# Xerox ANS COBOL (for BPM/CP-V)

**Xerox 550/560 and Sigma 5-9 Computers**

**Operations**

**Reference Manual**

90 15 01G

XEROX

# Xerox ANS COBOL (for BPM/CP-V)

## Xerox 550/560 and Sigma 5-9 Computers

## Operations Reference Manual

90 15 01G

May 1976

File No.: 1X23
XJ80, Rev. 0
Printed in U.S.A.

## REVISION

This publication documents the E07 version of the Xerox ANS COBOL compiler for BPM and CP-V. This is the G edition of the manual; it is identical to the F edition (90 15 01F, dated September 1973) including all revision packages (90 15 01F-1, 3/74; -2, 5/74; and -3, 6/75), and serves to consolidate them. Vertical lines in the outer margin indicate changes made in the most recent revision.

## RELATED PUBLICATIONS

| Title | Publication No. |
|---|---|
| Xerox 550 Computer/Reference Manual | 90 30 77 |
| Xerox 560 Computer/Reference Manual | 90 30 76 |
| Xerox Sigma 5 Computer/Reference Manual | 90 09 59 |
| Xerox Sigma 6 Computer/Reference Manual | 90 17 13 |
| Xerox Sigma 7 Computer/Reference Manual | 90 09 50 |
| Xerox Sigma 8 Computer/Reference Manual | 90 17 49 |
| Xerox Sigma 9 Computer/Reference Manual | 90 17 33 |
| Xerox Batch Processing Monitor (BPM)/BP,RT Reference Manual | 90 09 54 |
| Xerox Batch Processing Monitor (BPM)/OPS Reference Manual | 90 11 98 |
| Xerox Control Program-Five (CP-V)/TS Reference Manual | 90 09 07 |
| Xerox Control Program-Five (CP-V)/OPS Reference Manual | 90 16 75 |
| Xerox Control Program-Five (CP-V)/TS User's Guide | 90 16 92 |
| Xerox Control Program-Five (CP-V)/SM Reference Manual | 90 16 74 |
| Xerox ANS COBOL (for BPM/CP-V)/LN Reference Manual | 90 15 00 |
| Xerox ANS COBOL On-Line Debugger Reference Manual | 90 30 60 |
| Xerox Sort and Merge (for BPM/CP-V)/Reference Manual | 90 11 99 |
| Xerox Data Management System (DMS) (for BPM/CP-V)/Reference Manual | 90 17 38 |
| Xerox Extended Data Management System (EDMS)/Reference Manual | 90 30 12 |

Manual Content Codes: BP – batch processing, LN – language, OPS – operations, RP – remote processing, RT – real-time, SM – system management, SP – system programming, TP – transaction-processing, TS – time-sharing, UT – utilities.

# CONTENTS

## TABLES

# PREFACE

This manual describes the operations and characteristics of the Xerox ANS COBOL system (under BPM and CP-V) including the compiler, library subroutines, and pertinent compiler and run-time diagnostics.

It is assumed that the reader has a good working knowledge of the COBOL language as described in the Xerox ANS COBOL (for BPM/CP-V) Reference Manual and of the operation of the Xerox Control Program-Five and/or the Xerox Batch Processing Monitor.

# 1. PROGRAMMING HINTS

This chapter provides a number of useful hints for improving the efficiency of object programs.

## Description of Numeric Data Items

Avoid Mixed-Mode Arithmetic Statements. An arithmetic statement involving data items of more than one mode (binary, decimal, or floating) requires one or more relatively expensive conversions of the operands or the result. These conversions, which require run-time subroutines, are not needed when mixed-mode arithmetic statements are avoided.

Use Binary Rather Than Decimal Subscripts. The COBOL compiler requires that all subscripts be binary. The costly conversions of subscripts can be avoided if they are defined as binary rather than decimal.

Minimize Exponentiation. Exponentiation involves floating-point calculation mode even when there are no floating-point operands.

Use Binary Calculations if Possible. Binary calculations are faster than decimal or floating-point calculations and much faster than mixed mode. However, since binary items cannot contain a decimal point, their use is limited. If counters (i.e., input and output record counters) and subscripts are defined as binary data items, and other numeric data items as decimal, the number of costly conversions will generally be minimized without loss of the efficiencies of binary arithmetic.

Avoid Using Decimal Items Exceeding 15 Digits. Of the several ways to describe decimal items in COBOL, some permit the compiler to generate fewer instructions than others. For example, a data item containing 16, 17, or 18 decimal digits may require a double precision subroutine amounting to over 20 extra instructions not needed with items of 15 digits or less.

Specify Odd-Size Decimal Display Fields. Sigma pack and unpack instructions do not operate on even-sized decimal display fields. The compiler moves an even-sized display field to a work area in order to append a high-order zero, creating an odd-size field at a cost of three to six machine instructions. These extra instructions can be saved each time the field is referenced if it is odd-size to begin with.

Specify Packed Decimal if Possible. Packed decimal items occupy less space than decimal display items of the same size. Besides, they don't have to be packed and unpacked when used. Packed decimal, therefore, results in fewer instructions being generated for a given arithmetic statement.

Specify Signed Rather Than Unsigned Decimal Display Fields. The compiler must generate three instructions to get rid of the sign when a decimal field is described as unsigned. This applies each time the field is stored into.

To summarize, a decimal data item should be less than 16 digits long and have an odd number of digits. It should be described with a sign, and as packed decimal rather than decimal display.

### Examples of a Decimal Add

Example 1:

```
77    A    PIC 99.
77    B    PIC 99.

ADD A TO B.
```

Seventeen instructions will be generated.

Example 2:

```
77      A      PIC S999 USAGE IS COMP-3.
77      B      PIC S999 USAGE IS COMP-3.

ADD A TO B.
```

Three instructions will be generated.

## Table Handling

Use indexes rather than subscripts for referencing data items described with or subordinate to one or more OCCURS clauses. With a subscript, the displacement into the table must be calculated (subscript x entry size – entry size) each time the table item is referenced. With indexing, this calculation is made only once when the index is set. In addition, when a table is described with an INDEXED BY clause, the SEARCH statement can be used on that table, and the search routine generated by the compiler will be more efficient than one written by the programmer.

Subscripts, if used, should be in binary since decimal subscripts are converted to binary anyway.

## OCCURS DEPENDING ON Clause

Keep the use of this clause to a minimum. The OCCURS DEPENDING ON clause can be used effectively with variable length records to reduce the physical size of files and save I/O time. However, the clause will increase execution time because any reference to data item with an OCCURS DEPENDING ON clause requires that its size be calculated each time it is referenced. On balance, therefore, it is recommended that the use of OCCURS DEPENDING ON be kept to a minimum.

## Sort

If a program has an input or output procedure, or both, either the co-resident or the linked sort can be requested. The co-resident sort, which occupies core memory at the same time as the COBOL program, can significantly reduce the number of input/output operations and, hence, run-time. It should, therefore, be used when core memory is available.

When linked sort is used, the RELEASE statements in the COBOL input procedure build a file as an interface for the linked sort. When the input procedure is finished, sort replaces the COBOL program in memory (i.e., the COBOL object program is swapped out), and sorts the file created by the input procedure into a new file, whereupon the COBOL program is brought back into core memory, replacing the sort. The RETURN statements of the output procedure read the sorted file.

Co-resident sort avoids superfluous reading and writing of the two files used by sort. When the SORT verb is encountered, control is transferred to sort. Then, when sort wants to read a record, it gives control to the COBOL input procedure, which provides a record through the use of the RELEASE statement. When sort wants to write the sorted file, it gives control to the COBOL output procedure, which accepts the sorted record with RETURN statements. Thus, unnecessary input/output is avoided.

## I/O Considerations

Block Sequential Files. Blocking sequential files can shorten I/O and CPU times by reducing the number of physical records and increasing their size. It lessens start/stop times for tapes and compute time for setting up the I/O operations. A block size of 5000 to 7000 bytes is recommended.

Use Unlabeled or ANS-Labeled Tapes Rather Than Labeled Tapes. The monitor attaches control information to records written to labeled tape, but not to device tape or ANS-labeled tape. In addition, I/O on device tape of ANS-labeled tape is double-buffered. This allows I/O operations and CPU operations to overlap.

Avoid the INTO Option of the READ Statement and the FROM Option of the WRITE Statement. Working from record areas rather than moving the records to and from working storage reduces program run time. In some cases where a master file is updated, moving the input master record to the output master record can be avoided. This is accomplished with the SAME RECORD AREA statement that allows a record to be read, updated, and written with no record movement by the programmer. It also reduces the size of the program.

Block Relative Files. The monitor always reads or writes a minimum of one granule (512 words, 2048 bytes) from or to a relative file. The BLOCK CONTAINS clause of COBOL should be used to obtain a physical record size which is a multiple of granule size. This will insure optimum utilization of disk space and maximum speed. The COBOL I/O system will locate the proper granule/block and retrieve the user's record based on the relative record number supplied.

Note: To remove a record from a relative file place a X'FF00' in byte 1 of the record. This will signal the COBOL I/O system that this record is a null or deleted one. All records not written when the file is created are set to null by the system.

## Report Writer

A report restart facility can be provided by programming around the OPEN statement for the report file which is being restarted. No abort will occur. The programmer can accept a page start parameter from a control card, count the pages skipped and when the start page is reached the logic can then go to the OPEN and start printing. No other program logic needs to be altered.

The OPEN can also be circumvented for the case where there are multiple FDs for a given RD and the suppression of its output to a file is desired.

## COBOL/FORTRAN Interfaces

### COBOL to FORTRAN

It is possible for COBOL to "call" FORTRAN subprograms by means of the ENTER verb. However, certain setup routines must be called and temp variables initialized before the FORTRAN library routines can be utilized. This initialization process is triggered by the COBOL program having the statement:

```
ENTER FORTLINK
```

This is done prior to the first call to a FORTRAN subprogram.

In addition to the ENTER list of arguments, the COBOL program may contain a COMMON-STORAGE SECTION. This generates a DSECT which can be referenced in the FORTRAN subprograms by means of the statement:

```
COMMON/TALLY/TALLY, ....
```

Note that the first word of TALLY cannot be referenced in the COBOL program and is not initialized.

The COBOL CS option permits use of a name other than TALLY for the common-storage DSECT.

### FORTRAN to COBOL

Calling COBOL programs from FORTRAN programs has some restrictions. First, no arguments may be specified, data communication must be via named common, as indicated above. Second, COBOL does not have a verb equivalent to RETURN in FORTRAN. Third, the CALL statement in FORTRAN will use register 15 for its linkage, therefore it must be saved before the COBOL program starts executing.

In order to achieve the FORTRAN to COBOL linkage it is necessary to use a METASYMBOL routine to save register 15 upon entry into the COBOL routine and to exit from the COBOL routine. The entry point in the COBOL routine that is used in the FORTRAN CALL statement must be DEFed by using it in an ENTER COBOL statement.

The METASYMBOL routine should be coded like this:

```
                  SYSTEM     SIG7
                  DEF        SAVEREG
                  DEF        RETNREG
S15               RES        1
SAVEREG           STW,15     S15          SAVE REGISTER 15
                  B          *11          RETURN TO COBOL
RETNREG           B          *S15         RETURN TO FORTRAN
                  END
```

To use the METASYMBOL routine GO TO statements are required in the COBOL routine to go to the two entry points.

## Demonstration Job

The following job has been written to illustrate COBOL/FORTRAN interfaces.

```
!JOB XEROX,COBOL,7  . COBOL/FORTRAN
!TITLE COBOL TO FORTRAN
!COBOL LS,LO,GO
       IDENTIFICATION DEVISION.
       PROGRAM-ID.  COB:S.
       DATE-WRITTEN.  DEC 12, 1974.
       DATE-COMPILED.
       REMARKS.  COBOL TO FORTRAN AND ENTRY FOR FORTRAN.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.  XEROX-560.
       OBJECT-COMPUTER.  XEROX-560.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.

       DATA DIVISION.
       WORKING-STORAGE SECTION.
       77  I1 COMP VALUE 1.
       77  I2 COMP VALUE 2.
       77  I3 COMP VALUE 3.
       COMMON-STORAGE SECTION.
       77  J1 COMP VALUE 11.
       77  J2 COMP VALUE 12.
       77  J3 COMP VALUE 13.

       PROCEDURE DIVISION.

       S-1.
           ENTER FORTLINK.
           ENTER FTEST1 I1,I2,I3.
           ENTER FTEST2.
           STOP RUN.
           ENTER COBOL CTEST.
       CTEST.
           GO TO SAVEREG.
           DISPLAY 'CTEST ENTERED' UPON PRINTER.
           GO TO RETNREG.
```

```
!FORTRAN LS,LO,GO,S
      SUBROUTINE FTEST1 (I1, I2, I3)

C     THIS IS CALLED BY COBOL

      COMMON/TALLY/TALLY,J1,J2,J3
      OUTPUT I1,I2,I3,J1,J2,J3
      RETURN

      SUBROUTINE FTEST2

C     THIS CALLS COBOL

      OUTPUT 'FTEST2 RUNNING'
      CALL CTEST
      OUTPUT 'RETURN FROM CTEST'
      STOP

      END
```

# 2. COMPILER

This chapter describes various compilation options, the compiler outputs, and pertinent compile-time diagnostic messages.

## Compilation Initiation

A COBOL processor control command must initiate each Xerox ANS COBOL compilation job. The format of the command is

```
!COBOL s1,s2,s3,...,sn
```

where

$s_i$ may specify any of the following output options:

| | |
|---|---|
| BO | Permanent copy of the object program via the BO (Binary Output) device. |
| CS(name) | COMMON-STORAGE SECTION name (see "Object Program Structure" in Chapter 4). |
| DEBUG | Source program debugging statements (TRACE, EXHIBIT). |
| DIAG | Trivial diagnostic messages. |
| DMAP | Data Division Map. |
| DQ | Double quotation mark. |
| GO | Load-and-go copy of the object program via the GO device. |
| LIB(accounts) | Library accounts. |
| LO | An object program listing. |
| LS | A source program listing. |
| MAIN | Main program (see "Inter-Program Communication" in Chapter 3). |
| MAPS | Both Data Division Map and Procedure Division Map. |
| PMAP | Procedure Division Map. |
| SEG | Priority segments (see "Segmented Object Programs" in Chapter 3 and "Segmentation Feature" in Chapter 5). |
| SEQCHK | Sequence check. |
| SO | Source output. |
| SRTx | Co-resident sort. |
| SUB | Subprogram (suppresses generation of "END start"). |
| SYN | Compilation for syntax checking only (no code generation). |
| TEST | On-line debugger. |
| XREF | A cross-reference listing. |

The processor control command may be written in free form. Any number of spaces may appear between !COBOL and the specification string. Spaces are permitted before or after each option, but the option itself may not contain embedded spaces.

The specification string may be continued in one or more commands following the !COBOL control command. Continuation is specified by placing a semicolon at any point where a blank is legal. Position 1 of the continuation commands must be blank.

Specification sequence may vary. If no specifications are entered for the COBOL command, the options

LS, BO

are assumed. If any option is specified, all desired options must be specified.

## BO (Binary Object Deck)

This option specifies that relocatable object modules (ROMs) of the compiled program are to be produced in binary form.

## CS(name) (COMMON-STORAGE SECTION)

This option specifies the name to be used in the object program for the dummy program section that represents the COMMON-STORAGE SECTION. If this option is not specified, the name TALLY is used. If a name is used it is restricted to a maximum of 7 characters.

## DEBUG (Debugging Statements)

This option specifies that debugging statements TRACE and EXHIBIT are to be included in the compilation. Absence of this option enables debugging statements to be suppressed at compilation time. Thus it is not necessary to delete these statements from the source program when it is recompiled to obtain an operational object program. For a complete description of debugging statements refer to Chapter 12 of the Xerox ANS COBOL (for BPM/CP-V)/LN Reference Manual, 90 15 00.

## DIAG (Trivial Diagnostics)

This option specifies that trivial (warning) diagnostics also are to be listed along with the other diagnostics. These trivial diagnostics do not affect generation of the object program, but merely serve as warnings to the programmer. Examples of trivial diagnostics are

INCORRECT PUNCTUATION

EXTERNAL REFERENCE GENERATED

RIGHTMOST AND/OR FRACTIONAL DIGITS TRUNCATED

LEFTMOST DIGITS/CHARACTERS TRUNCATED

INTEGER AND FRACTIONAL DIGITS TRUNCATED

## DMAP (Data Division Map)

This option specifies that the Data Division Map is to be produced. This Map is an alphabetical list of the data-names along with their sizes and relative locations. Figure 1 shows a sample Data Division Map listing. The following information appears on the listing:

| | |
|---|---|
| Line number | Corresponds to the source line number where the data-name is defined. |
| Data-name | The data-name as it appears in the DATA DIVISION. |
| Relative location | The displacement from the origin of the base section in the object program. |
| | Example: If the origin of the base section in the object program was hexadecimal location '03C00' and the relative location of the data-name was '0058 3', this means the data-name begins in byte 3 of hexadecimal location '03C58'. |
| Size | The size of the data-name in bytes. |

```
 00015              000120 DATA DIVISION.
 00016              000130 FILE SECTION.
 00017              000140 FD  IN-FILE LABEL RECORDS ARE STANDARD DATA RECORD IS IN-REC.
 00018              000150  01 IN-REC.
 00019              000160     02 WEEK       PICTURE 9.
 00020              000170     02 DEPP       PICTURE 9.
 00021              000180     02 TYPE-RUN   PICTURE A(10).
 00022              000190     02 PROG       PICTURE X(4).
 00023              000200     02 DATE       PICTURE X(5).
 00024              000210     02 FILLER     PICTURE X(7).
 00025              000220     02 MINUTES    PICTURE 999V9.
 00026              000230     02 CHARGE     PICTURE 9999V9.
 00027              000231     02 FILLER   PICTURE X(43).
 00028              000240 FD  REP-FILE LABEL RECORDS ARE OMITTED REPORT IS USAGE-REPORT.
 00029              000250 FD  PRINT-FILE LABEL RECORD OMITTED DATA RECORD D-REC.
 00030              000260 01  D-REC PICTURE X(120).
 00031              000261 FD  TAPE-FILE
 00032              000262     LABEL RECORD IS LABEL1
 00033              000263     DATA RECORD IS TAPEREC.
 00034              000264 01  TAPEREC PICTURE X(10).
 00035              000265 01  LABEL1 PICTURE X(20).
 00036              000270 WORKING-STORAGE SECTION.
 00037              000280 77  MONTH PICTURE X(9).
 00038              000290 77  COUNT PICTURE 9 VALUE 1.
 00039              000300 77  CONT PICTURE X(11).
 00040              000310 77  SAVE-DEP PICTURE 9 VALUE 0.
 00041              000311 77  DEP PICTURE 9.
 00042              000320 01  DEP-NAMES.
 00043              000330     02 FILLER PICTURE A(11) VALUE 'ENGINEERING'.
 00044              000340     02 FILLER PICTURE A(11) VALUE 'SALES'.
 00045              000350     02 FILLER PICTURE A(11) VALUE 'ACCOUNTING'.
 00046              000360 01  D-NAMES REDEFINES DEP-NAMES.
 00047              000370     02 NAME PICTURE A(11) OCCURS 3 TIMES.
```

```
     COBOL DATA DIVISION MAP LISTING                    21:42 MAR 12, 1975          PAGE   1

LINE-NO     DATA-NAME                      REL-LOC  SIZE  RECORD-NAME            BASE-NAME

00026       CHARGE                         00008      5    IN-REC                 IN-FILE
00039       CONT                           00004     11                           WORKING-STORAGE
00038       COUNT                          00003      1                           WORKING-STORAGE
00046       D-NAMES                        0000A     33                           WORKING-STORAGE
00030       D-REC                          00000    120                           PRINT-FILE
00023       DATE                           00004      5    IN-REC                 IN-FILE
00041       DEP                            00008      1                           WORKING-STORAGE
00042       DEP-NAMES                      0000A     33                           WORKING-STORAGE
00020       DEPP                           00000 1    1    IN-REC                 IN-FILE
00017       IN-FILE                                                               FD - FILE
00018       IN-REC                         00000     80                           IN-FILE
00035       LABEL1                         00000     20                           TAPE-FILE
00025       MINUTES                        00007      4    IN-REC                 IN-FILE
00037       MONTH                          00000      9                           WORKING-STORAGE
00047       NAME                           0000A     11    D-NAMES                WORKING-STORAGE
00029       PRINT-FILE                                                            FD - FILE
00022       PROG                           00003      4    IN-REC                 IN-FILE
00028       REP-FILE                                                              FD - FILE
00040       SAVE-DEP                       00007      1                           WORKING-STORAGE
00048       TALLY                          00000      3                           COMMON-STORAGE
00033       TAPE-FILE                                                             FD - FILE
00034       TAPEREC                        00000     10                           TAPE-FILE
00021       TYPE-RUN                       00000 2   10    IN-REC                 IN-FILE
00019       WEEK                           00000      1    IN-REC                 IN-FILE
```

Figure 1. Sample Data Division Map Listing

| | |
|---|---|
| Record name | The name of the record (level 01) to which the data-name belongs. |
| Base name | The base section (corresponding to DSECTs in the object program) to which the data-name belongs. A base section is created for WORKING-STORAGE and each file-name defined in the source program. |

### DQ (Double Quotation Mark)

This option informs the compiler that the source program to be compiled uses the double quotation mark (") exclusively, instead of the single quotation mark ('). If this option is not specified, the single quotation mark is assumed. The Hollerith code for the double quotation mark is a multipunched 8-7 (hexadecimal 7F). The Hollerith code for the single quotation mark is a multipunched 8-5 (hexadecimal 7D).

### GO (Compile and Run)

This option specifies that the source program is to be compiled and then executed. The load-and-go copy of the object program is transmitted to the monitor GO file. The GO option also must be specified on the monitor !LOAD control command.

### LIB (Library Accounts)

This option specifies optional account numbers which may contain library source files that are needed to satisfy COPY statements in the source program. This permits library files of other accounts to be accessed. Up to three optional accounts may be specified.

Example: `!COBOL LS, LIB(ACC85011, TESTA, 90301)`

This LIB option instructs the compiler to search accounts ACC85011, TESTA, or 90301 for those library files that do not exist under the user's own account number.

Library files on labeled tape may also be accessed. (See "COBOL Library on Tape" in Chapter 3.)

### LO (Object Listing)

This option specifies that a listing of the object program is to be output on the LO device. Figure 2 illustrates a sample object listing, which is keyed to the source program by line number and resembles an assembly language listing.

### LS (Source Listing)

The source listing is output to the LO (Listing Output) device whenever the COBOL command specifies (explicitly or implicitly) the LS option.

Figure 3 depicts a sample Xerox ANS COBOL source program and Procedure Division Map listing with diagnostics immediately following the source lines containing errors. The COBOL processor control command is presented as the initial line of the listing. Each subsequent line contains a line number appearing in two parts separated by a period: the first number represents the position of the line in the source program as obtained from the SI (source input) device; the second number (subnumber) denotes lines inserted into the source program as a result of library retrieval statements (COPY or COPY REPLACING) in the source program.

Whenever the compiler detects an error in the source program, a diagnostic message and its message number are printed on the source listing immediately following the line containing the error. If the COBOL control command neither specifies nor implies the LS option, only the number of the line to which the diagnostic relates, the message number, and the message itself are printed. The number of diagnostic messages issued and the highest diagnostic severity level are printed at the end of the source listing. A complete listing of compiler diagnostics is shown in Chapter 5 of this manual.

```
        COBOL OBJECT CODE LISTING                           21:42 MAR 12, 1975

                                       *****  ROOT SEGMENT  *****

                                           DEF     TALLY
              00000                        DSECT   0                            SIZE IS 3
              00000                        ORG     TALLY
                                           DEF     INPUT-DEVICE
              00000                        DSECT   0                            SIZE IS 80
                                           DEF     I:INPUT-DEVICE
              00000                        DSECT   0                            SIZE IS 28
              00000                        ORG     I:INPUT-DEVICE
                                           DEF     F:INPUT-DEVICE
              00000     00000008           DATA,4  F:INPUT-DEVICE+X'00000000'
              00001     02000000 A         DATA,4  X'02000000'
              00002     00000000 A         DATA,4  X'00000000'
              00003     00000000 A         DATA,4  X'00000000'
              00004     00000000 A         DATA,4  X'00000000'
              00005     00000000 A         DATA,4  X'00000000'
              00006     00A00000           DATA,4  INPUT-DEVICE+X'00A00000'
              00000                        DSECT   2                            SIZE IS 208
              00000                        ORG     F:INPUT-DEVICE      *ZERO WORDS NOT PRINTED*
              00000     34000003 A         DATA,4  X'34000003'
              00001     10020009 A         DATA,4  X'10020009'
              00002     10000000           DATA,4  INPUT-DEVICE+X'10000000'
              00003     00A00000 X         DATA,4  C:ERA+X'00A00000'
              00004     00000000 X         DATA,4  C:ABA+X'00000000'
              00005     80000011 A         DATA,4  X'80000011'
              00006     00000016           DATA,4  F:INPUT-DEVICE+X'00000016'
              0000A     0000002C           DATA,4  F:INPUT-DEVICE+X'0000002C'
              00016     01000008 A         DATA,4  X'01000008'
              0001F     02000002 A         DATA,4  X'02000002'
              00022     03000002 A         DATA,4  X'03000002'
              00025     04000002 A         DATA,4  X'04000002'



              00069     77360000 X         UNPK,3  C:TLBL,3
              0006A     72A60001 X         LB,10   C:TLBL+X'1',3
              0006B     49A00019           OR,10   BASE+X'19'
              0006C     75A60001 X         STB,10  C:TLBL+X'1',3
00040         00008                                                     ORG     BASE+X'8'
              00008     01000000 X                                      DATA,4  BA(C:TLBL)+X'01000000'
              0006D     32300008           LW,3    BASE+X'8'
              0006E     222005A8           LI,2    BA(BASE)+X'5A8'
              0006F     61200000 A         MBS,2   0
              00070     EAB00025           BAL,11  *BASE+X'25'
              00025                                                     ORG     BASE+X'25'
              00025     00000071                                        DATA,4  BASE+X'00000071'
00042         00071     22200001 A         LI,2    1
              00072     76340000 X         PACK,3  C:TLBL,2
              00073     7F00003E           DST,0   BASE+X'3E'
              00074     76300016           PACK,3  BASE+X'16'
              00075     7D00003E           DC,0    BASE+X'3E'
              00076     6930007F           BNE     $+9
00043         00009                                                     ORG     BASE+X'9'
              00009     2B000000                                        DATA,4  BA(OUTPUTHREE)+X'2B000000'
              00077     32300009           LW,3    BASE+X'9'
              00078     222005A9           LI,2    BA(BASE)+X'5A9'
              00079     61200000 A         MBS,2   0
              0007A     22A00025 A         LI,10   37
              0007B     75A00003 A         STB,10  3
              0007C     32100003 A         LW,1    3
              0007D     61000061           MBS,0   BA(BASE)+X'61'
              0007E     6800007F           B       $+1
00043         0007F     EAB00026           BAL,11  *BASE+X'26'
              00026                                                     ORG     BASE+X'26'
              00026     00000080                                        DATA,4  BASE+X'00000080'
00047         00080     22E00001 A         LI,14   1
              00081     22600000           LI,6    I:INPUT-DEVICE
              00082     6AB00000 X         BAL,11  C:OPN
00048         00083     6AB00000 X         BAL,11  C:BBF
              00084     35B00025           STW,11  BASE+X'25'
              00085     6AB00067           BAL,11  BASE+X'67'
              00086     22E08002 A         LI,14   32770
              00087     22600000           LI,6    I:OUTPUTONE
```

Figure 2. Sample (Partial) Object Listing

```
   F00 COBOL SOURCE, DIAGNOSTIC AND PROCEDURE-MAP LISTING  21:43 MAR 12, 1975

 00000                   COBOL LS,LO,XREF,DIAG,PMAP
 00001                   000010 IDENTIFICATION DIVISION.
 00002                   000020 PROGRAM-ID. SEQUENTIAL-I-O-TEST.
 00003                          AUTHOR. XEROX CORPORATION.
 00004                   000040 DATE-WRITTEN.  DECEMBER 7 1974.
 00005                   000050 ENVIRONMENT DIVISION.
 00006                   000060 CONFIGURATION SECTION.
 00007                          SOURCE-COMPUTER. XEROX-560.
 00008                          OBJECT-COMPUTER. XEROX-560.
 00009                   000090 INPUT-OUTPUT SECTION.
 00010                   000100 FILE-CONTROL.
 00011                   000110     SELECT INPUT-DEVICE ASSIGN TO CARD-READER.
 00012                   000120     SELECT OUTPUTONE       ASSIGN TO MAGNETIC-TAPE.
 00013                   000130     SELECT OUTPUTTWO ASSIGN TO DISC.
 00014                   000140     SELECT OUTPUTHREE ASSIGN TO PRINTER.
 00015                   000145     SELECT OPTIONAL OP-FILE ASSIGN TO MAGNETIC-TAPE RESERVE 2
 00016                   000146       ALTERNATE AREAS.
 00017                   000147     SELECT ERROR-FILE ASSIGN TO
 00018                   000150 DATA DIVISION.
**** 022 **** NAME INVALID/OMITTED
**** 049 **** SYNTACTICAL ERROR
 00019                   000160 FILE SECTION.
 00020                   000170 FD INPUT-DEVICE LABEL RECORD OMITTED DATA RECORD INP.
 00021                   000180   01 INP PICTURE X(80).
 00022                   000190 FD OUTPUTONE LABEL RECORD STANDARD DATA RECORD OUT1.
 00023                   000210   01 OUT1 PICTURE X(80).
 00024                   000220 FD OUTPUTTWO LABEL RECORD STANDARD DATA RECORD OUT2.
 00025                   000240   01 OUT2 PICTURE X(80).
 00026                   000250 FD OUTPUTHREE LABEL RECORD OMITTED DATA RECORD OUT3.
 00027                   000260   01 OUT3 PICTURE X(80).
 00028                   000261 FD  OP-FILE LABEL RECORD IS STANDARD DATA RECORD IS OP-REC.
 00029                   000262 01  OP-REC PICTURE X(80).
 00030                   000263 FD  ERROR-FILE LABEL RECORD IS DATUM  DATA RECORD IS ERROR-REC.
 00031                   000264 01  ERROR-REC PICTURE X(80).
 00032                   000265 01  DATUM COPY LIB1.
 00032.00001                    01  DATUM.
 00032.00002                        02  DATA-0  PICTURE X.
 00032.00003                        02  DATA-1  PICTURE 9(5).
 00033                   000267 WORKING-STORAGE SECTION.
 00034                   000268   . 77 DATA-2 PICTURE 9(5) VALUE 123456.
**** 107 **** VALUE TRUNCATED ON LEFT
 00035                   000270 PROCEDURE DIVISION.
 00036                   000271 DECLARATIVES.
 00037                   000272 SEC-1 SECTION. USE AFTER STANDARD ERROR PROCEDURE ON ERROR-FILE.
 00038         0005E     000273 P1. DISPLAY ERROR-REC.
 00039                   000274 S2 SECTION. USE BEFORE BEGINNING FILE LABEL PROCEDURE ON OUTPUT.
 00040         00067     000275 P1. MOVE DATA-2 TO DATA-1.  MOVE ' ' TO DATA-0.
 00041                   000276 S3 SECTION. USE AFTER BEGINNING FILE LABEL PROCEDURE ON INPUT.
 00042         00071     000277 P1. IF DATA-1 = DATA-2 MOVE ' TEST TO READ AND CHECK USER LABEL
 00043         00077     000278-     'SUCCESS' TO OUT3 ELSE EXHIBIT NAMED DATA-1 DATA-2.
 00044                   000279 END DECLARATIVES.
 00045                   000280 SEC-4 SECTION.
 00046                   000281 START.
 00047         00080     000290     OPEN  INPUT INPUT-DEVICE.
 00048         00083     000300     OPEN OUTPUT OUTPUTONE, OUTPUTTWO,
 00049                   000310     OUTPUTHREE, ERROR-FILE.
 00050         0009B     000320     MOVE ' BEGIN SEQUENTIAL IO TEST ',TO OUT3.
**** 002 **** INCORRECT PUNCTUATION
**** 002 **** INCORRECT PUNCTUATION
 00051         000A2     000330     WRITE OUT3.
 00052                   000331 CHECK-USE-VERB-FORMAT-2.
 00053         000A7     000332     MOVE ' TEST TO READ AND CHECK USER LABEL FAILURE' TO OUT3.
 00054         000AE     000333     MOVE ' THIS IS RECORD 1' TO ERROR-REC.
 00055         000B5     000334     WRITE ERROR-REC CLOSE ERROR-FILE.
 00056         000C3     000335     ADD 5 TO DATA-1 OPEN INPUT ERROR-FILE. WRITE OUT3.
 00057         000DB     000336 P1. READ ERROR-FILE INTO OUT3 AT END GO TO GET-FIRST-INPUT.
 00058         000E7     000337     WRITE OUT3. GO TO P1.
 00059                   000340 GET-FIRST-INPUT.
 00060         000ED     000350     READ INPUT-DEVICE AT END GO TO CLOSE-INITIAL-INPUT.
 00061                   000360     WRITE OUT1 FROM INPUT.
**** 269 **** IDENTIFIER MISSING AFTER 'FROM'
 00062         000F2     000370     GO TO GET-FIRST-INPUT.
 00063                   000380 CLOSE-INITIAL-INPUT.
 00064         000F3     000390     CLOSE INPUT-DEVICE, OUTPUTONE.
 00065         000F9     000400     OPEN INPUT OUTPUTONE.
 00066                   000410 GET-SECOND-INPUT.
 00067         000FF     000420     READ OUTPUTONE       AT END GO TO CLOSE-SECOND-INPUT.
```

Figure 3. Sample Source Program and Procedure Division Map Listing

```
 00068        00104   000430     WRITE OUT2 FROM OUT1.
 00069        0010C   000440     GO TO GET-SECOND-INPUT.
 00070                000450 CLOSE-SECOND-INPUT.
 00071        0010D   000480     CLOSE OUTPUTONE, AND OUTPUTTWO.
**** 049 **** SYNTACTICAL ERROR
 00072        00110   000490     OPEN INPUT OUTPUTONE, OUTPUTTWO.
 00073                000500 COMPARE-RECORDS.
 00074        0011C   000510     READ OUTPUTONE        AT END GO TO TERMINAT.
**** 234 **** UNDEFINED PROCEDURE NAME - EXTERNAL REFERENCE GENERATED
 00075        00121   000520     READ OUTPUTTWO        AT END GO TO ERR.
 00076        00126   000530     IF OUT1 = OUT2 GO TO COMPARE-RECORDS.
 00077        0012B   000540     MOVE '  RECORD MISMATCH ' TO OUT3.
 00078        00132   000550     WRITE OUT3.
 00079        00137   000560     WRITE OUT3 FROM OUT1.
 00080        0013F   000570     WRITE OUT3 FROM OUT2.
 00081                000580     GOTO COMPARE-RECORDS.
**** 049 **** SYNTACTICAL ERROR
 00082                000590 ERR.
 00083        00147   000600     MOVE ' PREMATURE EOF ON DEVICE-2 ' TO OUT3.
 00084        0014E   000610     WRITE OUT3.
 00085                000620 TERMINATE.
**** 049 **** SYNTACTICAL ERROR
**** 003 **** AREA A VIOLATION
 00086        00153   000460     MOVE ' END SEQUENTIAL IO TEST ' TO OUT3.
 00087        0015A   000470     WRITE OUT3.
 00088        0015F   000630     CLOSE OUTPUTONE, OUTPUTTWO, OUTPUTHREE.
**** 159 **** EXTERNAL REFERENCE GENERATED

*** NUMBER OF DIAGNOSTIC MESSAGES   12 ***     HIGHEST SEVERITY LEVEL    7 ***
```

Figure 3. Sample Source Program and Procedure Division Map Listing (cont.)

## MAIN (Main Program)

Two or more source programs can be compiled separately and their object modules combined to form a single executable program. The MAIN option specifies that the source program to be compiled is the main program; its inclusion on the COBOL processor control command is for commentary purposes only.

## MAPS (Both Data Division Map and Procedure Division Map)

This option specifies that both the Data Division Map and the Procedure Division Map are to be produced.

## PMAP (Procedure Division Map)

This option specifies that the Procedure Division Map is to be produced. This Map appears as part of the Source Program listing. The relative starting location of each sentence in the PROCEDURE DIVISION is listed following the associated source line number. Figure 3 illustrates a sample source program and Procedure Division Map listing.

## SEG (Priority Segments)

This option specifies that the source program to be compiled is a segmented program, i.e., it contains Priority Segments. This option must be specified if a segmented object program is desired; otherwise, a nonsegmented object program is produced.

## SEQCHK (Sequence Check)

This option specifies that the sequence number field (columns 1 through 6) of the source program lines is tested for ascending sequence. If an out-of-sequence condition occurs, the compiler issues the diagnostic "SOURCE PROGRAM OUT OF SEQUENCE".

## SO (Source Output)

This option allows the user to write his source program out to a keyed file. The keys used are compatible with the Edit processor. When using this option, an ASSIGN control command for the system DCB M:SO must be provided.

## SRTx (Co-Resident Sort)

This option specifies that the SORT verb will be compiled with the co-resident sort code and a tree structure generated by the compiler. The proper element files must be loaded at load time to ensure execution of this code. The COBOL object program and the Sort processor will be loaded together to form one load module, thus eliminating the need for the COBOL program to be swapped in and out. (Refer to "Co-Resident Sort Feature" in Chapter 5.) The x can be either an S or an R, indicating that the programmer desires the sequential (tape, mixed tape/disk) or the Random (disk only) sorting technique to be used.

## SUB (Subprogram)

This option specifies that the source program to be compiled is a subprogram. No "END start" address will be generated by the compiler.

## SYN (Syntax Checking)

This option provides only for syntactical checking of the COBOL source program; code generation is bypassed, thereby saving machine time. It is recommended that this option be used for preliminary compilations, as most of the errors in the source program are detected during this pass. For the final compilation (i.e., with the SYN option deleted), remaining errors are detected during code generation.

## TEST (On-Line Debugger)

This option specifies that the compiled program is to be tested using the on-line debugger. It causes the computer to create all necessary files and linkages for the on-line debugger.

## XREF (Cross-Reference Listing)

This option specifies that a cross-reference listing of the COBOL source program is to be produced on the LO device. All nonreserved words defined in the source program are listed in alphanumeric order. Shown to the left of each word is the source line number of the statement where the word is initially defined. To the right, overflowing if need be to lines following, are the line numbers of statements in which references are made to the words. Figure 4 shows a sample cross-reference listing.

```
       COBOL CROSS-REFERENCE LISTING                         21:43 MAR 12, 1975  PAGE   1

EXTERNAL     C:ERR                          00088
00052        CHECK-USE-VERB-FORMAT-2
00063        CLOSE-INITIAL-INPUT            00060
00070        CLOSE-SECOND-INPUT             00067
00073        COMPARE-RECORDS                00076
00032.00002  DATA-0                         00040
00032.00003  DATA-1                         00040       00042       00043       00056
00034        DATA-2                         00040       00042       00043
00032        DATUM
00082        ERR                            00075
00030        ERROR-FILE                     00037       00049       00055       00056       00057
00031        ERROR-REC                      00030       00038       00054       00055
00059        GET-FIRST-INPUT                00057       00062
00066        GET-SECOND-INPUT               00069
00021        INP                            00020
00020        INPUT-DEVICE                   00047       00060       00064
00028        OP-FILE
00029        OP-REC                         00028
00026        OUTPUTHREE                     00049       00088
00022        OUTPUTONE                      00048       00064       00065       00067       00071
                                            00072       00074       00088
00024        OUTPUTTWO                      00048       00072       00075       00088
00023        OUT1                           00022       00068       00076       00079
00025        OUT2                           00024       00068       00076       00080
00027        OUT3                           00026       00043       00050       00051       00053
                                            00056       00057       00058       00077       00078
                                            00079       00080       00083       00084       00086
                                            00087
```

Figure 4. Sample Cross-Reference Listing

# 3. INTER-PROGRAM COMMUNICATION

## Introduction

Any given COBOL source program may be subdivided into two or more parts, each of which can be compiled independently. One of these subdivisions must be designated as the main or calling program at both compilation and execution times. The remaining subdivisions are designated as subprograms or called programs. Each subdivision of the total program, whether the calling program or a called program, has the format of a complete COBOL source program. Each subdivision must contain IDENTIFICATION, ENVIRONMENT, DATA, and PROCEDURE DIVISIONs.

## Rules for Usage

Successful usage of the feature requires observance of two alternative sets of rules. The first set is somewhat restrictive, but requires a minimal knowledge of the contents of the calling program and its subprograms and thus is less susceptible to programmer error.

1. The ENVIRONMENT DIVISIONs must all be complete with regard to the total program, and should be identical.

2. The FILE SECTIONs and REPORT SECTIONs must all be complete with regard to the total program, and should be identical.

3. If the programmer wishes to have data referenced by both the main and subprograms he can do it in one of two ways:

   a. Provide a LINKAGE SECTION and a PROCEDURE DIVISION USING statement in the called program and a CALL statement in the calling program. The LINKAGE SECTION will reference WORKING STORAGE items in the main program.

   b. Provide an identical COMMON-STORAGE SECTION in both the main and subprograms. Items in COMMON-STORAGE can then be referred to by both programs and can also be used as parameters in the ENTER statement when calling a Metasymbol or FORTRAN subprogram.

   See the Xerox ANS COBOL/LN Reference Manual, 90 15 00, Chapter 10, for more detailed information on this subject.

4. The PROCEDURE DIVISION of the calling program must contain all DECLARATIVES sections desired in the total program.

The second set of rules requires a careful and detailed analysis of the individual source programs but permit omission of repetitious entries, thus reducing the size of the programs and improving compilation time.

1. ENVIRONMENT DIVISION

   a. Calling Program

      The complete ENVIRONMENT DIVISION for the total program must be written.

   b. Subprograms

      Each subprogram must contain SELECT sentences only for those files referenced in its PROCEDURE DIVISION (and described in its DATA DIVISION).

2. DATA DIVISION

a. FILE SECTION

(1) Calling Program

The file and record descriptions for all files in the total program must be included.

(2) Subprograms

The file and record descriptions for all files referenced in the PROCEDURE DIVISION (and mentioned in an ENVIRONMENT DIVISION SELECT sentence) must be included.

b. REPORT SECTION

(1) Calling Program

The report descriptions of all reports used in the total program must appear.

(2) Subprograms

Each subprogram must contain only the descriptions of reports actually referenced therein. (The file description of the file containing the associated REPORT IS clause must also be present.)

Memory space is allocated and Data Control Blocks generated for the files described in the FILE SECTION of the main program. All subprograms making reference to reports or report data, when incorporated into the total program at run-time, refer to the areas reserved by the main program. Similarly, memory space is assigned in accordance with the COMMON-STORAGE SECTION description in the main program, and this area is shared by the main program and all associated subprograms when combined at run-time. The main program and each subprogram may have its own WORKING-STORAGE SECTION; data described therein is not shared, but is private to the program in which it is defined. However, WORKING-STORAGE items in a main program may be referred to in a subprogram by the use of a LINKAGE SECTION in the subprogram and the PROCEDURE DIVISION USING statement as mentioned previously in paragraph 3a above.

Program control can flow naturally between independent compilations employing the normal COBOL verbs GO TO and PERFORM. Only one additional statement is introduced into the Xerox ANS COBOL language to provide this natural flow. Any procedure point to which control may be passed by a separately compiled program must be declared as an external definition. The ENTER COBOL statement names those entry points (section- and paragraph-names) within the program that are to be visible to sequence control statements in other compilations.

# 4. OBJECT PROGRAM

The object program produced by the COBOL compiler is in Xerox standard object language format. It is output via the M:BO and/or M:GO system Data Control Blocks (DCBs) as directed by the options expressed in the COBOL control command. The compiler assumes either that the appropriate DCB has been pre-conditioned by ASSIGN commands to reflect the media on which the object program is to appear and the file-name(s) under which the object module is to be cataloged, or that those options have been deliberately permitted to default to the standard system conditions.

## Segmented Object Programs

A single COBOL source program can be so large that its object-time storage requirements exceed available computer memory. When such a situation occurs, the program may be partitioned into logical blocks called "overlays" or, in COBOL terminology, "Priority Segments". The logical structure of a program segmented in this manner resembles a simple tree. COBOL object programs employ the branch reference loading mode: each overlay is loaded into core storage when control reaches a reference to it during execution of the root or another overlay segment of the program. The SEG option must be specified as a COBOL control command option in order to produce a segmented program.

During compilation of a segmented program, only the root segment module is output via the M:BO and/or M:GO DCBs under the file-name contained in the DCB. It is possible to create permanent relocatable object modules (ROMs) by assigning the M:BO DCB to a file. The root and each overlay will then be output as permanent ROMs, as described below. Since the computer uses the M:LI DCB to write out the overlay segments for the GO file, the M:LI assignment should not be protected with a password. If it is, the user cannot access the overlays.

Overlay segment modules are output in individual files on disk; they are identified by the root segment module name with a two-digit suffix. For example, if the root segment name is OBJPROG, overlay segments are named OBJPROG01, OBJPROG02, OBJPROG03, and so on. In addition, the tree structure is specified in a TREE control command image, which is output on disk in a file that is also identified by the root segment module name with two zeros added, e.g., OBJPROG00. No tree structure is created for the BO output.

The TREE control command may be punched out by the PCL control command

```
COPY DC/OBJPROG00 TO CP
```

Similarly, each overlay segment module may be punched by the control command

```
COPY DC/OBJPROGnn TO CP(BIN)
```

where nn is the two-digit identifier for the segment, as explained above. (See "Segmentation Feature" in Chapter 3.)

## Object Program Structure

The object program is produced in one or more modules: one module comprises the entire program except for priority segments; one additional module is created for each priority-number used between the specified SEGMENT-LIMIT (or 50) and 99. A priority segment (overlay) module consists of a single (standard) control section and contains only the procedure code and literals of the relevant PROCEDURE DIVISION sections. The root segment module comprises multiple control sections. Figure 5 illustrates arrangement of the standard control section of the root segment module.

The root segment module may also contain a number of dummy program sections, which are created in the following instances:

1. COMMON-STORAGE SECTION. The COMMON-STORAGE SECTION of the DATA DIVISION of the source program is represented in the object program by a dummy program section whose name is supplied by the CS(name) control command option. If the CS(name) option is not specified, the name TALLY is used. The Special Register TALLY is generated as the initial entry in the COMMON-STORAGE dummy section produced by each COBOL compilation.

2. DCBs. A DCB (Data Control Block) appears in the object program for each file declared in the source program by a File Description (FD) file-name entry in the FILE SECTION of the DATA DIVISION, and is output as a dummy program section named F:file-name.

3. File Record Areas. A record area the size of the largest record defined is reserved for each file declared by an FD entry, and is represented in the object program by a dummy program section named file-name.

4. File Index Areas. Each file declared in the source program has associated with it one additional dummy program section named I:file-name, wherein five words are assigned for file control purposes. One additional word is allocated for each index-name mentioned in INDEXED BY options of the OCCURS clause in record descriptions pertaining to the file.

5. Report Table Area. Each report declared by a Record Description (RD) report-name entry in the REPORT SECTION of the DATA DIVISION of the source program is described in the object program by a table that is produced as a dummy program section named R:report-name.

6. Report Storage Area. The print lines, accumulators, control fields, and other data storage associated with each report are represented in the object program by a dummy program section named report-name.

7. WORKING-STORAGE SECTION. The WORKING-STORAGE SECTION in the DATA DIVISION has a definition (DEF) associated with it that indicates its beginning location. This definition is labeled DEF$WK.

These dummy program sections are illustrated by the load module map in Figure 6. Circled numbers are keyed to the itemized discussion above. Note that the map does not correlate with the sample object listing shown in Figure 2.

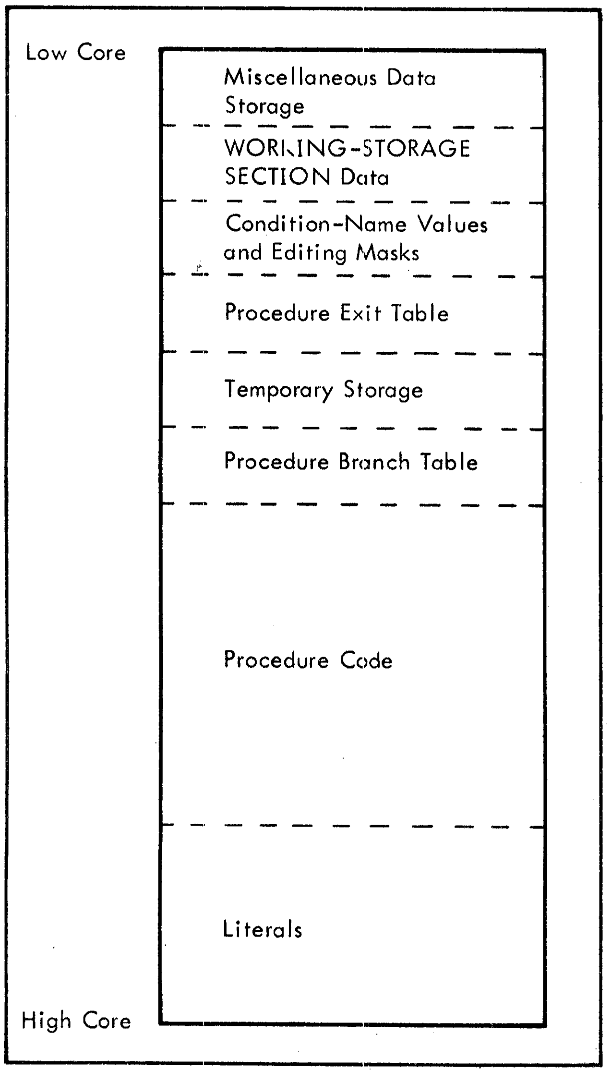
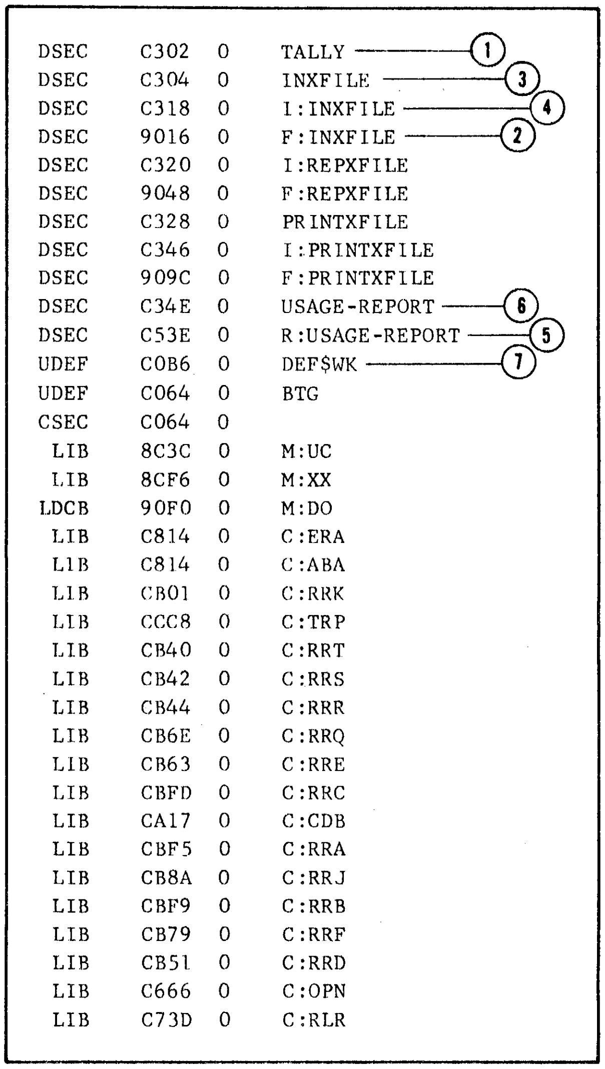


Figure 5. Standard Control Section of a Root Segment Module



```
DSEC   C302  0   TALLY ──────────①
DSEC   C304  0   INXFILE ───────────③
DSEC   C318  0   I:INXFILE ───────────④
DSEC   9016  0   F:INXFILE ─────────②
DSEC   C320  0   I:REPXFILE
DSEC   9048  0   F:REPXFILE
DSEC   C328  0   PRINTXFILE
DSEC   C346  0   I:PRINTXFILE
DSEC   909C  0   F:PRINTXFILE
DSEC   C34E  0   USAGE-REPORT ──────⑥
DSEC   C53E  0   R:USAGE-REPORT ──────⑤
UDEF   C0B6  0   DEF$WK ────────────⑦
UDEF   C064  0   BTG
CSEC   C064  0
 LIB   8C3C  0   M:UC
 LIB   8CF6  0   M:XX
LDCB   90F0  0   M:DO
 LIB   C814  0   C:ERA
 LIB   C814  0   C:ABA
 LIB   CB01  0   C:RRK
 LIB   CCC8  0   C:TRP
 LIB   CB40  0   C:RRT
 LIB   CB42  0   C:RRS
 LIB   CB44  0   C:RRR
 LIB   CB6E  0   C:RRQ
 LIB   CB63  0   C:RRE
 LIB   CBFD  0   C:RRC
 LIB   CA17  0   C:CDB
 LIB   CBF5  0   C:RRA
 LIB   CB8A  0   C:RRJ
 LIB   CBF9  0   C:RRB
 LIB   CB79  0   C:RRF
 LIB   CB51  0   C:RRD
 LIB   C666  0   C:OPN
 LIB   C73D  0   C:RLR
```

Figure 6. Load Module Map

# 5. PROGRAM COMPILATION AND EXECUTION

## Compilation of Large Source Programs

It is recommended that the following two monitor control commands be used for compilation of large source programs.

1. LIMIT control command

   Compilation of large source programs requires a large amount of temporary disk storage. For this reason the TSTORE option should be specified to allow the use of additional available disk storage.

   Example: !LIMIT (TSTORE, 2000)

   Also, temporary disk storage can be conserved by specifying that the compiler copy of the source program (from which the source listing is built) be saved on magnetic tape rather than on disk. The following control command permits this alternative assignment:

   ```
   !ASSIGN  F:W7, (LABEL, name), (SN, value), (OUTIN)
   ```

   where

   | | |
   |---|---|
   | name | specifies the name of the file. |
   | value | specifies the serial number of the tape reel to be used. |

2. POOL control command

   Compilation speed can be improved significantly by specifying additional buffers for use by the monitor.

   Example: !POOL (FPOOL, 8), (IPOOL, 8)

Examples of both the LIMIT and POOL control commands are presented in the deck setup in Figure 8.

## COBOL Work Files

The COBOL compiler uses 11 work files having the DCB names F:W0 through F:W10. To avoid confusion, the COBOL source program should not use those DCB names. In any event, it is good practice to place the !ASSIGN cards for user files after the !COBOL card and source deck, as illustrated throughout this manual.

## COBOL Library on Tape

Normally, library files are stored on disk. It is possible, however, to have library files on labeled tape. In this case, an ASSIGN control command for M:LI must be specified. For example,

```
!ASSIGN M:LI, (LABEL, name, account), (SN, value)
```

where

| | |
|---|---|
| name | specifies the name of a labeled file. |
| account | specifies the account under which the tape was created. |
| value | specifies the serial number of the tape reel to be used. |

## Print File Handling

If the BEFORE and/or AFTER ADVANCING clause is used in a COBOL source program, the data control block must indicate that the first position of the record is to be treated as a vertical-format-control character. If the file was assigned to the printer in the COBOL source program, the DCB will be pre-set with the VFC option. This eliminates the need for an ASSIGN control command.

If the ADVANCING option is not specified in a WRITE instruction addressed to a print file, the user is assumed to have indicated the vertical-format-control character at the source program level and stored this chapter in the first byte of the record to be printed. The action indicated by the control character is performed, and then the record is printed. The codes controlling the vertical format on the Xerox Buffered Line Printers, Models 7440/7445 and 7446, are shown in Table 1.

Table 1. Xerox Buffered Line Printers, Models 7440/7445 and 7446, Vertical-Format Control Codes

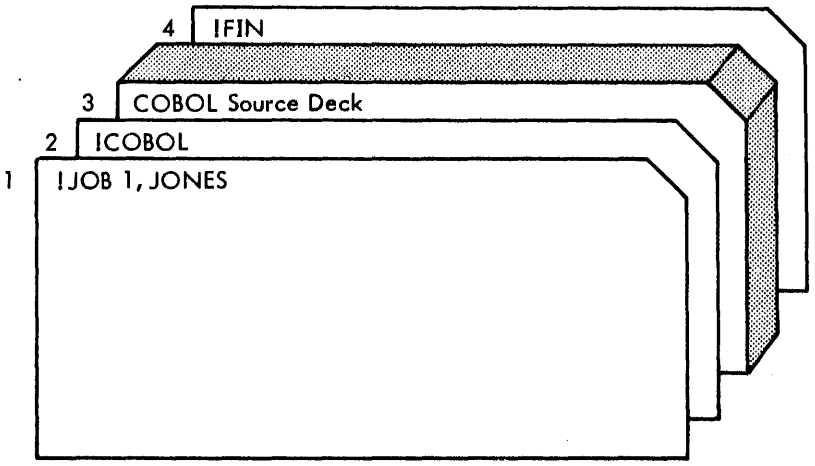
| Code (Hexadecimal) | Meaning |
|---|---|
| 40 | Print, single space. |
| 60 | Print, inhibit automatic upspace after printing. |
| C0 | Print, single space (same as 40). |
| C1 | Single space, print, single space. |
| C2 | Space 2 lines, print, single space. |
| C3 | Space 3 lines, print, single space. |
| ⋮ | ⋮ |
| CF | Space 15 lines, print, single space. |
| D0[t] | Skip to channel 0 (bottom of page), print, inhibit automatic upspace. |
| D1[t] | Skip to channel 1 (top of page), print, inhibit automatic upspace. |
| D2[t] | Skip to channel 2, print, inhibit automatic upspace. |
| D3[t] | Skip to channel 3, print, inhibit automatic upspace. |
| ⋮ | ⋮ |
| D7[t] | Skip to channel 7, print, inhibit automatic upspace. |
| E0 | Print, inhibit automatic upspace after printing (same as 60). |
| E1[t] | Space 1 line, print, inhibit automatic upspace after printing. |
| E2[t] | Space 2 lines, print, inhibit automatic upspace after printing. |
| ⋮ | ⋮ |
| EF[t] | Space 15 lines, print, inhibit automatic upspace after printing. |
| F0 | Skip to channel 0 (bottom of page), print, single space. |
| F1 | Skip to channel 1 ( top of page), print, single space. |
| F2 | Skip to channel 2, print, single space. |
| ⋮ | ⋮ |
| F7 | Skip to channel 7, print, single space. |

[t]Model 7446 only.

## Deck Structures

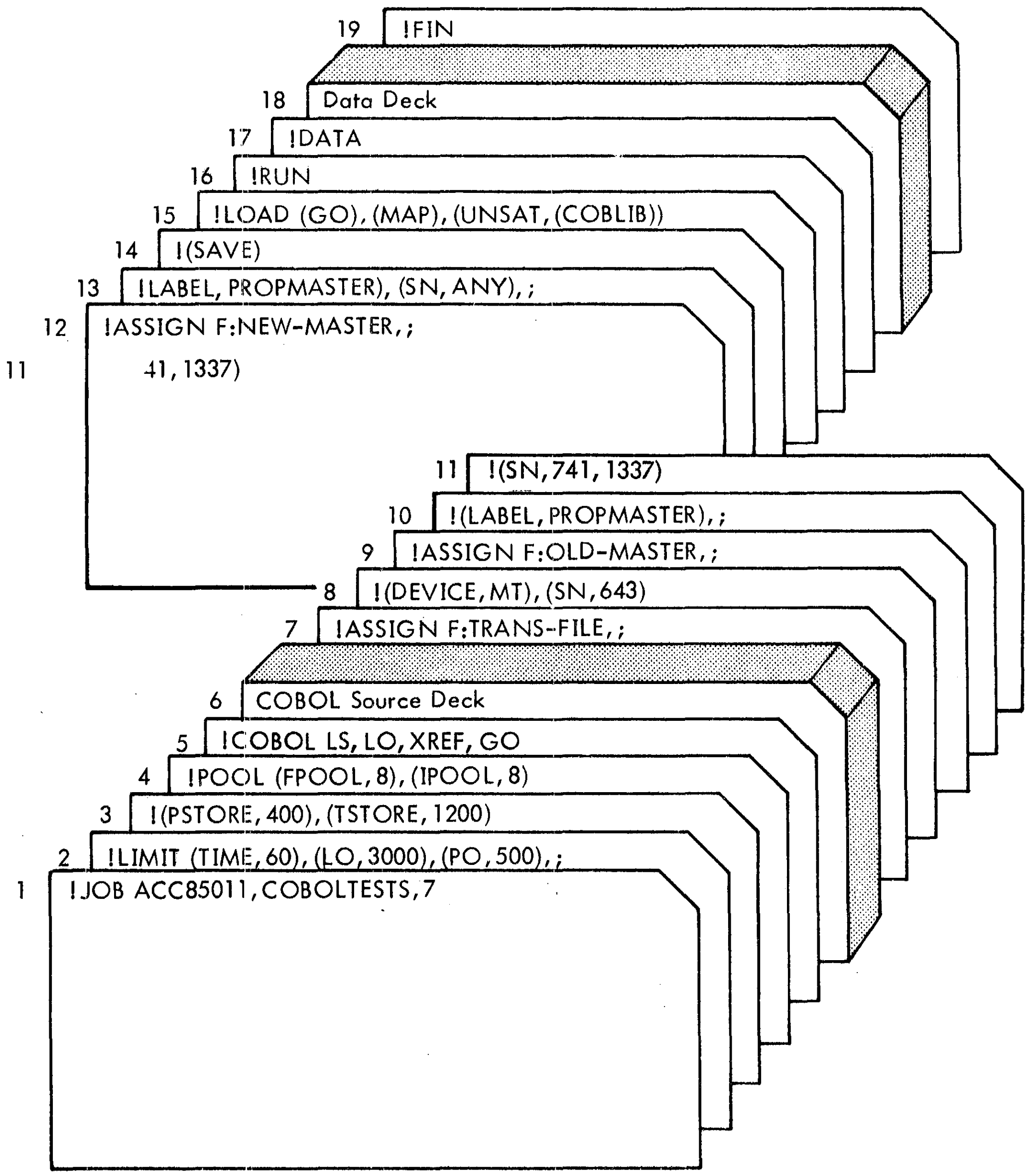
### Basic Setups

Figures 7, 8, and 9 show some of the ways in which COBOL program decks may be prepared for COBOL compilation and execution.



| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
| | 1 | Account number. |
| | JONES | Identifies the user. |
| 2 | !COBOL | Specifies that control is to be transferred to the COBOL processor. |
| 3 | | COBOL source deck. |
| 4 | !FIN | Signals the end of the job stack. |

Figure 7. Basic Setup – Compilation Only



```
19  !FIN
18  Data Deck
17  !DATA
16  !RUN
15  !LOAD (GO),(MAP),(UNSAT,(COBLIB))
14  !(SAVE)
13  !LABEL, PROPMASTER),(SN,ANY),;
12  !ASSIGN F:NEW-MASTER,;
11      41,1337)
11  !(SN,741,1337)
10  !(LABEL,PROPMASTER),;
9   !ASSIGN F:OLD-MASTER,;
8   !(DEVICE,MT),(SN,643)
7   !ASSIGN F:TRANS-FILE,;
6   COBOL Source Deck
5   !COBOL LS,LO,XREF,GO
4   !POOL (FPOOL,8),(IPOOL,8)
3   !(PSTORE,400),(TSTORE,1200)
2   !LIMIT (TIME,60),(LO,3000),(PO,500),;
1   !JOB ACC85011,COBOLTESTS,7
```

| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
| | ACC85011 | Account number. |
| | COBOLTESTS | Identifies the user. |
| | 7 | Indicates job priority. |
| 2 | !LIMIT | Control command that specifies the maximum values for various system resources used by the job. |
| | (TIME,60) | Specifies limit of 60 minutes execution time. |

Figure 8. Basic Setup – Compilation and Execution

| Card | Parameter | Description |
|---|---|---|
| | (LO,3000) | Specifies limit of 3000 pages listing output. |
| | (PO,500) | Specifies limit of 500 cards punch output. |
| | ; | Signals that this LIMIT command is continued on the following card. |
| 3 | !(PSTORE,400) | Specifies limit of 400 granules permanent disk storage. (This card is part of the preceding LIMIT command.) |
| | (TSTORE,1200) | Specifies limit of 1200 granules temporary disk storage. |
| 4 | !POOL | Control command that specifies additional buffers for use by the monitor. |
| | (FPOOL,8) | Specifies that 8 buffers are to be assigned to file management use. |
| | (IPOOL,8) | Specifies that 8 buffers are to be assigned to the file index pool. |
| 5 | !COBOL | Control command that specifies control is to be transferred to the COBOL processor. |
| | LS | Specifies that the source program is to be listed. |
| | LO | Specifies that the object program is to be listed. |
| | XREF | Specifies that the cross-reference listing is to be produced. |
| | GO | Specifies that the program is to be executed after compilation. |
| 6 | | COBOL source deck. |
| 7 | !ASSIGN | Control command that specifies the file and physical peripheral device to be used. |
| | F:TRANS-FILE | DCB name of TRANS-FILE file. |
| | ; | Signals that this ASSIGN command is continued on the following card. |
| 8 | !(DEVICE,MT) | Specifies that the file is to be assigned to a magnetic tape unit. (This card is part of the preceding ASSIGN command.) |
| | (SN,643) | Specifies that the input file is contained on reel number 643. |

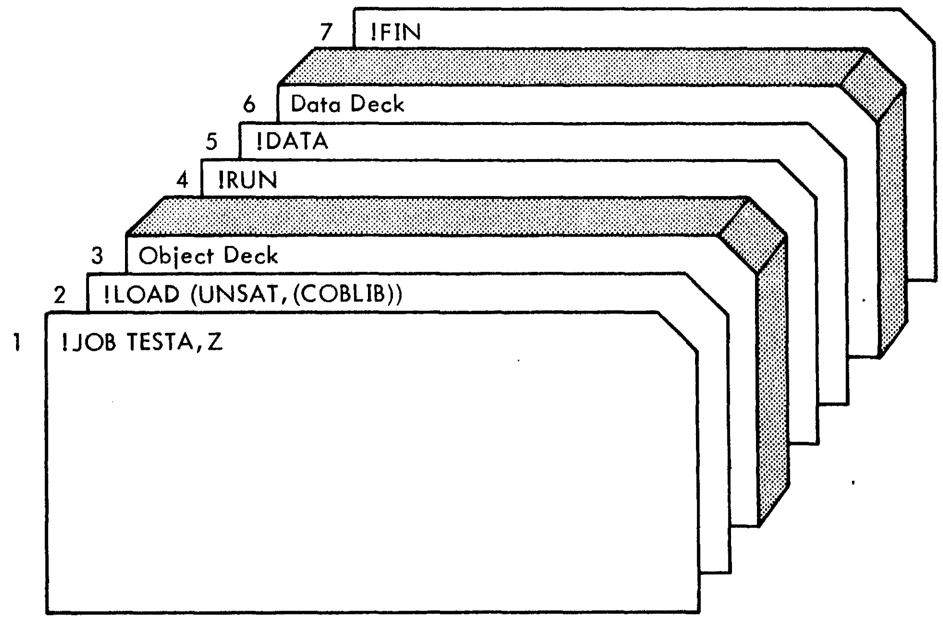
Figure 8. Basic Setup – Compilation and Execution (cont.)

| Card | Parameter | Description |
|---|---|---|
| 9 | !ASSIGN | Control command that specifies the file and physical peripheral device to be used. |
| | F:OLD-MASTER | DCB name of OLD-MASTER file. |
| | ; | Signals that this ASSIGN command is continued on the following card. |
| 10 | !(LABEL, PROPMASTER) | Specifies that the file is named PROPMASTER. (This card is part of the preceding ASSIGN command.) |
| | ; | Signals that this ASSIGN command is continued on the following card. |
| 11 | !(SN, 741, 1337) | Specifies that the input file is contained on two reels, numbers 741 and 1337. (This card is part of the preceding ASSIGN command.) |
| 12 | !ASSIGN | Control command that specifies the file and physical peripheral device to be used. |
| | F:NEW-MASTER | DCB name of NEW-MASTER file. |
| | ; | Signals that this ASSIGN command is continued on the following card. |
| 13 | !(LABEL, PROPMASTER) | Specifies that the output file is to be named PROPMASTER. (This card is part of the preceding ASSIGN command.) |
| | (SN, ANY) | Specifies that the output file is to be written on any available reel. |
| | ; | Signals that this ASSIGN command is continued on the following card. |
| 14 | !(SAVE) | Specifies that the file is to be saved. (This card is part of the preceding ASSIGN command.) |
| 15 | !LOAD | Control command that directs the resident loader to form a relocatable load module. |
| | (GO) | Specifes that data from the user temporary GO file is to be included in the root of the load module. |
| | (MAP) | Specifies that all external references and definitions for the load module are to be listed. |
| | (UNSAT, (COBLIB)) | Specifies that the COBOL library (identified by account number COBLIB) is to be searched for external definitions required for the load module. |

Figure 8. Basic Setup – Compilation and Execution (cont.)

| Card | Parameter | Description |
|---|---|---|
| 16 | !RUN | Control command that specifies the program is to be executed. |
| 17 | !DATA | Control command that specifies a data deck is to follow. |
| 18 | | Data deck. |
| 19 | !FIN | Signals the end of the job stack. |

Figure 8. Basic Setup – Compilation and Execution (cont.)



| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
| | TESTA | Account number. |
| | Z | Identifies the user. |
| 2 | !LOAD | Control command that directs the resident loader to form a relocatable load module. |
| | (UNSAT,(COBLIB)) | Specifies that the COBOL library (identified by account number COBLIB) is to be searched for external definitions required for the load module. |
| 3 | | Object deck. |
| 4 | !RUN | Control command that specifies the program is to be executed. |

Figure 9. Basic Setup – Execution with Object Deck

| Card | Parameter | Description |
|---|---|---|
| 5 | !DATA | Control command that specifies a data deck is to follow. |
| 6 | | Data deck. |
| 7 | !FIN | Signals the end of the job stack. |

Figure 9. Basic Setup – Execution with Object Deck (cont.)

## Segmentation Feature

To combine segmented programs into a single executable program, the desired overlay structure must be communicated to the loader. This may be done in the usual way by a TREE control command or semiautomatically by a PTREE command, which references the files containing the TREE commands generated by individual compilations. Refer to "Segmented Object Programs" in Chapter 2.

Figures 10, 11, and 12 show how a COBOL program with priority segments is set up for compilation and execution.

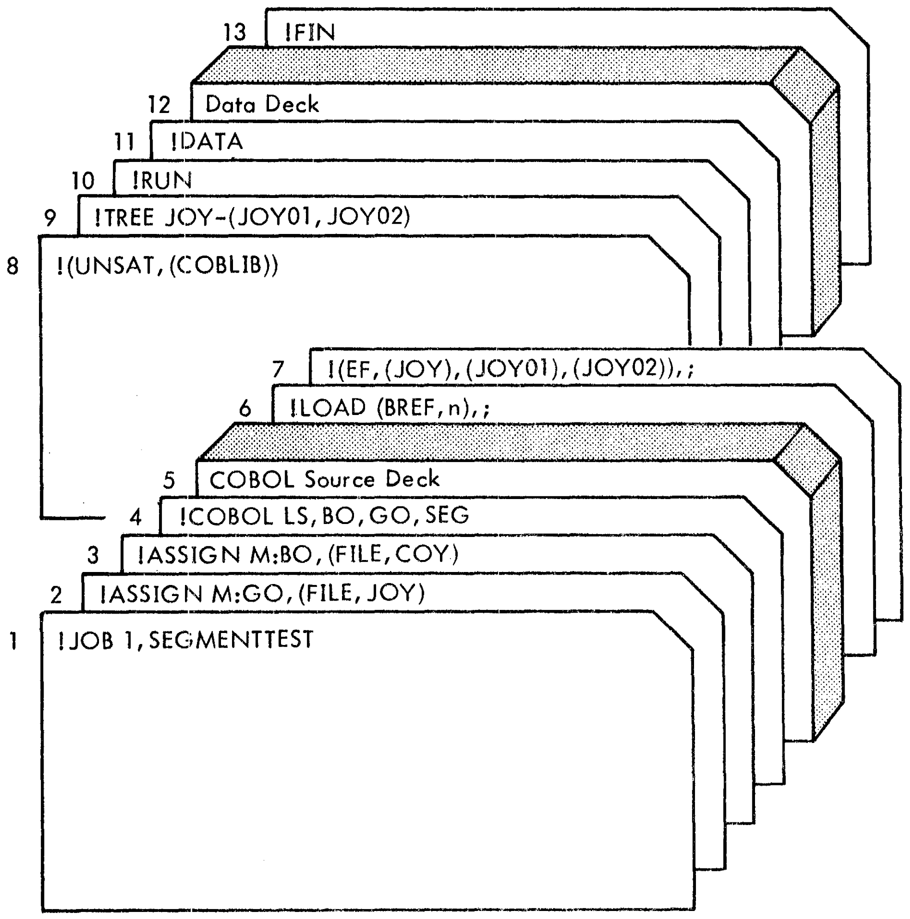


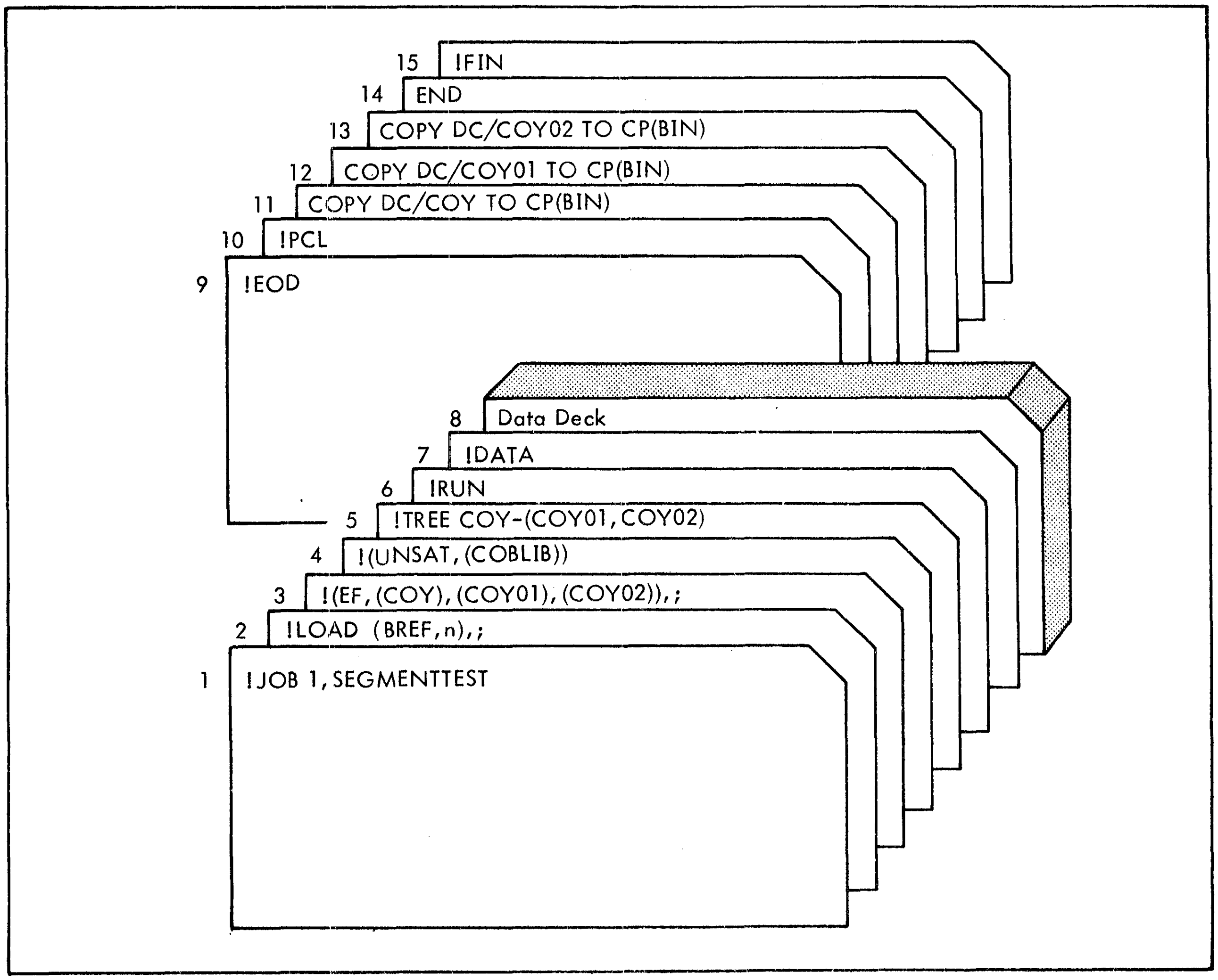
Figure 10. Segmentation Feature – Compilation and Execution

| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
| | 1 | Account number. |
| | SEGMENTTEST | Identifies the user. |
| 2 | !ASSIGN | Control command that specifies the file and physical peripheral device to be used. |
| | M:GO | The file is the system GO file. |
| | (FILE, JOY) | Specifies that the file is to be a disk file named JOY. |
| 3 | !ASSIGN | Control command that specifies the file and physical peripheral device to be used. |
| | M:BO | The file is the system BO file. |
| | (FILE, COY) | Specifies the file is to be a disk file named COY. |
| 4 | !COBOL | Control command that specifies control is to be transferred to the COBOL processor. |
| | LS | Specifies that the source program is to be listed. |
| | BO | Specifies that permanent relocatable object modules are to be produced. |
| | GO | Specifies that the program is to be executed after compilation. |
| | SEG | Specifies that the program contains priority segments. |
| 5 | | COBOL source deck. |
| 6 | !LOAD | Control command that directs the resident loader to form a relocatable load module. |
| | (BREF, n) | Specifies that the overlay structure is to be set up for the branch referencing loading mode. The parameter "n" (if present) is a decimal value specifying the maximum number of interbranch references within the program. If "n" is absent or zero, a total of 11 words per segment are reserved in the reference loading table (two words per reference). |
| | ; | Signals that this LOAD command is continued on the following card. |
| 7 | !(EF, (JOY), (JOY01), (JOY02)) | Specifies that the modules of the root segment (JOY) and the two overlay segments (JOY01 and JOY02) are to be included in the load module. (This card is part of the preceding LOAD command.) |
| | ; | Signals that the LOAD command is continued on the following card. |

Figure 10. Segmentation Feature – Compilation and Execution (cont.)

| Card | Parameter | Description |
|---|---|---|
| 8 | !(UNSAT, (COBLIB)) | Specifies that the COBOL library (identified by account number COBLIB) is to be searched for external definitions required for the load module. (This card is part of the preceding LOAD command.) |
| 9 | !TREE | Control command that specifies the overlay structure of the load module. |
|  | JOY-(JOY01, JOY02) | Specifies that module JOY is the root segment and modules JOY01 and JOY02 are overlay segments. |
| 10 | !RUN | Control command that specifies the program is to be executed. |
| 11 | !DATA | Control command that specifies a data deck is to follow. |
| 12 |  | Data deck. |
| 13 | !FIN | Signals the end of the job stack. |

Figure 10. Segmentation Feature – Compilation and Execution (cont.)



15 !FIN
14 END
13 COPY DC/COY02 TO CP(BIN)
12 COPY DC/COY01 TO CP(BIN)
11 COPY DC/COY TO CP(BIN)
10 !PCL
9 !EOD
8 Data Deck
7 !DATA
6 !RUN
5 !TREE COY-(COY01,COY02)
4 !(UNSAT, (COBLIB))
3 !(EF, (COY), (COY01), (COY02)),;
2 !LOAD (BREF,n),;
1 !JOB 1,SEGMENTTEST

Figure 11. Segmentation Feature – Load from BO File, Execute, and Punch BO File

| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
|  | 1 | Account number. |
|  | SEGMENTTEST | Identifies the user. |
| 2 | !LOAD | Control command that directs the resident loader to form a relocatable load module. |
|  | (BREF,n) | Specifies that the overlay structure is to be set up for the branch referencing loading mode. The parameter "n" (if present) is a decimal value specifying the maximum number of interbranch references within the program. If "n" is absent or zero, a total of 11 words per segment are reserved in the reference loading table (two words per reference). |
|  | ; | Signals that this LOAD command is continued on the following card. |
| 3 | !(EF,(COY),(COY01),(COY02)) | Specifies that the modules of the root segment (COY) and the two overlay segments (COY01 and COY02) are to be included in the load module. (This card is part of the preceding LOAD command.) |
|  | ; | Signals that the LOAD command is continued on the following card. |
| 4 | !(UNSAT,(COBLIB)) | Specifies that the COBOL library (identified by account number COBLIB) is to be searched for external definitions required for the load module. (This card is part of the preceding LOAD command.) |
| 5 | !TREE | Control command that specifies the overlay structure of the load module. |
|  | COY-(COY01,COY02) | Specifies that module COY is the root segment and modules COY01 and COY02 are overlay segments. |
| 6 | !RUN | Control command that specifies the program is to be executed. |
| 7 | !DATA | Control command that specifies a data deck is to follow. |
| 8 |  | Data deck. |
| 9 | !EOD | Defines the end of the data deck. |
| 10 | !PCL | Initiates the Peripheral Conversion Language (PCL) processor. |
| 11 | COPY DC/COY TO CP(BIN) | Punches a binary deck for root COY. |

Figure 11. Segmentation Feature – Load from BO File, Execute, and Punch BO File (cont.)

| Card | Parameter | Description |
|---|---|---|
| 12 | COPY DC/COY01 TO CP(BIN) | Punches a binary deck for first overlay segment. |
| 13 | COPY DC/COY02 TO CP(BIN) | Punches a binary deck for second overlay segment. |
| 14 | END | Terminates PCL operations. |
| 15 | !FIN | Signals the end of the job stack. |

Figure 11. Segmentation Feature – Load from BO File, Execute, and Punch BO File (cont.)

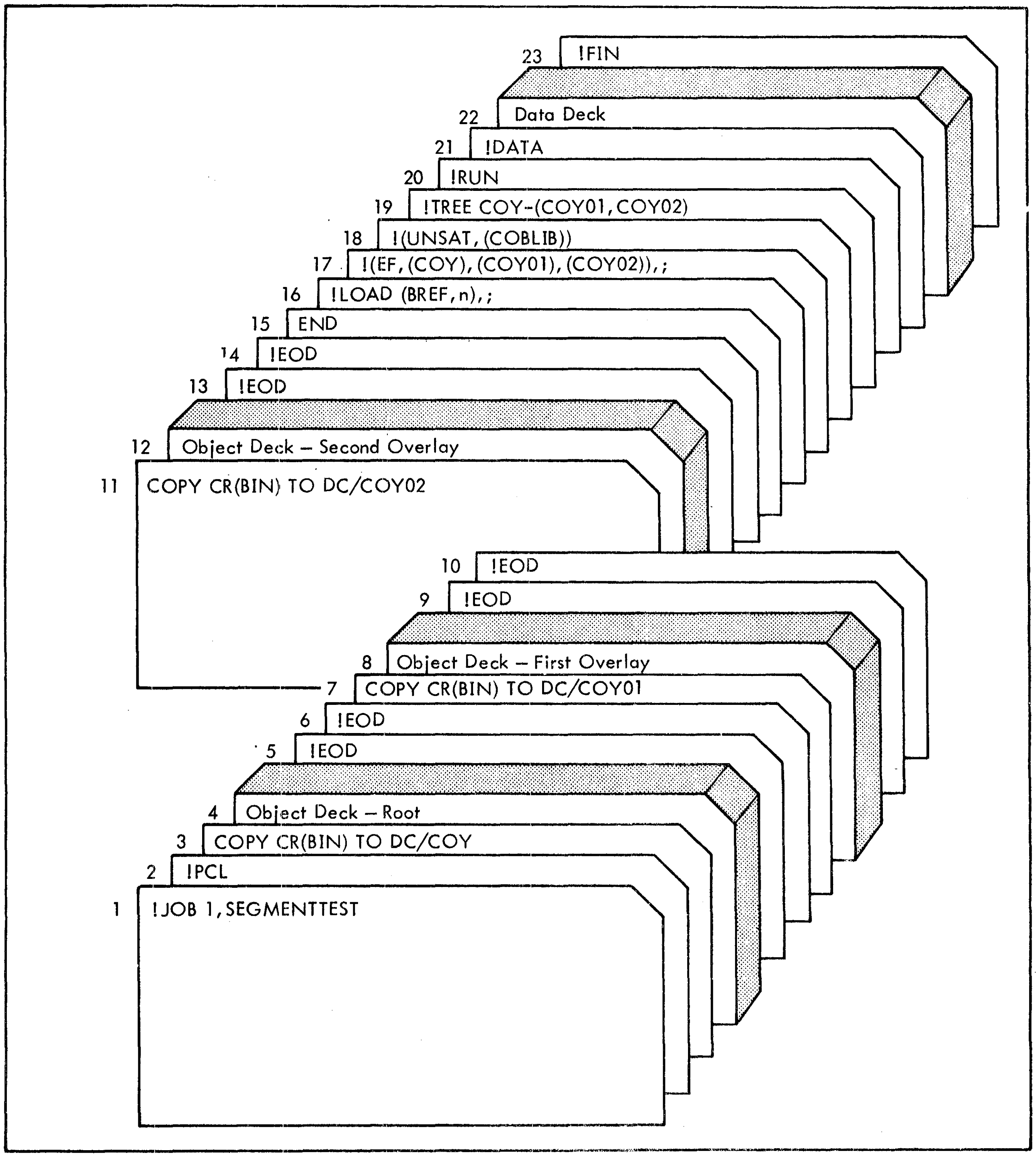


Figure 12. Segmentation Feature – Execution from Object Decks

| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
| | 1 | Account number. |
| | SEGMENTTEST | Identifies the user. |
| 2 | !PCL | Initiates the Peripheral Conversion Language (PCL) processor. |
| 3 | COPY CR(BIN) TO DC/COY | Copies object deck from card reader to disk file named COY. |
| 4 | | Object deck for root segment. |
| 5 | !EOD | Signals PCL of the end of the root segment card deck. |
| 6 | !EOD | |
| 7 | COPY CR(BIN) TO DC/COY01 | Copies object deck from card reader to disk file named COY01. |
| 8 | | Object deck for first overlay segment. |
| 9 | !EOD | Signals PCL of the end of the overlay segment card deck. |
| 10 | !EOD | |
| 11 | COPY CR(BIN) TO DC/COY02 | Copies object deck from card reader to disk file named COY02. |
| 12 | | Object deck for second overlay segment. |
| 13 | !EOD | Signals PCL of the end of the overlay segment card deck. |
| 14 | !EOD | |
| 15 | END | Terminates PCL operations. |
| 16 | !LOAD | Control command that directs the resident loader to form a relocatable load module. |
| | (BREF, n) | Specifies that the overlay structure is to be set up for the branch referencing loading module. The parameter "n" |

Figure 12. Segmentation Feature – Execution from Object Decks (cont.)

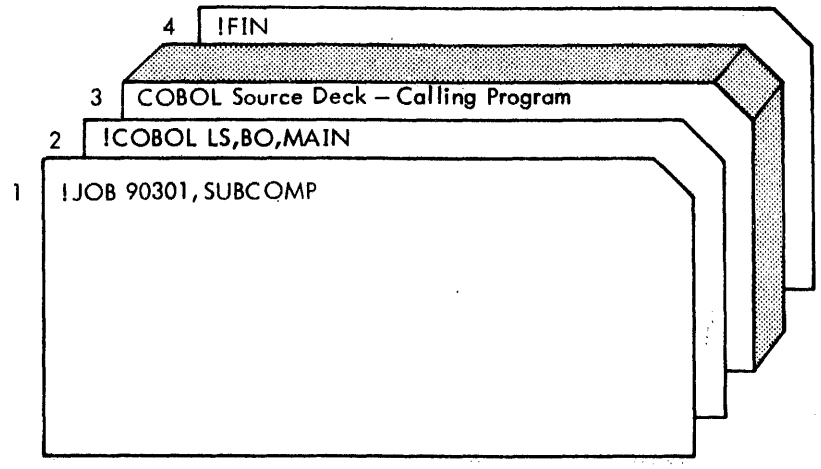
| Card | Parameter | Description |
|---|---|---|
| | (BREF,n) -(cont.) | (if present) is a decimal value specifying the maximum number of interbranch references within the program. If "n" is absent or zero, a total of 11 words per segment are reserved in the reference loading table (two words per reference). |
| | ; | Signals that this LOAD command is continued on the following card. |
| 17 | !(EF,COY),(COY01),COY02)) | Specifies that the modules of the root segment (COY) and the two overlay segments (COY01 and COY02) are to be included in the load module. (This card is part of the preceding LOAD command.) |
| | ; | Signals that the LOAD command is continued on the following card. |
| 18 | !(UNSAT,(COBLIB)) | Specifies that the COBOL library (identified by account number COBLIB) is to be searched for external definitions required for the load module. (This card is part of the preceding LOAD command.) |
| 19 | !TREE | Control command that specifies the overlay structure of the load module. |
| | COY-(COY01,COY02) | Specifies that module COY is the root segment and modules COY01 and COY02 are the overlay segments. |
| 20 | !RUN | Control command that specifies the program is to be executed. |
| 21 | !DATA | Control command that specifies a data deck is to follow. |
| 22 | | Data deck. |
| 23 | !FIN | Signals the end of the job stack. |

Figure 12. Segmentation Feature – Execution from Object Decks (cont.)

## Inter-Program Communication (Subcompile Feature)

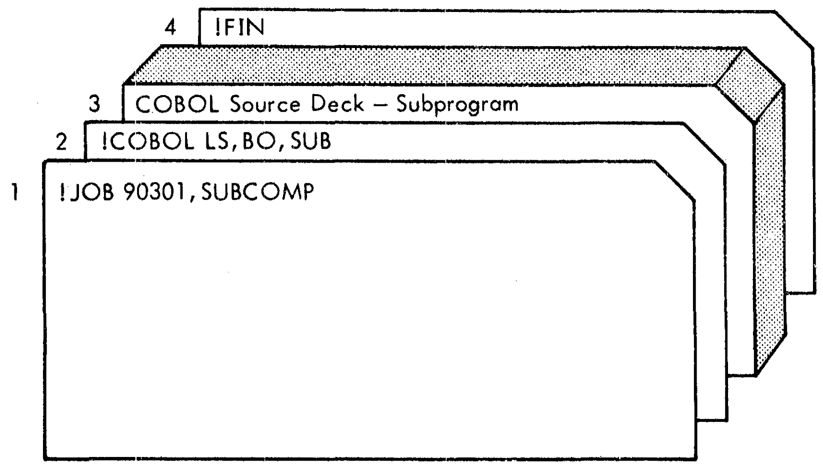
A single logical problem solution expressed in COBOL may be subdivided into two or more source programs that can be compiled separately and whose resultant object modules can be subsequently combined into a single executable program. Rules for such program subdivision are explained in Chapter 3. Briefly stated, one of the subdivisions must be designated as the main or calling program at both compilation and load times, and the remaining subdivisions must be denoted as subprograms or called programs (SUB option) at compilation time.

Figures 13, 14, 15, and 16 show how two COBOL programs are compiled separately and how the resultant object modules are then combined into a single executable program.



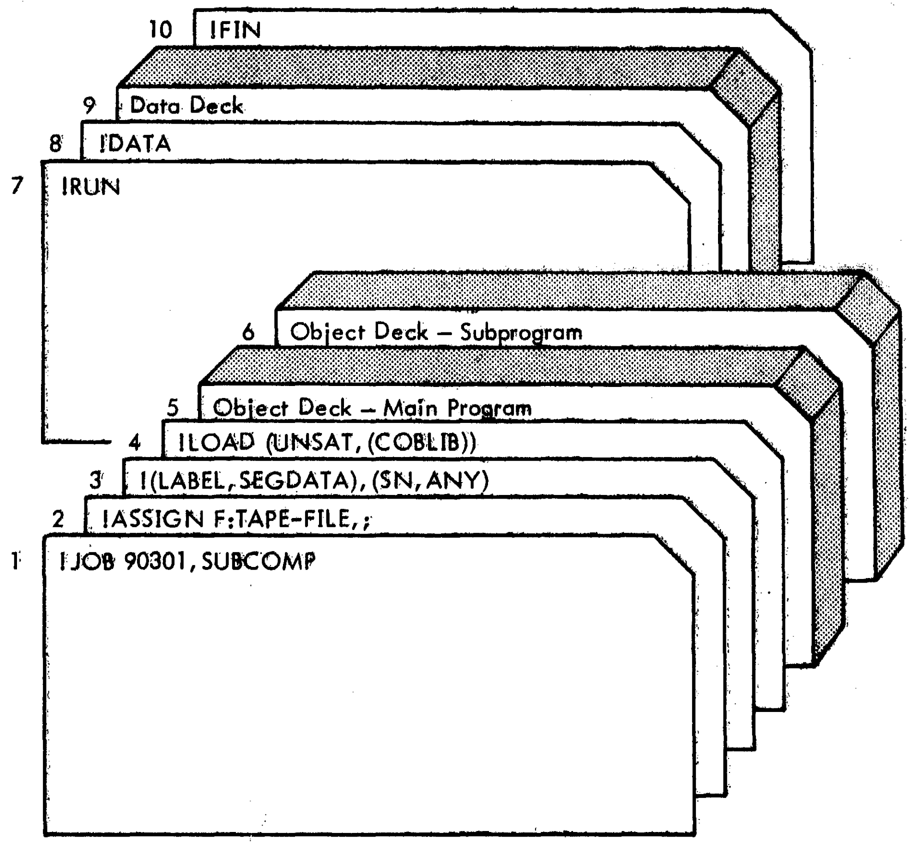
| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
| | 90301 | Account number. |
| | SUBCOMP | Identifies the user. |
| 2 | !COBOL | Control command that specifies control is to be transferred to the COBOL processor. |
| | LS | Specifies that the source program is to be listed. |
| | BO | Specifies that the binary object deck is to be produced. |
| | MAIN | Specifies that this program is to be compiled as the main or calling program. |
| 3 | | COBOL source deck of the main program. |
| 4 | !FIN | Signals the end of the job stack. |

Figure 13. Inter-Program Communication – Compilation of Main or Calling Program



| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
| | 90301 | Account number. |
| | SUBCOMP | Identifies the user. |
| 2 | !COBOL | Control command that specifies control is to be transferred to the COBOL processor. |
| | LS | Specifies that the source program is to be listed. |
| | BO | Specifies that the binary object deck is to be produced. |
| | SUB | Specifies that this program is to be compiled as the subprogram or called program. |
| 3 | | COBOL source deck of the subprogram. |
| 4 | !FIN | Signals the end of the job stack. |

Figure 14. Inter-Program Communication -- Compilation of Subprogram or Called Program



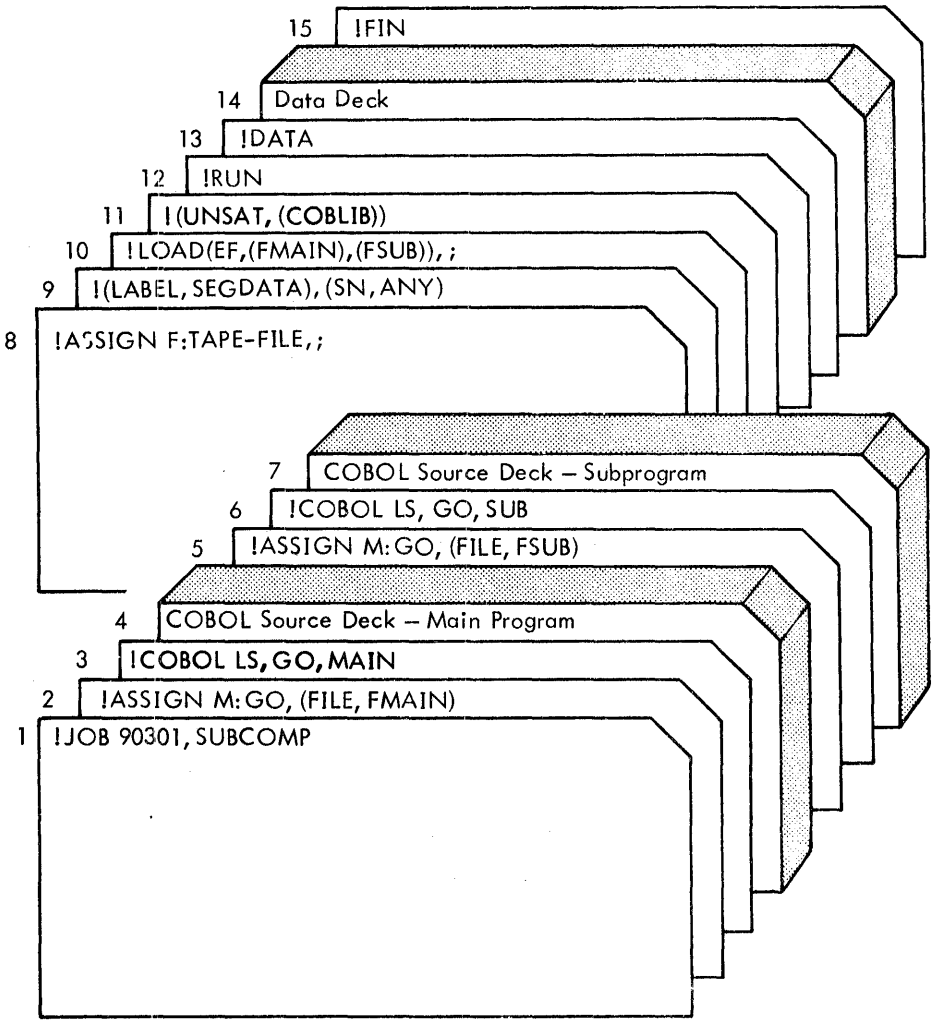
10 !FIN
9 Data Deck
8 !DATA
7 !RUN
6 Object Deck – Subprogram
5 Object Deck – Main Program
4 !LOAD (UNSAT, (COBLIB))
3 !(LABEL, SEGDATA), (SN, ANY)
2 !ASSIGN F:TAPE-FILE, ;
1 !JOB 90301, SUBCOMP

| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
| | 90301 | Account number. |
| | SUBCOMP | Identifies the user. |
| 2 | !ASSIGN | Control command that specifies the file and physical peripheral device to be used. |
| | F:TAPE-FILE | DCB name of TAPE-FILE file. |
| | ; | Signals that this ASSIGN command is continued on the following card. |
| 3 | !(LABEL, SEGDATA) | Specifies that the file is to be named SEGDATA. (This card is part of the preceding ASSIGN command.) |
| | (SN, ANY) | Specifies that the file is to be written on any available reel. |
| 4 | !LOAD | Control command that directs the resident loader to form a relocatable load module. |
| | (UNSAT, (COBLIB)) | Specifies that the COBOL library (identified by the account name COBLIB) is to be searched for external definitions required for the load module. |
| 5 | | Object deck of the main program. |
| 6 | | Object deck of the subprogram. |
| 7 | !RUN | Control command that specifies the program is to be executed. |

Figure 15. Inter-Program Communication – Execution with Object Decks

| Card | Parameter | Description |
|---|---|---|
| 8 | !DATA | Control command that specifies a data deck is to follow. |
| 9 | | Data deck. |
| 10 | !FIN | Signals the end of the job stack. |

Figure 15. Inter-Program Communication – Execution with Object Decks (cont.)



15 !FIN
14 Data Deck
13 !DATA
12 !RUN
11 !(UNSAT, (COBLIB))
10 !LOAD(EF,(FMAIN),(FSUB)), ;
9 !(LABEL, SEGDATA), (SN, ANY)
8 !ASSIGN F:TAPE-FILE, ;
7 COBOL Source Deck – Subprogram
6 !COBOL LS, GO, SUB
5 !ASSIGN M:GO, (FILE, FSUB)
4 COBOL Source Deck – Main Program
3 !COBOL LS, GO, MAIN
2 !ASSIGN M:GO, (FILE, FMAIN)
1 !JOB 90301, SUBCOMP

| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
| | 90301 | Account number. |
| | SUBCOMP | Identifies the user. |

Figure 16. Inter-Program Communication – Compilation and Execution

| Card | Parameter | Description |
|---|---|---|
| 2 | !ASSIGN | Control command that specifies the file and physical peripheral device to be used. |
| | M:GO | The file is the system GO file. |
| | (FILE,FMAIN) | Specifies that the GO file (containing the module of the main or calling program) is to be a disk file named FMAIN. |
| 3 | !COBOL | Control command that specifies control is to be transferred to the COBOL processor. |
| | LS | Specifies that the source program is to be listed. |
| | GO | Specifies that the program is to be executed after compilation. |
| | MAIN | Specifies that this program is to be compiled as the main or calling program. |
| 4 | | COBOL source deck of the main program. |
| 5 | !ASSIGN | Control command that specifies the file and physical peripheral device to be used. |
| | M:GO | The file is the system GO file. |
| | (FILE, FSUB) | Specifies that the GO file (containing the module of the subprogram or called program) is to be a disk file named FSUB. |
| 6 | !COBOL | Specifies that control is to be transferred to the COBOL processor. |
| | LS | Specifies that the source program is to be listed. |
| | GO | Specifies that the program is to be executed after compilation. |
| | SUB | Specifies that this program is to be compiled as the subprogram or called program. |
| 7 | | COBOL source deck of the subprogram. |
| 8 | !ASSIGN | Control command that specifies the file and physical peripheral device to be used. |
| | F:TAPE-FILE | DCB name of TAPE-FILE file. |
| | ; | Signals that this ASSIGN command is continued on the following card. |
| 9 | !(LABEL, SEGDATA) | Specifies that the file is to be named SEGDATA. (This card is part of the preceding ASSIGN command.) |
| | (SN,ANY) | Specifies that the output file is to be written on any available reel. |
| 10 | !LOAD | Control command that directs the resident loader to form a relocatable load module. |
| | (EF, (FMAIN), (FSUB)) | Specifies that modules of the main program (FMAIN) and the subprogram (FSUB) are to be included in the load module. |
| | ; | Signals that this LOAD command is continued on the following card. |
| 11 | !(UNSAT, (COBLIB)) | Specifies that the COBOL library (identified by account number COBLIB) is to be searched for external definitions required for the load module. (This card is part of the preceding LOAD command.) |

Figure 16. Inter-Program Communication – Compilation and Execution (cont.)

| Card | Parameter | Description |
|---|---|---|
| 12 | !RUN | Control command that specifies the program is to be executed. |
| 13 | !DATA | Control command that specifies a data deck is to follow. |
| 14 | | Data deck. |
| 15 | !FIN | Signals the end of the job stack. |

Figure 16. Inter-Program Communication – Compilation and Execution (cont.)

## ENTER Statement Feature

The ENTER statement allows the COBOL program to enter any non-COBOL subroutine that the loader can load at object time: for example, a closed machine-language subroutine or a FORTRAN subroutine. The subroutine name must be defined as an entry point.

Figures 17 and 18 show how the subroutine object deck is combined with the COBOL program for compilation and execution.

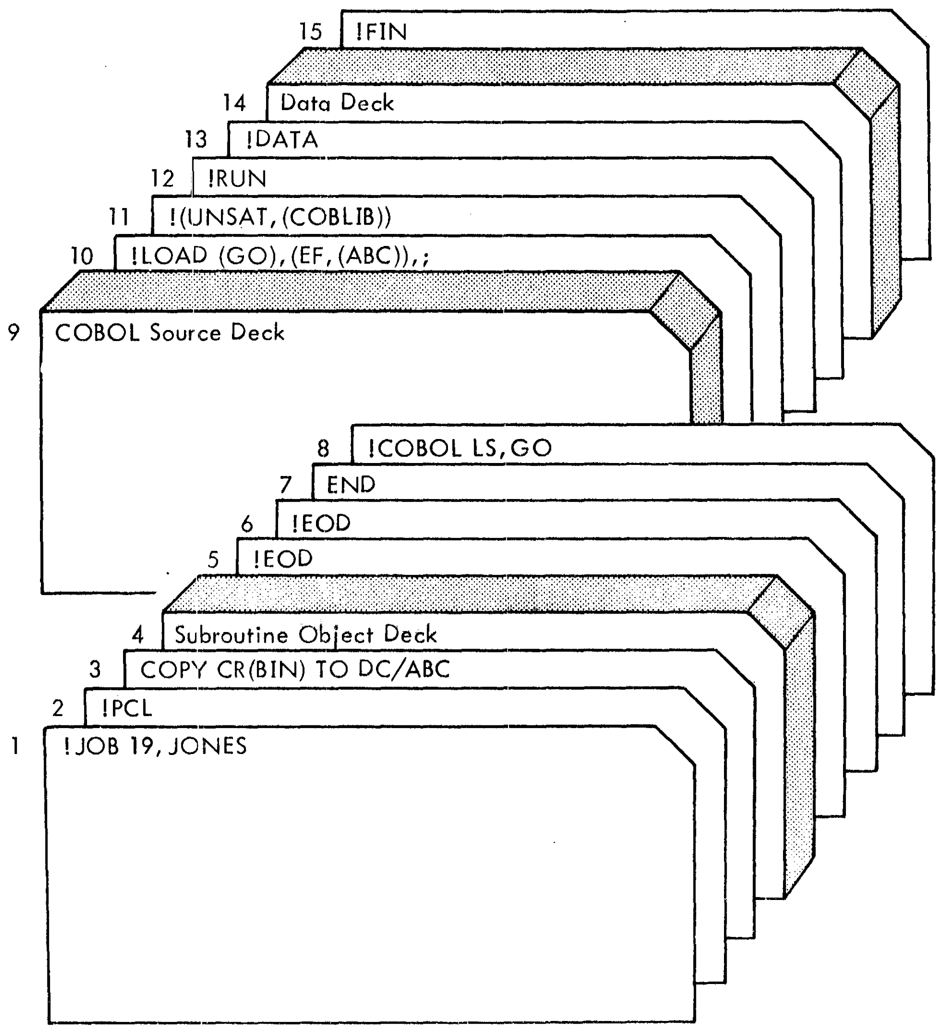
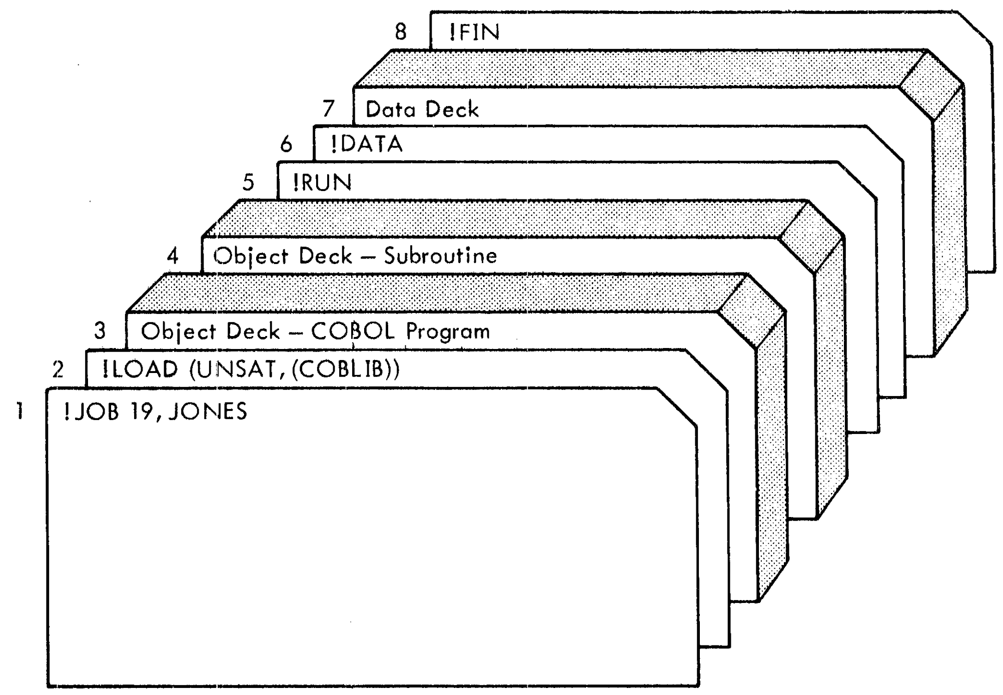


Figure 17. ENTER Statement Feature – Compilation and Execution

| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
|  | 19 | Account number. |
|  | JONES | Identifies the user. |
| 2 | !PCL | Initiates Peripheral Control Language (PCL) processor. |
| 3 | COPY CR(BIN) TO DC/ABC | Copies object deck from card reader to disk file name ABC. |
| 4 |  | Subroutine object deck. |
| 5 | !EOD | Signals PCL of the end of the subroutine object deck. |
| 6 | !EOD |  |
| 7 | END | Terminates PCL operations. |
| 8 | !COBOL | Control command that specifies control is to be transferred to the COBOL processor. |
|  | LS | Specifies that the source program is to be listed. |
|  | GO | Specifies that the program is to be executed after compilation. |
| 9 |  | COBOL source deck. |
| 10 | !LOAD | Control command that directs the resident loader to form a relocatable load module. |
|  | (GO) | Specifies that data from the user temporary GO file is to be included in the root of the load module. |
|  | (EF, (ABC)) | Specifies that the module of file ABC is to be included in the load module. |
|  | ; | Signals that this LOAD command is continued on the following card. |
| 11 | !(UNSAT, (COBLIB)) | Specifies that the COBOL library (identified by account number COBLIB) is to be searched for external definitions required for the load module. (This card is part of the preceding LOAD command.) |
| 12 | !RUN | Control command that specifies the program is to be executed. |
| 13 | !DATA | Control command that specifies a data deck is to follow. |
| 14 |  | Data deck. |
| 15 | !FIN | Signals the end of the job stack. |

Figure 17. ENTER Statement Feature – Compilation and Execution (cont.)


8 !FIN
7 Data Deck
6 !DATA
5 !RUN
4 Object Deck – Subroutine
3 Object Deck – COBOL Program
2 !LOAD (UNSAT, (COBLIB))
1 !JOB 19, JONES

| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
| | 19 | Account number. |
| | JONES | Identifies the user. |
| 2 | !LOAD | Control command that directs the resident loader to form a relocatable load module. |
| | (UNSAT, (COBLIB)) | Specifies that the COBOL library (identified by account number COBLIB) is to be searched for external definitions required for the load module. |
| 3 | | Object deck of the COBOL program. |
| 4 | | Object deck of the subroutine. |
| 5 | !RUN | Control command that specifies the program is to be executed. |
| 6 | !DATA | Control command that specifies a data deck is to follow. |
| 7 | | Data deck. |
| 8 | !FIN | Signals the end of the job stack. |

Figure 18. ENTER Statement Feature – Execution with Object Decks

## Co-Resident Sort Feature

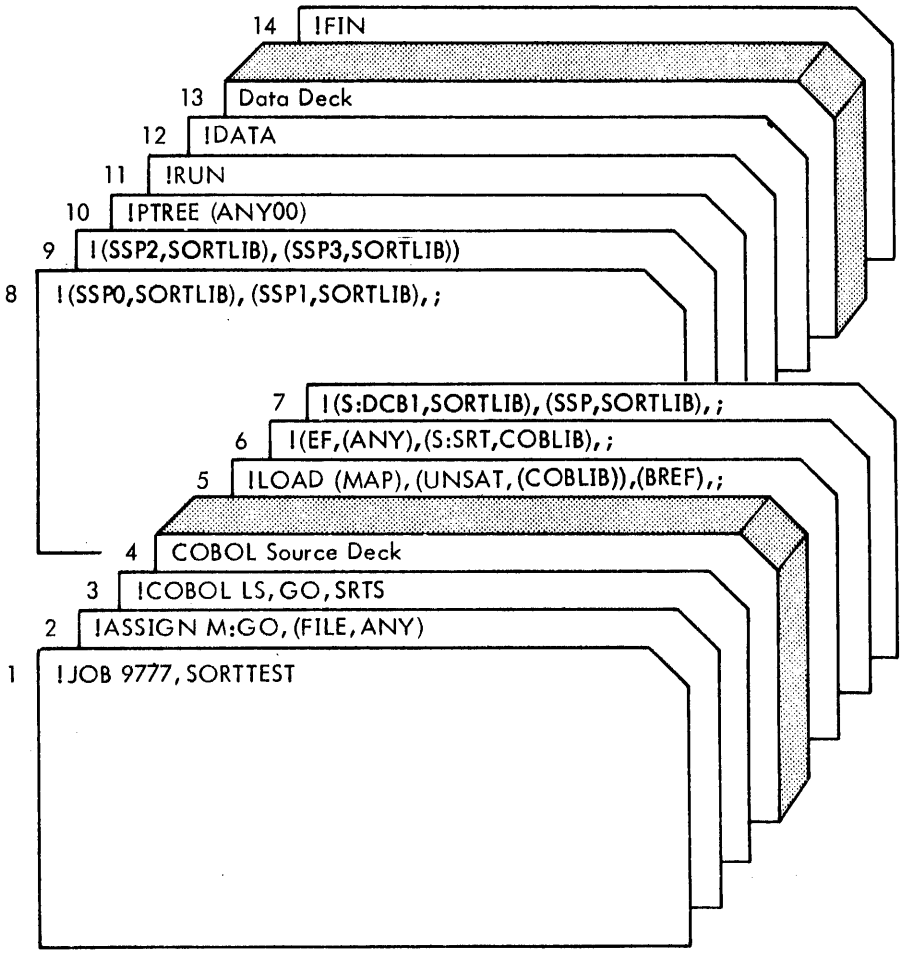
To use the co-resident sort feature, the names of the Sort processor modules must be communicated to the loader along with the sort TREE structure. The sort TREE structure is generated by the COBOL compiler and is stored in a file on disk under the user program name with two zeros added (for example, ANY00). The user can access this compiler-built TREE structure with a PTREE control command, or he can bypass the compiler-built TREE file and actually supply the same TREE structure with a TREE control command. (The TREE control command is especially useful in changing the TREE structure.) The compiler does not build a TREE for a segmented program that uses the co-resident sort feature.

Figures 19 and 20 show how to compile, load, and execute a COBOL program using co-resident sort. Notice the PTREE control command in this figure (see card 10). Instead of using this command to access the compiler-built TREE structure, the user could have substituted the following TREE control commands to supply the same TREE structure –

```
!TREE ANY–S:SRT–S:DCB1–SSP–(SSP0,SSP1,SSP2,SSP3)
```

if SRTS is specified on the COBOL control command, or if SRTR is specified:

```
!TREE ANY–S:SRT–S:DCB1–SRP–(SRP0,SRP1,SRP2,SRP3)
```



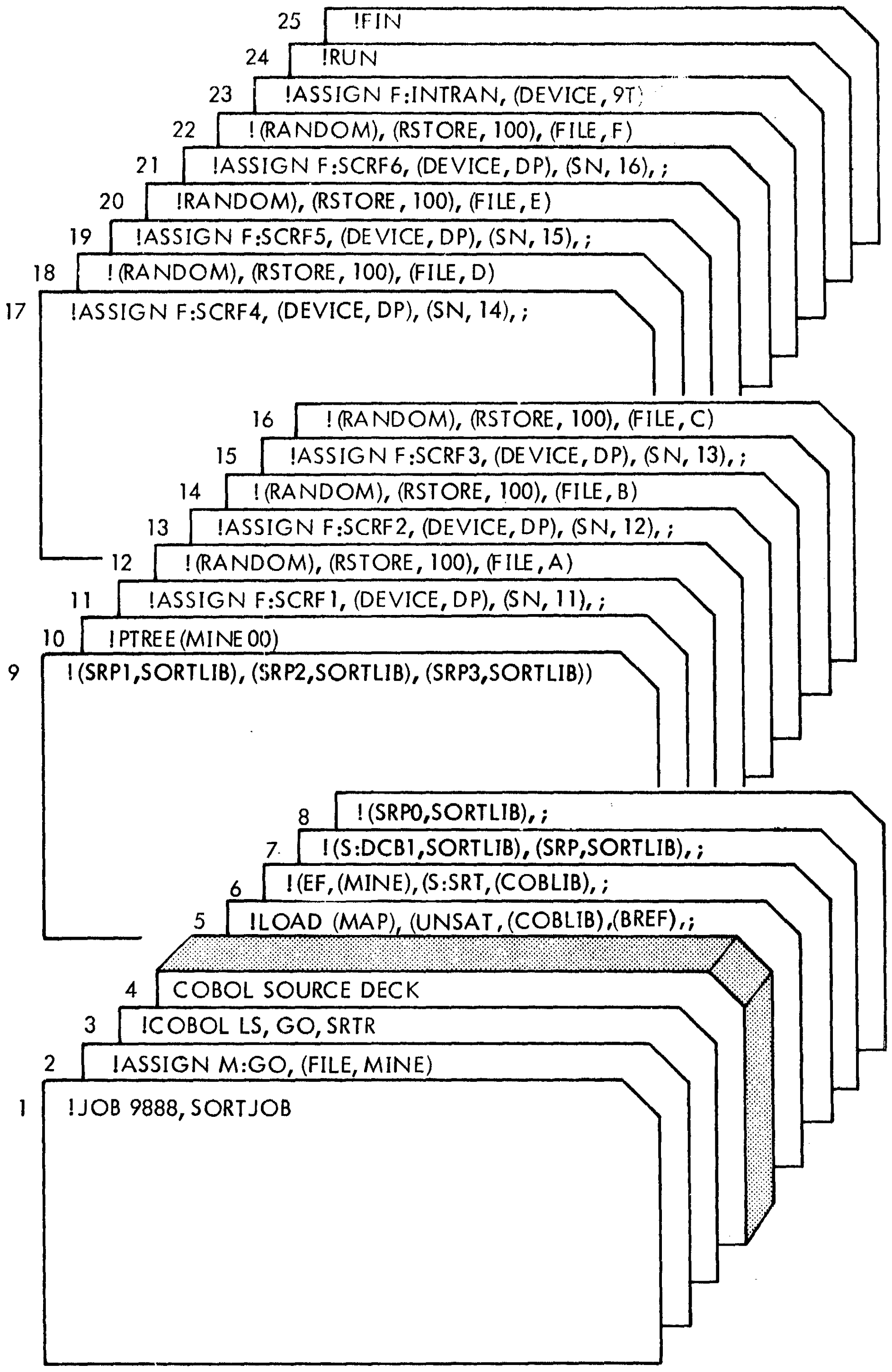
```
14  !FIN
13  Data Deck
12  !DATA
11  !RUN
10  !PTREE (ANY00)
 9  !(SSP2,SORTLIB), (SSP3,SORTLIB))
 8  !(SSP0,SORTLIB), (SSP1,SORTLIB),;
 7  !(S:DCB1,SORTLIB), (SSP,SORTLIB),;
 6  !(EF,(ANY),(S:SRT,COBLIB),;
 5  !LOAD (MAP), (UNSAT, (COBLIB)),(BREF),;
 4  COBOL Source Deck
 3  !COBOL LS,GO,SRTS
 2  !ASSIGN M:GO, (FILE, ANY)
 1  !JOB 9777, SORTTEST
```

| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of a job stack. |
| | 9777 | Account number. |
| | SORTTEST | Identifies the user. |

Figure 19. Co-Resident Sort Feature – Compilation and Execution (Sequential Sort Technique)

| Card | Parameter | Description |
|---|---|---|
| 2 | !ASSIGN | Control command that specifies the file and physical peripheral device to be used. |
| | M:GO | The file is the system GO file. |
| | (FILE, ANY) | Specifies that the file is to be a disk file named ANY. |
| 3 | !COBOL | Control command that specifies control is to be transferred to the COBOL processor. |
| | LS | Specifies that the source program is to be listed. |
| | GO | Specifies that the program is to be executed after compilation. |
| | SRTS | Specifies that co-resident sort code is to be generated. |
| 4 | | COBOL source deck. |
| 5 | !LOAD | Control command that directs the resident loader to form a relocatable load module. |
| | (MAP) | Gives complete listing of external references and definitions for the load module. |
| | (UNSAT, (COBLIB)) | Specifies that the COBOL library (identified by account name COBLIB) is to be searched for external definitions required for the load module. |
| | (BREF) | Specifies that the branch reference mode of loading is to be used. |
| | ; | Signals that this LOAD command is continued on the following card. |
| 6 | !(EF, (ANY), (S:SRT,COBLIB) | These cards are all continuation of the LOAD command. They specify that the modules of the root segment ANY, the module S:SRT, and the Sort modules S:DCB1, SSP, SSP0, SSP1, SSP2, and SSP3 are to be included in the load module. S:SRT can be found in account COBLIB, and the Sort module can be found in account SORTLIB. |
| 7 | !(S:DCB1,SORTLIB), (SSP,SORTLIB) | |
| 8 | !(SSP0,SORTLIB), (SSP1,SORTLIB) | |
| 9 | !(SSP2,SORTLIB), (SSP3,SORTLIB)) | |
| 10 | !PTREE (ANY00) | Control command that is used to obtain the TREE control command from the user's file (named ANY00, which is the name of the program with 00 appended to it). |
| 11 | !RUN | Control command that specifies the program is to be executed. |
| 12 | !DATA | Control command that specifies a data deck is to follow. |
| 13 | | Data deck. |
| 14 | !FIN | Signals the end of the job stack. |

Figure 19. Co-Resident Sort Feature – Compilation and Execution (Sequential Sort Technique) (cont.)



```
25  !FIN
24  !RUN
23  !ASSIGN F:INTRAN, (DEVICE, 9T)
22  !(RANDOM), (RSTORE, 100), (FILE, F)
21  !ASSIGN F:SCRF6, (DEVICE, DP), (SN, 16), ;
20  !RANDOM), (RSTORE, 100), (FILE, E)
19  !ASSIGN F:SCRF5, (DEVICE, DP), (SN, 15), ;
18  !(RANDOM), (RSTORE, 100), (FILE, D)
17  !ASSIGN F:SCRF4, (DEVICE, DP), (SN, 14), ;
16  !(RANDOM), (RSTORE, 100), (FILE, C)
15  !ASSIGN F:SCRF3, (DEVICE, DP), (SN, 13), ;
14  !(RANDOM), (RSTORE, 100), (FILE, B)
13  !ASSIGN F:SCRF2, (DEVICE, DP), (SN, 12), ;
12  !(RANDOM), (RSTORE, 100), (FILE, A)
11  !ASSIGN F:SCRF1, (DEVICE, DP), (SN, 11), ;
10  !PTREE (MINE00)
9   !(SRP1,SORTLIB), (SRP2,SORTLIB), (SRP3,SORTLIB))
8   !(SRP0,SORTLIB), ;
7   !(S:DCB1,SORTLIB), (SRP,SORTLIB), ;
6   !(EF, (MINE), (S:SRT, (COBLIB), ;
5   !LOAD (MAP), (UNSAT, (COBLIB), (BREF), ;
4   COBOL SOURCE DECK
3   !COBOL LS, GO, SRTR
2   !ASSIGN M:GO, (FILE, MINE)
1   !JOB 9888, SORTJOB
```

Figure 20. Co-Resident Sort Feature – Compilation and Execution (Random Sort Technique)

| Card | Parameter | Description |
|---|---|---|
| 1 | !JOB | Signals the beginning of the job stack. |
| 2 | !ASSIGN | Controls command that specifies files characteristics. |
| | M:GO | The device is the system GO file. |
| | (FILE,MINE) | This file is a disk file called MINE. |
| 3 | !COBOL | Call and transfer Control to the COBOL compiler. |
| | LS | List the source program. |
| | GO | Build a load module of the object program on the GO file. |
| | SRTR | Specifies that co-resident sort code is to be generated. |
| 4 | | The COBOL source program deck. |
| 5 | !LOAD | Directs the resident loader to form a relocatable load module. |
| | (MAP) | List the external references and definitions for the load module. |
| | (UNSAT,COBLIB) | Satisfy external references from COBOL run-time library. |
| | (BREF) | Specifies that the branch reference mode of loading is to be used. |
| | ; | Signals that the load command continues on the next card. |
| 6 | (EF, (MINE), (S:SRT,COBLIB) | These cards are all a continuation of the LOAD command. They specify that the modules of the root segment MINE, the module S:SRT, and Sort modules S:DCB1, SRP, SRP0, SRP1, SRP2, and SRP3 are to be included in the load modules. The module S:SRT can be found in account COBLIB, and the Sort modules can be found in account SORTLIB. |
| 7 | (S:DCB1,SORTLIB), (SRP,SORTLIB) | |
| 8 | (SRP0,SORTLIB) | |
| 9 | (SRP1,SORTLIB), (SRP2,SORTLIB), (SRP3,SORTLIB)) | |
| 10 | !PTREE (MINE00) | This command will obtain the TREE command from the user's file named MINE00. |
| 11 | !ASSIGN,F:SCRF1,(DEVICE,DP), (SN,11) | These cards assign the first six of the 17 Sort work file DCBs to six private disk pack files. These files are assigned to RANDOM storage, using 100 granules on each pack. |
| 12 | !(RANDOM),(RSTORE,100),(FILE,A) | |
| 13 | !ASSIGN F:SCRF2,(DEVICE,DP), (SN,12) | |
| 14 | !(RANDOM),(RSTORE,100),(FILE,B) | |
| 15 | !ASSIGN F:SCRF3,(DEVICE,DP), (SN,13) | |
| 16 | !(RANDOM,(RSTORE,100), (FILE,C) | |

Figure 20. Co-Resident Sort Feature – Compilation and Execution (Random Sort Technique) (cont.)

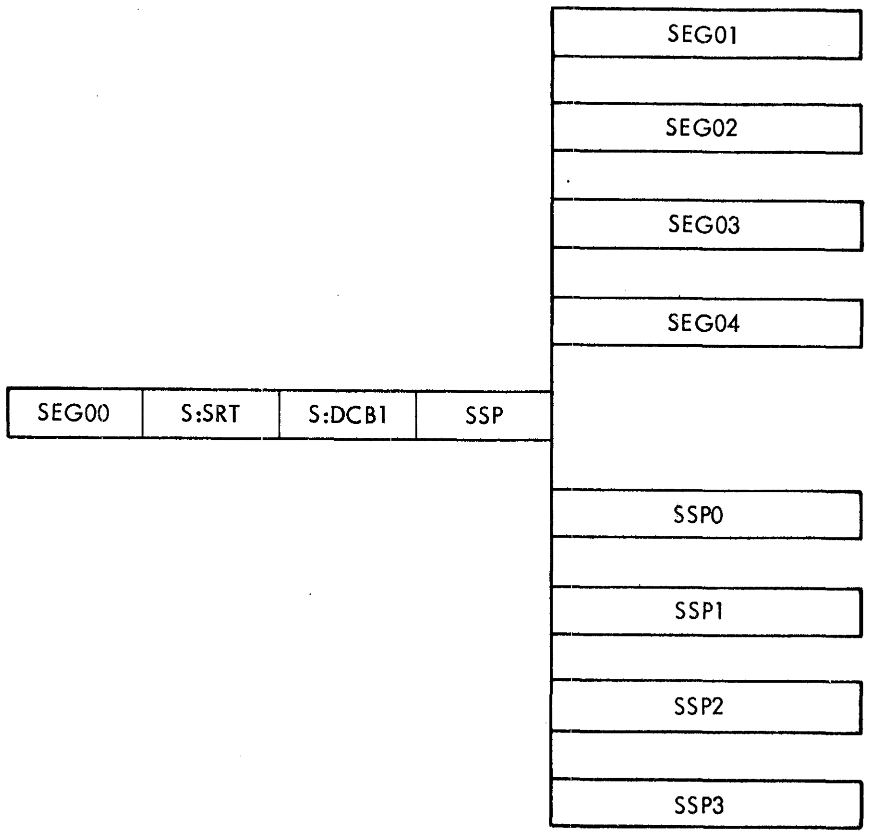
| Card | Parameter | Description |
|---|---|---|
| 17 | !ASSIGN F:SCRF4,(DEVICE,DP), (SN,14), | |
| 18 | !(RANDOM), (RSTORE, 100), (FILE, D) | |
| 19 | !ASSIGN F:SCRF5,(DEVICE,DP), (SN,15), | |
| 20 | !(RANDOM), (RSTORE, 100), (FILE, E) | |
| 21 | !ASSIGN F:SCRF6,(DEVICE,DP),(SN,16), | |
| 22 | !(RANDOM),(RSTORE,100),(FILE,F) | |
| 23 | !ASSIGN F:INTRAN,(DEVICE,9T) | The input data file for this program is called INTRAN and can be found on an unlabeled 9-track device tape. |
| 24 | !RUN | This command requests that the compiled object program be executed. |
| 25 | !FIN | Signals the end of the job stack. |

Figure 20. Co-Resident Sort Feature – Compilation and Execution (Random Sort Technique) (cont.)

The three examples shown below illustrate the use of the co-resident sort with a segmented COBOL program.

Example 1:

Tree structure



where:

SEG00 is the COBOL root program (section 1) that contains the references to the various overlay segments as well as the section (section 2) that contains the SORT verb and the Input and Output procedure sections.

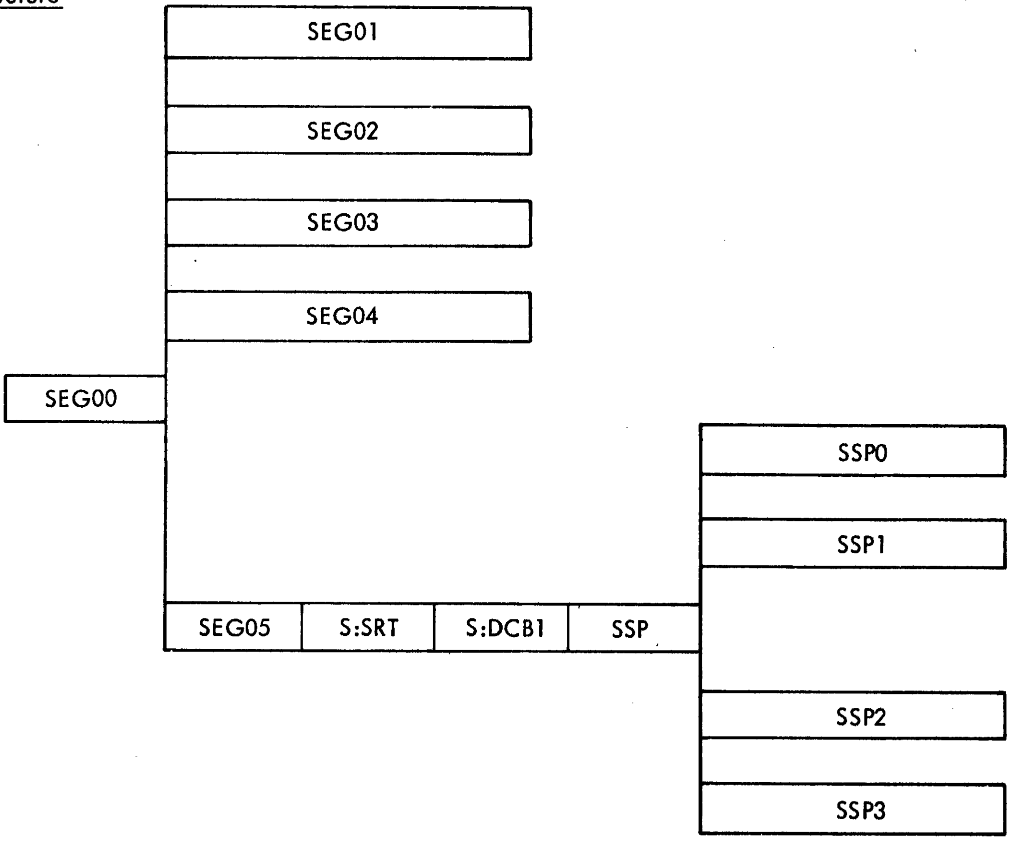
SEG01 to SEG04 are the overlay segments (section numbers above 49).

Job Control Cards

```
!SEG03, SEG04, SSP0, SSP1, SSP2, SSP3)
!TREE SEG00 - S:SRT - S:DCB1 - SSP - (SEG01, SEG02,;
!LOAD. . ., (BREF)
```

Example 2:

Tree Structure



where

SEG00 is the COBOL root program (section 1) that contains all the references to the various overlay segments.

SEG01 to SEG04 are the overlay segments (section numbers above 49).

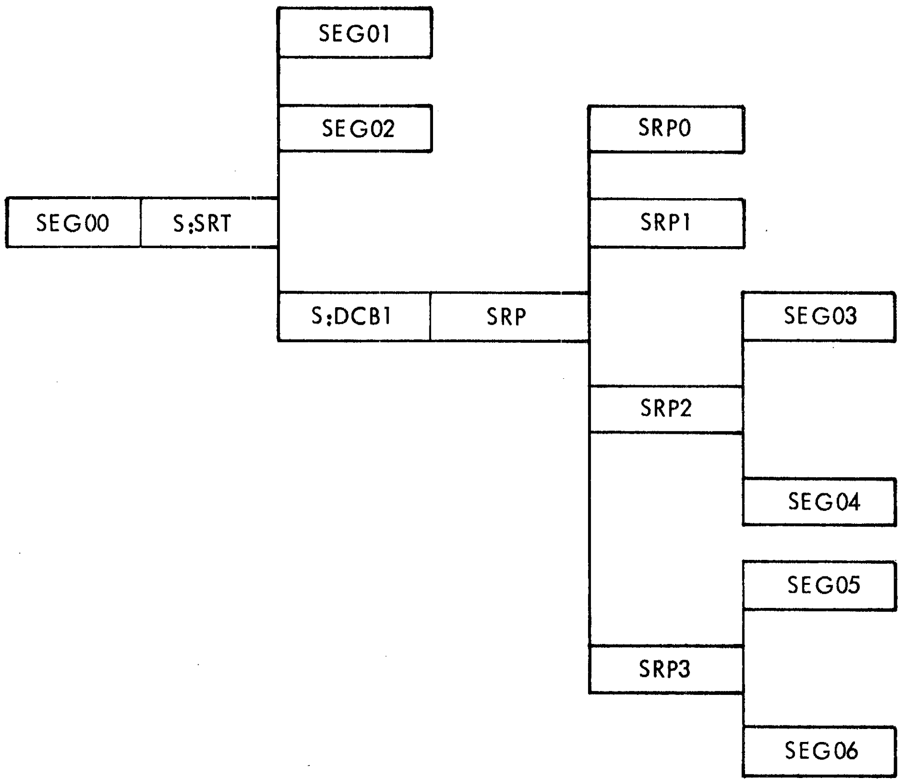
SEG05 is the overlay segment (section 80) that contains the SORT verb and the Input and Output procedure sections.

Job Control Cards

```
!S:DCB1 - SSP - (SSP0, SSP1, SSP2, SSP3)
!TREE SEG00 - (SEG01, SEG02, SEG03, SEG04, SEG05 - S:SRT-;
!LOAD. . ., (BREF)
```

Example 3:

Tree Structure



where

SEG00 is the COBOL root program (section 1) that contains references to the various overlay segments as well as the section (section 2) that contains the SORT verb and the Input and Output procedure section.

| | |
|---|---|
| SEG01 - 02 | are overlay segments (section numbers above 49). |
| SEG03 - 04 | are overlay segments in the Input Procedure (SRP2 of SORT) (section numbers above 49). |
| SEG05 - 06 | are overlay segments in the Output Procedure (SRP3 of SORT) (section numbers above 49). |

Job Control Cards

```
        !-(SEG03,SEG04),SRP3-(SEG05,SEG06)))

    !TREE SEG00 - S:SRT - (SEG01,SEG02,S:DCB1-SRP-(SRP0,SRP1,SRP2,;

!LOAD. . . , (BREF)
```

## Debug Module Object Time Switch

The object time switch dynamically activates the debugging code inserted by the compiler. If the switch is on, all the effects of the debugging language written in the source program are permitted. If the switch is off, all the effects described in the COBOL Reference Manual are inhibited. Recompilation of the source program is not required to provide or eliminate this facility.

The object time switch is normally in the ON position. In order to deactivate the debug code (turn switch to OFF position), the following option in the !RUN control command should be used:

```
!RUN (START,NO$DBG)
```

# 6. XEROX ANS COBOL COMPILER DIAGNOSTICS

Table 2 lists all diagnostic messages produced by the COBOL compiler. Certain diagnostics are associated with a Strength Code having the following significance:

| | |
|---|---|
| P – Precautionary | These diagnostics, which are produced only when the DIAG control command option is specified, indicate that a trivial error (or possibility of an error) unaffecting program execution has been detected. |
| F – Fatal | A serious error has been detected. Compilation is not completed and no object program is produced. |

The object program severity level (in hexadecimal) associated with each diagnostic is also shown. Execution of a program bearing a severity level of 7 or greater is not recommended.

Table 2. Xerox ANS COBOL Compiler Diagnostics

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 001 | SOURCE PROGRAM OUT OF SEQUENCE<br>The sequence number field (columns 1 through 6) of the source program lines is tested for ascending sequence only when the SEQCHK control command option is specified. | | 4 |
| 002 | INCORRECT PUNCTUATION | P | 0 |
| 003 | AREA A VIOLATION | | 1 |
| 004 | NAME/NUMERIC LITERAL EXCEEDS 30 CHARACTERS – TRUNCATED | | 4 |
| 005 | INVALID CHARACTER(S) | | 4 |
| 006 | QUOTE MARK OMITTED<br>Either the terminating quote of a non-numeric literal has been omitted or a nonblank character has occurred prior to a quote on a continuation line. | | 4 |
| 007 | NON-NUMERIC LITERAL EXCEEDS 255 CHARACTERS – TRUNCATED | | 4 |
| 008 | RESERVED WORD USED INCORRECTLY – TREATED AS A NAME<br>A reserved word has been encountered in a COBOL division within which it is inapplicable. It receives preliminary treatment as a name. | | 2 |
| 009 | DIVISION HEADER INCORRECT OR OMITTED | | 2 |
| 010 | PERIOD OMITTED | | 2 |
| 011 | REQUIRED SECTION OMITTED | | 4 |
| 012 | SECTION OUT OF ORDER | | 2 |
| 013 | SECTION DUPLICATED | | 2 |
| 014 | REQUIRED PARAGRAPH OMITTED | | 2 |
| 015 | PARAGRAPH OUT OF ORDER | | 2 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 016 | PARAGRAPH DUPLICATED | | 2 |
| 017 | REQUIRED CLAUSE OMITTED – COMPILATION ABORTED | F | |
| 018 | CLAUSE DUPLICATED | | 2 |
| 019 | PROCEDURE DIVISION STRUCTURED INCORRECTLY<br>A section header has not preceded the initial PROCEDURE DIVISION statements but has occurred later. This condition conflicts with rules that govern structuring of the PROCEDURE DIVISION, but is harmless in itself. | | 1 |
| 020 | REQUIRED WORD MISSING | | 2 |
| 021 | MISSING COBOL DIVISION(S) – COMPILATION ABORTED | F | |
| 022 | NAME INVALID/OMITTED | | 7 |
| 023 | INVALID LITERAL | | 7 |
| 024 | INVALID SUBSCRIPT | | 7 |
| 025 | CLOSING PARENTHESIS OMITTED | | 4 |
| 026 | INVALID NUMBER | | 7 |
| 027 | ILLEGAL CURRENCY SIGN | | 4 |
| 028 | ILLEGAL PRIORITY-NUMBER | | 4 |
| 029 | INCORRECT SWITCH-NAME | | 4 |
| 030 | INVALID 'ALL' LITERAL | | 7 |
| 031 | CONDITION-NAME OMITTED | | 7 |
| 032 | INCOMPLETE 'SAME' CLAUSE | | 7 |
| 033 | INVALID/OMITTED QUALIFIER | | 7 |
| 034 | UNSELECTED FILE<br>An FD or SD entry has no corresponding SELECT sentence in the FILE-CONTROL paragraph. This is a violation of COBOL rules but is harmless in this implementation if an !ASSIGN command is provided for the file at execution time. | | 4 |
| 035 | INVALID LEVEL-NUMBER | | 7 |
| 036 | INVALID/OMITTED DATA-NAME | | 7 |
| 037 | SECTION HEADER INCORRECT | | 4 |
| 038 | SOURCE WORDS BYPASSED | | 7 |
| 039 | INVALID INDEXING | | 7 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 040 | FD REPORT CLAUSE REQUIRED – COMPILATION ABORTED<br>An RD entry has not been associated with any file via a REPORT clause on an FD entry. Thus, it is impossible to produce the report. | F | |
| 041 | INVALID PICTURE | | 7 |
| 042 | 'TYPE' AND/OR 'NEXT GROUP' OMITTED OR WRITTEN IMPROPERLY | | 7 |
| 043 | CLAUSE WRITTEN ILLEGALLY | | 7 |
| 044 | EXCESSIVE REPETITION COUNT IN PICTURE | | 7 |
| 045 | INVALID REPETITION COUNT | | 7 |
| 046 | ILLEGAL CHARACTER(S) IN PICTURE – 'B' SUBSTITUTED | | 7 |
| 047 | ILLEGAL COMBINATION OF PICTURE SYMBOLS – DISCARDED | | 7 |
| 048 | EXCESSIVE SIZE SPECIFIED FOR EDITED FIELD – TRUNCATED | | 7 |
| 049 | SYNTACTICAL ERROR | | 7 |
| 050 | CONDITIONAL STATEMENT INVALID IN CONTEXT<br>A conditional statement has been written at a point where only imperative statements are permissible, e.g., following AT END. | | 7 |
| 051 | INCORRECT SUBSCRIPTING/INDEXING | | 7 |
| 052 | INCORRECT CLASS TEST | | 7 |
| 053 | INCORRECT SIGN TEST | | 7 |
| 054 | INCORRECT ARITHMETIC OR LOGICAL EXPRESSION | | 7 |
| 055 | CONDITION TOO LIBERAL FOR THIS FORMAT OF 'SEARCH' STATEMENT | | 7 |
| 056 | INCORRECT ARITHMETIC-EXPRESSION | | 7 |
| 057 | SECTION-NAME OMITTED | P | 0 |
| 058 | PARAGRAPH-NAME OMITTED | P | 0 |
| 059 | NULL PROCEDURE | | 1 |
| 060 | PREMATURE END OF PROCEDURE DIVISION<br>A period has not been encountered as the last source language element preceding the end of the source program. | | 2 |
| 061 | STATEMENT TOO COMPLEX FOR ANALYSIS<br>Too many levels of nested conditions and/or levels of parenthetical groupings and/or logical connectors have been specified for the statement. A set of simpler statements should be provided to accomplish the desired effect. | | 7 |
| 062 | EXCESSIVE NEGATION | | 6 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 063 | NEGATIVE INTEGER – MUST BE UNSIGNED OR POSITIVE | | 7 |
| 064 | INTEGER VALUE TOO GREAT | | 7 |
| 065 | MNEMONIC-NAME SYNONYM<br><br>A mnemonic-name and a data-name have been given identical names. This condition is a violation of COBOL rules, but is harmless unless the name has been employed in an ambiguous reference. | | 2 |
| 066 | SPECIFICATION CONFLICT<br><br>Either conflicting USAGEs or an illegal combination of USAGE and BLANK WHEN ZERO or JUSTIFIED RIGHT has been specified. The first specification encountered is used; others are discarded. | | 4 |
| 067 | MULTIPLE VALUE CLAUSES<br><br>More than one VALUE clause has been specified in a data entry; the first is used, others are discarded. | | 4 |
| 068 | ILLEGAL USE OF 'REDEFINES'<br><br>The level-number of the data entry bearing the REDEFINES clause has not corresponded to any level-number within the potential redefinition scope. The REDEFINES clause is ignored. | | 7 |
| 069 | INCORRECT QUALIFICATION<br><br>An incorrect qualification in conjunction with a REDEFINES or RENAMES clause has been encountered. The REDEFINES clause is obeyed, since the data-name is not essential to its resolution. The scope of the RENAMES is set arbitrarily to 1 byte and its origin is assumed to be the base of the current record. | | 7 |
| 070 | ILLEGAL DATA HIERARCHY<br><br>Level-66 entries have not been specified as the last entries in a data hierarchy. All entries between the last level-66 entry and the beginning of the next record are discarded. | | 7 |
| 071 | INVALID 'RENAMES' SCOPE<br><br>The extent of a RENAMES scope has been indeterminable; 1 byte is assumed. | | 7 |
| 072 | MISPLACED 'RENAMES' CLAUSE<br><br>The RENAMES clause has not been associated with level-number 66. The RENAMES declaration is ignored. | | 7 |
| 073 | CONDITION-NAME ENTRY LACKS 'VALUE' CLAUSE<br><br>No VALUE clause has been specified on a level-88 entry. The entry is deleted. | | 7 |
| 074 | CONDITION-NAME ENTRY BEARS INVALID CLAUSE(S)<br><br>Clauses other than the VALUE clause have been encountered on a level-88 entry. These clauses are ignored. | | 7 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 075 | MISPLACED 'REDEFINES' CLAUSE<br>The REDEFINES clause has appeared before the occurrence of any legitimate potential redefinition point in the current data hierarchy. The clause is ignored. | | 7 |
| 076 | ILLEGAL USE OF 'OCCURS . . . DEPENDING ON'<br>On variable-length records the variable portion of the record must follow the fixed portion. If this rule is violated (that is, if a fixed item or group follows the last variable group of the record), any reference to the fixed item or group will be unpredictable. | | 7 |
| 077 | NESTING OF 'OCCURS' EXCEEDS 3 LEVELS<br>An attempt to define a table of more than three dimensions has been detected. The OCCURS clause is ignored. | | 7 |
| 078 | MISPLACED 'OCCURS' CLAUSE | | 7 |
| 079 | USAGE CONFLICT BETWEEN GROUP AND SUBORDINATE ITEMS<br>A conflict has occurred between the stated USAGE of a group and a subordinate data entry. The description of the subordinate item is accepted. | | 6 |
| 080 | MISPLACED 'PICTURE' CLAUSE<br>A PICTURE has been specified in conjunction with one of the USAGE types having predetermined characteristics, e.g., COMPUTATIONAL, COMPUTATIONAL-1, COMPUTATIONAL-2, and INDEX. The PICTURE clause is discarded. | P | 0 |
| 081 | ILLEGAL 'BLANK WHEN ZERO' CLAUSE<br>A BLANK WHEN ZERO clause has been found to be in conjunction with a PICTURE that precludes it, i.e., that is not unsigned numeric DISPLAY or numeric edited. The BLANK WHEN ZERO clause is ignored. | | 6 |
| 082 | ILLEGAL 'JUSTIFIED RIGHT' CLAUSE<br>A JUSTIFIED RIGHT clause has been specified on a group item or an elementary item that is not alphanumeric. The JUSTIFIED RIGHT clause is ignored. | | 6 |
| 083 | 'VALUE' CLAUSE WITHIN SCOPE OF 'REDEFINES'<br>The VALUE is accepted and used in the object program. | | 6 |
| 084 | NESTED 'VALUE' CLAUSES<br>The VALUE is accepted and used in the object program. | | 6 |
| 085 | 'VALUE' CLAUSE INCONSISTENT WITH CLASS OF ENTRY<br>The VALUE is not accepted. | | 7 |
| 086 | 'OCCURS . . . DEPENDING ON' ILLEGAL WITHIN SCOPE OF 'REDEFINES' | | 7 |
| 087 | 'PICTURE' CLAUSE ILLEGAL ON GROUP ENTRY<br>The PICTURE is ignored. | | 6 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 088 | NON-UNIQUE DATA REFERENCE | | 7 |
| 089 | NON-UNIQUE PROCEDURE-NAME | | 7 |
| 090 | INVALID 'DEPENDING ON' FIELD<br>The 'GO TO . . . DEPENDING ON' data item has not been specified as a numeric field. The statement is discarded. | | 7 |
| 091 | NON-CONTIGUOUS DATA ITEM FOLLOWING DATA STRUCTURE<br>Level 77 is changed to level 01. | | 4 |
| 092 | LEVEL 66 ILLEGAL FOLLOWING LEVEL 77, OR 01 | | 7 |
| 093 | INVALID DATA USAGE | | 7 |
| 094 | MAXIMUM SIZE EXCEEDED FOR NUMERIC OPERAND | | 7 |
| 095 | UNDEFINED DATA REFERENCE | | 7 |
| 096 | INVALID DATA REFERENCE<br>A condition-name or mnemonic-name has been referenced where a data item is expected. | | 7 |
| 097 | VALUE ILLEGAL WITHOUT COLUMN NO.<br>The value is discarded. | | 7 |
| 098 | NUMERIC VALUE ILLEGAL IN REPORT SECTION<br>The value is discarded. | | 7 |
| 099 | PRIORITY SEGMENTATION IS NOT HONORED IN THIS COMPILATION<br>Priority segmentation has been indicated in the source program without specification of the SEG control command option. | | 2 |
| 100 | ILLEGAL LEVEL-NUMBER SEQUENCE<br>The level-number is accepted. | | 2 |
| 101 | UNDEFINED KEY<br>No data entry has been specified to satisfy a KEY clause reference. The key name is disregarded. | | 7 |
| 102 | SIZE OF DATA ENTRY INDETERMINATE<br>Sufficient information has not been provided to determine the size of a data entry. | | 7 |
| 103 | SIZES OF REDEFINING AND REDEFINED AREAS UNEQUAL<br>The size of the largest of these areas is used. | | 7 |
| 104 | RENAMES DATA-NAME MISSING<br>No valid RENAMES clause has been specified on a level-66 data entry. | | 7 |
| 105 | VALUE LITERAL CONFLICTS WITH CLASS OF DATA ENTRY<br>The value is ignored. | | 7 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 106 | VALUE TRUNCATED ON RIGHT<br>Insufficient storage space has been allocated to the value. | | 4 |
| 107 | VALUE TRUNCATED ON LEFT<br>Insufficient storage space has been allocated to the value. | | 4 |
| 108 | DUPLICATE DATA-NAMES WHICH CANNOT BE UNIQUELY REFERENCED | | 2 |
| 109 | EXCESSIVE NESTING OF LIBRARY RETRIEVAL STATEMENTS – COMPILATION ABORTED | F | |
| 110 | PICTURE INCOMPATIBLE WITH USAGE<br>A PICTURE (possibly containing editing characters) incompatible with USAGE COMPUTATIONAL-3 has been specified. The USAGE is discarded. | | 6 |
| 111 | INCORRECT 'GO TO' STRUCTURE | | 7 |
| 112 | 'USE' STATEMENT OMITTED FROM DECLARATIVES SECTION | | 7 |
| 113 | POSSIBLE MISUSE OF RESERVED WORD | | 4 |
| 114 | NO CORRESPONDING DATA ITEMS IN A 'CORRESPONDING' STATEMENT | | 7 |
| 115 | IDENTIFIER IN 'CORRESPONDING' STATEMENT IS AN ELEMENTARY ITEM | | 7 |
| 116 | COMPILER LIMITATION EXCEEDED – STATEMENT INCOMPLETELY COMPILED<br>This message occurs when a PERFORM statement (format 4) is too lengthy in its entirety to be compiled. | | 7 |
| 117 | INVALID LIBRARY RETRIEVAL STATEMENT – COMPILATION ABORTED<br>The library file does not exist. | F | |
| 118 | NUMBER OF RENAMED FILES EXCEEDS COMPILER CAPACITY | | B |
| 119 | ASSEMBLY PHASE TABLE OVERFLOW – COMPILATION ABORTED<br>There are probably too many section and paragraph definitions. | F | |
| 120 | FILLER MEANINGLESS ON LEVEL 77 – ACCEPTED | | 4 |
| 121 | CONFLICT BETWEEN 'BLOCK CONTAINS' CLAUSE AND RECORD SIZE | | 7 |
| 122 | CANNOT PROCESS DATA STRUCTURE IN CORE AVAILABLE – COMPILATION ABORTED<br>This is a general message indicating that a compiler data storage area has overflowed. | F | |
| 123 | REPORT FIELD OVERLAP – DATA ITEM TRUNCATED<br>The report line probably contains conflicting COLUMN NUMBER assignments. | | 4 |
| 124 | REPORT STATEMENTS BYPASSED | | 7 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 125 | CONFLICT BETWEEN 'RECORD CONTAINS' CLAUSE AND RECORD SIZE<br>The computed record size is used. | | 4 |
| 126 | VACUOUS'ROUNDED' OPTION – IGNORED<br>Arithmetic operation has not developed digits of lesser significance than the rightmost digit position of the result data item. Thus, rounding is not effected. | P | 0 |
| 127 | 'SELECT'SENTENCES DUPLICATED | | 4 |
| 128 | ILLEGAL NUMERIC-EDITED USAGE<br>A usage conflict has occurred. A warning is issued. | P | 0 |
| 129 | ILLEGAL FLOATING-POINT USAGE FOR INTEGER<br>An integer value has been used. A warning is issued. | P | 0 |
| 130 | ILLEGAL BINARY/FLOATING POINT USAGE<br>A usage conflict has occurred. A warning is issued. | P | 0 |
| 131 | ILLEGAL INDEX DATA USAGE<br>This is treated as a binary (COMPUTATIONAL) data item. A warning is issued. | P | 0 |
| 132 | ILLEGAL NON-INTEGER USAGE<br>An integer portion of the data item is used. | | 4 |
| 133 | ILLEGAL COMPUTATIONAL-3 USAGE<br>A usage conflict has occurred. A warning is issued. | P | 0 |
| 134 | ILLEGAL ALPHANUMERIC USAGE<br>A usage conflict has occurred. A warning is issued. | P | 0 |
| 135 | ILLEGAL ALPHANUMERIC-EDITED USAGE<br>A usage conflict has occurred. A warning is issued. | P | 0 |
| 136 | MAXIMUM OF 3 IDENTIFIERS ONLY MAY BE VARIED – ENTIRE "PERFORM" STATEMENT DELETED | | 7 |
| 137 | ILLEGAL ELEMENTARY ITEM USAGE | | 7 |
| 138 | ILLEGAL INDEX-NAME USAGE<br>This is treated as a binary (COMPUTATIONAL) data item. | | 7 |
| 139 | PARAGRAPH BOTH ALTERED AND PERFORMED<br>The ALTER and PERFORM statements have been generated. This is a warning of high error probability. | | 6 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 140 | ALTERED PARAGRAPH NOT 'GO TO'<br><br>A GO TO statement has not been specified as the sole contents of a paragraph that is the subject of an ALTER statement. The termination point of the paragraph is preset to transfer control to C:ERR. If control reaches the paragraph subsequent to the execution of the ALTER statement, control is transferred as specified by the ALTER statement following execution of the statements contained in the paragraph (assuming that no single statement has caused transfer of control). | | 4 |
| 141 | INVALID PROCEDURE REFERENCE<br><br>In most cases, the statemenet is deleted. In some instances a reference to C:ERR is substituted for the incorrect procedure-name. | | 7 |
| 142 | INVALID SECTION-NAME REFERENCE<br><br>A section-name has been referenced in an ALTER statement, where only paragraphs may be mentioned. The statement is deleted. | | 7 |
| 143 | EXTERNAL NAME ALTERED TO PROCEED TO OVERLAY<br><br>An undefined procedure-name, which is therefore presumed to be an external definition, has been altered to proceed to a point in an overlay segment. Unless the ALTER statement itself is in the same overlay segment, C:ERR is substituted for the target procedure-name. | | 7 |
| 144 | INVALID PARAGRAPH-NAME REFERENCE<br><br>A paragraph-name has occurred where only a section-name is permissible, e.g., as the INPUT or OUTPUT PROCEDURE of a SORT. This is a warning of the probability of error. The requested action is taken. | | 7 |
| 145 | INTEGER VALUE ILLEGAL IN CONTEXT | | 7 |
| 146 | 'SET' USED WITH NON-INDEXED FIELD | | 7 |
| 147 | 'GO TO' INITIALIZED AT C:ERR<br><br>Comment only. GO TO statements are preset to transfer control to C:ERR if an ALTER statement has not provided a legitimate transfer point prior to its execution. | P | 0 |
| 148 | INVALID FILE-NAME<br><br>A file-name has not appeared as the operand of a statement requiring one. The statement is deleted. | | B |
| 149 | INVALID RECORD-NAME<br><br>A record-name has not appeared as the operand of a statement requiring one. The statement is deleted. | | 7 |
| 150 | LABEL/ERROR CHECK IN DECLARATIVES SECTION<br><br>An input-output statement that requires execution of a DECLARATIVES procedure has occurred within a DECLARATIVES procedure. The statement is generated, but may yield erroneous results (e.g., a loop) at execution time. | | 7 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 151 | INVALID REVERSED/NO REWIND OPTION<br>Specified input/output statement options are incompatible with the access mode of the file. The options are deleted. | | 7 |
| 152 | AT END/INVALID KEY OPTION INCOMPATIBLE WITH ACCESS MODE<br>Specified input/output statement options are incompatible with the access mode of the file. A warning is issued. The statement is generated as written. | | 7 |
| 153 | 'SEEK' USED WITH UN-KEYED FILE<br>Specification of the SEEK statement is incompatible with the organization and access mode of the file. The statement is deleted. | | 7 |
| 154 | INVALID KEY<br>The ACTUAL KEY has been undefined, defined twice, or judged incompatible with the access mode of the file. | | 7 |
| 155 | MAXIMUM DISPLAY SIZE EXCEEDED<br>The aggregate size of operands in a DISPLAY, EXHIBIT, ACCEPT, or STOP 'literal' statement has exceeded 254 characters. The display line is truncated. | | 6 |
| 156 | ILLEGAL SUBSCRIPTED 'DEPENDING ON' FIELD<br>Subscripts are ignored. | | 7 |
| 157 | NON-TABLE ITEM SEARCH<br>The statement is deleted. | | B |
| 158 | 'SEARCH ALL' UNORDERED TABLE ILLEGAL<br>A warning is issued. A serial search of the entire table is generated. | | B |
| 159 | EXTERNAL REFERENCE GENERATED<br>This message is commentary only and indicates generation of a reference to an assumed external definition. | P | 0 |
| 160 | UNDEFINED PARAMETER-NAME<br>The presence of qualification indicates that this external reference is not intentional. A reference to C:ERR is substituted. | | 7 |
| 161 | CONDITION-NAME USED AS PARAMETER<br>A reference to the conditional variable is substituted for the condition-name. | | 7 |
| 162 | DIMENSIONED PARAMETER<br>Parameters are not permitted to be subscripted/indexed. Subscripts are ignored and a reference to the first occurrence is generated. | | 7 |
| 163 | INDEX-NAME USED AS PARAMETER<br>A warning is issued. Reference to the index-name in generated. | | 7 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 164 | SUBSCRIPTS/INDICES APPLIED TO UNDIMENSIONED DATA ITEM<br>The subscripts/indices are ignored. | | 7 |
| 165 | INVALID SUBSCRIPTS/INDICES<br>The subscripts/indices are ignored and reference is made to the first occurrence. | | 7 |
| 166 | EXCESSIVE SUBSCRIPTS/INDICES<br>The excess subscripts/indices are discarded. | | 7 |
| 167 | MAXIMUM SUBSCRIPT SIZE EXCEEDED<br>The offending subscript is replaced by a value of 1 so that the first occurrence is referenced. | | 7 |
| 168 | FRACTION USED AS SUBSCRIPT<br>A data item, which bears fractional places only, has been used as a subscript. The offending subscript is replaced by a value of 1 so that the first occurrence is referenced. | | 7 |
| 169 | SIGNIFICANCE LOST WHEN ALIGNED<br>A data item whose PICTURE contains trailing Ps has been employed as a subscript. The scaled value is used. | | 7 |
| 170 | INCORRECT SUBSCRIPT/INDEX | | 7 |
| 171 | FLOATING POINT SUBSCRIPT – INTEGER VALUE ONLY USED | | 6 |
| 172 | SUBSCRIPTED TABLE ITEM<br>Subscripting has been specified where indexing should be employed. The subscripted reference is generated. | | 4 |
| 173 | SUBSCRIPT INCREMENT/DECREMENT USED<br>The increment/decrement has been applied to the subscript and a subscripted reference is generated. | | 6 |
| 174 | INEFFECTIVE DIGITS TRUNCATED<br>A decimal item used a subscript of sufficient size that insignificant digits may be truncated by the subscript calculation. | | 4 |
| 175 | NON-INTEGER SUBSCRIPT – INTEGER VALUE USED | | 4 |
| 176 | DIMENSIONED SUBSCRIPT<br>The value in the first occurrence of the array whose name has been given as a subscript is employed in the subscript calculation. | | 7 |
| 177 | INSUFFICIENT SUBSCRIPTS/INDICES<br>A value of 1 is assumed for each missing subscript/index. | | 7 |
| 178 | DIMENSIONED DATA NOT SUBSCRIPTED/INDEXED<br>A value of 1 is assumed for each missing subscript/index. | | 7 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 179 | MAXIMUM SORT KEY LENGTH EXCEEDED – 255 CHARACTER USED | | 7 |
| 180 | INVALID CS NAME – IGNORED<br>The CS (COMMON-STORAGE) control command parameter has been written incorrectly. | | B |
| 181 | INVALID CONTROL COMMAND OPTION – IGNORED<br>An unrecognizable control command option has been encountered and is ignored. | | 6 |
| 182 | ILLEGAL RELATION TEST. ONLY CONDITION-NAME TEST GENERATED<br>A relation test involving a condition-name test has been written improperly. The condition-name test is generated but the balance of the conditional statement is deleted. | | 7 |
| 183 | ILLEGAL OPERAND IN COMPARISON – COMPARISON DELETED<br>An illegal comparand (object) has been detected. The comparison is deleted. | | 7 |
| 184 | ILLEGAL SUBJECT IN RELATION TEST. STATEMENT DELETED<br>An illegal subject has been detected. The entire conditional statement is deleted. | | 7 |
| 185 | ILLEGAL RELATION TEST<br>An illegal implication has been detected. The entire conditional statement is deleted. | | 7 |
| 186 | RELEASE/RETURN NOT AN INPUT/OUTPUT PROCEDURE<br>The RELEASE/RETURN statement is deleted. | | 7 |
| 187 | SORT STATEMENT WITHIN INPUT/OUTPUT PROCEDURE – DELETED | | B |
| 188 | SORT KEY NOT IN SORT-FILE RECORD DESCRIPTION<br>The incorrect SORT key specification is ignored. | | B |
| 189 | NO SORT KEYS<br>The SORT statement is deleted. | | B |
| 190 | EXCESSIVE SORT KEYS<br>Excessive SORT keys (the maximum is 16) are ignored. | | B |
| 191 | INVALID REPORT RECORD<br>The GENERATE statement has not referenced a report-name or a report record-name. The statement is deleted. | | B |
| 192 | INVALID DATA REFERENCE – EXPRESSION DELETED<br>An expression operand has not been defined. The expression is deleted. | | B |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 193 | INVALID EXPRESSION OPERAND – EXPRESSION DELETED<br>An illegal arithmetic operand has occurred within an expression. The expression is deleted. | | B |
| 194 | INVALID EXPRESSION – DELETED<br>An expression has been malformed and is deleted. | | B |
| 195 | UNBALANCED EXPRESSION – DELETED<br>An imbalance of operators and operands has been detected in an expression. The expression is deleted. | | B |
| 196 | SUM ADDENDS NOT DEFINED IN A DETAIL OR OTHER SUM ITEM<br>The undefined SUM clause operands are deleted. | | B |
| 197 | INCOMPATIBLE LINE NUMBERS GIVEN IN 'PAGE LIMITS' CLAUSE<br>Compilation is continued in accordance with the dictates of the source program, but erroneous results are likely if the object program is executed. | | B |
| 198 | NO FD, SD ENTRY ASSOCIATED WITH A 'SELECT' CLAUSE – COMPILATION ABORTED | F | |
| 199 | DUPLICATE FD/SD ENTRIES | | B |
| 200 | CONFLICT BETWEEN "ACCESS MODE" AND "ACTUAL KEY" – RANDOM ACCESS ASSUMED | | 6 |
| 201 | CONFLICT BETWEEN "ACCESS MODE" AND "ACTUAL KEY" – SEQUENTIAL ACCESS ASSUMED | | 6 |
| 202 | MAXIMUM ACTUAL KEY SIZE EXCEEDED – 255 CHARACTERS USED | | 6 |
| 203 | "END DECLARATIVES" STATEMENT MISSING | | B |
| 204 | MAXIMUM NUMBER OF SELECT STATEMENTS EXCEEDED – COMPILATION ABORTED | F | |
| 205 | MORE THAN 3 FD'S ASSOCIATED WITH 1 RD – IGNORED | | 7 |
| 206 | VALUE CLAUSE NOT ALLOWED – COMPILATION ABORTED | F | |
| 207 | LEVEL 66 DATA ENTRY BEARS INVALID CLAUSE(S) | | 7 |
| 208 | EXCESSIVE CHARACTERS IN PICTURE STRING – TRUNCATED | | 7 |
| 209 | A "RENAMING" STATEMENT CANNOT BE HONORED | | B |
| 210 | RIGHTMOST AND/OR FRACTIONAL DIGITS TRUNCATED | P | 0 |
| 211 | LEFTMOST DIGITS/CHARACTERS TRUNCATED | P | 0 |
| 212 | INTEGER AND FRACTIONAL DIGITS TRUNCATED | P | 0 |
| 213 | LEVEL 77 ILLEGAL IN FILE SECTION – DATA ENTRY DISCARDED | | 7 |
| 214 | DUPLICATE OR INVALID RD NAME – COMPILATION ABORTED | F | |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 215 | VALUE CLAUSE WITHIN SCOPE OF OCCURS<br>The VALUE clause is not permitted within the scope of an OCCURS clause. | | 7 |
| 216 | OCCURS ILLEGAL ON LEVEL 01 OR 77 | | 7 |
| 217 | DECLARATIVE IS NOT APPROPRIATE ON FILE WITH LABEL RECORDS OMITTED | | 7 |
| 218 | ILLEGAL CONTINUATION CHARACTER – IGNORED<br>An illegal character in column 7 was encountered. | | 4 |
| 219 | DECLARED DATA STORAGE EXCEEDS AVAILABLE CORE STORAGE | | 7 |
| 220 | DUPLICATE DECLARATIVES HAVE BEEN SPECIFIED | | 7 |
| 221 | INTEGER PERFORM COUNT LIMIT OF (2**19)-1 EXCEEDED – VALUE TRUNCATED | | 7 |
| 222 | COMPILER LIMIT OF 9 REPORT CONTROL FIELDS EXCEEDED – COMPILATION ABORTED | F | |
| 223 | USAGE NOT SPECIFIED – NUMERIC DISPLAY ASSUMED | | 4 |
| 224 | KEYED FILE BLOCKING PRE-EMPTED BY MONITOR – CLAUSE IGNORED<br>The BLOCK CONTAINS clause may not be specified for a keyed file. | | 3 |
| 225 | SIZE IN NUMERIC PICTURE GREATER THAN 31 – RESULTS ARE UNPREDICTABLE<br>Numeric items may not exceed PICTURE 9(31). If this is a filler item, change to PICTURE X(n). | | 2 |
| 226 | CAUTION NO RECORD DESCRIBED – VALID IF REPORT CLAUSE PRESENT | P | 0 |
| 227 | WARNING – PROCEDURE NAME PASSED IN ENTER STATEMENT IN AN OVERLAY SEGMENT | P | 0 |
| 228 | OPTION OF DEBUGGING MISSING/INVALID | | 7 |
| 229 | COPY REPLACING STATEMENT INCORRECTLY STRUCTURED | | 7 |
| 230 | DEVICE NOT SPECIFIED – CONSOLE ASSUMED<br>The ACCEPT statement did not specify a device. | | 2 |
| 231 | IDENTIFIER NOT SPECIFIED FOR "ACCEPT" STATEMENT | | 7 |
| 232 | USAGE NOT SPECIFIED – DISPLAY ASSUMED<br>DISPLAY was not specified in a USAGE clause in a report group entry. | | 1 |
| 233 | MAXIMUM DCB SIZE EXCEEDED – 3 INSN/OUTSNS GENERATED<br>The value of "integer" in a SELECT statement is too large; the value of 3 is used. | | B |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 234 | UNDEFINED PROCEDURE NAME – EXTERNAL REFERENCE GENERATED | | 7 |
| 235 | SOURCE INPUT EXCEEDS 72 CHARACTERS – TRUNCATED | | 7 |
| 236 | REMAINDER NOT ALLOWED ON DIVIDE WITH MULTIPLE RECEIVING FIELDS | | 7 |
| 237 | SUBSCRIPTED 'DEPENDING ON' DATA-NAME – COMPILATION ABORTED<br>The data-name in an OCCURS DEPENDING ON clause may not be subscripted. | F | |
| 238 | 'OCCURS DEPENDING ON' ENTRIES EXCEEDED LIMIT 15 – COMPILATION ABORTED<br>A maximum of 15 variable groups is allowed for each record description. | F | |
| 239 | OPTION OF DELIMITED MISSING/INVALID | | 7 |
| 240 | IDENTIFIER MISSING/INVALID AFTER 'IN'/'OF' | | 7 |
| 241 | REQUIRED WORD 'RUN' OR LITERAL MISSING AFTER 'STOP' | | 7 |
| 242 | REQUIRED WORD 'INTO' OR 'BY' MISSING | | 7 |
| 243 | FILE-NAME OR REQUIRED WORD(S) 'REVERSED'/'NO REWIND' MISSING | | 7 |
| 244 | REQUIRED WORD(S) MISSING AFTER 'TALLYING' OR 'REPLACING' | | 7 |
| 245 | REQUIRED WORD 'TALLYING' OR 'REPLACING' MISSING | | 7 |
| 246 | REQUIRED WORD 'TO' MISSING | | 7 |
| 247 | REQUIRED WORD 'DEPENDING' MISSING | | 7 |
| 248 | REQUIRED WORD 'TIMES'/'UNTIL'/'VARYING' MISSING | | 7 |
| 249 | REQUIRED WORD 'INTO'/'END'/'INVALID' MISSING | | 7 |
| 250 | REQUIRED WORD 'FROM' MISSING | | 7 |
| 251 | REQUIRED WORD 'UNTIL' MISSING | | 7 |
| 252 | REQUIRED WORD 'ELSE' MISSING | | 7 |
| 253 | REQUIRED WORD 'WHEN' MISSING | | 7 |
| 254 | REQUIRED WORD(S) 'LOCK' OR 'NO REWIND' MISSING | | 7 |
| 255 | REQUIRED WORD(S) 'TO'/'UP BY'/'DOWN BY' MISSING | | 7 |
| 256 | REQUIRED WORD 'INPUT'/'OUTPUT'/'I-O' MISSING | | 7 |
| 257 | FILE-NAME MISSING | | 7 |
| 258 | LITERAL MISSING AFTER 'ALL'/'LEADING'/'FIRST' | | 7 |

Table 2. Xerox ANS COBOL Compiler Diagnostics (cont.)

| Message Number | Message | Strength | Severity Level |
|---|---|---|---|
| 259 | LITERAL MISSING AFTER 'BY' | | 7 |
| 260 | IDENTIFIER/INTEGER/MNEMONIC-NAME MISSING AFTER 'BEFORE/ AFTER ADVANCING' | | 7 |
| 261 | RECORD-NAME MISSING AFTER 'WRITE' | | 7 |
| 262 | IDENTIFIER/LITERAL/INDEX-NAME MISSING AFTER 'FROM'/ 'TO'/'BY' | | 7 |
| 263 | IDENTIFIER MISSING AFTER 'TO' | | 7 |
| 264 | IDENTIFIER MISSING AFTER 'TO'/'GIVING' | | 7 |
| 265 | IDENTIFIER MISSING AFTER 'COMPUTE' | | 7 |
| 266 | IDENTIFIER MISSING AFTER 'INSPECT' | | 7 |
| 267 | IDENTIFIER MISSING AFTER 'DEPENDING ON' | | 7 |
| 268 | IDENTIFIER MISSING AFTER 'INTO' | | 7 |
| 269 | IDENTIFIER MISSING AFTER 'FROM' | | 7 |
| 270 | IDENTIFIER MISSING AFTER 'SEARCH' | | 7 |
| 271 | IDENTIFIER MISSING AFTER 'ACCEPT' | | 7 |
| 272 | PROCEDURE-NAME MISSING | | 7 |
| 273 | MISSING/INCORRECT STATEMENT AFTER 'AT END'/'INVALID KEY'/'SIZE ERROR' | | 7 |
| 274 | REQUIRED WORD(S) 'NEXT SENTENCE' MISSING | | 7 |
| 275 | SUBROUTINE-NAME MISSING AFTER 'ENTER' | | 7 |
| 276 | IDENTIFIER/LITERAL INVALID OR MISSING | | 7 |
| 277 | IDENTIFIER/INDEX-NAME MISSING AFTER 'VARYING'/'SET'/'AFTER' | | 7 |
| 278 | IDENTIFIER/LITERAL MISSING AFTER 'INTO'/'FROM'/'BY' | | 7 |
| 279 | NUMBER OF USE STATEMENTS EXCEEDS 64–COMPILATION ABORTED | F | |
| 280 | SIZE FOR THIS SECTION HAS EXCEEDED 65K | | 7 |
| 281 | UNDEFINED/INVALID REPORT NAME | | 2 |
| 282 | NOT ENOUGH DYNAMIC MEMORY--COMPILATION ABORTED | F | B |
| 283 | SOURCE IMAGE EXCEEDED 80 CHARACTERS – TRUNCATED | P | 0 |
| 284 | SOURCE IMAGE EXCEEDED 140 CHARACTERS--COMPILATION ABORTED | F | B |
| 285 | INVALID/MISSING REPORT RECORD--COMPILATION ABORTED | F | B |
| 286 | UNDEFINED CONTROL FIELD, IGNORED | | 7 |
| 287 | INVALID DATA USAGE IN CLASS TEST | | 4 |

# 7. RUN-TIME SUBROUTINES, SERVICES AND DIAGNOSTICS

## Library Subroutines

Table 3 shows subroutines contained in the system library that may be referenced by COBOL object programs.

Table 3. COBOL Object Program Subroutines

| Element File | Entry Points | Function |
|---|---|---|
| C:ALT | | ALTER of an overlay segment handler |
| | C:ALT | |
| C:BIS | | Binary search subroutine |
| | C:BIS | |
| C:CBP | | Alphanumeric comparison overlap handler |
| | C:CBP | |
| C:CHKPT | | Checkpoint routines |
| | C:CKP | Record checkpoints |
| | C:INT | !INT key-in entry point |
| | C:MIN | CLOCK-UNITS (minutes) value |
| | C:TIM | Timer interrupt routine |
| C:CONV | | Data conversion subroutines |
| | C:CBD | Binary to packed decimal |
| | C:CDB | Packed decimal to binary |
| | C:CDE | Packed decimal to floating-point short format |
| | C:CDF | Packed decimal to floating-point long format |
| | C:CED | Floating-point short format to packed decimal |
| | C:CFD | Floating-point long format to packed decimal |
| | C:DBD | Binary to unpacked decimal |
| | C:DED | Floating-point short format to unpacked decimal |
| | C:DFD | Floating-point long format to unpacked decimal |
| C:DECL | | I/O label declaratives handler |
| | C:ABF | After beginning file label |
| | C:ABR | After beginning reel label |
| | C:AEF | After ending file label |
| | C:AER | After ending reel label |
| | C:BBF | Before beginning file label |
| | C:BBR | Before beginning reel label |
| | C:BEF | Before ending file label |
| | C:BER | Before ending reel label |
| | C:CLD | Close |

Table 3. COBOL Object Program Subroutines (cont.)

| Element File | Entry Points | Function |
|---|---|---|
| C:DECL (cont.) | C:ERD | Error declaratives |
| | C:OPD | Open |
| | C:RLD | Read |
| | C:WLD | Write |
| C:DPD | | Double precision division |
| | C:DPD | |
| C:DPM | | Double precision multiplication |
| | C:DPM | |
| C:ERR | | Illegitimate control transfers handler |
| | C:ERR | |
| C:EXP | | Interface for exponentiation routines |
| | C:EXP | |
| C:INS | | Run-time routine for INSPECT |
| | C:INS | |
| C:LIO | | Input/output handlers |
| | C:ABA | Abnormal return |
| | C:CIB | Close input buffer |
| | C:CLS | Close a DCB |
| | C:ERA | Error return |
| | C:OPN | Open a DCB |
| | C:RLR | Read |
| | C:WLR | Write |
| | C:WOB | Write output block |
| C:NTS | | Run-time routine for UNSTRING |
| | C:NTS | |
| C:NCRS | | Bypassing co-resident sort |
| | C:SRT | |
| C:RND | | Arithmetic rounding subroutine |
| | C:RND | |
| C:STN | | Run-time routine for STRING |
| | C:STN | |
| C:SZT | | Size error testing |
| | C:SZT | |

Table 3. COBOL Object Program Subroutines (cont.)

| Element File | Entry Points | Function |
|---|---|---|
| C:TRC | | Trace control subroutine |
| | C:TRC | |
| | C:TRX | |
| C:TRP | | Trap handler |
| | C:TRP | Trap processor |
| | C:TRN | Abort suppression flag |
| C:RRG | | Report Writer subroutines |
| | C:RRA | Return point from Declarative routine and GROUP INDICATE presetting |
| | C:RRB | Return point from summing (control footing level) |
| | C:RRC | Return point from print line formation |
| | C:RRD | Return point from SUM counter resetting |
| | C:RRE | Return point from control field preservation |
| | C:RRF | Return point from summing (detail level) |
| | C:RRG | GENERATE entry point |
| | C:RRH | TERMINATE entry point |
| | C:RRI | INITIATE entry point |
| | C:RRJ | Return point from GROUP INDICATE clearing |
| | C:RRK | Entry point for erroneous use of report with no prior INITIATE statement |
| | C:RRQ | Return point from control break testing – no break |
| | C:RRR | Return point upon control break at level 1 |
| | C:RRS | Return point upon control break at level 2 |
| | C:RRT | Return point upon control break at level 3 |
| | C:RRU | Return point upon control break at level 4 |
| | C:RRV | Return point upon control break at level 5 |
| | C:RRW | Return point upon control break at level 6 |
| | C:RRX | Return point upon control break at level 7 |
| | C:RRY | Return point upon control break at level 8 |
| | C:RRZ | Return point upon control break at level 9 |
| C:VPL | | Variable records handler |
| | C:VPL | |

## Subprogram Calls

The ENTER subroutine-name statement as implemented in the Xerox ANS COBOL language causes generation of a calling sequence to the external definition subroutine-name. All such calling sequences are issued in the form of Xerox Standard Calling Sequences:

1. The number of arguments is passed in register 14.

2. The linking register is 15.

Each parameter is represented in the pointer word vector by a single word whose format is

| | | | |
|---|---|---|---|
| 0 | Code | Must be zero | Address |
| 0 | 1 8 | 9 12 | 13 31 |

where

Bit 0 indicates whether the Address field is indirect. (COBOL issues direct references only; thus, this bit is always zero.)

Code field } Address field } are filled as follows:

| Data Type | COBOL Description | Bit Value of Code Field | Contents of Address Field |
|---|---|---|---|
| Binary | INDEX or COMPUTATIONAL | 00000010 | WA (parameter) |
| Floating short | COMPUTATIONAL-1 | 00000100 | WA (parameter) |
| Floating long | COMPUTATIONAL-2 | 00001000 | WA (parameter) |
| Packed decimal | COMPUTATIONAL-3 | 100xxxx0† | BA (parameter) |
| EBCDIC | DISPLAY | 10100000 | BA (parameter) |
| DCB (Data Control Block) | file-name | 10100011 | WA (parameter) |
| Program location | procedure-name | 00000001 | WA (parameter) |

†xxxx (bits 4 through 7) indicates decimal length is in the same format as the Decimal Instructions.

## Special Interfaces to Hardware and Monitor Services

The cababilities described in this section are implemented in the form of assembly language routines that can be added to the run-time library at the user's option. Each routine is independent and any combination of services can be elected for a given installation. In general, these routines contain Xerox defined entry points which the user programmer cites by symbolic name in an ENTER verb in his COBOL syntax. The ENTER for each routine must generally contain a string of data names which (at run-time) contain parameter values defining the nature of the service to be provided. The order of the parameters is strictly defined and it is the user's responsibility to provide the correct values. In effect, these routines are "super-macros" for providing services not available in the Xerox ANS COBOL language.

The specifications below define the service to be provided and the anticipated ENTER syntax required. Some of the individual services may be implemented, as subsections of one run-time routine, thereby requiring a somewhat smaller total number of machine language programs in the run-time library.

The services provided are

1. Delete a record from a keyed file.

   The user's file must have been the subject of an "OPEN INPUT-OUTPUT" statement. In this case, the value to be used as the monitor key will be picked up from the user's data area and used in an M:DELREC call.

   User Syntax:

   <u>ENTER</u> <u>DELETER</u> file-name, data-name-1, data-name-2[, procedure-name]

   where

   DELETER is the entry point in the run-time routine.

   file-name is the appropriate FD name.

   data-name-1 contains the monitor record key (DISPLAY).

   data-name-2 contains the length of data-name-1 (COMPUTATIONAL).

   procedure-name is to be executed if the specified key is not found in the file. This parameter is optional.

2. Get the monitor key and actual record size of the last processed record in a file.

   The required values are abstracted from the KBUF and ARS areas of the DCB.

   User Syntax:

   <u>ENTER</u> <u>LASTKEY</u> file-name, data-name-1, data-name-2[, data-name-3]

   where

   LASTKEY is the entry point in the run-time routine.

   file-name is the appropriate FD name.

   data-name-1 is the area into which the monitor key of the last record read or written will be inserted (DISPLAY).

   data-name-2 is the area into which the length of the monitor key will be inserted (COMPUTATIONAL). This value will be zero if no key was found.

   data-name-3 is the area into which the actual size, in bytes, of the last record read or written will be inserted (COMPUTATIONAL). This parameter is optional.

3. Set a file to keyed sequential access and position to a specified key value. The user who wishes to process a keyed file sequentially is required to specify "ACCESS IS SEQUENTIAL". The user is expected to open the file as sequential and then call upon this routine to redefine it as "keyed sequential". The file will then be positioned to the specified key value and a return will be made to the user program where reading will proceed sequentially. If a record exists whose key matches the specified value, it will be the first record accessed by the next sequential read. If the specified key is not in the file, the next sequential read will access the first record with a higher key value.

User Syntax:

ENTER START file-name, data-name-1, data-name-2

where

START is the entry point in the run-time routine.

file-name is the appropriate FD name (SEQUENTIAL).

data-name-1 contains the monitor record key (DISPLAY).

data-name-2 contains the length of data-name-1 (COMPUTATIONAL).

Note that the redefinition of the consecutive file to keyed sequential occurs by executing an M:CLOSE and then an M:OPEN.

4. Skip n records in a file.

Uses the monitor PRECORD routine to skip forward over the specified number of records. If the file is monitor formatted, n logical records will be bypassed; if user formatted, n physical records will be skipped. This routine does not allow skipping from the middle of a physical record in a user-formatted file, nor does it account for blocks already in memory due to double buffering. The user is responsible for reducing his skip count in such a situation.

User Syntax:

ENTER SKIP file-name, data-name-1[, procedure-name]

where

SKIP is the entry point in the run-time routine.

file-name is the appropriate FD name.

data-name-1 contains the number of records to skip (COMPUTATIONAL). A negative number indicates reversed skipping. The number of records yet to be skipped will be stored in data-name-1 upon completion.

procedure-name is to be executed if either of the following abnormal conditions occurs: end-of-file, end-of-tape (user-formatted file). The number of records yet to be skipped is placed in the actual record size field (ARS) of the associated DCB. This parameter is optional.

5. Close and release disk file to the monitor.

This routine closes and releases disk files to the monitor when the COBOL programmer is through using them.

User Syntax:

ENTER RELFILES file-name-1, file-name-2, ..., file-name-n

where

RELFILES is the entry point in the run-time routine.

file-name-1 is an appropriate FD name.

:

file-name-n is the last of multiple files to be closed and released.

6. Get current date, time, and sense switch settings.

This routine picks up the current date, time, and sense switch settings, and makes them available to the user program. The routine optionally picks up the current number of lines per page from a printer DCB. This value is set by the (LINES, value) parameter in the !ASSIGN card.

User Syntax:

<u>ENTER</u> <u>GETCOM</u> data-name-1[, print-file-name]

where

GETCOM is the entry point in the run-time routine.

data-name-1 is a 26-byte area (DISPLAY) into which will be inserted the following information:

| | |
|---|---|
| bytes 1-6 | the pseudo-switch settings; 0 is off, 1 is on. |
| bytes 7-8 | blank. |
| bytes 9-24 | time and date, in the monitor form HH:MM MON DD, 'YY (hours, minutes, month, day, year). |
| bytes 25-26 | number of lines per page in the printer DCB. |

print-file-name is the appropriate FD name corresponding to the printer DCB.

7. Transform data to new collating sequence.

Allows the user to translate up to 255 bytes of data to any specified collating sequence. The user is responsible for constructing a 256-byte table containing the target collating sequence. The run-time routine uses the Translate Byte String instruction to accomplish the transformation. The target translation table is defined by the user in much the same way that key translation is specified in the Sigma Sort.

User Syntax:

<u>ENTER</u> <u>TRANSFORM</u> data-name-1, data-name-2, data-name-3

where

TRANSFORM is the entry point in the run-time routine.

data-name-1 contains up to 255 bytes of data to be transformed (DISPLAY).

data-name-2 contains the actual length of the byte string to be transformed (COMPUTATIONAL).

data-name-3 is a 256-byte translation table containing the target collating sequence. Data-name-1 and data-name-3 must start on word boundaries.

8. Set a data area to zero.

Allows the user to background large data areas to EBCDIC zero ('F0').

User Syntax:

<u>ENTER</u> <u>SETZERO</u> data-name-1, data-name-2

where

SETZERO is the entry point in the run-time routine.

data-name-1 is the area to set to zero (DISPLAY).

data-name-2 is the byte length of the area to be set to zero (COMPUTATIONAL). Maximum value is 32,767.

9. Set a data area to blanks.

   Allows the user to background large data areas to EBCDIC blank ('40').

   User Syntax:

   ENTER SETBLANK data-name-1, data-name-2

   where

   | | |
   |---|---|
   | SETBLANK | is the entry point in the run-time routine. |
   | data-name-1 | is the area to be set to blanks (DISPLAY). |
   | data-name-2 | is the byte length of the area to be set to blanks (COMPUTATIONAL). Maximum value is 32,767. |

10. Signal operator to change printer form or punch card stock.

    Allows the user to request a change in the form used on the specified output device (card punch or line printer). Any message, up to 255 bytes long, may be inserted into the output symbiont stream. The message, generally directions to the computer operator, is automatically intercepted at actual print (or punch) time, directed to the operator's console, and the output symbiont is suspended. Upon performance of the action specified in the user programmer's message, the symbiont can be restarted and printing or punching continued. Note that a second message is required later to cause restoration of a "standard" form for the next job.

    User Syntax:

    ENTER FORMESS file-name, data-name-1, data-name-2

    where

    | | |
    |---|---|
    | FORMESS | is the entry point in the run-time routine. |
    | file-name | is the appropriate output FD name. |
    | data-name-1 | is the message to be inserted in the print or punch output symbiont (DISPLAY). |
    | data-name-2 | is the length of the message (COMPUTATIONAL). Maximum value is 255. |

## COBOL Error Codes

In addition to the error and abnormal returns documented in the appropriate BPM/CP-V monitor reference manuals, code numbers 80 through 9F (hexadecimal) are reserved for the COBOL compiler and object programs. Table 4 defines these codes.

Table 4. COBOL Error Codes

| Code (Hexadecimal) | Procedure Name | Meaning |
|---|---|---|
| 01 | OPEN | Opening a DCB with insufficient information. |
| 03 | OPEN | Nonexistent name. |
| 04 | PRECORD READ | Beginning of file. |
| 07 | READ | Lost data (buffer size smaller than record read). |

Table 4. COBOL Error Codes (cont.)

| Code (Hexadecimal) | Procedure Name | Meaning |
|---|---|---|
| 0A | CLOSE | Closing an unopened file. |
| 13 | WRITE<br>DELREC | Requested key not found on an UPDATE file. |
| 14 | OPEN | Insufficient information to identify a file. |
| 15 | WRITE<br>DELREC | Illegal sequence of operations on an INOUT file. |
| 16 | WRITE | NEWKEY option specified on already existing key. |
| 17 | WRITE | NEWKEY option not specified on key for OUT or OUTIN files. |
| 18 | WRITE | KEY not in proper sequence. |
| 1C | READ<br>WRITE<br>PRECORD | End of tape. |
| 1D | READ<br>PRECORD | Beginning of tape. |
| 2E | OPEN | Opening an open file. |
| 80 | READ/WRITE | Request to READ/WRITE an unopened file. |
| 82 | OPEN | Unable to obtain dynamic area for blocking/deblocking. |
| 86 | READ | Logical Record read is larger than maximum size (MAXSIZE) specification in COBOL program. |
| 87 | READ | The sum of the record prefix count is not equal to the block prefix count. |
| 88 | READ | The block prefix count does not agree with the actual record size read by the system (ARS). |
| 89 | WRITE | User is attempting to write a logical record that is too large for his blocking buffer (a logical record cannot be greater than eight bytes smaller than maximum blocksize). |
| 8C | GCP [1] | Common page not available. |
| 8D | Object Pro-gram Sort [2] | Sort error. |
| 8F | OPEN | Opening a locked file. |
| 90 | Report Writer [3]<br>GENERATE<br>TERMINATE | Report not initiated. |
| 91 | Report Writer [3]<br>INITIATE | Report already initiated. |

Table 4. COBOL Error Codes (cont.)

| Code (Hexadecimal) | Procedure Name | Meaning |
|---|---|---|
| 92 | Report Writer[3] | Incompatible line spacing. |
| 99 | C:VPL | Value of data-name in OCCURS DEPENDING ON clause exceeds the maximum specified. |
| 9A | Object Program ENTER | Invalid calling sequence for run-time library subroutine. |
| 9E | Object Program[4] | Erroneous transfer of control. |
| 9F | Compiler | Internal I/O errors. |

Notes:

1 SR1 contains zero.

2 R6 contains one of the following values:

| | |
|---|---|
| 01 | (Sort error – in and out record count out of balance) |
| 02 | (Sort aborted – I/O error) |
| 03 | (Sort aborted – specification error) |
| 04 | (Sort aborted – registers give reason) |
| 05 | (Sort aborted – memory overflow) |
| 06 | (Sort aborted – illegal own-code action request) |
| 07 | (Reserved for future use) |
| 08 | (Sort aborted – illegal decimal key) |
| 09 | (Sort error – sequence error in output file) |

3 SR1 contains location of call to C:RRG; SR3 (bytes 1 through 3) contains address of Report Table (R:report-name).

4 No register settings are significant.

The only COBOL run-time diagnostic is of the form

```
PROGRAM ABORTED--ERROR CODE nn nn

xxxxx IS FD-NAME

REL. INST. LOCATION IS yyyyyyyy
```

where

| | |
|---|---|
| nn nn | is the appropriate 4-digit error code and subcode number. |
| xxxxx | is the name of the file. |
| yyyyyyyy | is the relative location of the instruction causing the error. |

In each case that an error is incurred, the action taken is to abort the current job. The STEP condition code is set to 6. (Refer to the CP-V Batch Processing Reference Manual, 90 17 64, for the STEP command.)

The error code is contained in byte 0 of SR3. Except where footnoted above, the DCB address is contained in bytes 1 through 3 of SR3 and the location following the associated CAL1 instruction is communicated in SR1.

# APPENDIX. REFERENCE TABLES

This appendix contains the following reference material:

## STANDARD SYMBOLS AND CODES

The symbol and code standards described in this publication are applicable to all Xerox computer products, both hardware and software. They may be expanded or altered from time to time to meet changing requirements.

The symbols listed here include two types: graphic symbols and control characters. Graphic symbols are displayable and printable; control characters are not. Hybrids are SP, the symbol for a blank space; and DEL, the delete code, which is not considered a control command.

Three types of code are shown: (1) the 8-bit Xerox Standard Computer Code, i.e., the Extended Binary-Coded-Decimal Interchange Code (EBCDIC); (2) the 7-bit American National Standard Code for Information Interchange (ANSCII); and (3) the Xerox standard card code.

## STANDARD CHARACTER SETS

1. EBCDIC

   57-character set: uppercase letters, numerals, space, and & - / . < > ( ) + | $ * : ; , % # @ ' =

   63-character set: same as above plus ¢ ! _ ? " ¬

   89-character set: same as 63-character set plus lowercase letters

2. ANSCII

   64-character set: uppercase letters, numerals, space, and ! " $ % & ' ( ) * + , - . / \ ; : = < > ? @ [ ] ^ #

   95-character set: same as above plus lowercase letters and { } ¦ ~ `

## CONTROL CODES

In addition to the standard character sets listed above, the symbol repertoire includes 37 control codes and the hybrid code DEL (hybrid code SP is considered part of all character sets). These are listed in the table titled Standard Symbol-Code Correspondences.

## SPECIAL CODE PROPERTIES

The following two properties of all standard codes will be retained for future standard code extensions:

1. All control codes, and only the control codes, have their two high-order bits equal to "00". DEL is not considered a control code.

2. No two graphic EBCDIC codes have their seven low-order bits equal.

## STANDARD 8-BIT COMPUTER CODES (EBCDIC)

|  |  | Most Significant Digits |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hexadecimal |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|  | Binary | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| Least Significant Digits 0 | 0000 | NUL | DLE | ds |  | SP | & | - | ▨ |  |  |  |  |  |  |  | 0 |
| 1 | 0001 | SOH | DC1 | ss |  | ▨ | ▨ | / | ▨ | a | j |  | \\ [1] | A | J |  | 1 |
| 2 | 0010 | STX | DC2 | fs |  | ▨ | ▨ | ▨ | ▨ | b | k | s | { [1] | B | K | S | 2 |
| 3 | 0011 | ETX | DC3 | si |  | ▨ | ▨ | ▨ | ▨ | c | l | t | } [1] | C | L | T | 3 |
| 4 | 0100 | EOT | DC4 |  |  | ▨ | ▨ | ▨ | ▨ | d | m | u | [ [1] | D | M | U | 4 |
| 5 | 0101 | HT | LF NL |  |  | Will not be assigned |  |  |  | e | n | v | ] [1] | E | N | V | 5 |
| 6 | 0110 | ACK | SYN |  |  | ▨ | ▨ | ▨ | ▨ | f | o | w |  | F | O | W | 6 |
| 7 | 0111 | BEL | ETB |  |  | ▨ | ▨ | ▨ | ▨ | g | p | x |  | G | P | X | 7 |
| 8 | 1000 | EOM BS | CAN |  |  | ▨ | ▨ | ▨ | ▨ | h | q | y |  | H | Q | Y | 8 |
| 9 | 1001 | ENQ | EM |  |  | ▨ | ▨ | ▨ | ▨ | i | r | z |  | I | R | Z | 9 |
| A | 1010 | NAK | SUB |  |  | ¢ [2] | ! | ^ [1] | : |  |  |  |  | ▨ | ▨ | ▨ | ▨ |
| B | 1011 | VT | ESC |  |  | . | $ | , | # |  |  |  |  | ▨ | ▨ | ▨ | ▨ |
| C | 1100 | FF | FS |  |  | < | * | % | @ |  |  |  |  | Will not be assigned |  |  |  |
| D | 1101 | CR | GS |  |  | ( | ) | _ | ' |  |  |  |  | ▨ | ▨ | ▨ | ▨ |
| E | 1110 | SO | RS |  |  | + | ; | > | = |  |  |  |  | ▨ | ▨ | ▨ | ▨ |
| F | 1111 | SI | US |  |  | \| [2] | ¬ [2] | ? | " |  |  |  |  | ▨ | ▨ | ▨ | DEL |
|  |  | 3 |  |  |  | 4 |  |  |  | 5 |  |  |  |  |  |  |  |

NOTES:

1 The characters ^ \ { } [ ] are ANSCII characters that do not appear in any of the EBCDIC-based character sets, though they are shown in the EBCDIC table.

2 The characters ¢ | ¬ appear in the 63- and 89-character EBCDIC sets but not in either of the ANSCII-based sets. However, Xerox software translates the characters c into ANSCII characters as follows:

| EBCDIC | = | ANSCII |
|---|---|---|
| ¢ |  | ` (6-0) |
| \| |  | ¦ (7-12) |
| ¬ |  | ~ (7-14) |

3 The EBCDIC control codes in columns 0 and 1 and their binary representation are exactly the same as those in the ANSCII table, except for two interchanges: LF/NL with NAK, and HT with ENQ.

4 Characters enclosed in heavy lines are included only in the standard 63- and 89-character EBCDIC sets.

5 These characters are included only in the standard 89-character EBCDIC set.

## STANDARD 7-BIT COMMUNICATION CODES (ANSCII)[1]

|  |  | Most Significant Digits |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|
| Decimal (rows) | (col's.) → | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| ↓ | Binary [1] | x000 | x001 | x010 | x011 | x100 | x101 | x110 | x111 |
| Least Significant Digits 0 | 0000 | NUL | DLE | SP | 0 | @ | P | ` | p |
| 1 | 0001 | SOH | DC1 | ! [5] | 1 | A | Q | a | q |
| 2 | 0010 | STX | DC2 | " | 2 | B | R | b | r |
| 3 | 0011 | ETX | DC3 | # | 3 | C | S | c | s |
| 4 | 0100 | EOT | DC4 | $ | 4 | D | T | d | t |
| 5 | 0101 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | 0110 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | 0111 | BEL | ETB | ' | 7 | G | W | g | w |
| 8 | 1000 | BS | CAN | ( | 8 | H | X | h | x |
| 9 | 1001 | HT | EM | ) | 9 | I | Y | i | y |
| 10 | 1010 | LF NL | SUB | * | : | J | Z | j | z |
| 11 | 1011 | VT | ESC | + | ; | K | [ [5] | k | { |
| 12 | 1100 | FF | FS | , | < | L | \\ | l | ¦ |
| 13 | 1101 | CR | GS | - |  | M | ] [5] | m | } |
| 14 | 1110 | SO | RS | . | > | N | [4] ^ [5] | n | ~ [4] |
| 15 | 1111 | SI | US | / | ? | O | _ [4] | ·o | DEL |
|  |  | 2 |  | 3 |  |  |  |  |  |

NOTES:

1 Most significant bit, added for 8-bit format, is either 0 or even parity.

2 Columns 0-1 are control codes.

3 Columns 2-5 correspond to the 64-character ANSCII set.
Columns 2-7 correspond to the 95-character ANSCII set.

4 On many current teletypes, the symbol

| | | | |
|---|---|---|---|
| ^ | is | ↑ | (5-14) |
| _ | is | ← | (5-15) |
| ~ | is | ESC or ALTMODE control | (7-14) |

and none of the symbols appearing in columns 6-7 are provided. Except for the three symbol differences noted above, therefore, such teletypes provide all the characters in the 64-character ANSCII set. (The Xerox 7015 Remote Keyboard Printer provides the 64-character ANSCII set also, but prints ^ as Λ.)

5 On the Xerox 7670 Remote Batch Terminal, the symbol

| | | | |
|---|---|---|---|
| ! | is | \| | (2-1) |
| [ | is | ¢ | (5-11) |
| ] | is | ! | (5-13) |
| ^ | is | ¬ | (5-14) |

and none of the symbols appearing in columns 6-7 are provided. Except for the four symbol differences noted above, therefore, this terminal provides all the characters in the 64-character ANSCII set.

# STANDARD SYMBOL-CODE CORRESPONDENCES

| EBCDIC[†] | | Symbol | Card Code | ANSCII[††] | Meaning | Remarks |
|---|---|---|---|---|---|---|
| Hex. | Dec. | | | | | |
| 00 | 0 | NUL | 12-0-9-8-1 | 0-0 | null | 00 through 23 and 2F are control codes. |
| 01 | 1 | SOH | 12-9-1 | 0-1 | start of header | |
| 02 | 2 | STX | 12-9-2 | 0-2 | start of text | |
| 03 | 3 | ETX | 12-9-3 | 0-3 | end of text | |
| 04 | 4 | EOT | 12-9-4 | 0-4 | end of transmission | |
| 05 | 5 | HT | 12-9-5 | 0-9 | horizontal tab | |
| 06 | 6 | ACK | 12-9-6 | 0-6 | acknowledge (positive) | |
| 07 | 7 | BEL | 12-9-7 | 0-7 | bell | |
| 08 | 8 | BS or EOM | 12-9-8 | 0-8 | backspace or end of message | EOM is used only on Xerox Keyboard/Printers Models 7012, 7020, 8091, and 8092. |
| 09 | 9 | ENQ | 12-9-8-1 | 0-5 | enquiry | |
| 0A | 10 | NAK | 12-9-8-2 | 1-5 | negative acknowledge | |
| 0B | 11 | VT | 12-9-8-3 | 0-11 | vertical tab | |
| 0C | 12 | FF | 12-9-8-4 | 0-12 | form feed | |
| 0D | 13 | CR | 12-9-8-5 | 0-13 | carriage return | |
| 0E | 14 | SO | 12-9-8-6 | 0-14 | shift out | |
| 0F | 15 | SI | 12-9-8-7 | 0-15 | shift in | |
| 10 | 16 | DLE | 12-11-9-8-1 | 1-0 | data link escape | |
| 11 | 17 | DC1 | 11-9-1 | 1-1 | device control 1 | |
| 12 | 18 | DC2 | 11-9-2 | 1-2 | device control 2 | |
| 13 | 19 | DC3 | 11-9-3 | 1-3 | device control 3 | |
| 14 | 20 | DC4 | 11-9-4 | 1-4 | device control 4 | |
| 15 | 21 | LF or NL | 11-9-5 | 0-10 | line feed or new line | |
| 16 | 22 | SYN | 11-9-6 | 1-6 | sync | |
| 17 | 23 | ETB | 11-9-7 | 1-7 | end of transmission block | |
| 18 | 24 | CAN | 11-9-8 | 1-8 | cancel | |
| 19 | 25 | EM | 11-9-8-1 | 1-9 | end of medium | |
| 1A | 26 | SUB | 11-9-8-2 | 1-10 | substitute | Replaces characters with parity error. |
| 1B | 27 | ESC | 11-9-8-3 | 1-11 | escape | |
| 1C | 28 | FS | 11-9-8-4 | 1-12 | file separator | |
| 1D | 29 | GS | 11-9-8-5 | 1-13 | group separator | |
| 1E | 30 | RS | 11-9-8-6 | 1-14 | record separator | |
| 1F | 31 | US | 11-9-8-7 | 1-15 | unit separator | |
| 20 | 32 | ds | 11-0-9-8-1 | | digit selector | 20 through 23 are used with Sigma EDIT BYTE STRING (EBS) instruction – not input/output control codes. |
| 21 | 33 | ss | 0-9-1 | | significance start | |
| 22 | 34 | fs | 0-9-2 | | field separation | |
| 23 | 35 | si | 0-9-3 | | immediate significance start | |
| 24 | 36 | | 0-9-4 | | | 24 through 2E are unassigned. |
| 25 | 37 | | 0-9-5 | | | |
| 26 | 38 | | 0-9-6 | | | |
| 27 | 39 | | 0-9-7 | | | |
| 28 | 40 | | 0-9-8 | | | |
| 29 | 41 | | 0-9-8-1 | | | |
| 2A | 42 | | 0-9-8-2 | | | |
| 2B | 43 | | 0-9-8-3 | | | |
| 2C | 44 | | 0-9-8-4 | | | |
| 2D | 45 | | 0-9-8-5 | | | |
| 2E | 46 | | 0-9-8-6 | | | |
| 2F | 47 | | 0-9-8-7 | | | |
| 30 | 48 | | 12-11-0-9-8-1 | | | 30 through 3F are unassigned. |
| 31 | 49 | | 9-1 | | | |
| 32 | 50 | | 9-2 | | | |
| 33 | 51 | | 9-3 | | | |
| 34 | 52 | | 9-4 | | | |
| 35 | 53 | | 9-5 | | | |
| 36 | 54 | | 9-6 | | | |
| 37 | 55 | | 9-7 | | | |
| 38 | 56 | | 9-8 | | | |
| 39 | 57 | | 9-8-1 | | | |
| 3A | 58 | | 9-8-2 | | | |
| 3B | 59 | | 9-8-3 | | | |
| 3C | 60 | | 9-8-4 | | | |
| 3D | 61 | | 9-8-5 | | | |
| 3E | 62 | | 9-8-6 | | | |
| 3F | 63 | | 9-8-7 | | | |

[†]Hexadecimal and decimal notation.

[††]Decimal notation (column-row).

## STANDARD SYMBOL-CODE CORRESPONDENCES (cont.)

| EBCDIC[†] | | Symbol | Card Code | ANSCII[††] | Meaning | Remarks |
|---|---|---|---|---|---|---|
| Hex. | Dec. | | | | | |
| 40 | 64 | SP | blank | 2-0 | blank | |
| 41 | 65 | | 12-0-9-1 | | | 41 through 49 will not be assigned. |
| 42 | 66 | | 12-0-9-2 | | | |
| 43 | 67 | | 12-0-9-3 | | | |
| 44 | 68 | | 12-0-9-4 | | | |
| 45 | 69 | | 12-0-9-5 | | | |
| 46 | 70 | | 12-0-9-6 | | | |
| 47 | 71 | | 12-0-9-7 | | | |
| 48 | 72 | | 12-0-9-8 | | | |
| 49 | 73 | | 12-8-1 | | | |
| 4A | 74 | ¢ or ` | 12-8-2 | 6-0 | cent or accent grave | Accent grave used for left single quote. On model 7670, ` not available, and ¢ = ANSCII 5-11. |
| 4B | 75 | . | 12-8-3 | 2-14 | period | |
| 4C | 76 | < | 12-8-4 | 3-12 | less than | |
| 4D | 77 | ( | 12-8-5 | 2-8 | left parenthesis | |
| 4E | 78 | + | 12-8-6 | 2-11 | plus | |
| 4F | 79 | \| or ¦ | 12-8-7 | 7-12 | vertical bar or broken bar | On Model 7670, ¦ not available, and \| = ANSCII 2-1. |
| 50 | 80 | & | 12 | 2-6 | ampersand | |
| 51 | 81 | | 12-11-9-1 | | | 51 through 59 will not be assigned. |
| 52 | 82 | | 12-11-9-2 | | | |
| 53 | 83 | | 12-11-9-3 | | | |
| 54 | 84 | | 12-11-9-4 | | | |
| 55 | 85 | | 12-11-9-5 | | | |
| 56 | 86 | | 12-11-9-6 | | | |
| 57 | 87 | | 12-11-9-7 | | | |
| 58 | 88 | | 12-11-9-8 | | | |
| 59 | 89 | | 11-8-1 | | | |
| 5A | 90 | ! | 11-8-2 | 2-1 | exclamation point | On Model 7670, ! is \|. |
| 5B | 91 | $ | 11-8-3 | 2-4 | dollars | |
| 5C | 92 | * | 11-8-4 | 2-10 | asterisk | |
| 5D | 93 | ) | 11-8-5 | 2-9 | right parenthesis | |
| 5E | 94 | ; | 11-8-6 | 3-11 | semicolon | |
| 5F | 95 | ~ or ¬ | 11-8-7 | 7-14 | tilde or logical not | On Model 7670, ~ is not available, and ¬ = ANSCII 5-14. |
| 60 | 96 | - | 11 | 2-13 | minus, dash, hyphen | |
| 61 | 97 | / | 0-1 | 2-15 | slash | |
| 62 | 98 | | 11-0-9-2 | | | 62 through 69 will not be assigned. |
| 63 | 99 | | 11-0-9-3 | | | |
| 64 | 100 | | 11-0-9-4 | | | |
| 65 | 101 | | 11-0-9-5 | | | |
| 66 | 102 | | 11-0-9-6 | | | |
| 67 | 103 | | 11-0-9-7 | | | |
| 68 | 104 | | 11-0-9-8 | | | |
| 69 | 105 | | 0-8-1 | | | |
| 6A | 106 | ⁀ | 12-11 | 5-14 | circumflex | On Model 7670 ⁀ is ¬. On Model 7015 ⁀ is ∧ (caret). |
| 6B | 107 | , | 0-8-3 | 2-12 | comma | |
| 6C | 108 | % | 0-8-4 | 2-5 | percent | |
| 6D | 109 | _ | 0-8-5 | 5-15 | underline | Underline is sometimes called "break character"; may be printed along bottom of character line. |
| 6E | 110 | > | 0-8-6 | 3-14 | greater than | |
| 6F | 111 | ? | 0-8-7 | 3-15 | question mark | |
| 70 | 112 | | 12-11-0 | | | 70 through 79 will not be assigned. |
| 71 | 113 | | 12-11-0-9-1 | | | |
| 72 | 114 | | 12-11-0-9-2 | | | |
| 73 | 115 | | 12-11-0-9-3 | | | |
| 74 | 116 | | 12-11-0-9-4 | | | |
| 75 | 117 | | 12-11-0-9-5 | | | |
| 76 | 118 | | 12-11-0-9-6 | | | |
| 77 | 119 | | 12-11-0-9-7 | | | |
| 78 | 120 | | 12-11-0-9-8 | | | |
| 79 | 121 | | 8-1 | | | |
| 7A | 122 | : | 8-2 | 3-10 | colon | |
| 7B | 123 | # | 8-3 | 2-3 | number | |
| 7C | 124 | @ | 8-4 | 4-0 | at | |
| 7D | 125 | ' | 8-5 | 2-7 | apostrophe (right single quote) | |
| 7E | 126 | = | 8-6 | 3-13 | equals | |
| 7F | 127 | " | 8-7 | 2-2 | quotation mark | |

[†]Hexadecimal and decimal notation.

[††]Decimal notation (column-row).

| EBCDIC† | | Symbol | Card Code | ANSCII†† | Meaning | Remarks |
|---|---|---|---|---|---|---|
| Hex. | Dec. | | | | | |
| 80 | 128 | | 12-0-8-1 | | | 80 is unassigned. |
| 81 | 129 | a | 12-0-1 | 6-1 | | 81-89, 91-99, A2-A9 comprise the |
| 82 | 130 | b | 12-0-2 | 6-2 | | lowercase alphabet. Available |
| 83 | 131 | c | 12-0-3 | 6-3 | | only in standard 89- and 95- |
| 84 | 132 | d | 12-0-4 | 6-4 | | character sets. |
| 85 | 133 | e | 12-0-5 | 6-5 | | |
| 86 | 134 | f | 12-0-6 | 6-6 | | |
| 87 | 135 | g | 12-0-7 | 6-7 | | |
| 88 | 136 | h | 12-0-8 | 6-8 | | |
| 89 | 137 | i | 12-0-9 | 6-9 | | |
| 8A | 138 | | 12-0-8-2 | | | 8A through 90 are unassigned. |
| 8B | 139 | | 12-0-8-3 | | | |
| 8C | 140 | | 12-0-8-4 | | | |
| 8D | 141 | | 12-0-8-5 | | | |
| 8E | 142 | | 12-0-8-6 | | | |
| 8F | 143 | | 12-0-8-7 | | | |
| 90 | 144 | | 12-11-8-1 | | | |
| 91 | 145 | j | 12-11-1 | 6-10 | | |
| 92 | 146 | k | 12-11-2 | 6-11 | | |
| 93 | 147 | l | 12-11-3 | 6-12 | | |
| 94 | 148 | m | 12-11-4 | 6-13 | | |
| 95 | 149 | n | 12-11-5 | 6-14 | | |
| 96 | 150 | o | 12-11-6 | 6-15 | | |
| 97 | 151 | p | 12-11-7 | 7-0 | | |
| 98 | 152 | q | 12-11-8 | 7-1 | | |
| 99 | 153 | r | 12-11-9 | 7-2 | | |
| 9A | 154 | | 12-11-8-2 | | | 9A through A1 are unassigned. |
| 9B | 155 | | 12-11-8-3 | | | |
| 9C | 156 | | 12-11-8-4 | | | |
| 9D | 157 | | 12-11-8-5 | | | |
| 9E | 158 | | 12-11-8-6 | | | |
| 9F | 159 | | 12-11-8-7 | | | |
| A0 | 160 | | 11-0-8-1 | | | |
| A1 | 161 | | 11-0-1 | | | |
| A2 | 162 | s | 11-0-2 | 7-3 | | |
| A3 | 163 | t | 11-0-3 | 7-4 | | |
| A4 | 164 | u | 11-0-4 | 7-5 | | |
| A5 | 165 | v | 11-0-5 | 7-6 | | |
| A6 | 166 | w | 11-0-6 | 7-7 | | |
| A7 | 167 | x | 11-0-7 | 7-8 | | |
| A8 | 168 | y | 11-0-8 | 7-9 | | |
| A9 | 169 | z | 11-0-9 | 7-10 | | |
| AA | 170 | | 11-0-8-2 | | | AA through B0 are unassigned. |
| AB | 171 | | 11-0-8-3 | | | |
| AC | 172 | | 11-0-8-4 | | | |
| AD | 173 | | 11-0-8-5 | | | |
| AE | 174 | | 11-0-8-6 | | | |
| AF | 175 | | 11-0-8-7 | | | |
| B0 | 176 | | 12-11-0-8-1 | | | |
| B1 | 177 | \ | 12-11-0-1 | 5-12 | backslash | |
| B2 | 178 | { | 12-11-0-2 | 7-11 | left brace | |
| B3 | 179 | } | 12-11-0-3 | 7-13 | right brace | |
| B4 | 180 | [ | 12-11-0-4 | 5-11 | left bracket | On Model 7670, [ is ¢. |
| B5 | 181 | ] | 12-11-0-5 | 5-13 | right bracket | On Model 7670, ] is !. |
| B6 | 182 | | 12-11-0-6 | | | B6 through BF are unassigned. |
| B7 | 183 | | 12-11-0-7 | | | |
| B8 | 184 | | 12-11-0-8 | | | |
| B9 | 185 | | 12-11-0-9 | | | |
| BA | 186 | | 12-11-0-8-2 | | | |
| BB | 187 | | 12-11-0-8-3 | | | |
| BC | 188 | | 12-11-0-8-4 | | | |
| BD | 189 | | 12-11-0-8-5 | | | |
| BE | 190 | | 12-11-0-8-6 | | | |
| BF | 191 | | 12-11-0-8-7 | | | |

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

## STANDARD SYMBOL-CODE CORRESPONDENCES (cont.)

| EBCDIC[†] | | Symbol | Card Code | ANSCII[††] | Meaning | Remarks |
|---|---|---|---|---|---|---|
| Hex. | Dec. | | | | | |
| C0 | 192 | | 12-0 | | | C0 is unassigned. |
| C1 | 193 | A | 12-1 | 4-1 | | C1-C9, D1-D9, E2-E9 comprise the uppercase alphabet. |
| C2 | 194 | B | 12-2 | 4-2 | | |
| C3 | 195 | C | 12-3 | 4-3 | | |
| C4 | 196 | D | 12-4 | 4-4 | | |
| C5 | 197 | E | 12-5 | 4-5 | | |
| C6 | 198 | F | 12-6 | 4-6 | | |
| C7 | 199 | G | 12-7 | 4-7 | | |
| C8 | 200 | H | 12-8 | 4-8 | | |
| C9 | 201 | I | 12-9 | 4-9 | | |
| CA | 202 | | 12-0-9-8-2 | | | CA through CF will not be assigned. |
| CB | 203 | | 12-0-9-8-3 | | | |
| CC | 204 | | 12-0-9-8-4 | | | |
| CD | 205 | | 12-0-9-8-5 | | | |
| CE | 206 | | 12-0-9-8-6 | | | |
| CF | 207 | | 12-0-9-8-7 | | | |
| D0 | 208 | | 11-0 | | | D0 is unassigned. |
| D1 | 209 | J | 11-1 | 4-10 | | |
| D2 | 210 | K | 11-2 | 4-11 | | |
| D3 | 211 | L | 11-3 | 4-12 | | |
| D4 | 212 | M | 11-4 | 4-13 | | |
| D5 | 213 | N | 11-5 | 4-14 | | |
| D6 | 214 | O | 11-6 | 4-15 | | |
| D7 | 215 | P | 11-7 | 5-0 | | |
| D8 | 216 | Q | 11-8 | 5-1 | | |
| D9 | 217 | R | 11-9 | 5-2 | | |
| DA | 218 | | 12-11-9-8-2 | | | DA through DF will not be assigned. |
| DB | 219 | | 12-11-9-8-3 | | | |
| DC | 220 | | 12-11-9-8-4 | | | |
| DD | 221 | | 12-11-9-8-5 | | | |
| DE | 222 | | 12-11-9-8-6 | | | |
| DF | 223 | | 12-11-9-8-7 | | | |
| E0 | 224 | | 0-8-2 | | | E0, E1 are unassigned. |
| E1 | 225 | | 11-0-9-1 | | | |
| E2 | 226 | S | 0-2 | 5-3 | | |
| E3 | 227 | T | 0-3 | 5-4 | | |
| E4 | 228 | U | 0-4 | 5-5 | | |
| E5 | 229 | V | 0-5 | 5-6 | | |
| E6 | 230 | W | 0-6 | 5-7 | | |
| E7 | 231 | X | 0-7 | 5-8 | | |
| E8 | 232 | Y | 0-8 | 5-9 | | |
| E9 | 233 | Z | 0-9 | 5-10 | | |
| EA | 234 | | 11-0-9-8-2 | | | EA through EF will not be assigned. |
| EB | 235 | | 11-0-9-8-3 | | | |
| EC | 236 | | 11-0-9-8-4 | | | |
| ED | 237 | | 11-0-9-8-5 | | | |
| EE | 238 | | 11-0-9-8-6 | | | |
| EF | 239 | | 11-0-9-8-7 | | | |
| F0 | 240 | 0 | 0 | 3-0 | | |
| F1 | 241 | 1 | 1 | 3-1 | | |
| F2 | 242 | 2 | 2 | 3-2 | | |
| F3 | 243 | 3 | 3 | 3-3 | | |
| F4 | 244 | 4 | 4 | 3-4 | | |
| F5 | 245 | 5 | 5 | 3-5 | | |
| F6 | 246 | 6 | 6 | 3-6 | | |
| F7 | 247 | 7 | 7 | 3-7 | | |
| F8 | 248 | 8 | 8 | 3-8 | | |
| F9 | 249 | 9 | 9 | 3-9 | | |
| FA | 250 | | 12-11-0-9-8-2 | | | FA through FE will not be assigned. |
| FB | 251 | | 12-11-0-9-8-3 | | | |
| FC | 252 | | 12-11-0-9-8-4 | | | |
| FD | 253 | | 12-11-0-9-8-5 | | | |
| FE | 254 | | 12-11-0-9-8-6 | | | |
| FF | 255 | DEL | 12-11-0-9-8-7 | | delete | Special – neither graphic nor control symbol. |

[†]Hexadecimal and decimal notation.

[††]Decimal notation (column-row).

## HEXADECIMAL ARITHMETIC

ADDITION TABLE

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 |
| 2 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 |
| 3 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 |
| 4 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 |
| 5 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 |
| 6 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 8 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 9 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A |
| C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B |
| D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C |
| E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D |
| F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E |

MULTIPLICATION TABLE

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 04 | 06 | 08 | 0A | 0C | 0E | 10 | 12 | 14 | 16 | 18 | 1A | 1C | 1E |
| 3 | 06 | 09 | 0C | 0F | 12 | 15 | 18 | 1B | 1E | 21 | 24 | 27 | 2A | 2D |
| 4 | 08 | 0C | 10 | 14 | 18 | 1C | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C |
| 5 | 0A | 0F | 14 | 19 | 1E | 23 | 28 | 2D | 32 | 37 | 3C | 41 | 46 | 4B |
| 6 | 0C | 12 | 18 | 1E | 24 | 2A | 30 | 36 | 3C | 42 | 48 | 4E | 54 | 5A |
| 7 | 0E | 15 | 1C | 23 | 2A | 31 | 38 | 3F | 46 | 4D | 54 | 5B | 62 | 69 |
| 8 | 10 | 18 | 20 | 28 | 30 | 38 | 40 | 48 | 50 | 58 | 60 | 68 | 70 | 78 |
| 9 | 12 | 1B | 24 | 2D | 36 | 3F | 48 | 51 | 5A | 63 | 6C | 75 | 7E | 87 |
| A | 14 | 1E | 28 | 32 | 3C | 46 | 50 | 5A | 64 | 6E | 78 | 82 | 8C | 96 |
| B | 16 | 21 | 2C | 37 | 42 | 4D | 58 | 63 | 6E | 79 | 84 | 8F | 9A | A5 |
| C | 18 | 24 | 30 | 3C | 48 | 54 | 60 | 6C | 78 | 84 | 90 | 9C | A8 | B4 |
| D | 1A | 27 | 34 | 41 | 4E | 5B | 68 | 75 | 82 | 8F | 9C | A9 | B6 | C3 |
| E | 1C | 2A | 38 | 46 | 54 | 62 | 70 | 7E | 8C | 9A | A8 | B6 | C4 | D2 |
| F | 1E | 2D | 3C | 4B | 5A | 69 | 78 | 87 | 96 | A5 | B4 | C3 | D2 | E1 |

# INDEX

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.

XEROX

## Reader Comment Form

We would appreciate your comments and suggestions for improving this publication

| Publication No. | Rev. Letter | Title | Current Date |
|---|---|---|---|
| | | | |

How did you use this publication?

☐ Learning ☐ Installing ☐ Sales
☐ Reference ☐ Maintaining ☐ Operating

Is the material presented effectively?

☐ Fully Covered ☐ Well Illustrated ☐ Well organized ☐ Clear

What is your overall rating of this publication?

☐ Very Good ☐ Fair ☐ Very Poor
☐ Good ☐ Poor

What is your occupation?

Your other comments may be entered here. Please be specific and give page, column, and line number references where applicable. To report errors, please use the Xerox Software Improvement or Difficulty Report (1188) instead of this form.

Your name & Return Address

Thank You For Your Interest (fold & fasten as shown on back, no postage needed if mailed in U S A )

CUT ALONG LINE

PLEASE FOLD AND TAPE--
NOTE: U. S. Postal Service will not deliver stapled forms

FOLD ALONG LINE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY MAIL**
FIRST CLASS PERMIT NO. 59153 LOS ANGELES,CA 90045

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
5250 W. CENTURY BOULEVARD
LOS ANGELES, CA 90045

ATTN: PROGRAMMING PUBLICATIONS

FOLD ALONG LINE

**Honeywell**

**Honeywell Information Systems**
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

22228, 3C1178, Printed in U.S.A.