 } }
   [ AND { data-name-2      { IS EQUAL TO } }
         {                  { IS =        } }
         { condition-name-2                 }
                 { identifier-4   }
                 { literal-2      }
                 { arithmetic-expression-2 } } ] ...

{ imperative-statement-2 }
{ NEXT SENTENCE          }
```

SET STATEMENT

Format 1

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

Format 2

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

Format 3

```
SET figurative-constant  { IN FAC OF }  { display-item }  { ON  }
                         { OF        }  { identifier-5 }  { OFF }
```

SORT STATEMENT

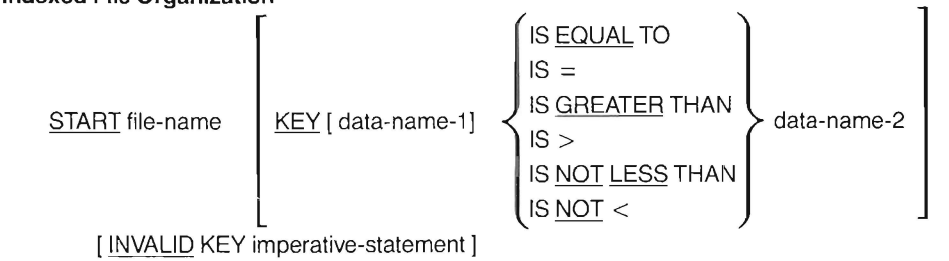
```
SORT file-name-1 ON { ASCENDING  }  KEY { data-name-1 } ...
                    { DESCENDING }

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

    [ WITH DUPLICATES IN ORDER ]
    [ COLLATING SEQUENCE IS alphabet-name ]

    { INPUT  PROCEDURE IS section-name-1 [ { THROUGH } section-name-2 ] }
    {                                     { THRU    }                  }
    { USING  file-name-2 [ file-name-3 ] ...                           }
    { OUTPUT PROCEDURE IS section-name-3 [ { THROUGH } section-name-4 ] }
    {                                     { THRU    }                  }
    { GIVING file-name-4                                               }
```

START STATEMENT

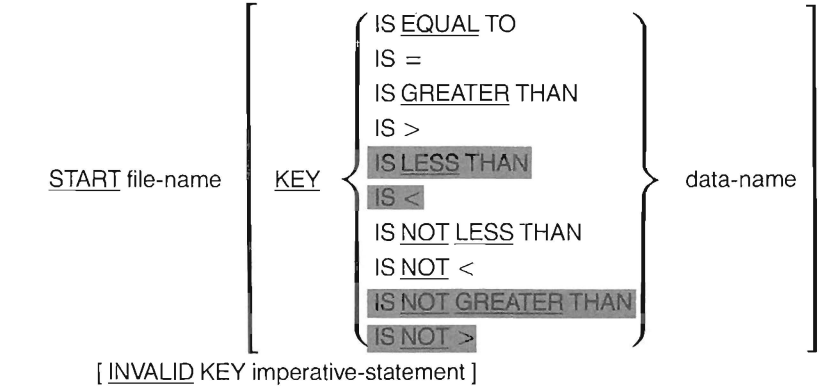
Consecutive File Organization

```
START file-name
```

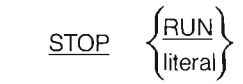
Indexed File Organization



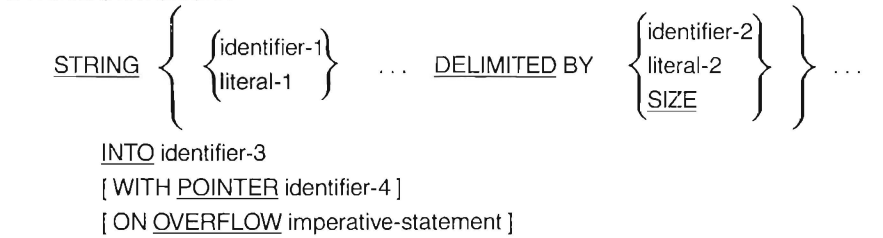
Relative File Organization



STOP STATEMENT



STRING STATEMENT



SUBTRACT STATEMENT

Format 1

```
SUBTRACT  { identifier-1 }  [ identifier-2 ]  ... FROM identifier-m [ ROUNDED ]
          { literal-1    }  [ literal-2   ]
    [ ON SIZE ERROR imperative-statement ]
```

Format 2

```
SUBTRACT  { identifier-1 }  [ identifier-2 ]  ... FROM  { identifier-m }
          { literal-1    }  [ literal-2   ]             { literal-m    }
    GIVING identifier-n [ ROUNDED ]
        [ ON SIZE ERROR imperative-statement ]
```

Format 3

```
SUBTRACT  { CORRESPONDING }  identifier-1 FROM identifier-2 [ ROUNDED ]
          { CORR          }
    [ ON SIZE ERROR imperative-statement ]
```

UNSTRING STATEMENT

```
UNSTRING identifier-1
  [ DELIMITED 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 ]
```

WRITE STATEMENT

Consecutive File Organization

```
WRITE record-name [ FROM identifier-1]
  [ { BEFORE }  ADVANCING  { { identifier-2 } { LINE  } }  ]
    { AFTER  }             { { integer     } { LINES } }
                           { user-figurative-constant   }
                           { PAGE                       }
```

Indexed File Organization

```
WRITE record-name [ FROM identifier ]
    [ TIMEOUT OF { data-name-1 } [ SECOND  ]
                 { integer     }   [ SECONDS ] ]
        [ HOLDER-ID IN data-name-2 ]
            { imperative-statement }
            { NEXT SENTENCE        } ]
    [ INVALID KEY imperative-statement ]
```

Relative File Organization

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

GENERAL FORMATS FOR CONDITIONS

Condition:

Simple condition
NOT Simple condition
Combined condition
NOT Combined condition

Simple conditions:

Class, Condition-name, Figurative-constant, Modified-data-tag, Relation, Sign, and Switch-status.

Combined condition:

```
condition-1 { AND } condition-2
            { OR  }
```

Relational operator:

```
{ IS [ NOT ] GREATER THAN }
{ IS [ NOT ] LESS THAN    }
{ IS [ NOT ] EQUAL TO     }
{ IS [ NOT ] <            }
{ IS [ NOT ] >            }
{ IS [ NOT ] =            }
```

Class condition:

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

Condition-name and Switch-status condition:

condition-name

Figurative-constant condition:

figurative-constant { IN / OF } { identifier / FAC OF display-item } IS [NOT] { ON / OFF }

Modified-data-tag condition:

FAC OF display-item ALTERED

Relation condition:

{ identifier-1 / literal-1 / arithmetic-expression-1 } [relational-operator] { identifier-2 / literal-2 / arithmetic-expression-2 }

Sign condition:

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

Abbreviated combined relation condition:

relation-condition { { AND / OR } [NOT] [relational-operator] object } ...

MISCELLANEOUS FORMATS

QUALIFICATION

Format 1

{ data-name-1 / condition-name } [{ OF / IN } data-name-2] ...

Format 2

paragraph-name [{ OF / IN } section-name]

Format 3

text-name [{ OF / IN } library-name [{ OF / IN / ON } volume-name]]

SUBSCRIPTING

```
{data-name     }
{condition-name}  ( subscript-1 [ subscript-2 [ subscript-3 ] ] )
```

INDEXING

```
{data-name     }  ( ( {index-name-1 [ ± literal-2 ]}
{condition-name}    {literal-1                     }

    [ {index-name-2 [ ± literal-4 ]} ]
    [ {literal-3                   } ]

    [ {index-name-3 [ ± literal-6 ]} ]  ] ) )
    [ {literal-5                   } ]
```

IDENTIFIER

Format 1

```
data-name-1 [ {OF} data-name-2 ] ... [ ( subscript-1 [ subscript-2
              {IN}                     [ subscript-3 ] ] ) ]
```

Format 2

```
data-name-1 [ {OF} data-name-2 ] ...
              {IN}
    [ ( {index-name-1 [ ± literal-2 ]}
        {literal-1                  }
      [ {index-name-2 [ ± literal-4 ]}
        {literal-3                  }
      [ {index-name-3 [ ± literal-6 ]}  ] ] ) ]
        {literal-5                  }
```

GLOSSARY

Alphabet-Name	A user-defined word, in the SPECIAL-NAMES paragraph of the Environment Division, that assigns a name to a specific character set and/or collating sequence.
Area A	This area occupies character positions 8 through 11 on a line of a COBOL source program. It is reserved for the beginning of division headers, section names, paragraph names, level indicators, and certain level numbers.
Area B	This area occupies character positions 12 through 72. It contains all remaining source code in a COBOL source program.
Arithmetic Expression	An identifier of a numeric elementary item, a numeric literal, such identifiers and literals separated by arithmetic operators, two arithmetic expressions separated by an arithmetic operator, or an arithmetic expression enclosed in parentheses.
Character-String	A sequence of contiguous characters which form a COBOL word, a literal, a PICTURE character-string, or a comment-entry.
COBOL Word	A character-string of not more than 30 characters which forms a user-defined word, a system-name, or a reserved word.
Comment-Entry	An entry in the Identification Division that may be any combination of characters from the computer's character set.
Comment Line	A source program line represented by an asterisk (*) in the indicator area of the line and any characters from the computer's character set in area A and area B of the line. The comment line serves only for documentation in a program. A special form of comment line represented by a slant (/) in the indicator area of the line and any characters from the computer's character set in area A and area B of that line causes page ejection prior to printing the comment.
Condition	When the term 'condition' (condition-1, condition-2, . . .) appears in these language specifications in or in reference to 'condition' (condition-1, condition-2, . . .) of a general format, it is a conditional expression consisting of either a simple condition optionally parenthesized, or a combined condition consisting of the syntactically correct combination of simple conditions, logical operators, and parentheses, for which a truth value can be determined.

Condition-Name	A user-defined word that assigns a name to a subset of values that a conditional variable may assume; or a user-defined word assigned to a status of an implementor-defined switch or device. When 'condition-name' is used in the general formats, it represents a unique data item reference consisting of a syntactically correct combination of a condition-name, together with qualifiers and subscripts, as required for uniqueness of reference.
Data Item	A unit of data (excluding literals) defined by the COBOL program.
Data-Name	A user-defined word that names a data item described in a data description entry. When used in the general formats, 'data-name' represents a word which must not be subscripted or qualified unless specifically permitted by the rules of the format.
Declarative Sentence	A compiler directing sentence consisting of a single USE statement terminated by the separator period.
Display-Item	A data-name that is defined within a Record Description Entry for a Workstation Screen.
Figurative Constant	Reserved words that are used to name and reference specific constant values. Specifically, they are zero(s or es), space(s), high-value(s), low-value(s), and quote(s).
File-Name	A user-defined word that names a file described in a file description entry or a sort-merge file description entry within the File Section of the Data Division.
Hexadecimal Value	A hexadecimal value can be either two or four hexadecimal characters. The hexadecimal value must be enclosed in quotation marks. A hexadecimal character is any of the characters, '0', '1', . . . , '9', or 'A', . . . , 'F'.
Identifier	A syntactically correct combination of a data-name, with its qualifiers or subscripts, as required for uniqueness of reference, that names a data item. The rules for 'identifier' associated with the general formats may, however, specifically prohibit qualification or subscripting.
Imperative Statement	A statement that begins with an imperative verb and specifies an unconditional action to be taken. An imperative statement may consist of a sequence of imperative statements.
Implementor-Name	A system-name that refers to a particular feature available on that implementor's computing system.

Index	A computer storage area or register, the content of which represents the identification of a particular element in a table.
Index-Name	A user-defined word that names an index associated with a specific table.
Indicator Area	This area occupies column seven on a line in a COBOL source program. It is used to indicate continuation lines and comment lines.
Integer	A numeric literal or a numeric data item that does not include any digit position to the right of the decimal point. When the term 'integer' appears in general formats, integer must not be a numeric data item, and must not be signed, nor zero unless explicitly allowed by the rules of that format.
Language-Name	A system-name that specifies a particular programming language.
Level-Number	A user-defined word, expressed as a one or two digit number, which indicates the hierarchical position of a data item or the special properties of a data description entry. Level-numbers in the range 1 through 49 indicate the position of a data item in the hierarchical structure of a logical record. Level-numbers in the range 1 through 9 may be written either as a single digit or as a zero followed by a significant digit. Level-numbers 77 and 88 identify special properties of a data description entry.
Library-Name	A user-defined word that names a VS COBOL library that is to be used by the compiler for a given source program compilation.
Literal	A character-string whose value is implied by the ordered set of characters comprising the string. A literal can be numeric or nonnumeric.
Mnemonic-Name	A user-defined word that is associated in the Environment Division with a specific implementor-name.
Nonnumeric Literal	A literal bounded by quotation marks. The string of characters may include any character in the computer's character set.
Numeric Literal	A literal composed of one or more numeric characters that may contain either a decimal point, or an algebraic sign, or both. The decimal point must not be the rightmost character. The algebraic sign, if present, must be the leftmost character.

Paragraph-Name	A user-defined word that identifies and begins a paragraph in the Procedure Division.
Parameter-Reference-Name	A name that identifies a specific GETPARM request.
Procedure-Name	A user-defined word which is used to name a paragraph or section in the Procedure Division. It consists of a paragraph-name that may be qualified or a section-name.
Program-Name	In the Identification Division, a user-defined word that identifies a COBOL source program.
Record Description Entry	The total set of data description entries associated with a particular record.
Record-Name	A user-defined word that names a record described in a record description entry in the Data Division of a COBOL program.
Routine-Name	A user-defined word that defines a procedure written in a language other than COBOL.
Section-Name	A user-defined word which names a section in the Procedure Division.
Segment-Number	A user-defined word which classifies sections in the Procedure Division for purposes of segmentation. Segment-numbers may contain only the characters '0', '1', . . . , '9'. A segment-number may be expressed either as a one or two digit number.
Sentence	A sequence of one or more statements, the last of which is terminated by a period.
Sequence Number Area	This area occupies the character positions 1 through 6 on a line in a COBOL source program. It consists of six digits in the sequence area and labels each source program line. The EDITOR assigns these numbers automatically when the COBOL source text is entered.
Statement	A syntactically valid combination of words and symbols, beginning with a verb written in a COBOL source program.
Subscript	An integer whose value identifies a particular item in a table.
System-Name	A COBOL word which is used to communicate with the operating environment.

Table-Name	A user-defined word that references a numeric or alpha-numeric data item whose data description contains an OCCURS clause.
User-Figurative Constant	A user-defined word that is used to name and reference a hexadecimal character.
Volume-Name	A user-defined word that names a VS volume.

VS COBOL RESERVED WORDS

ACCEPT
ACCESS
ADD
ADVANCING
AFTER
ALARM
ALL
ALPHABETIC
ALSO
ALTER
ALTERED
ALTERNATE
AND
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR

BEFORE
BINARY
BLANK
BLOCK
BLOCKS
BOTTOM
BUFFER
BY

CALL
CANCEL
CD
CF
CH
CHARACTER
CHARACTERS
CLOCK-UNITS
CLOSE
COBOL
CODE
CODE-SET
COLLATING
COLUMN
COMMA
COMMUNICATION
COMP
COMPRESSED
COMPUTATIONAL
COMPUTE
CONFIGURATION
CONTAINS
CONTROL
CONTROLS
CONVERSION
COPY
CORR
CORRESPONDING
COUNT
CURRENCY
CURSOR

DATA
DATABASE-NAME
DATE
DATE-COMPILED
DATE-WRITTEN
DAY
DE
DEADLOCK
DEBUG-CONTENTS
DEBUG-ITEM
DEBUG-LINE
DEBUG-NAME
DEBUG-SUB-1
DEBUG-SUB-2
DEBUG-SUB-3
DEBUGGING
DECIMAL-POINT
DECLARATIVES
DELETE
DELETION
DELIMITED
DELIMITER
DEPENDING
DESCENDING
DESTINATION
DETAIL
DISABLE
DISPLAY
DISPLAY-WS
DIVIDE
DIVISION
DOWN
DUPLICATES
DYNAMIC

EGI
ELSE
EMI
ENABLE
END
END-OF-PAGE
ENTER
ENVIRONMENT
EOP
EQUAL
ERASE
ERROR
ESI
EVERY
EXCEPTION
EXCLUSIVE
EXIT
EXTEND
EXTENSION-RIGHTS

FAC
FD
FIGURATIVE-CONSTANTS
FILE
FILE-CONTROL
FILENAME
FILLER
FINAL
FIRST
FOOTING
FOR
FREE
FROM

GENERATE
GIVING
GO
GREATER
GROUP

HEADING
HIGH-VALUE
HIGH-VALUES
HOLD

VS COBOL RESERVED WORDS (Continued)

HOLDER-ID

IDENTIFICATION
IF
IN
INDEX
INDEXED
INDICATE
INITIAL
INITIATE
INPUT
INPUT-OUTPUT
INSPECT
INSTALLATION
INTO
INVALID
INVOKE
I-O
I-O-CONTROL
IS

JUST
JUSTIFIED

KEY
KEYS

LABEL
LAST
LEADING
LEFT
LENGTH
LESS
LIBRARY
LIMIT
LIMITS
LINAGE
LINAGE-COUNTER
LINE
LINE-COUNTER
LINES
LINKAGE
LIST
LOCK
LOW-VALUE
LOW-VALUES

MEMORY
MERGE
MESSAGE
MODE
MODIFIABLE
MODIFY
MODULES
MOVE
MULTIPLE
MULTIPLY

NATIVE
NEGATIVE
NEXT
NO
NODISPLAY
NO-MOD
NOT
NUMBER
NUMERIC

OBJECT
OBJECT-COMPUTER
OCCURS
OF
OFF
OMITTED
ON
ONLY
OPEN
OPTIONAL
OR
ORDER
ORDER-AREA
ORGANIZATION
OUTPUT
OVERFLOW

PAGE
PAGE-COUNTER
PERFORM
PF
PFKEY
PFKEYS
PH
PIC
PICTURE
PLUS
POINTER
POSITION
POSITIVE
PRINTING
PRIOR
PROCEDURE
PROCEDURES
PROCEED
PROGRAM
PROGRAM-ID
PROTECT

QUEUE
QUOTE
QUOTES

RANDOM
RANGE
RD
READ
READY
RECEIVE
RECORD
RECORDS
REDEFINES
REEL
REFERENCES
RELATIVE
RELEASE
REMAINDER
REMOVAL
RENAMES
REPLACING
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RESET
RESTART
RETRIEVAL
RETURN
RETURN-CODE
REVERSED
REWIND
REWRITE

VS COBOL RESERVED WORDS (Continued)

RF
RH
RIGHT
ROLL
ROLLBACK
ROUNDED
ROW
RUN

SAME
SD
SEARCH
SECOND
SECONDS
SECTION
SECURITY
SEGMENT
SEGMENT-LIMIT
SELECT
SEND
SENTENCE
SEPARATE
SEQUENCE
SEQUENTIAL
SET
SETTING
SHARED
SIGN
SIZE
SORT
SORT-MERGE
SOURCE
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STANDARD-1
START
STATUS
STOP
STRING
SUB-QUEUE-1
SUB-QUEUE-2
SUB-QUEUE-3
SUBTRACT
SUM
SUPPRESS
SWITCH-1
SWITCH-2
SWITCH-3
SWITCH-4
SWITCH-5
SWITCH-6
SWITCH-7
SYMBOLIC
SYNC
SYNCHRONIZED

TABLE
TALLYING
TAPE
TERMINAL
TERMINATE
TEXT
THAN
THEN
THROUGH
THRU
TIME
TIMEOUT
TIMES
TO
TOP
TRACE
TRAILING
TYPE

UNIT
UNSTRING
UNTIL
UP
UPDATE
UPON
USAGE
USE
USING

VALUE
VALUES
VARYING
VOLUME

WANG-VS
WHEN
WITH
WORDS
WORKING-STORAGE
WRITE

ZERO
ZEROES
ZEROS

+
–
*
/
**
<
>
=

HEXADECIMAL TO DECIMAL CONVERSION

Use this table to convert a hexadecimal number to a decimal number. The place value of each digit in a hexadecimal number is given in columns one to six. Determine each place value and then add the decimal values. For example, to determine the decimal equivalent of hex ABC, find the decimal value of A in the third hexadecimal column (2,560), of B in the second column (176), of C in the first column (12), and take their sum (2748).

HEXADECIMAL COLUMNS

Hex	Sixth Dec	Fifth Dec	Fourth Dec	Third Dec	Second Dec	First Dec	Hex
0	0	0	0	0	0	0	0
1	1,048,576	65,536	4,096	256	16	1	1
2	2,097,152	131,072	8,192	512	32	2	2
3	3,145,728	196,608	12,288	768	48	3	3
4	4,194,304	262,144	16,384	1,024	64	4	4
5	5,242,880	327,680	20,480	1,280	80	5	5
6	6,291,456	393,216	24,576	1,536	96	6	6
7	7,340,032	458,752	28,672	1,792	112	7	7
8	8,388,608	524,288	32,768	2,048	128	8	8
9	9,437,184	589,824	36,864	2,304	144	9	9
A	10,485,760	655,360	40,960	2,560	160	10	A
B	11,534,336	720,896	45,056	2,816	176	11	B
C	12,582,912	786,432	49,152	3,072	192	12	C
D	13,631,488	851,968	53,248	3,328	208	13	D
E	14,680,064	917,504	57,344	3,584	224	14	E
F	15,728,640	983,040	61,440	3,840	240	15	F

POWERS OF 2 AND 16

POWERS OF 2		POWERS OF 16	
Value	Exponent	Value	Exponent
1	0	1	0
2	1	16	1
4	2	256	2
8	3	4,096	3
16	4	65,536	4
32	5	1,048,576	5
64	6	16,777,216	6
128	7	268,435,456	7
256	8	4,294,967,296	8
512	9	68,719,476,736	9
1,024	10	1,099,511,627,776	A
2,048	11	17,592,186,044,416	B
4,096	12	281,474,976,710,656	C
8,192	13	4,503,599,627,370,496	D
16,384	14	72,057,594,037,927,936	E
32,768	15	1,152,921,504,606,846,976	F
65,536	16		
131,072	17		
262,144	18		
524,288	19		
1,048,576	20		
2,097,152	21		
4,194,304	22		
8,388,608	23		

FIELD ATTRIBUTE CHARACTERS

DISPLAY ATTRIBUTES				HEXADECIMAL CHARACTERS
BRIGHT	MODIFY	ALL	NOLINE	80
BRIGHT	MODIFY	UPPERCASE	NOLINE	81
BRIGHT	MODIFY	NUMERIC	NOLINE	82
BRIGHT	PROTECT	ALL	NOLINE	84
BRIGHT	PROTECT	UPPERCASE	NOLINE	85
BRIGHT	PROTECT	NUMERIC	NOLINE	86
DIM	MODIFY	ALL	NOLINE	88
DIM	MODIFY	UPPERCASE	NOLINE	89
DIM	MODIFY	NUMERIC	NOLINE	8A
DIM	PROTECT	ALL	NOLINE	8C
DIM	PROTECT	UPPERCASE	NOLINE	8D
DIM	PROTECT	NUMERIC	NOLINE	8E
BLINK	MODIFY	ALL	NOLINE	90
BLINK	MODIFY	UPPERCASE	NOLINE	91
BLINK	MODIFY	NUMERIC	NOLINE	92
BLINK	PROTECT	ALL	NOLINE	94
BLINK	PROTECT	UPPERCASE	NOLINE	95
BLINK	PROTECT	NUMERIC	NOLINE	96
BLANK	MODIFY	ALL	NOLINE	98
BLANK	MODIFY	UPPERCASE	NOLINE	99
BLANK	MODIFY	NUMERIC	NOLINE	9A
BLANK	PROTECT	ALL	NOLINE	9C
BLANK	PROTECT	UPPERCASE	NOLINE	9D
BLANK	PROTECT	NUMERIC	NOLINE	9E
BRIGHT	MODIFY	ALL	LINE	A0
BRIGHT	MODIFY	UPPERCASE	LINE	A1
BRIGHT	MODIFY	NUMERIC	LINE	A2
BRIGHT	PROTECT	ALL	LINE	A4
BRIGHT	PROTECT	UPPERCASE	LINE	A5
BRIGHT	PROTECT	NUMERIC	LINE	A6
DIM	MODIFY	ALL	LINE	A8
DIM	MODIFY	UPPERCASE	LINE	A9
DIM	MODIFY	NUMERIC	LINE	AA
DIM	PROTECT	ALL	LINE	AC
DIM	PROTECT	UPPERCASE	LINE	AD
DIM	PROTECT	NUMERIC	LINE	AE
BLINK	MODIFY	ALL	LINE	B0
BLINK	MODIFY	UPPERCASE	LINE	B1
BLINK	MODIFY	NUMERIC	LINE	B2
BLINK	PROTECT	ALL	LINE	B4
BLINK	PROTECT	UPPERCASE	LINE	B5
BLINK	PROTECT	NUMERIC	LINE	B6
BLANK	MODIFY	ALL	LINE	B8
BLANK	MODIFY	UPPERCASE	LINE	B9
BLANK	MODIFY	NUMERIC	LINE	BA
BLANK	PROTECT	ALL	LINE	BC
BLANK	PROTECT	UPPERCASE	LINE	BD
BLANK	PROTECT	NUMERIC	LINE	BE

TRANSLATION TABLE

Dec	Hex	Binary	ASCII Printer Graphics	ASCII Display Graphics	EBCDIC
0	00	00000000			
1	01	00000001			
2	02	00000010			
3	03	00000011			
4	04	00000100			
5	05	00000101			
6	06	00000110			
7	07	00000111			
8	08	00001000			
9	09	00001001			
10	0A	00001010			
11	0B	00001011			
12	0C	00001100		!!	
13	0D	00001101			
14	0E	00001110			
15	0F	00001111			
16	10	00010000		a	
17	11	00010001		e	
18	12	00010010		i	
19	13	00010011		o	
20	14	00010100		u	
21	15	00010101		a	
22	16	00010110		e	
23	17	00010111		i	
24	18	00011000		o	
25	19	00011001		u	
26	1A	00011010		a	
27	1B	00011011		e	
28	1C	00011100		u	
29	1D	00011101		A	
30	1E	00011110		O	
31	1F	00011111		U	
32	20	00100000	space	space	
33	21	00100001	!	!	
34	22	00100010	"	"	
35	23	00100011	#	#	
36	24	00100100	$	$	
37	25	00100101	%	%	
38	26	00100110	&	&	
39	27	00100111	'	'	

TRANSLATION TABLE (continued)

Dec	Hex	Binary	ASCII Printer Graphics	ASCII Display Graphics	EBCDIC
40	28	00101000	(	(	
41	29	00101001	)	)	
42	2A	00101010	*	*	
43	2B	00101011	+	+	
44	2C	00101100	,	,	
45	2D	00101101	-	-	
46	2E	00101110	.	.	
47	2F	00101111	/	/	
48	30	00110001	0	0	
49	31	00110000	1	1	
50	32	00110010	2	2	
51	33	00110011	3	3	
52	34	00110100	4	4	
53	35	00110101	5	5	
54	36	00110110	6	6	
55	37	00110111	7	7	
56	38	00111000	8	8	
57	39	00111001	9	9	
58	3A	00111010	:	:	
59	3B	00111011	;	;	
60	3C	00111100	<	<	
61	3D	00111101	=	=	
62	3E	00111110	>	>	
63	3F	00111111	?	?	
64	40	01000000	@	@	
65	41	01000001	A	A	
66	42	01000010	B	B	
67	43	01000011	C	C	
68	44	01000100	D	D	
69	45	01000101	E	E	
70	46	01000110	F	F	
71	47	01000111	G	G	
72	48	01001000	H	H	
73	49	01001001	I	I	
74	4A	01001010	J	J	¢
75	4B	01001011	K	K	.
76	4C	01001100	L	L	<
77	4D	01001101	M	M	(
78	4E	01001110	N	N	+
79	4F	01001111	O	O	!

TRANSLATION TABLE (continued)

Dec	Hex	Binary	ASCII Printer Graphics	ASCII Display Graphics	EBCDIC
80	50	01010000	P	P	&
81	51	01010001	Q	Q	
82	52	01010010	R	R	
83	53	01010011	S	S	
84	54	01010100	T	T	
85	55	01010101	U	U	
86	56	01010110	V	V	
87	57	01010111	W	W	
88	58	01011000	X	X	
89	59	01011001	Y	Y	
90	5A	01011010	Z	Z	!
91	5B	01011011	[	[	$
92	5C	01011100	¢	¢	*
93	5D	01011101	]	]	)
94	5E	01011110	•	•	;
95	5F	01011111	–		
96	60	01100000			—
97	61	01100001	a	a	/
98	62	01100010	b	b	
99	63	01100011	c	c	
100	64	01100100	d	d	
101	65	01100101	e	e	
102	66	01100110	f	f	
103	67	01100111	g	g	
104	68	01101000	h	h	
105	69	01101001	i	i	
106	6A	01101010	j	j	
107	6B	01101011	k	k	,
108	6C	01101100	l	l	%
109	6D	01101101	m	m	–
110	6E	01101110	n	n	>
111	6F	01101111	o	o	?
112	70	01110000	p	p	
113	71	01110000	q	q	
114	72	01110010	r	r	
115	73	01110011	s	s	
116	74	01110100	t	t	
117	75	01110101	u	u	
118	76	01110110	v	v	
119	77	01110111	w	w	

TRANSLATION TABLE (continued)

Dec	Hex	Binary	ASCII Printer Graphics	ASCII Display Graphics	EBCDIC
120	78	01111000	x	x	
121	79	01111001	y	y	
122	7A	01111010	z	z	:
123	7B	01111011			#
124	7C	01111100			@
125	7D	01111101		e	'
126	7E	01111110		c	=
127	7F	01111111			"
128	80	10000000			
129	81	10000001			a
130	82	10000010			b
131	83	10000011			c
132	84	10000100			d
133	85	10000101			e
134	86	10000110			f
135	87	10000111			g
136	88	10001000			h
137	89	10001001			i
138	8A	10001010			
139	8B	10001011			
140	8C	10001100			
141	8D	10001101			
142	8E	10001110			
143	8F	10001111			
144	90	10010000			
145	91	10010001			j
146	92	10010010			k
147	93	10010011			l
148	94	10010100			m
149	95	10010101			n
150	96	10010110			o
151	97	10010111			p
152	98	10011000			q
153	99	10011001			r
154	9A	10011010			
155	9B	10011011			
156	9C	10011100			
157	9D	10011101			
158	9E	10011110			
159	9F	10011111			

TRANSLATION TABLE (continued)

Dec	Hex	Binary	ASCII Printer Graphics	ASCII Display Graphics	EBCDIC
160	A0	10100000			
161	A1	10100001			
162	A2	10100010			s
163	A3	10100011			t
164	A4	10100100			u
165	A5	10100101			v
166	A6	10100110			w
167	A7	10100111			x
168	A8	10101000			y
169	A9	10101001			z
170	AA	10101010			
171	AB	10101011			
172	AC	10101100			
173	AD	10101101			
174	AE	10101110			
175	AF	10101111			
176	B0	10110000			
177	B1	10110001			
178	B2	10110010			
179	B3	10110011			
180	B4	10110100			
181	B5	10110101			
182	B6	10110110			
183	B7	10110111			
184	B8	10111000			
185	B9	10111001			
186	BA	10111010			
187	BB	10111011			
188	BC	10111100			
189	BD	10111101			
190	BE	10111110			
191	BF	10111111			
192	C0	11000000			{
193	C1	11000001			A
194	C2	11000010			B
195	C3	11000011			C
196	C4	11000100			D
197	C5	11000101			E
198	C6	11000110			F
199	C7	11000111			G

TRANSLATION TABLE (continued)

Dec	Hex	Binary	ASCII Printer Graphics	ASCII Display Graphics	EBCDIC
200	C8	11001000			H
201	C9	11001001			I
202	CA	11001010			
203	CB	11001011			
204	CC	11001100			
205	CD	11001101			
206	CE	11001110			
207	CF	11001111			
208	D0	11010000			}
209	D1	11010001			J
210	D2	11010010			K
211	D3	11010011			L
212	D4	11010100			M
213	D5	11010101			N
214	D6	11010110			O
215	D7	11010111			P
216	D8	11011000			Q
217	D9	11011001			R
218	DA	11011010			
219	DB	11011011			
220	DC	11011100			
221	DD	11011101			
222	DE	11011110			
223	DF	11011111			
224	E0	11100000			
225	E1	11100001			
226	E2	11100010			S
227	E3	11100011			T
228	E4	11100100			U
229	E5	11100101			V
230	E6	11100110			W
231	E7	11100111			X
232	E8	11101000			Y
233	E9	11101001			Z
234	EA	11101010			

WANG

Title VS COBOL QUICK REFERENCE GUIDE

Publications Number 800-6200-05

Customer Comment Form

Help Us Help You

We've worked hard to make this document useful, readable, and technically accurate. Did we succeed? Only you can tell us! Your comments and suggestions will help us improve our technical communications. Please take a few minutes to let us know how you feel.

Please rate the quality of this publication in each of the following areas.

	VERY GOOD	GOOD	FAIR	POOR	VERY POOR
Technical Accuracy — Does the system work the way the manual says it does?	☐	☐	☐	☐	☐
Readability — Is the manual easy to read and understand?	☐	☐	☐	☐	☐
Clarity — Are the instructions easy to follow?	☐	☐	☐	☐	☐
Examples — Were they helpful, realistic? Were there enough of them?	☐	☐	☐	☐	☐
Organization — Was it logical? Was it easy to find what you needed to know?	☐	☐	☐	☐	☐
Illustrations — Were they clear and useful?	☐	☐	☐	☐	☐
Physical Attractiveness — What did you think of the printing, binding, etc?	☐	☐	☐	☐	☐

What errors or faults did you find in the manual? (Please include page numbers) _____

Do you have any other comments or suggestions? _____

Name _____

Company _____

Street _____

City _____

State/Country _____

Zip Code _____ **Telephone** _____

Thank you for your help.

Printed in U.S.A. 14-3151 8-83-5C

WANG

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 16 LOWELL, MA

POSTAGE WILL BE PAID BY ADDRESSEE

WANG LABORATORIES, INC.
TECHNICAL PUBLICATIONS
ONE INDUSTRIAL AVENUE
LOWELL, MASSACHUSETTS 01851

Attention: Technical Writing Department

Cut along dotted line.

WANG LABORATORIES, INC.
51 Middlesex St.
No. Chelmsford, MA 01863

To Order by Phone Call
(800) 225-0234
From Mass., Hawaii, and Alaska Call
(617) 256-1400
TELEX 951-743

ORDER FORM FOR WANG MANUALS AND DOCUMENTATION

Customer Number _____

Bill To _____

_____

_____

Ship To _____

_____

_____

Customer Contact _____ Phone _____

Date _____ Purchase Order Number _____

Taxable Yes ☐ No ☐ Tax Exempt Number _____

Salesperson's Name _____ Employee No. _____ RDB No. _____

Document Number	Description	QTY	Unit Price	Total Price

	Subtotal	
Authorized Signature _____ Date _____	Less Applicable Discount	
	Subtotal	
☐ Check this box if you would like a free copy of the	Local Sales Tax	
WANG CUSTOMER RESOURCE CATALOG (700-7647)	**Total Amount**	

Wang Terms and Conditions

1. **TAXES** — Prices are exclusive of all sales, use, and like taxes.
2. **DELIVERY** — Delivery will be F.O.B. Wang's plant. Customer will be billed for freight charges; and unless customer specifies otherwise, all shipments will go best way surface as determined by Wang. Wang shall not assume any liability in connection with the shipment nor shall the carrier be construed to be an agent of Wang. If the customer requests that Wang arrange for insurance the customer will be billed for the insurance charges.
3. **PAYMENT** — Terms are net 30 days from date of invoice. Unless otherwise stated by customer, partial shipments will generate partial invoices.
4. **PRICES** — The prices shown are subject to change without notice. Individual document prices may be found in the *Wang Customer Resource Catalog* (700-7647).
5. **LIMITATION OF LIABILITY** — In no event shall Wang be liable for loss of data or for special, incidental or consequential damages in connection with or arising out the of use of or information contained in any manuals or documentation furnished hereunder.

Printed in U.S.A. 14-3165 12-82-2C

WANG

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 16 LOWELL, MA

POSTAGE WILL BE PAID BY ADDRESSEE

WANG LABORATORIES, INC.
Supplies Division
c/o Order Entry Dept.
M/S 1711
800 Chelmsford Street
Lowell, MA 01851

Cut along dotted line.

WANG

ONE INDUSTRIAL AVENUE
LOWELL, MASSACHUSETTS 01851
TEL. (617) 459-5000
TWX 710-343-6769, TELEX 94-7421

Printed in U.S.A.
800-6200-05
5-84-10M