60497300

CONTROL DATA CORPORATION

COBOL VERSION 5 INSTANT MANUAL

CDC® OPERATING SYSTEMS: NOS 1 NOS/BE 1

60497300

CONTROL DATA CORPORATION

COBOL VERSION 5 INSTANT MANUAL

CDC® OPERATING SYSTEMS: NOS 1 NOS/BE 1

REVISION RECORD

REVISION	DESCRIPTION
A (12/30/76)	Original Release.
B (02/06/81)	This revision reflects COBOL 5.3 (feature 1250) at PSR level 528. Changes include an interface to Advanced Access Methods 2.1, CYBER Database Control System 2.1 and Common Memory Manager (CMM).

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Front Cover	-
Title Page	-
ii	B
iii/iv	B
v	B
vi	B
vii	B
1	B
2	B
2.1/2.2	B
3	A
4	A
5 thru 56	B
57/58	B
59 thru 68	B
69 thru 74	A
Back Cover	-

PREFACE

This instant provides a convenient summary of the COBOL Version 5.3 language which operates under control of the following operating systems:

- NOS 1 for the CONTROL DATA® CYBER 170 Series; CYBER 70 Models 71, 72, 73, 74; and 6000 Series Computer Systems
- NOS/BE 1 for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, 74; and 6000 Series Computer Systems

COBOL 5 is designed to be a superset of the language specified in the American National Standard X3.23-1974, COBOL. Extensions to the standard language are indicated in this instant by shading.

This instant provides a brief description of the major COBOL language features. The instant is intended for programmers familiar with COBOL 5.

More detailed information can be found in the publications listed below.

Publication	Publication Number
COBOL Version 5 Reference Manual	60497100
COBOL Version 5 User's Guide	60497200
COBOL Version 5 Report Writer User's Guide	60496900

CDC manuals can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

SPECIAL FEATURES

In addition to supporting the full definition of 1974 ANS COBOL (X3.23-1974), the COBOL5 compiler supports the following additional features:

- Direct (Hashed), Actual Key, and Word Addressable files
- INITIALIZE statement
- Inter-program communication with other languages such as FORTRAN and COMPASS
- Dynamic paragraph trace facility which includes the current CPU utilization as each paragraph is entered
- Symbolic dump of the Data Division (through the Termination Dump facility) at user request and/or at program termination showing data-names together with their contents
- Interface to the CYBER Database Control System (CDCS) using the DDL sub-schema
- Structured programming support via language extensions derived from the draft for the next ANS standard for COBOL
- Interface to the Message Control System (MCS) as well as interactive input/output via ACCEPT/DISPLAY
- File name substitution at run time through the file equivalence parameter of the execution call statement
- Specification by programmer of portions of working storage to reside in Extended Core Storage (ECS)
- User selectable dynamic table bounds checking
- Access to part of a data item through use of reference modification

CONTENTS

PROGRAM EFFICIENCY HINTS

The following options improve compilation time performance:

- Use the SY parameter of the COBOL 5 control statement if only compilation is desired.
- Use the TAF parameter of the COBOL 5 control statement to prevent loading of unnecessary modules when the job is to be executed using TAF.
- Avoid using the DB parameters of the COBOL 5 control statement unless program debugging is desired.
- Avoid using the LBZ parameter of the COBOL 5 control statement; if some fields have leading blanks, use the INSPECT statement.
- Do not restrict field length through either the use of the CM parameter in the job statement or the use of an MFL statement (NOS only).
- Do not use RFL statements.

The following options improve execution time performance:

- Use same size, same sign convention, and same decimal point location for sending and receiving fields.
- Use index-names rather than subscripts.
- Use the SET statement to increment and decrement index-name values.
- Use the SYNCHRONIZED RIGHT clause for numeric data frequently referenced.
- Use the SAME RECORD AREA clause to save moves.
- Use the VALUE clause whenever possible to initialize WORKING-STORAGE instead of a MOVE statement.
- Use a binary table search if the data items in the table are ordered sequentially and the table contains more than eight entries. Use a serial search if the table contains less than nine entries.
- Make alphanumeric table and item sizes a multiple of 10 characters.
- Align tables and items on word boundaries through the use of the SYNCHRONIZED clause, level 77 items, or automatic level 01 alignment.

- Construct overlays (sections greater than 49) in such a manner that the overlays are executed only once.
- Give careful consideration to any decision to utilize the internal COBOL SORT.
- Represent subscripts and counters in binary (COMP-1).
- Place the most likely condition first for OR in a compound IF statement. Place the least likely condition first for AND in a compound IF statement.
- Restrict arithmetic items other than COMPUTATIONAL-1 or COMPUTATIONAL-4 to 9 digits or less.
- Do not manipulate large table entries in their table locations; move the matching argument to a work area.
- Avoid the use of unblocked data files.
- Avoid the use of multi-level subscripting.
- Avoid character comparison with items of unequal size.
- Avoid all on SIZE ERROR clauses on any arithmetic operation.
- Avoid passing parameters when calling another program; use the Common-Storage Section for shared data.

NOTATION

[ ]	Enclosed elements are optional.
{ }	Only one element must be selected.
[ ] ... or { } ...	Repeat enclosed elements as needed.

COBOL reserved words have preassigned meanings and appear in capitals.

COBOL reserved words that are underlined are required; words not underlined can be omitted.

Terms in lowercase letters represent words or symbols supplied by the programmer.

Commas and semicolons are used optionally to improve readability; periods are required where shown.

At least one space must follow all punctuation symbols.

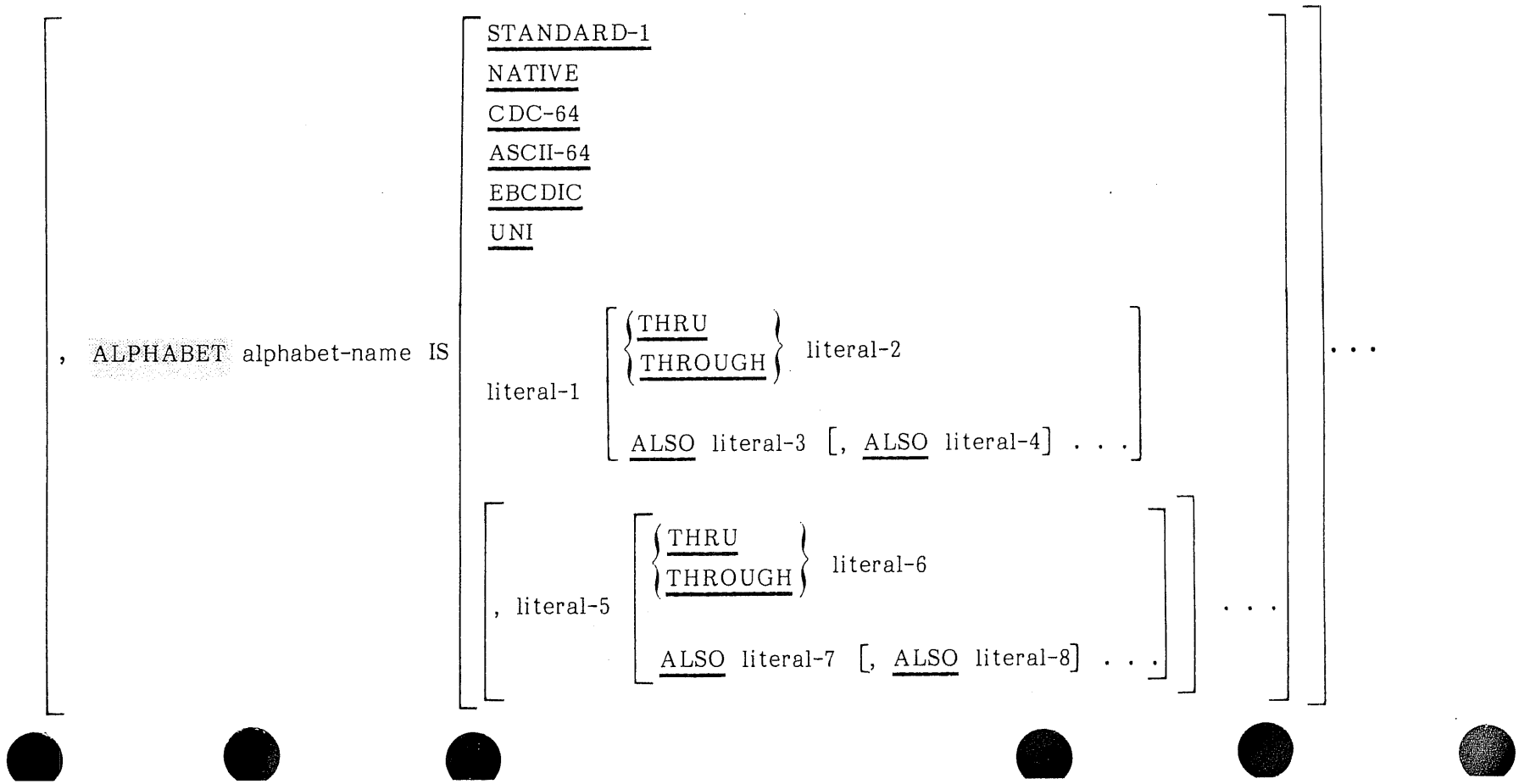
ENVIRONMENT DIVISION

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.†
SOURCE-COMPUTER.  [ computer-name
                  [, WITH DEBUGGING MODE] ].

OBJECT-COMPUTER.  [ computer-name
                  [, PROGRAM COLLATING SEQUENCE IS alphabet-name]
                  [, SEGMENT-LIMIT IS segment-number] ].
```

† The entire CONFIGURATION SECTION is optional.

SPECIAL-NAMES.

[, implementor-name IS mnemonic-name]...



[, CURRENCY SIGN IS literal]

[, DECIMAL-POINT IS COMMA]

[, { QUOTE IS / QUOTES ARE } APOSTROPHE]

[, SIGN CONTROL IS { LEADING / TRAILING } [SEPARATE CHARACTER]]

[, SUB-SCHEMA IS sub-schema-name]

[, SWITCH-n [IS mnemonic-name] [[ON STATUS IS condition-name-1] [, OFF STATUS IS condition-name-2] / [OFF STATUS IS condition-name-2] [, ON STATUS IS condition-name-1]]]...

INPUT-OUTPUT SECTION.

File-Control Entry

Format 1 (Sequential File Organization)

FILE-CONTROL.

```
SELECT [OPTIONAL] file-name
  ASSIGN TO implementor-name-1 [, implementor-name-2] . . .
[; ORGANIZATION IS SEQUENTIAL]
[; ACCESS MODE IS SEQUENTIAL]
[; FILE STATUS IS data-name]
[; RESERVE integer [AREA ]]
                   [AREAS]
[; USE literal] .
```

Format 2 (Relative File Organization)

```
FILE-CONTROL.

    SELECT file-name
     ASSIGN TO implementor-name-1 [, implementor-name-2] . . .
     ; ORGANIZATION IS RELATIVE
     [ ; ACCESS MODE IS { SEQUENTIAL [, RELATIVE KEY IS data-name] }
                        { {RANDOM  }                               }
                        { {DYNAMIC }   , RELATIVE KEY IS data-name }  ]
     [; FILE STATUS IS data-name]
     [; RESERVE integer [AREA ]]
                        [AREAS]
     [; USE literal].
```

Format 3 (Indexed File Organization, Direct File Organization, Actual Key File Organization)

```
FILE-CONTROL.
    SELECT file-name
        ASSIGN TO implementor-name-1 [,implementor-name-2]...
        ;ORGANIZATION IS { INDEXED    }
                         { DIRECT     }
                         { ACTUAL-KEY }
        :RECORD KEY IS data-name
        [;ACCESS MODE IS { SEQUENTIAL }]
                         { RANDOM     }
                         { DYNAMIC    }
        [;ALTERNATE RECORD KEY IS data-name-1 [WITH DUPLICATES ASCENDING]
            [{ OMITTED } WHEN data-name-2 CONTAINS CHARACTER FROM literal]
             { USE     }
            [OMITTED WHEN KEY IS { SPACES }]
                                 { ZEROS  }
        [;FILE STATUS IS data-name]
        [;USE literal .]
```

Format 4 (Word-Address File Organization)

```
FILE-CONTROL.
    SELECT file-name
        ASSIGN TO implementor-name-1 [, implementor-name-2] . . .
      ; ORGANIZATION IS WORD-ADDRESS
      ; WORD-ADDRESS KEY IS data-name
      [ ; ACCESS MODE IS { SEQUENTIAL } ]
                         { RANDOM     }
                         { DYNAMIC    }
      [; FILE STATUS IS data-name]
      [ ; RESERVE integer [AREA ] ]
                          [AREAS]
      [; USE literal].
```

I-O-CONTROL.

```
[; APPLY input-output-technique ON file-name-1 [, file-name-2] . . .]

[; MULTIPLE FILE TAPE CONTAINS {file-name-1       } [POSITION integer-1]
                               {pseudo-file-name-1}
    [, {file-name-2       } [POSITION integer-2]] . . .                ]...
       {pseudo-file-name-2}

[; SAME [{RECORD   }] AREA FOR file-name-1 [, file-name-2] . . .]...
        {SORT     }
        {SORT-MERGE}

[; RERUN [ON {file-name-1       }] EVERY {{[END OF] {REEL}}       OF file-name-2}...
             {implementor-name  }        {{      {UNIT}}                        }
                                         {integer-1 RECORDS                     }
                                         {condition-name                        }
```

DATA DIVISION

```
DATA DIVISION.
[ FILE SECTION.]
[ COMMON-STORAGE SECTION.]
[ WORKING-STORAGE SECTION.]
[ SECONDARY-STORAGE SECTION.]
[ LINKAGE SECTION.]
[ COMMUNICATIONS SECTION.]
[ REPORT SECTION.]
```

File Description Entry (File Section Only)

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

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

[; EXTERNAL]

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

    [; VALUE OF implementor-name-1 IS {data-name-1} [, implementor-name-2 IS {data-name-2}] . . .]
                                      {literal-1  }                         {literal-2  }

[; LINAGE IS {data-name-1} LINES [, WITH FOOTING AT {data-name-2}]
             {integer-1  }                          {integer-2  }

   [, LINES AT TOP {data-name-3}] [, LINES AT BOTTOM {data-name-4}]]
                   {integer-3  }                     {integer-4  }
```

```
[ { ; RECORD CONTAINS [integer-1 TO] integer-2 CHARACTERS [DEPENDING ON data-name ] }
  { RECORD IS VARYING IN SIZE [ [FROM integer-1]                                    }
  { [TO integer-2] CHARACTERS ] [DEPENDING ON data-name]                            } ]

[ ; RECORDING MODE IS { DECIMAL } ]
                      { BINARY  }

[ ; { REPORT IS  } report-name-1 [, report-name-2]. . . ] .
    { REPORTS ARE }

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

Sort-Merge File Description Entry (File Section Only)

SD file-name

```
[ ; RECORD { CONTAINS [integer-1 TO] integer-2 CHARACTERS                  }
           { IS VARYING IN SIZE [ [FROM integer-3 ]                        }
           { [TO integer-4] CHARACTERS ]                                   }
           { [DEPENDING ON data-name-1]. . .                               } ]
```

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

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

Communication Description Entry (Communication Section Under NOS Only)

Format 1

```
CD cd-name; FOR INITIAL INPUT
     [[; SYMBOLIC QUEUE IS data-name-1 ]
          [;SYMBOLIC SUB-QUEUE-1          IS    data-name-2 ]
          [;SYMBOLIC SUB-QUEUE-2          IS    data-name-3 ]
          [;SYMBOLIC SUB-QUEUE-3          IS    data-name-4 ]
          [;MESSAGE DATE                  IS    data-name-5 ]
          [;MESSAGE TIME                  IS    data-name-6 ]
          [;SYMBOLIC SOURCE               IS    data-name-7 ]
          [;TEXT LENGTH                   IS    data-name-8 ]
          [;END KEY                       IS    data-name-9 ]
          [;STATUS KEY                    IS    data-name-10 ]
          [;MESSAGE COUNT                 IS    data-name-11 ]]
     [ data-name-1, data-name-2,.....data-name-11 ] ].
```

Format 2

```
CD  cd-name; FOR OUTPUT
    [; DESTINATION COUNT          IS      data-name-1]
    [; TEXT LENGTH                IS      data-name-2]
    [; STATUS KEY                 IS      data-name-3]
        [; DESTINATION TABLE OCCURS  integer-2  TIMES
          [; INDEXED  BY  index-name-1  [,index-name-2]  ...]]
    [; ERROR KEY                  IS      data-name-4 ]
    [; SYMBOLIC DESTINATION       IS      data-name-5 ]
```

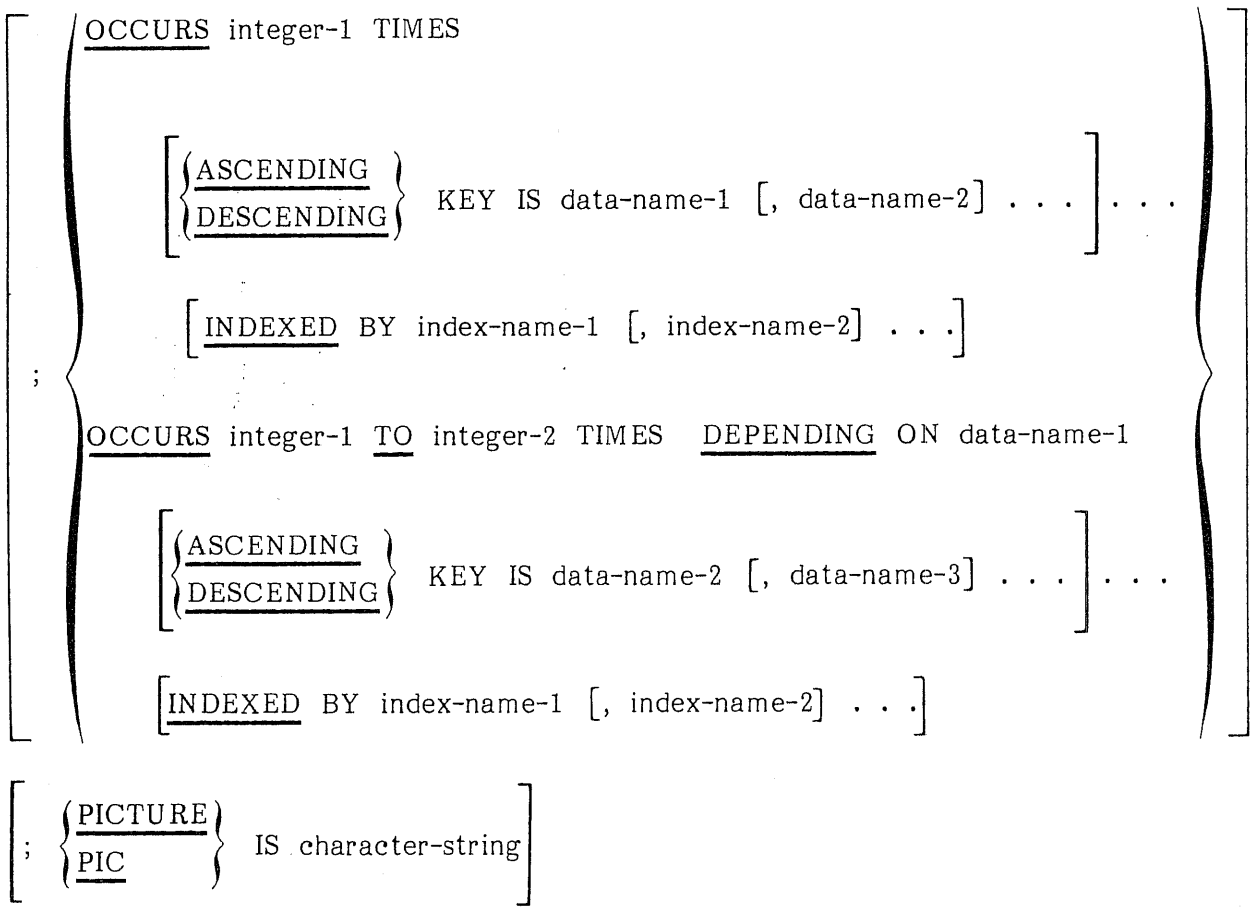
Report Description Entry (Report Section Only)

```
RD Report-name

      [; CODE literal ]
      [; { CONTROL IS  } { data-name-1 [, data-name-2 ...]                    } ]
         { CONTROLS ARE} { FINAL [, data-name-1 [, data-name-2]...]           }
      [; PAGE [LIMIT IS ] integer-1 [LINE ] [,HEADING integer-2 ]
              [LIMITS ARE]          [LINES]
         [, FIRST DETAIL integer-3 ] [, LAST DETAIL integer-4]
         [, FOOTING integer-5 ]                                               ]
      { report-group-description entry } ...
```

Data Description Entry (File, Common-Storage, Working-Storage, Secondary-Storage, Linkage, and Communications Sections)

Format 1

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

    [; BLANK WHEN ZERO]

    [; { JUSTIFIED } RIGHT]
       { JUST      }
```


OCCURS integer-1 TIMES
ASCENDING
DESCENDING
KEY IS data-name-1 [, data-name-2] . . .
INDEXED BY index-name-1 [, index-name-2] . . .
OCCURS integer-1 TO integer-2 TIMES DEPENDING ON data-name-1
ASCENDING
DESCENDING
KEY IS data-name-2 [, data-name-3] . . .
INDEXED BY index-name-1 [, index-name-2] . . .
PICTURE
PIC
IS character-string

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

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

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

[; VALUE IS literal] .
```

Format 2

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

Format 3

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

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

Report Description Entry (Report Section Only)

```
RD report-name

   [; CODE literal]

   [ ; {CONTROL IS  } { data-name-1 [, data-name-2] . . .                          } ]
       {CONTROLS ARE} { FINAL [, data-name-1 [, data-name-2] . . . ]               }
```

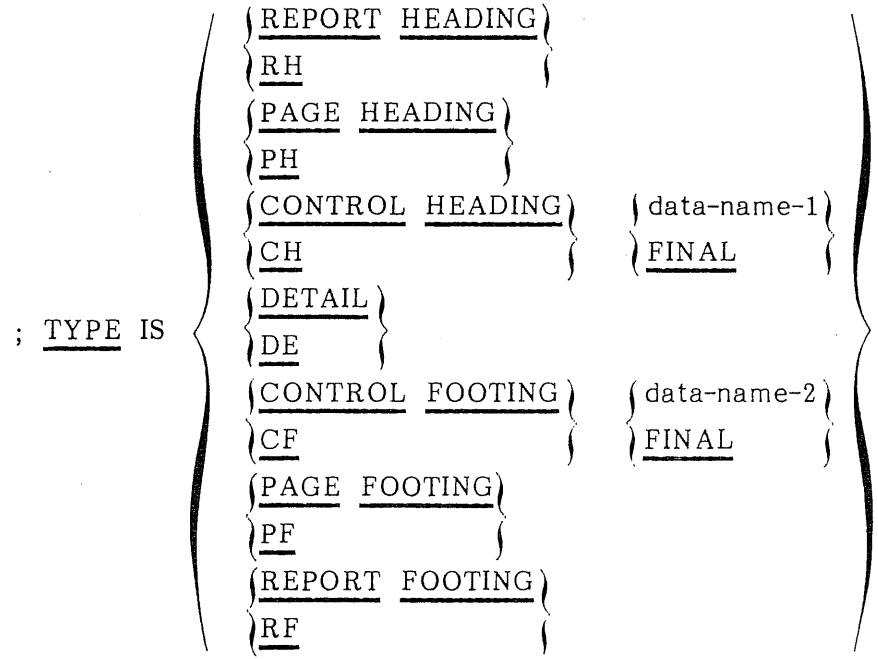
```
[; PAGE [LIMIT IS / LIMITS ARE] integer-1 [LINE / LINES] [, HEADING integer-2]
    [, FIRST DETAIL integer-3] [, LAST DETAIL integer-4]
    [, FOOTING integer-5]]
```

```
{report-group-description entry} . . .
```

Report Group Description Entry (Report Section Only)

Format 1

```
01 [data-name-1]
    [; LINE NUMBER IS {integer-1 [ON NEXT PAGE] / PLUS integer-2}]
    [; NEXT GROUP IS {integer-1 / PLUS integer-2 / NEXT PAGE}]
```



[; [USAGE IS] DISPLAY].

Format 2

```
level-number [data-name]

[ ; LINE NUMBER IS { integer-1 [ON NEXT PAGE] } ]
                   { PLUS integer-2            }

[; [USAGE IS] DISPLAY].
```

Format 3

```
level-number [data-name]

  [; BLANK WHEN ZERO]

  [; COLUMN NUMBER IS integer]

  [; GROUP INDICATE]

  [ ; { JUSTIFIED } RIGHT ]
      { JUST      }
```

```
[ ; LINE NUMBER IS { integer-1 [ON NEXT PAGE] } ]
                   { PLUS integer-2           }

; { PICTURE } IS character-string
  { PIC     }

{ ; SOURCE IS identifier                                                                    }
{ ; VALUE IS literal                                                                        }
{ ; SUM identifier-1 [, identifier-2] . . . [UPON data-name-1 [, data-name-2] . . . ] . . . }
{   [RESET ON { data-name-3 }]                                                              }
{             { FINAL       }                                                             }

[ ; [USAGE IS] DISPLAY] .
```

PROCEDURE DIVISION

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

[DECLARATIVES.

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

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

END DECLARATIVES.]

{section-name SECTION [segment-number].

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

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

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

```
ACCEPT identifier [FROM mnemonic-name]

ACCEPT identifier FROM { DATE        }
                       { DAY         }
                       { DAY-OF-WEEK }
                       { TIME        }

ACCEPT cd-name MESSAGE COUNT

ADD { literal-1    } [, { literal-2    }] . . . TO identifier-m [ROUNDED] [, identifier-n [ROUNDED]] . . .
    { identifier-1 }    { identifier-2 }

    [; ON SIZE ERROR imperative-statement]

ADD { literal-1    } { , literal-2    } [, literal-3    ] . . . GIVING identifier-m [ROUNDED]
    { identifier-1 } { , identifier-2 } [, identifier-3 ]

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

ADD { CORRESPONDING } identifier-1 TO identifier-2 [ROUNDED] [, identifier-3 [ROUNDED] . . .]
    { CORR          }

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

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

[, procedure-name-3 TO [PROCEED TO] procedure-name-4] ...

CALL {identifier / literal} [USING data-name-1 [,data-name-2] ...] [; ON OVERFLOW imperative-statement]

CANCEL {identifier-1 / literal-1} [, identifier-2 / , literal-2] ...

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

CLOSE relation-name [WITH LOCK] ...

COMPUTE identifier-1 [ROUNDED] [, identifier-2 [ROUNDED]] ...

{FROM / = / EQUALS} arithmetic-expression [; ON SIZE ERROR imperative-statement]

```
COMPUTE {identifier-3}...
        {FROM    }
        {=       } boolean expression
        {EQUALS  }

CONTINUE

COPY text-name [{OF} library-name]
               {IN}

    [REPLACING {, {== pseudo-text-1 ==} BY {== pseudo-text-2 ==}}...].
               {  {identifier-1       }    {identifier-2       }
               {  {literal-1          }    {literal-2          }
               {  {word-1             }    {word-2             }

DELETE {file-name RECORD [; INVALID KEY imperative statement]}
       {FILE {file-name}...                                  }

DISABLE {INPUT  } [TERMINAL]} cd-name WITH KEY {identifier-1}
        {OUTPUT }                              {literal-1   }

DISPLAY {literal-1   } [, literal-2   ]... [UPON mnemonic-name] [WITH NO ADVANCING]
        {identifier-1}  [, identifier-2]

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

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

```
DIVIDE {identifier-1} INTO {identifier-2} GIVING identifier-3 [ROUNDED]
       {literal-1   }      {literal-2   }
    [, identifier-4 [ROUNDED]] . . . [; ON SIZE ERROR imperative-statement]

DIVIDE {identifier-1} BY {identifier-2} GIVING identifier-3 [ROUNDED]
       {literal-1   }    {literal-2   }
    [, identifier-4 [ROUNDED]] . . . [; ON SIZE ERROR imperative-statement]

DIVIDE {identifier-1} INTO {identifier-2} GIVING identifier-3 [ROUNDED]
       {literal-1   }      {literal-2   }
    REMAINDER identifier-4 [; ON SIZE ERROR imperative-statement]

DIVIDE {identifier-1} BY {identifier-2} GIVING identifier-3 [ROUNDED]
       {literal-1   }    {literal-2   }
    REMAINDER identifier-4 [; ON SIZE ERROR imperative-statement]

ENABLE {INPUT          [TERMINAL]} cd-name WITH KEY {identifier-1}
       {OUTPUT                   }                 {literal-1   }

                  [ COMPASS  ]
ENTER             [ FORTRAN-X ] routine-name
                  [ FTN5     ]

[ USING {data-name-1     } [, data-name-2     ]
        {file-name-1     }  [, file-name-2     ]
        {procedure-name-1}  [, procedure-name-2] ...
        {literal-1       }  [, literal-2      ]  ]

EXIT [PROGRAM].
```

```
GENERATE { data-name   }
         { report-name }

GO TO [procedure-name-1]

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

IF condition; [THEN] { statement-1   } { ; ELSE statement-2 . . . [; END-IF] }
                     { NEXT SENTENCE } { ; ELSE NEXT SENTENCE                 }
                                       { ; END-IF                             }
```

Conditional expressions include:

```
{identifier-1                }   { IS [NOT] GREATER THAN  }   {identifier-2                }
{literal-1                   }   { IS [NOT] ≥             }   {literal-2                   }
{arithmetic-expression-1     }   { IS [NOT] LESS THAN     }   {arithmetic-expression-2     }
                                 { IS [NOT] ≤             }
                                 { IS [NOT] EQUAL TO      }
                                 { IS [NOT] =             }
                                 { IS UNEQUAL TO          }
                                 { EQUALS                 }
                                 { EXCEEDS                }
```

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

identifier IS [NOT] { NUMERIC      }
                    { ALPHABETIC   }

                    { IS [NOT] EQUAL TO }
boolean expression-1{ IS [NOT] =        } boolean expression-2
                    { IS UNEQUAL TO     }
                    { EQUALS            }

condition-name

INITIALIZE identifier-1 [, identifier-2] . . .

    [ REPLACING { ALPHABETIC          } DATA BY {identifier-3}
                { ALPHANUMERIC        }         {literal     }
                { NUMERIC             }
                { ALPHANUMERIC-EDITED }
                { NUMERIC-EDITED      } ]

INITIATE report-name-1 [, report-name-2] . . .
```

INSPECT identifier-1 TALLYING

```
{ , identifier-2 FOR { , { { ALL       }      { literal-1    }  [ { BEFORE } INITIAL { literal-2    } ] } ... } ...
                       { { LEADING     }      { identifier-3 }    { AFTER  }         { identifier-4 }   }
                       { { CHARACTERS  }                                                             }
```

INSPECT identifier-1 REPLACING

```
{ CHARACTERS BY { literal-4    }  [ { BEFORE } INITIAL { literal-5    } ]                                                                                  }
{               { identifier-6 }    { AFTER  }         { identifier-7 }   }                                                                                  }
{ { , { ALL     } { , { literal-3    } BY { literal-4    }  [ { BEFORE } INITIAL { literal-5    } ] } ... } ...                                          }
{ { , { LEADING } { , { identifier-5 }    { identifier-6 }    { AFTER  }         { identifier-7 }   } } ... } ...                                          }
{ { , { FIRST   } }                                                                                                                                    }
```

```
INSPECT identifier-1 TALLYING

{ , identifier-2 FOR { , { {ALL       } {literal-1   } } [ {BEFORE} INITIAL {literal-2   } ] } ... } ...
                     {   {LEADING   } {identifier-3} }   { {AFTER } }       {identifier-4} ]
                     {   CHARACTERS }                 }

[BEFORE] REPLACING
[AFTER ]

{ CHARACTERS BY {literal-4   } [ {BEFORE} INITIAL {literal-5   } ]                                                                  }
{               {identifier-6} [ {AFTER } }       {identifier-7} ]                                                                 }
{ { , {ALL      } } { , {literal-3   } BY {literal-4   } [ {BEFORE} INITIAL {literal-5   } ] } ... } ...                           }
{     {LEADING  } }     {identifier-5}    {identifier-6}   { {AFTER } }       {identifier-7} ]   }     }     }
{     {FIRST    } }                                        }                                  }     }     }
```

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

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

     [COLLATING SEQUENCE IS alphabet-name]

     USING file-name-2, file-name-3 [, file-name-4] . . .

     { OUTPUT PROCEDURE IS section-name-1 [ { THRU      } section-name-2 ] }
     {                                     { THROUGH   }                  }
     { GIVING file-name-5                                                 }

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

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

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

    [; ON SIZE ERROR imperative-statement]

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

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

     { INPUT file-name-1 [REVERSED      ] [, file-name-2 [REVERSED      ]] . . . }
     {                   [WITH NO REWIND]                [WITH NO REWIND]        }
     {                                                                           }
OPEN { OUTPUT file-name-3 [WITH NO REWIND] [, file-name-4 [WITH NO REWIND]] . . .} . . .
     {                                                                           }
     { I-O file-name-5 [, file-name-6] . . .                                     }
     {                                                                           }
     { EXTEND file-name-7 [, file-name-8] . . .                                  }

OPEN {INPUT relation-name [WITH NO REWIND] . . .} . . .
     {I-O relation-name                          }
```

```
PERFORM [procedure-name-1 [{THRU | THROUGH} procedure-name-2]] [; imperative-statement; END-PERFORM]

PERFORM [procedure-name-1 [{THRU | THROUGH} procedure-name-2]] {identifier-1 | integer-1} TIMES
    [; imperative-statement; END-PERFORM]

PERFORM [procedure-name-1 [{THRU | THROUGH} procedure-name-2]] [; WITH TEST {BEFORE | AFTER}]
    UNTIL condition-1 [imperative-statement; END-PERFORM]

PERFORM [procedure-name-1 [{THRU | THROUGH} procedure-name-2]] [; WITH TEST {BEFORE | AFTER}]
    VARYING {identifier-1 | index-name-1} FROM {identifier-2 | index-name-2 | literal-1}
    BY {identifier-3 | literal-2} UNTIL condition-1
    [AFTER {identifier-4 | index-name-3} FROM {identifier-5 | index-name-4 | literal-3}
    BY {identifier-6 | literal-4} UNTIL condition-2] ...
    [imperative-statement; END-PERFORM]
```

PURGE cd-name

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

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

READ relation-name [NEXT] RECORD [; AT END imperative-statement]

READ relation-name RECORD [; KEY IS data-name] [; INVALID KEY imperative-statement]

RECEIVE cd-name {MESSAGE / SEGMENT} INTO identifier-1 [; NO DATA imperative statement]

REPLACE {, == pseudo-text-1==BY=pseudo-text-2== } ...

REPLACE OFF

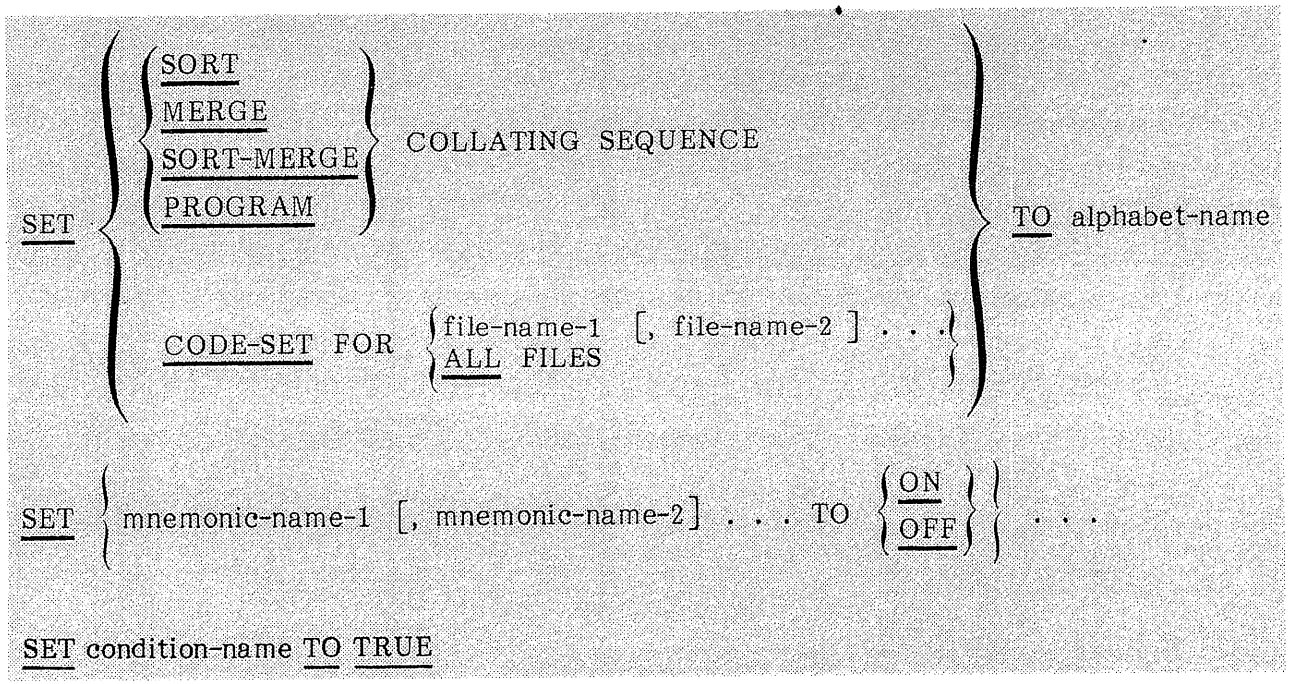
RELEASE record-name [FROM identifier]

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

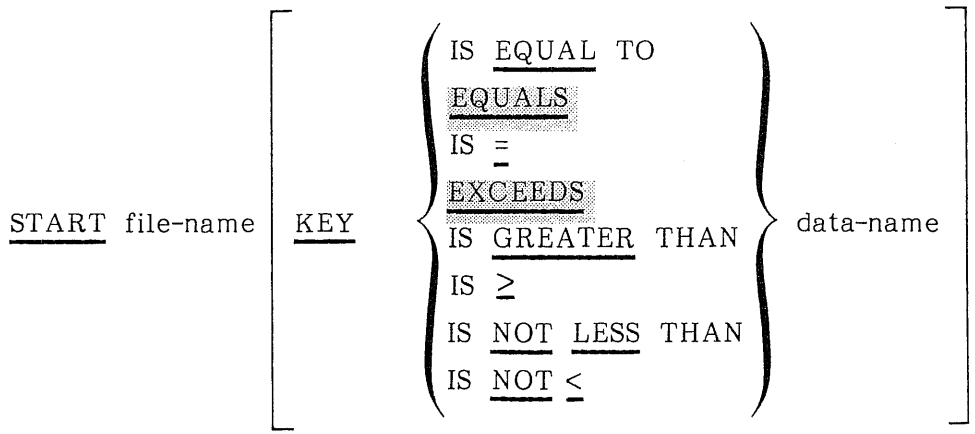
REWRITE record-name [FROM identifier] [; INVALID KEY imperative-statement]

```
SEARCH identifier-1 [VARYING {index-name-1 }]  [; AT END imperative-statement-1]
                             {identifier-2 }

    ; WHEN condition-1 {imperative-statement-2}  [; WHEN condition-2 {imperative-statement-3}] . . .
                       {NEXT SENTENCE        }                      {NEXT SENTENCE        }

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

    ; WHEN { data-name-1     { EQUALS       } { identifier-3     } }
         {                  { IS EQUAL TO  } { literal-2        } }
         { condition-name-1 { IS =         } { arithmetic-expression-1 } }

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

  {imperative-statement-2} [; END-SEARCH]
  {NEXT SENTENCE        }
```

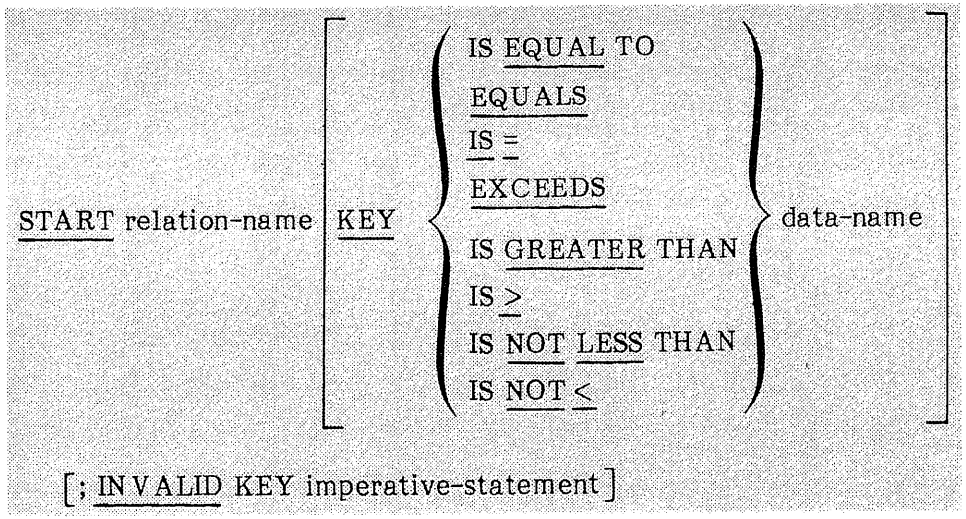
```
SEND cd-name FROM identifier-1

SEND cd-name  FROM identifier-1  { WITH identifier-2 }
                                 { WITH ESI        }
                                 { WITH EMI        }
                                 { WITH EGI        }

[ { BEFORE } ADVANCING { { { identifier-3 }          [ LINE ] } }
  { AFTER  }           { { { integer      }          [ LINES ] } }
                       { { { mnemonic-name } }                  }
                       { { { PAGE         } }                  } ]

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

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


SET
SORT
MERGE
SORT-MERGE
PROGRAM
COLLATING SEQUENCE
CODE-SET FOR
file-name-1 [, file-name-2] . . .
ALL FILES
TO alphabet-name
SET
mnemonic-name-1 [, mnemonic-name-2] . . . TO
ON
OFF
. . .
SET condition-name TO TRUE

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

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

[WITH DUPLICATES IN ORDER]

[COLLATING SEQUENCE IS alphabet-name]

{ INPUT PROCEDURE IS section-name-1 [ { THRU    } section-name-2 ] }
{                                     { THROUGH }                  }
{ USING file-name-2 [, file-name-3] . . .                          }

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



[; INVALID KEY imperative-statement]



STOP { RUN / literal }

```
STRING {identifier-1} [, identifier-2] . . . DELIMITED BY {identifier-3}
       {literal-1   }  [, literal-2   ]                   {literal-3   }
                                                          {SIZE        }

    [, {identifier-4} [, identifier-5] . . . DELIMITED BY {identifier-6}] . . .
       {literal-4   }  [, literal-5   ]                   {literal-6   }
                                                          {SIZE        }

    INTO identifier-7 [WITH POINTER identifier-8] [; ON OVERFLOW imperative-statement]

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

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

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

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

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

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

[, identifier-3 [ROUNDED]] . . . [; ON SIZE ERROR imperative-statement]

SUPPRESS PRINTING

TERMINATE report-name-1 [, report-name-2] . . .

UNSTRING identifier-1

[DELIMITED BY [ALL] { identifier-2 / literal-1 } [, OR [ALL] { identifier-3 / literal-2 }] . . .]

INTO identifier-4 [, DELIMITER IN identifier-5] [, COUNT IN identifier-6]

[, identifier-7 [, DELIMITER IN identifier-8] [, COUNT IN identifier-9]] . . .

[WITH POINTER identifier-10] [TALLYING IN identifier-11]

[; ON OVERFLOW imperative-statement]

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

```
USE BEFORE REPORTING identifier.
```

```
USE FOR DEBUGGING ON

{ [ALL REFERENCES OF] identifier-1 }  [  [ALL REFERENCES OF] identifier-2 ]
{ procedure-name-1                 }  [  procedure-name-2                 ]
{ file-name-1                      }  [, file-name-2                      ]  . . .
{ cd-name-1                        }  [  cd-name-2                        ]
{ ALL PROCEDURES                   }  [  ALL PROCEDURES                   ]
```

```
USE FOR HASHING ON file-name-1 [, file-name-2] . . .
```

```
USE FOR ACCESS CONTROL [ ON { INPUT      } ]
                           { I-O        }
                           { INPUT I-O  }
                           { I-O INPUT  }

    KEY IS data-name [ FOR { realm-name-1 [, realm-name-2]... } ]
                           { REALMS                             }

USE FOR DEADLOCK ON { realm-name-1 [, realm-name-2]... }
                    { REALMS                            }
```

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

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

    [ ; AT { END-OF-PAGE } imperative-statement ]
           { EOP        }
```

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

COBOL5 CONTROL STATEMENT

The COBOL5 control statement consists of the word COBOL5 optionally followed by a parameter list used to specify compilation selections. Parameters can be specified in any order. A comma is the only valid parameter separator. The complete control statement is terminated by either a period or a right parenthesis. Default parameter values might be changed by individual installations.

COBOL5.
COBOL5(parameter list) [comments]

- ANSI (ANSI Extension Diagnosis)

Omitted	Non-ANSI extensions allowed
ANSI ANSI=T	Non-ANSI extensions diagnosed as trivial errors
ANSI=F	Non-ANSI extensions diagnosed as fatal errors
ANSI=NOEDIT	Numeric display items are not edited by the DISPLAY statement
ANSI=77LEFT	Level 77 items are stored SYNC LEFT
ANSI=AUDIT	Equivalent to selecting both ANSI=NOEDIT and ANSI=77LEFT. Non-ANSI reserved words are not recognized as reserved words

- APO (Apostrophe Character)

Omitted	Nonnumeric literals delimited by quotation mark character
APO	Nonnumeric literals delimited by apostrophe character

- B (Binary Output)

Omitted	Binary output on file LGO
B	Binary output on file BIN
B=0	No binary output produced
B=lfn	Binary output on file lfn

- BL (Burstable Listing)

Omitted	Triple space separates listing sections
BL	Page eject occurs between listing sections

- CC1 (COMP Equate to COMP-1)

Omitted	COMP data items stored and processed as COMP items
CC1	COMP data items stored and processed as COMP-1 items

- D (Database Sub-Schema File Identification)

Omitted D=0	No SUB-SCHEMA clause allowed in source program
D	Sub-schema for CDCS interface on file with same name as sub-schema
D=lfn	Sub-schema for CDCS interface on file lfn

- DB (Debugging Selection)

Omitted DB=0	No DB parameter options selection
DB=B	Executable code produced regardless of all errors in source program
DB=DL	Debugging lines compiled as executable code
DB=RF	Reference modification values are checked during execution to ensure that values are within bounds
DB=SB	Subscript and index references checked during execution for out-of-bounds references
DB=TR	Program execution flow traced
DB	Equivalent to DB=DL/SB/B

Slashes are used to separate multiple options selected for the DB parameter.

- E (Error File Name)

Omitted E=0	Error information written on file OUTPUT
E	Error information written to file ERR
E=lfn	Error information written on file lfn

- EL (Error Level Reported)

Omitted EL=W	W, F and C level errors listed
EL EL=F	F and C level errors listed
EL=T	T, W, F, and C level errors listed
EL=C	C level errors listed

- ET (Error Termination)

Omitted	Next control statement executed after program termination
ET=F	Compiler aborted by F or C level errors
ET=T	Compiler aborted by T, W, F, or C level errors
ET=W	Compiler aborted by W, F, or C level errors
ET=C	Compiler aborted by C level errors

- FDL (Fast Dynamic Loader Processing)

Omitted	All subprograms must be resident at the same time. CALL statement must specify a literal with first 7 characters unique in run unit. CDCS sub-schema cannot be used by subprograms
FDL	Equivalent to FDL=FDLFILE
FDL=lfn	Literal, identifier, or program name longer than 7 characters allowed in CALL statement. CDCS sub-schema can be used in subprograms. FDL file on file lfn.

- FIPS

Omitted	No FIPS diagnostics issued
FIPS	Equivalent to FIPS=4
FIPS=n	Language features above the specified FIPS level are diagnosed; n specifies level 1, 2, 3, or 4

  The parameters ANSI and EL=T must be specified to obtain a listing of FIPS diagnostics.

- I (Input File Name)

Omitted	Source program on file INPUT
I	Source program on file COMPILE
I=lfn	Source program on file lfn

- L (Listing File Name)

Omitted	Source listing and selected listings on file OUTPUT
L	Source listing and selected listings on file LIST
L=0	No listing produced
L=lfn	Source listing and selected listings on file lfn

- LBZ (Leading Blank Zero)

Omitted	Numeric fields with leading blanks treated as errors
LBZ	Leading blanks in numeric fields treated as zeros

- LO (Listing Options)

Omitted LO=S	Source program listed
LO=-S	Source program not listed
LO=M	Data map listed
LO=O	Object code and COMPASS mnemonics listed

LO=R	Cross reference map listed
LO=0	No listing produced
LO	Equivalent to LO=S/M/R

Slashes are used to separate multiple options selected for the LO parameter.

- MSB (Main Subroutine Indicator)

Omitted	Source program compiled normally
MSB	Source program compiled as subroutine with COBOL initiation

The MSB parameter should be used only when the COBOL program is called by a program written in a language other than COBOL.

- PD (Print Density)

Omitted PD=6	Listings specified by E and L parameters single spaced at 6 lines per inch
PD PD=8	Listings specified by E and L parameters single spaced at 8 lines per inch
PD=3	Listings specified by E and L parameters double spaced at 6 lines per inch
PD=4	Listings specified by E and L parameters double spaced at 8 lines per inch

- PS (Page Size)

Omitted	Number of lines on output page calculated by system
PS=n	Number of lines on output page indicated by n

- PSQ (Program Sequence)

Omitted	Compiler-generated sequence numbers used for all diagnostics

PSQ — Sequence numbers in columns 1 through 6 of each line used for all diagnostics

- PW (Page Width)

  Omitted — Output lines 136 characters in length

  PW — Output lines 72 character in length

  PW=n — Output lines n characters in length, 136 maximum

- SB (Subcompile Indicator)

  Omitted — Program compiled as main program

  SB — Program compiled as subprogram

- SY (Syntax Check)

  Omitted — Source program compiled and executable code generated

  SY — Source program checked for correct syntax; no executable code generated

- TAF (TAF Program)

  Omitted — Program runs in non-TAF environment

  TAF — Program runs as NOS TAF task

- TDF (Termination Dump Indicator)

  Omitted — No termination dump

  TDF — Termination dump is written to file TDFILE

  TDF=lfn — Termination dump is written to file lfn

- U (Update File Name)

  Omitted
  U=0 — No update file created

  U — COMPASS line images written on file COMPS

  U=lfn — COMPASS line images written on file lfn

- UCI (Unpack COMP-1 Items)

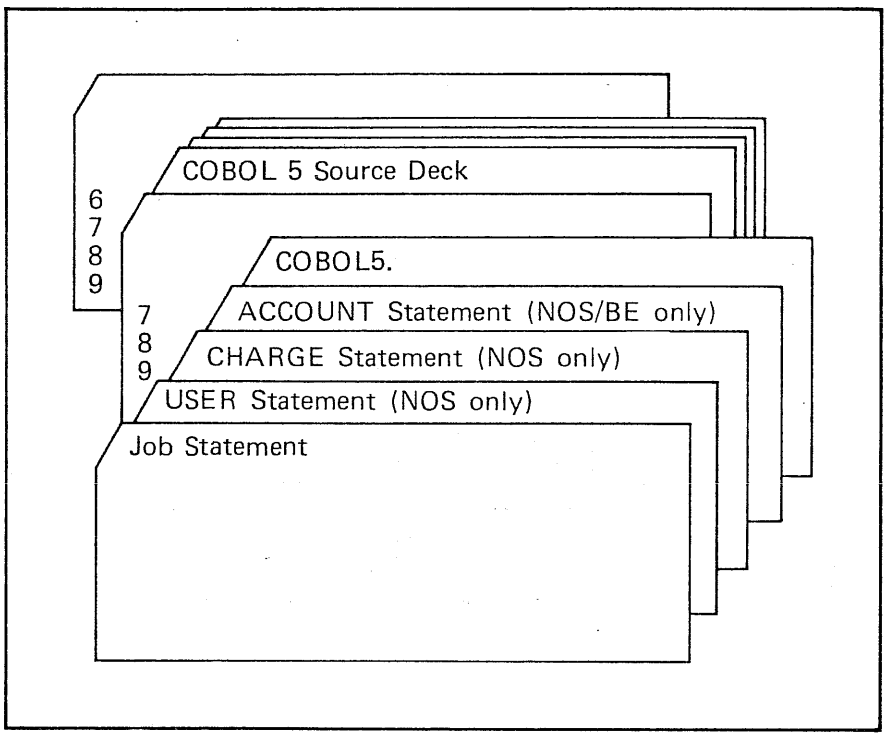
Omitted	COMP-1 items processed in COMP-1 format
UCI	COMP-1 items converted to integer format before processing

- X (Copy Text File Name)

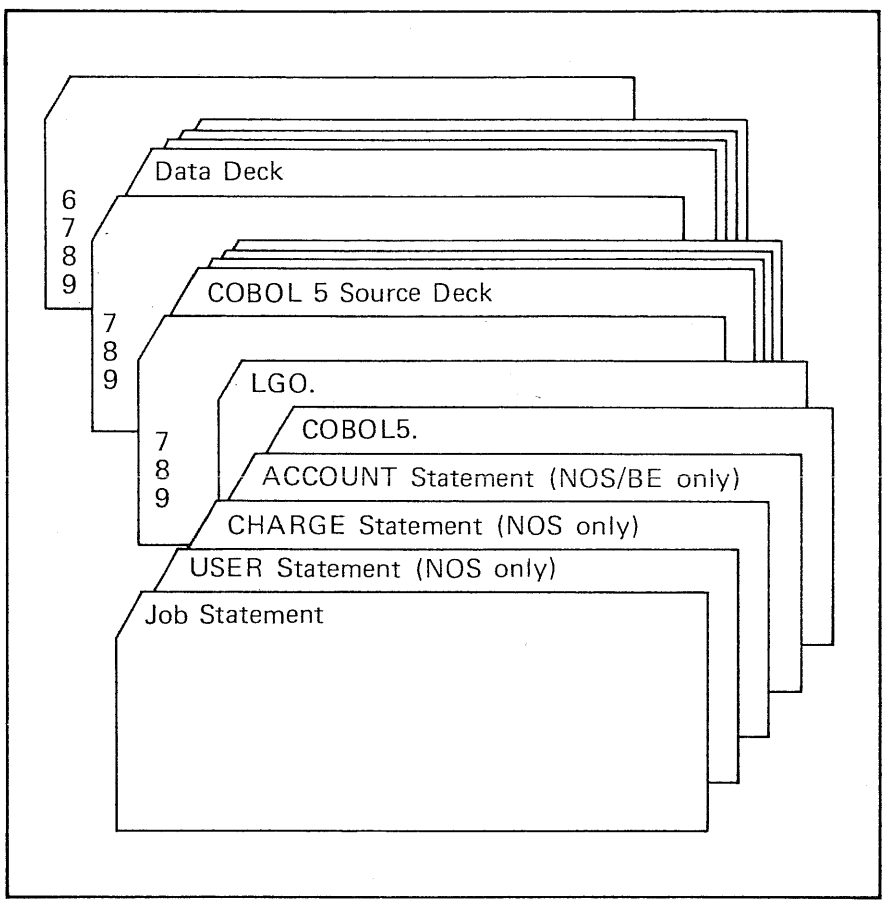
Omitted X=0	UPDATE source library on file OLDPL
X	UPDATE source library on file NEWPL
X=lfn	UPDATE source library on file lfn

SAMPLE COBOL 5 DECK STRUCTURES

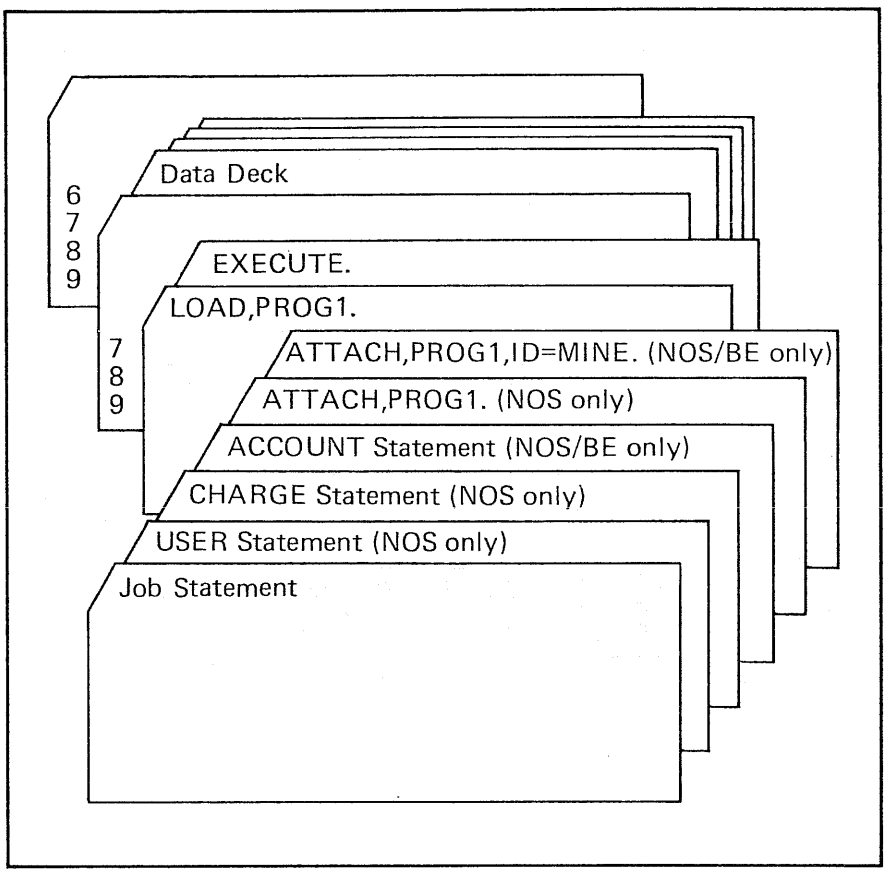
Compiling a COBOL 5 source program.



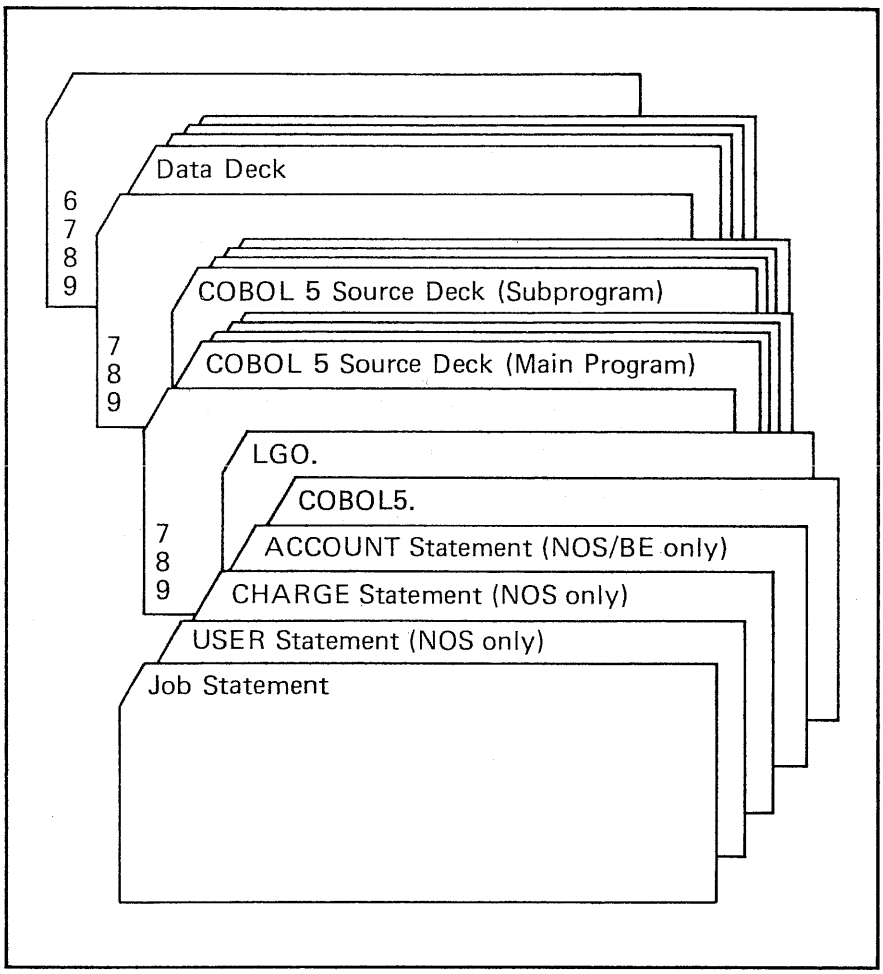
Compiling and executing a COBOL 5 source program.



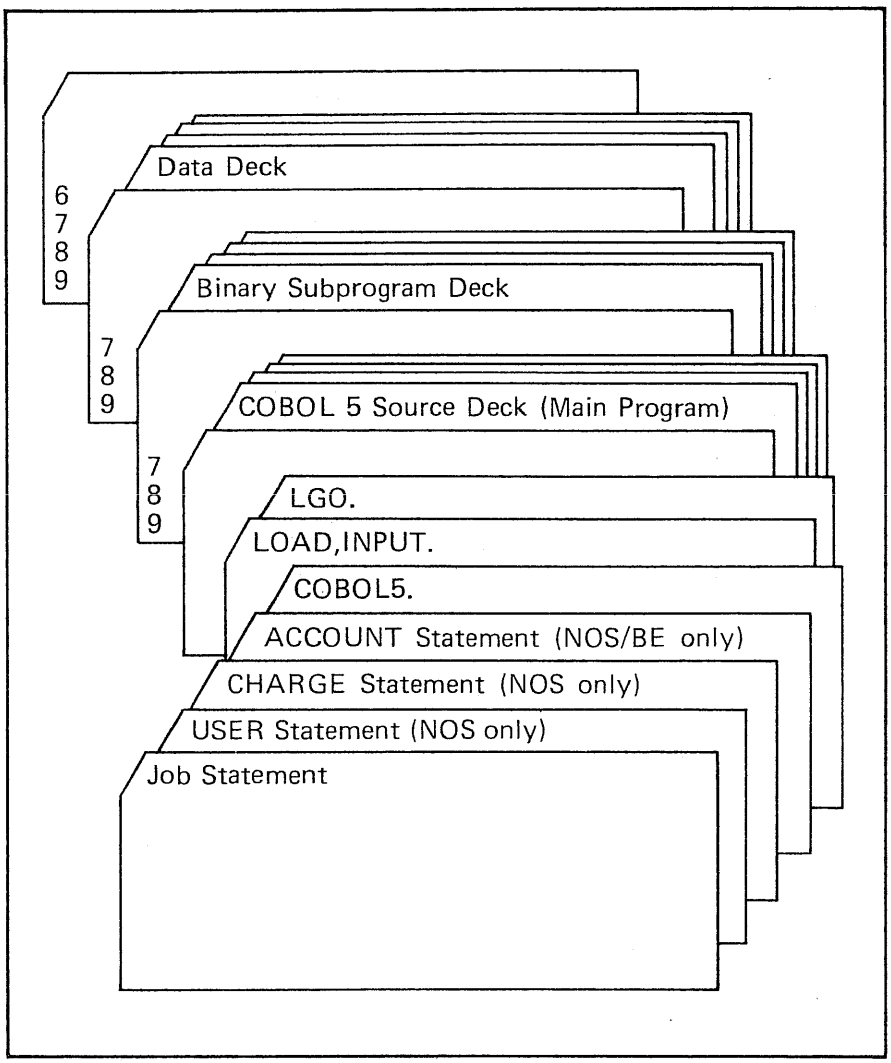
Executing a COBOL 5 object program.



Compiling and executing a COBOL 5 main program and a COBOL 5 subprogram.



Compiling and executing a COBOL 5 main program with a previously compiled subprogram.



COBOL 5 RESERVED WORD LIST

ACCEPT
ACCESS
ACTUAL-KEY
ADD
ADDRESS
ADVANCING
AFTER
ALL
ALPHABET
ALPHABETIC
ALPHANUMERIC
ALPHANUMERIC-EDITED
ALSO
ALTER
ALTERNATE
AND
ANY
APOSTROPHE
APPLY
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR

BEFORE
BEGINNING
BITS
BLANK
BLOCK
BOOLEAN
BOOLEAN-AND
BOOLEAN-EXOR
BOOLEAN-OR
BOTTOM
BY

CALL
CANCEL
CD
CF
CH
CHARACTER
CHARACTERS
CLOCK-UNITS
CLOSE
COBOL
CODE
CODE-SET
COLLATING
COLUMN
COMMA
COMMON-STORAGE
COMMUNICATION
COMP
COMP-1
COMP-2
COMP-3
COMP-4
COMPUTATIONAL
COMPUTATIONAL-1
COMPUTATIONAL-2
COMPUTATIONAL-3
COMPUTATIONAL-4
COMPUTE
CONFIGURATION
CONTAINS
CONTROL
CONTROLS
CONVERSION
COPY
CORR
CORRESPONDING
COUNT
CURRENCY

DATA
DATE
DATE-COMPILED
DATE-WRITTEN
DAY
DAY-OF-WEEK
DE
DEADLOCK
DEBUG-CONTENTS
DEBUG-ITEM
DEBUG-LINE
DEBUG-NAME
DEBUG-NUMERIC-CONTENTS
DEBUG-SUB-1
DEBUG-SUB-2
DEBUG-SUB-3
DEBUGGING
DECIMAL-POINT
DECLARATIVES
DELETE
DELIMITED
DELIMITER
DEPENDING
DESCENDING
DESTINATION
DETAIL

DIRECT
DISABLE
DISPLAY
DIVIDE
DIVISION
DOWN
DUPLICATES
DYNAMIC

EGI
ELSE
EMI
ENABLE
END
END-IF
END-OF-PAGE
END-PERFORM
END-SEARCH
ENDING
ENTER
ENVIRONMENT
EOP
EQUAL
EQUALS
ERROR
ESI
EVERY
EXCEEDS
EXCEPTION
EXIT
EXTEND
EXTERNAL

FD
FILE
FILE-CONTROL
FILES
FILLER
FINAL
FIRST
FOOTING
FOR
FROM

GENERATE
GIVING
GO
GREATER
GROUP

HASHED-VALUE
HASHING
HEADING
HIGH-VALUE
HIGH-VALUES

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

JUST
JUSTIFIED

KEY

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

MEMORY
MERGE
MESSAGE
MODE
MODULES
MOVE
MULTIPLE
MULTIPLY

NATIVE
NEGATIVE
NEXT
NO
NOT
NUMBER
NUMERIC
NUMERIC-EDITED

OBJECT-COMPUTER
OBJECT-PROGRAM
OCCURS
OF
OFF
OMITTED
ON
OPEN
OPTIONAL
OR
ORDER
ORGANIZATION
OTHER
OUTPUT
OVERFLOW

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

QUEUE
QUOTE
QUOTES

RANDOM
RD
READ
REALMS
RECEIVE
RECORD
RECORDING
RECORDS
REDEFINES
REEL
REFERENCES
RELATIVE
RELEASE
REMAINDER
REMOVAL
RENAMES
REPLACE
REPLACING
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RESET
RETURN
REVERSED
REWIND
REWRITE
RF
RH
RIGHT
ROUNDED
RUN

SAME
SD
SEARCH
SECONDARY-STORAGE
SECTION
SECURITY
SEGMENT
SEGMENT-LIMIT
SELECT
SEND
SENTENCE
SEPARATE
SEQUENCE

SEQUENTIAL
SET
SIGN
SIZE
SORT
SORT-MERGE
SOURCE
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STANDARD-1
START
STATUS
STOP
STRING
SUB-SCHEMA
SUB-QUEUE-1
SUB-QUEUE-2
SUB-QUEUE-3
SUBTRACT
SUM
SUPERVISOR
SUPPRESS
SUSPEND
SYMBOLIC
SYNC
SYNCHRONIZED

TABLE
TALLYING
TAPE
TERMINAL
TERMINATE
TEST
TEXT
THAN
THEN
THROUGH
THRU
TIME
TIMES
TO
TOP
TRACE-ON
TRACE-OFF
TRAILING
TRUE
TYPE

UNEQUAL
UNIT
UNSTRING
UNTIL
UP
UPON
USAGE
USE
USING

VALUE
VALUES
VARYING

WHEN
WITH
WORD-ADDRESS
WORDS
WORKING-STORAGE
WRITE

ZERO
ZEROES
ZEROS

+
-
*
/
**
:
=

STANDARD CHARACTER SETS

COBOL	Display Code (octal)	CDC			ASCII		
		Graphic	Hollerith Punch (026)	External BCD Code	Graphic Subset	Punch (029)	Code (octal)
	00†	: (colon)††	8-2	00	: (colon)††	8-2	072
A	01	A	12-1	61	A	12-1	101
B	02	B	12-2	62	B	12-2	102
C	03	C	12-3	63	C	12-3	103
D	04	D	12-4	64	D	12-4	104
E	05	E	12-5	65	E	12-5	105
F	06	F	12-6	66	F	12-6	106
G	07	G	12-7	67	G	12-7	107
H	10	H	12-8	70	H	12-8	110
I	11	I	12-9	71	I	12-9	111
J	12	J	11-1	41	J	11-1	112
K	13	K	11-2	42	K	11-2	113
L	14	L	11-3	43	L	11-3	114
M	15	M	11-4	44	M	11-4	115
N	16	N	11-5	45	N	11-5	116
O	17	O	11-6	46	O	11-6	117
P	20	P	11-7	47	P	11-7	120
Q	21	Q	11-8	50	Q	11-8	121
R	22	R	11-9	51	R	11-9	122
S	23	S	0-2	22	S	0-2	123
T	24	T	0-3	23	T	0-3	124
U	25	U	0-4	24	U	0-4	125
V	26	V	0-5	25	V	0-5	126
W	27	W	0-6	26	W	0-6	127
X	30	X	0-7	27	X	0-7	130
Y	31	Y	0-8	30	Y	0-8	131
Z	32	Z	0-9	31	Z	0-9	132
0	33	0	0	12	0	0	060
1	34	1	1	01	1	1	061
2	35	2	2	02	2	2	062
3	36	3	3	03	3	3	063
4	37	4	4	04	4	4	064
5	40	5	5	05	5	5	065
6	41	6	6	06	6	6	066
7	42	7	7	07	7	7	067
8	43	8	8	10	8	8	070
9	44	9	9	11	9	9	071
+	45	+	12	60	+	12-8-6	053
-	46	-	11	40	-	11	055
*	47	*	11-8-4	54	*	11-8-4	052
/	50	/	0-1	21	/	0-1	057
(	51	(	0-8-4	34	(	12-8-5	050
)	52	)	12-8-4	74	)	11-8-5	051
$	53	$	11-8-3	53	$	11-8-3	044
=	54	=	8-3	13	=	8-6	075
blank	55	blank	no punch	20	blank	no punch	040
, (comma)	56	, (comma)	0-8-3	33	, (comma)	0-8-3	054
. (period)	57	. (period)	12-8-3	73	. (period)	12-8-3	056
	60	≡	0-8-6	36	#	8-3	043
	61	[	8-7	17	[	12-8-2	133
	62	]	0-8-2	32	]	11-8-2	135
	63	%††	8-6	16	%††	0-8-4	045
" (quote)	64	≠	8-4	14	" (quote)	8-7	042
	65	⌐	0-8-5	35	_ (underline)	0-8-5	137
	66	v	11-0	52	!	12-8-7	041
	67	∧	0-8-7	37	&	12	046
	70	↑	11-8-5	55	' (apostrophe)	8-5	047
	71	↓	11-8-6	56	?	0-8-7	077
<	72	<	12-0	72	<	12-8-4	074
>	73	>	11-8-7	57	>	0-8-6	076
	74	≤	8-5	15	@	8-4	100
	75	≥	12-8-5	75	\	0-8-2	134
	76	¬	12-8-6	76	~ (circumflex)	11-8-7	136
; (semicolon)	77	; (semicolon)	12-8-7	77	; (semicolon)	11-8-6	073

†Twelve zero bits at the end of a 60-bit word in a zero byte record are an end-of-record mark rather than two colons.
††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank (55_8).

CDC CHARACTER SET COLLATING SEQUENCE								
Collating Sequence Decimal/Octal		CDC Graphic	Display Code	External BCD	Collating Sequence Decimal/Octal		CDC Graphic	Display Code	External BCD
00	00	blank	55	20	32	40	H	10	70
01	01	≤	74	15	33	41	I	11	71
02	02	%	63 †	16 †	34	42	v	66	52
03	03	[	61	17	35	43	J	12	41
04	04	→	65	35	36	44	K	13	42
05	05	≡	60	36	37	45	L	14	43
06	06	^	67	37	38	46	M	15	44
07	07	↑	70	55	39	47	N	16	45
08	10	↓	71	56	40	50	O	17	46
09	11	>	73	57	41	51	P	20	47
10	12	≥	75	75	42	52	Q	21	50
11	13	¬	76	76	43	53	R	22	51
12	14	.	57	73	44	54	]	62	32
13	15	)	52	74	45	55	S	23	22
14	16	;	77	77	46	56	T	24	23
15	17	+	45	60	47	57	U	25	24
16	20	$	53	53	48	60	V	26	25
17	21	*	47	54	49	61	W	27	26
18	22	-	46	40	50	62	X	30	27
19	23	/	50	21	51	63	Y	31	30
20	24	,	56	33	52	64	Z	32	31
21	25	(	51	34	53	65	:	00 †	none†
22	26	=	54	13	54	66	0	33	12
23	27	≠	64	14	55	67	1	34	01
24	30	<	72	72	56	70	2	35	02
25	31	A	01	61	57	71	3	36	03
26	32	B	02	62	58	72	4	37	04
27	33	C	03	63	59	73	5	40	05
28	34	D	04	64	60	74	6	41	06
29	35	E	05	65	61	75	7	42	07
30	36	F	06	66	62	76	8	43	10
31	37	G	07	67	63	77	9	44	11

†In installations using the 63-graphic set, the % graphic does not exist. The : graphic is display code 63, External BCD code 16.

ASCII CHARACTER SET COLLATING SEQUENCE

Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code	Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code
00	00	blank	55	20	32	40	@	74	40
01	01	!	66	21	33	41	A	01	41
02	02	"	64	22	34	42	B	02	42
03	03	#	60	23	35	43	C	03	43
04	04	$	53	24	36	44	D	04	44
05	05	%	63†	25	37	45	E	05	45
06	06	&	67	26	38	46	F	06	46
07	07	'	70	27	39	47	G	07	47
08	10	(	51	28	40	50	H	10	48
09	11	)	52	29	41	51	I	11	49
10	12	*	47	2A	42	52	J	12	4A
11	13	+	45	2B	43	53	K	13	4B
12	14	,	56	2C	44	54	L	14	4C
13	15	-	46	2D	45	55	M	15	4D
14	16	.	57	2E	46	56	N	16	4E
15	17	/	50	2F	47	57	O	17	4F
16	20	0	33	30	48	60	P	20	50
17	21	1	34	31	49	61	Q	21	51
18	22	2	35	32	50	62	R	22	52
19	23	3	36	33	51	63	S	23	53
20	24	4	37	34	52	64	T	24	54
21	25	5	40	35	53	65	U	25	55
22	26	6	41	36	54	66	V	26	56
23	27	7	42	37	55	67	W	27	57
24	30	8	43	38	56	70	X	30	58
25	31	9	44	39	57	71	Y	31	59
26	32	:	00†	3A	58	72	Z	32	5A
27	33	;	77	3B	59	73	[	61	5B
28	34	<	72	3C	60	74	\	75	5C
29	35	=	54	3D	61	75	]	62	5D
30	36	>	73	3E	62	76	^	76	5E
31	37	?	71	3F	63	77	_	65	5F

†In installations using a 63-graphic set, the % graphic does not exist. The : graphic is display code 63.

64 CHARACTER EBCDIC SUBSET COLLATING SEQUENCE					
Collating Sequence Decimal/Octal		Graphic	EBCDIC Punch	Display Code	EBCDIC Code
00	00	blank	no punch	55	40
01	01	.	12-8-3	57	4B
02	02	<	12-8-4	72	4C
03	03	(	12-8-5	51	4D
04	04	+	12-8-6	45	4E
05	05	¡	12-8-7	66	4F
06	06	&	12	67	50
07	07	$	11-8-3	53	5B
08	10	*	11-8-4	47	5C
09	11	)	11-8-5	52	5D
10	12	;	11-8-6	77	5E
11	13	¬	11-8-7	76	5F
12	14	-	11	46	60
13	15	/	0-1	50	61
14	16	.	0-8-3	56	6B
15	17	%	0-8-4	63	6C
16	20	_	0-8-5	65	6D
17	21	>	0-8-6	73	6E
18	22	?	0-8-7	71	6F
19	23	:	8-2	00	7A
20	24	#	8-3	60	7B
21	25	@	8-4	74	7C
22	26	'	8-5	70	7D
23	27	=	8-6	54	7E
24	30	"	8-7	64	7F
25	31	¢	12-8-2/12-0	61	4A
26	32	A	12-1	01	C1
27	33	B	12-2	02	C2
28	34	C	12-3	03	C3
29	35	D	12-4	04	C4
30	36	E	12-5	05	C5
31	37	F	12-6	06	C6

64 CHARACTER EBCDIC SUBSET COLLATING SEQUENCE (Contd)				
Collating Sequence Decimal/Octal	Graphic	EBCDIC Punch	Display Code	EBCDIC Code
32 40	G	12-7	07	C7
33 41	H	12-8	10	C8
34 42	I	12-9	11	C9
35 43	!	11-8-2/11-0	62	5A
36 44	J	11-1	12	D1
37 45	K	11-2	13	D2
38 46	L	11-3	14	D3
39 47	M	11-4	15	D4
40 50	N	11-5	16	D5
41 51	O	11-6	17	D6
42 52	P	11-7	20	D7
43 53	Q	11-8	21	D8
44 54	R	11-9	22	D9
45 55	none	0-8-2	75	E0
46 56	S	0-2	23	E2
47 57	T	0-3	24	E3
48 60	U	0-4	25	E4
49 61	V	0-5	26	E5
50 62	W	0-6	27	E6
51 63	X	0-7	30	E7
52 64	Y	0-8	31	E8
53 65	Z	0-9	32	E9
54 66	0	0	33	F0
55 67	1	1	34	F1
56 70	2	2	35	F2
57 71	3	3	36	F3
58 72	4	4	37	F4
59 73	5	5	40	F5
60 74	6	6	41	F6
61 75	7	7	42	F7
62 76	8	8	43	F8
63 77	9	9	44	F9

UNIVAC 1108 COLLATING SEQUENCE [UNI]					
Collating Sequence Decimal/Octal		1108 Graphic	Card Punch	Display Code	CYBER Graphic
00	00	@	8-7	61	[
01	01	[	12-8-5	75	≥
02	02	]	11-8-5	70	↑
03	03	∀	12-8-7	77	:
04	04	Δ	11-8-7	73	>
05	05	blank	no punch	55	blank
06	06	A	12-1	01	A
07	07	B	12-1	02	B
08	10	C	12-3	03	C
09	11	D	12-4	04	D
10	12	E	12-5	05	E
11	13	F	12-6	06	F
12	14	G	12-7	07	G
13	15	H	12-8	10	H
14	16	I	12-9	11	I
15	17	J	11-1	12	J
16	20	K	11-2	13	K
17	21	L	11-3	14	L
18	22	M	11-4	15	M
19	23	N	11-5	16	N
20	24	O	11-6	17	O
21	25	P	11-7	20	P
22	26	Q	11-8	21	Q
23	27	R	11-9	22	R
24	30	S	0-2	23	S
25	31	T	0-3	24	T
26	32	U	0-4	25	U
27	33	V	0-5	26	V
28	34	W	0-6	27	W
29	35	X	0-7	30	X
30	36	Y	0-8	31	Y
31	37	Z	0-9	32	Z

UNIVAC 1108
COLLATING SEQUENCE [UNI] (Contd)

Collating Sequence Decimal/Octal		1108 Graphic	Card Punch	Display Code	CYBER Graphic
32	40	)	12-8-4	52	)
33	41	–	11	46	–
34	42	+	12	45	+
35	43	<	12-8-6	76	¬
36	44	=	8-3	54	=
37	45	>	8-6	63	%
38	46	&	8-2	00	:
39	47	$	11-8-3	53	$
40	50	*	11-8-4	47	*
41	51	(	0-8-4	51	(
42	52	%	0-8-5	65	→
43	53	:	8-5	74	≤
44	54	?	12-0	72	<
45	55	!	11-0	66	∨
46	56	,	0-8-3	56	,
47	58	\	0-8-6	60	≡
48	60	0	0	33	0
49	61	1	1	34	1
50	62	2	2	35	2
51	63	3	3	36	3
52	64	4	4	37	4
53	65	5	5	40	5
54	66	6	6	41	6
55	67	7	7	42	7
56	70	8	8	43	8
57	71	9	9	44	9
58	72	'	8-4	64	≠
59	73	;	11-8-6	71	↓
60	74	/	0-1	50	/
61	75	.	12-8-3	57	.
62	76	∪	0-8-7	67	∧
63	77	≠	0-8-2	62	]

CONTROL DATA CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO.
MINNEAPOLIS, MINN. 55440

SALES OFFICES AND SERVICE CENTERS
IN MAJOR CITIES THROUGHOUT THE WORLD