# CHAPTER 5

# DIBOL COMPILER (COMP)

The Compiler converts a DIBOL source program into a binary program and reserves storage space for the constants, variables, and statements used by the program.

The Compiler outputs a source program compilation listing and a storage map listing of the records and fields used by the program. Turn on the printer before running the Compiler.

## 5.1 COMP OPERATING PROCEDURES

To execute the Compiler program, type:

```
     RUN COMP[,filnaml...,filnam7][/xx]
```

where:

filnaml...,filnam7
: are file(s) to be compiled into one binary program. If no files are specified, the program in the edit buffer is compiled.

/xx
: is one or a combination of the following option switches:

- /N stops output of the compilation listing and the storage map listing.
- /G compiles the program and, if no errors are detected, executes the binary program; implies /N. The message LOADING is displayed when compiling is successfully completed. If INIT SYS is used in the program, the program must have an END statement to be compiled and executed with the /G option.
- /T enables the TRACE function; implies /G.

/D transfers control to DDT; implies /G.

/O creates a binary program that requires less memory space by eliminating the TRACE feature and accurate error reporting. Execution speed of the compiled program is increased by as much as 20%. This option can be combined with /N or /G.

The /O option saves memory space as follows:

- Saves one location for each executable statement.
- Saves one location for each label.
- Uses one location for each ON ERROR statement.

Use the /O option on thoroughly debugged programs.

Unless the /N or /G option is specified in the RUN COMP command, the Compiler outputs a two-part compilation listing (Data Division and Procedure Division) of the source program and a storage map either on the printer or on the device specified in START, PROC, or END.

The Compiler underscores the number of the line where an error occurs and inserts a caret (^) pointing to the error. Other errors are listed on the storage map. Errors must be corrected before the program can be executed.

The Compiler displays the number of errors as nn ERRORS

### 5.1.1 Source Program Compilation Listing

```
COS DIBOL    12-JUL-78   WED     COMPILATION LISTING   V 8.00  PAGE 01
          DATA DIVISION   OPTIONAL COMPILATION STATEMENT

0100   START                         ;Optional compilation statement.
0110   RECORD INBUF                  ;Record named INBUF.
0120          STOCKN, D4             ;Numeric field named STOCKN.
0130          DESC, A25              ;Alphanumeric field named DESC.
0140          UCOST, D5              ;Five-character numeric field.
0150          QORDER, D4             ;Four-character numeric field.
0160          , D9                   ;Unreferencable unnamed field.
0170   RECORD OUTBUF                 ;Record named OUTBUF.
0180          , D4                   ;Unnamed numeric field.
0190          , A25                  ;Twenty-five character field.
0200          , D5                   ;Unnamed field.
0210          , D4                   ;Temporary storage field.
0220          ECOST, D9              ;Numeric field named ECOST.
0230   RECORD                        ;Unnamed record-temporary storage
0240                                 ;cannot be directly referenced.
0250          TITLE, A6, 'OVRHED'    ;Field initialized to 'OVRHED'.
```

```
COS DIBOL    12-JUL-78   WED     COMPILATION LISTING   V 8.00  PAGE 02
    PROCEDURE  DIVISION   END DATA DIVISION-BEGIN PROCEDURE DIVISION

0260    PROC                             ;Beginning of Procedure Division.
0270             INIT(1,I,TITLE)         ;Opens TITLE on channel 1-input.
0280             INIT(2,O, 'OUTPUT')     ;'OUTPUT' on channel 2-output.
0290    LOOP,    XMIT(1,INBUF,EOF)       ;Transfer INBUF to EOF.
0300             OUTBUF=INBUF            ;INBUF moved to OUTBUF.
0310             IF(STOCKN.LT.100) GO TO LOOP ;Conditional statement.
0320             ECOST=UCOST*QORDER      ;UCOST times QORDER moved to ECOST.
0340             XMIT(2,OUTBUF)          ;Transfer OUTBUF onto channel 2.
0350                                     ;
0360             GO TO LOOP              ;Branch control to LOOP.
0370    EOF,     FINI (2)                ;Identifies end of logical unit.
0380             FINI (1)                ;Writes record and closes file.
0390             STOP                    ;Stops program execution.
0400    END                              ;Marks the end of the program.
```

### 5.1.2 Storage Map Listing

```
COS DIBOL   12-JUL-78   WED    STORAGE MAP LISTING     V 8.00  PAGE 03

#         NAME      TYPE       DIM  SIZE   ORIGIN

0001      INBUF     RECORD     01    49    20000
0002      STOCKN    DECMAL     01    04    20002
0003      DESC      ALPHA      01    25    20006
0004      UCOST     DECMAL     01    05    20037
0005      QORDER    DECMAL     01    04    20044
0006      OUTBUF    RECORD     01    49    20062
0007      ECOST     DECMAL     01    09    20132
0010      TITLE     ALPHA      01    06    20146
0011      ..1       DECMAL     01    01    20154
0012      ..2       DECMAL     01    01    20155
0013      ..OUTP    ALPHA      01    06    20156
0014      LOOP      LABEL      00    01    10110
0015      EOR       LABEL      00    01    10144
0016      ..1000    DECMAL     01    04    20164

0014 labels

NO ERRORS DETECTED.   08 K CORE REQUIRED [3956 FREE LOCS -14 BUFFERS]
```

The storage map lists the record and field names and the labels as they were processed by the Compiler. The information is arranged in six columns with the following headings:

| | |
|---|---|
| # | contains the internal number of the name in column 2. This number is only used in machine-level programming. |

| | |
|---|---|
| NAME | is the name (field name, record name, program label) or literal used in the compiled program. Literals are numeric or alphanumeric characters which appear in the Procedure Division of the source program. Only the first four characters of a numeric literal are used. Each numeric literal is preceded by two periods (..) to distinguish as an internal name. Numeric literals with four characters or less appear only once on the storage map even though they may occur more than once in the program. Numeric literals with more than four characters are listed each time they occur in the program. Record literals begin with a double quote and end with a single quote. |
| TYPE | describes the use of name in the program. |

| | |
|---|---|
| ALPHA | used as the name of an alphanumeric field or as an alphanumeric literal. |
| DECMAL | used as the name of a numeric field or as a numeric literal. |
| RECORD | used as a record name or as record literal. |
| LABEL | used as a program label. |
| REDEF | is multiply defined (redefined). All attempts at definition after the first are flagged as errors in the compiler listing. |
| UNDEF*** | is an undefined label referenced by the program. For example: GO TO TAG1 in a program where TAG1 does not appear as a label.<br><br>This error is output to the printer even if the /N option is in effect. The line number where the label is used is displayed. |

| | |
|---|---|
| DIM | contains the array dimension (number of fields) of the alphanumeric or numeric labels. The column is meaningless for other types of labels. |
| SIZE | lists the size of the name. The size of a RECORD is the number of characters in all its labels plus 2. |
| ORIGIN | gives the octal byte memory address of the name. |

The number of labels used, number of errors detected, memory required, and free locations are listed at the bottom of the storage map. You cannot get this information if you suppress listing of storage map.

Maximum number of labels allowed in a 16K-byte system is 365; in 24K-byte or larger systems, 511.

Use the SAVE command to store the binary program.

## 5.2 CONDITIONAL COMPILATION PROCEDURE (CCP)

The Conditional Compilation Procedure (CCP) is a feature which permits you to include statements in a source program which will be compiled only if you elect to have those statements compiled.

Statements included in a program for conditional compilation are enclosed within angle brackets as in the following example.

```
      RECORD A
B1,    D5
C1,    A4
PROMPT, D1
      RECORD N
NAME, A6
      PROC
<PROMPT
      XMIT(8,"ENTER NAME:')
>
      XMIT(7,N)
      STOP
      END
```

The left angle bracket (<) is followed by a control variable (in this case PROMPT). Unless the control variable is turned on before the left angle bracket is encountered, statements between the angle brackets will be ignored. A right angle bracket marks the end of a conditional area and is on a line by itself. The command to turn on a control variable is as follows:

```
=control variable
```

The above program requires the operator to type in a name on the keyboard. If this same program is recompiled with the control variable PROMPT on, it produces a DIBOL program which first displays a message to the operator.

```
      RECORD A
B1,    D5
C1,    A4
PROMPT, D1
      RECORD N
NAME, A6
      PROC
=PROMPT          ;Turn on prompt.
<PROMPT
      XMIT(8,"ENTER NAME:')
>
      XMIT(7,N)
      STOP
      END
```

Conditional compilation can also be used to debug statements in a source program. Once the program has been tested, the control variable can be removed by deleting the command to turn it on.