Digital Equipment Corporation
Maynard, Massachusetts

digital

PDP-8/I

DIBOL Programming

A Self-Instruction Manual

DEC-08-WDRA-D

PDP-8/I DIBOL PROGRAMMING A SELF-INSTRUCTION MANUAL

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Printing September 1970

DEC	PDP
FLIP CHIP	FOCAL
DIGITAL	COMPUTER LAB

CONTENTS

ILLUSTRATIONS

TABLES

INTRODUCTION

This self-instruction manual is designed to present, as simply as possible, a complete course in DIBOL programming.

Each DIBOL program is divided into two sections — the data section and the procedure section. The data section defines data elements used in the program. The procedure section contains the file-handling data manipulation instructions and implies the order of program operation. This manual explains each of these sections in an easy-to-learn question and answer format.

The DIBOL Self-Instruction Manual was designed for comprehension in one sitting, so that the student could immediately begin writing DIBOL programs. In this way, knowledge gained is reinforced through application.

Upon completion of this manual, the student should be able to write a DIBOL program for the application problem given in Appendix I.

SECTION I

A DIBOL PROGRAM BY DEDUCTION

The DIBOL language is actually simple and straightforward. This section is intended to give the student a frame of reference so the value of the following section will be readily apparent.

The general knowledge of this section puts the specialized knowledge of the following section in perspective. In this section, you will learn how to design a flow chart, and how to read a DIBOL program with a fair degree of comprehension.

Following is a statement of what a program should do. (Turn to Page 9 and fold out Foldout #1.)

PROGRAM: Information is stored in records located on a magnetic tape. Each record contains 64 characters. A read operation reads one record at a time (starting with the first record in the file). Since the teletypewriter will print over 64 characters per line, we want to list on the teletypewriter all information on the magnetic tape file, printing one record on each line. After printing the last record on the tape file, we want to stop the program.

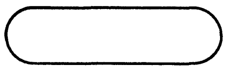
QUESTION: Does the flowchart on Foldout #1 illustrate the logic outlined in the verbal statement of our program?

ANSWER: Yes, the flowchart is accurate.

***

The shapes of the boxes in the flowchart denote different functions, such as comparison, reading/writing, beginning/end, and internal data arrangement. Using the flowchart, answer the following questions:

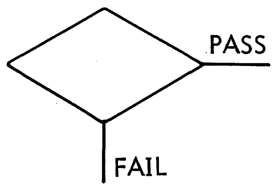
QUESTION: What is the function of this symbol?



ANSWER: It denotes the beginning/end of the program logic flow.

***

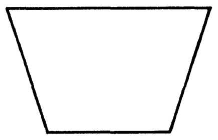
QUESTION: What is the function of this symbol?



ANSWER: It denotes the testing of a condition and, depending upon the outcome of the test, shows the action to be taken.

***

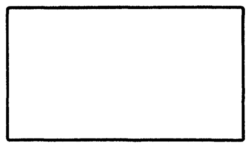
QUESTION: What is the function of this symbol?



ANSWER: It denotes an Input (reading) or an Output (writing) operation to be performed by the computer. (In the program, there is an internal device assignment so the computer would issue a read/write command to the proper input/output device.)

***

QUESTION: What is the function denoted by this symbol?



ANSWER: It denotes explicit commands such as device initialization, move data, etc.

***

Below the flowchart on Foldout #1 is the DIBOL-coded program which accomplishes the functions diagrammed. Note that it requires only ten statements to accomplish the outlined task.

The following dialogue is designed to help the reader understand the function of each statement in the DIBOL program.

QUESTION: From the flowchart, is each character of information passed directly from the magnetic tape to the teletypewriter?

ANSWER: No. A complete record composed of 64 characters is read into the computer memory before any data is written (outputted).

***

QUESTION: Since we know of no reason why data records should be restricted in length to 64 characters, how does the computer know how much memory to reserve for the storage of the data record?

ANSWER: The programmer must tell the computer how much memory will be required to store input data.

***

QUESTION: As a matter of convention, in a program, the area of memory reserved for record storage precedes the processing instructions. From the DIBOL-coding on Foldout #1, which statement allocates 64 characters of storage?

ANSWER: Statement 3; (A1, A64).

***

QUESTION: Statement number 3 says the block of storage labelled A1 will be reserved for 64 alphanumeric characters. What is A1?

ANSWER: It is a label. It could just as well be called XX, YY, or ZOT. It serves as a name which the programmer can reference from the procedure section of the program (note: it is not so referenced in the sample program).

***

QUESTION: The BLOCK statement (#2) gives a label A to an area of computer memory available for input. If several different input devices are being used, several different BLOCK statements (with their respective field-definition statements following) could appear. In order to designate two 16 alphanumeric character fields and one

QUESTION: (Cont) 32 alphanumeric character field instead of the present 64-character field, write the appropriate block and field-definition statements.

ANSWER:

```
BLOCK A
        A1, A16
        A2, A16
        A3, A32
```

***

QUESTION: A compiler-statement is a non-executable DIBOL statement. Such a statement gives the compiling program information necessary to properly interpret notations made by the programmer. Compiler statements tell the compiler program when to begin and end encoding DIBOL source statements, and when to begin converting DIBOL statements into actual machine procedures. Look at the sample DIBOL program and see if you can guess which are the compiler statements.

ANSWER: Statements 1, 4, and 10. Statements 1 and 10 tell the program the bounds of the DIBOL coding. Statement 4 tells the compiler program to interpret the following lines of coding as procedure to be executed by the computer.

***

QUESTION: The data section of a DIBOL program describes the data elements used in the program and allocates memory. Which statements in Foldout #1 comprise the data section?

ANSWER: Statements 2 and 3 comprise the data section.

***

QUESTION: Which statements are actual processing instructions?

ANSWER: Statements 5 through 9 are processing statements.

***

QUESTION: What is the function of the word, LOOP, (in statement 8)?

ANSWER: LOOP is a label denoting a point in the processing cycle to which the program branches. In this case, the program branches to statement 6.

***

QUESTION: Statement 5 is the first processing instruction. From the flowchart, what is accomplished by this statement?

ANSWER: It tells the computer what tape unit will be used and whether the file will be read (input) or written (output). This process is called file-initialization.

***

> In this case, tape transport #2 is initialized as an input (IN) file. (We will discuss the meaning of the V in a later section.)

QUESTION: From the flowchart (and following the INIT statement in the program), what does the XMIT statement (6) do?

ANSWER: It causes a read operation (reads a tape record) from transport 2. More fully, the XMIT causes data transmission. It can either read (IN) or write (OUT) depending upon the file-initialization. If a file is initialized as an input (IN), the XMIT statement would cause a read operation from the file. If a file is initialized as an output (OUT), the XMIT statement would cause a write operation on to the file. The data is either read from, or written into the BLOCK as specified in the XMIT statement (in this example, BLOCK A).

***

QUESTION: For XMIT (2, A) to cause a record to be written, what would the initialization statement look like?

ANSWER: INIT (2, V, OUT).

***

QUESTION: For XMIT (2, A, EOF) to cause a record to be read, what would the initialization statement look like?

ANSWER: INIT (2, V, IN). (As in statement 5 in our example.)

***

QUESTION: Statement 6 says: Read a record from device number 2, storing the data from that record in the area labelled A. When no more records are available, i.e., the End-of-File (EOF) has been reached, then jump to the instruction labelled EOF.

QUESTION: (Cont) Noting that only READ instructions have pointers to which the program will branch when an end-of-file (EOF) is reached, what does statement 7 do?

ANSWER: Since there is no end-of-file pointer (just a device number and the BLOCK label) this must be a write command (writing data from storage block A to device 8).

***

QUESTION: Note, that it is only necessary to initialize a file-oriented device such as a magnetic tape drive. Devices such as teletype, line printers, and paper-tape readers do not need initialization. Is device 8 a file-oriented device?

ANSWER: No. It is a teletype and therefore does not need initialization. Note, under the present DIBOL configuration, each device is assigned its own device code. Device assignments are:

1, 2, 3, 4	mag tape (DECtape)
5	paper tape in
6	line printer
7	teletype in
8	teletype out

***

QUESTION: What does statement 8 do? (Go to LOOP)

ANSWER: It is an unconditioned command for the computer to branch to the instruction labeled LOOP.

***

QUESTION: What is EOF?

ANSWER: It is the label for the end-of-file routine referenced in statement 6 (LOOP; XMIT (2, I, EOF)). This label indicates the location for program transfer at the end of the input file. In this example, program control would transfer to statement 9.

***

QUESTION: What would you guess to be the function of Statement 9?

ANSWER: It is a FINIsh statement with respect to the file on device 2. Actually, to write a tape-file, an end-of-file mark would be put on the tape, and the tape would be

ANSWER: (Cont) rewound. In the case of our input file, the tape is just rewound. Only file-oriented devices require a FINI statement.

***

QUESTION: Why was device 8 not issued a FINI command?

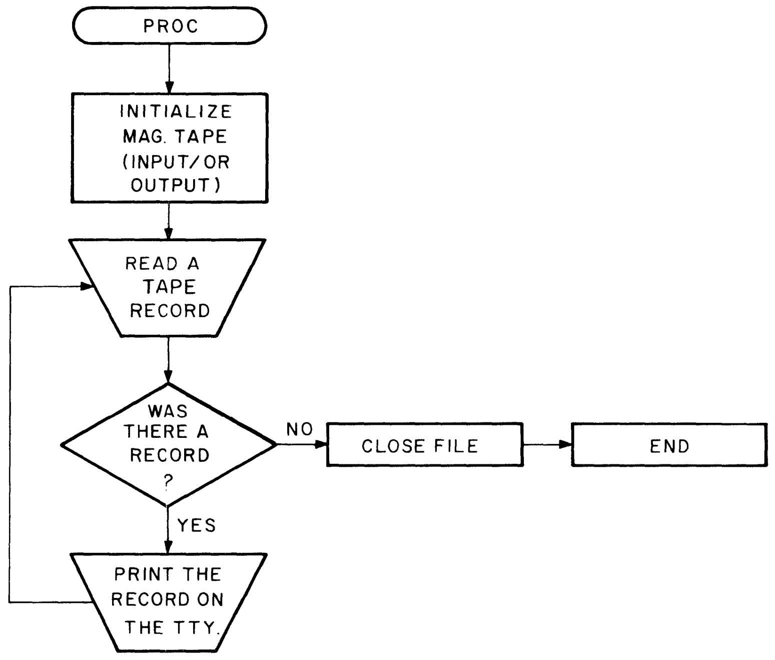
ANSWER: It is not a file-oriented device. The only file-oriented device on a DIBOL configuration is magnetic tape.

***

FOLDOUT #1

SAMPLE PROBLEM

FLOWCHART FOR PROBLEM



PROC
INITIALIZE MAG. TAPE (INPUT/OR OUTPUT)
READ A TAPE RECORD
WAS THERE A RECORD ?
NO
CLOSE FILE
END
YES
PRINT THE RECORD ON THE TTY.

08-0584

DIBOL PROGRAM FOR PROBLEM

```
START                                       ;1
BLOCK A                                     ;2
        A1, A64                             ;3
PROC                                        ;4
        INIT (2,V,IN)                       ;5
LOOP,   XMIT (2,A,EOF)                      ;6
        XMIT (8,A)                          ;7
        GO TO LOOP                          ;8
EOF,    FINI (2)                            ;9
END                                         ;10
$
```

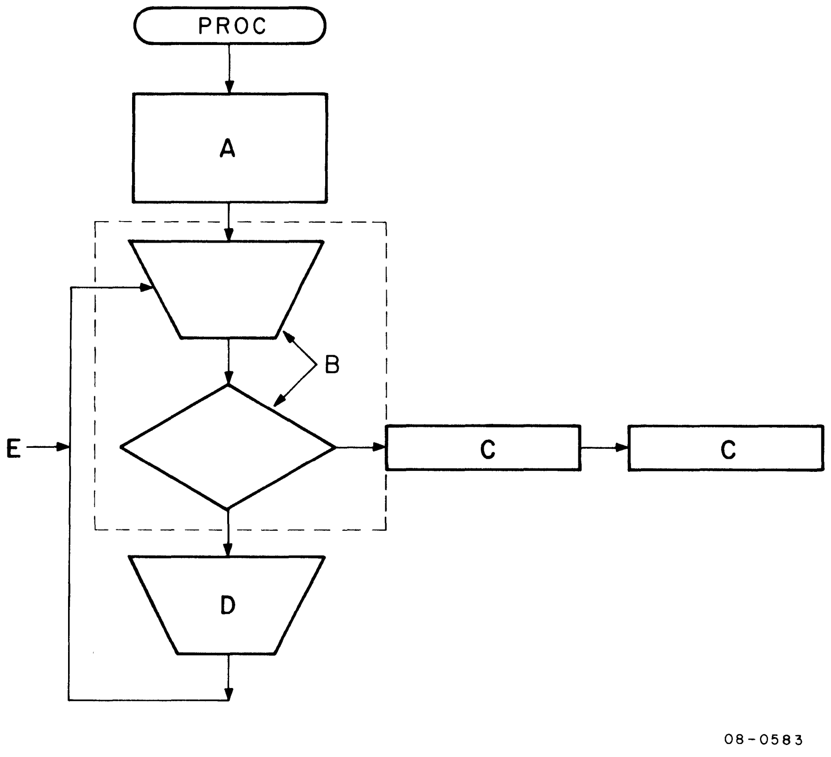
SUMMARY

SECTION I

To review the initial problem, i.e., printing a 64-character record from a tape file on to a Teletype until the end-of-file, examine the following lines of the DIBOL program.

```
START                      ;Compiler statement, non-executable
BLOCK A                    ;indicates the beginning of the contiguous area
                              for the data elements that comprise BLOCK A.
        A1, A64            ;Data statement A1 is an alphanumeric record 64
                              characters long.
PROC                       ;Compiler statement.-begins procedure section.
        INIT (2,IN)        ;Initialize DECtape 2 as an input file
LOOP, XMIT (2,A,EOF)       ;Read a record from DECtape 2 into the area as-
                              signed to BLOCK A.  When End-of-File is
                              reached, it causes a program transfer.
        XMIT (8,A)         ;Write the record on to Device 8 (teletype) from
                              BLOCK A.
        GO TO LOOP         ;Go to statement that reads another record
EOF, FINI (2)              ;At end of file, the tape on DECtape 2 is rewound
END                        ;Compiler statement — indicates the end of program
$                          ;Every DIBOL program must be terminated by a $
                              sign.
```

QUESTION: Below is the same flowchart diagram as listed on the foldout. Mark in the appropriate statement number corresponding to the flowchart function.



ANSWER:

A=5
B=6
C=9
D=7
E=8

***

SECTION II

DIBOL IN DEPTH

The student should now have a general knowledge of the elements that make up a DIBOL-coded program. Thus making the information in this section more meaningful.

START
PROC
END

There are two basic sections in a DIBOL program. First is the data section which describes all data and causes allocation of memory storage. Second, there is processing section, which contains the executable instructions.

QUESTION: What is the statement that separates the data section from the processing section?

ANSWER: The PROC statement.

***

QUESTION: Is PROC an executable statement? If not, what is it?

ANSWER: PROC is not an executable statement. From the previous section we recognize it as a compiler statement.

***

QUESTION: What is a compiler statement?

ANSWER: A compiler statement is a message to the compiler program indicating the nature of the DIBOL-language statements. A compiler instruction is not executable by the DIBOL program.

***

There are three kinds of statements in a DIBOL program:

1) Compiler statements
2) Data statements
3) Procedure statements

All DIBOL programs consist of a START statement (which is a compiler statement), followed by the data section (composed of data statements), followed by a PROC statement (a compiler statement), followed by the procedure section (composed of procedure statements), followed by an END statement (a compiler statement).

QUESTION: What are the three required compiler instructions in a DIBOL program?

ANSWER: START, PROC, and END.

QUESTION: What are the three compiler statements and two sections that make up a DIBOL source-program (in the order in which they appear)?

ANSWER:

```
START                      (compiler instruction)
          data section
PROC                       (compiler instruction)
          processing section
END                        (compiler instruction)
```

***

BLOCK

QUESTION: Where is the data section in a DIBOL program?

ANSWER: The data section is between the START and PROC statements.

***

QUESTION: From the sample program, in which section does the BLOCK statement reside?

ANSWER: The BLOCK statement resides in the data section.

***

The BLOCK statement designates the beginning of a group of data statements. It may or may not give that group a name and controls where in memory the block of data will be stored. A BLOCK statement must be followed by one or more data statements. (A data statement defines all data elements with respect to type and size.) The general format for a BLOCK statement is:

```
(BLOCK     block name,    X-overlay and/or C-Clear)
 required  optional       optional
```

QUESTION: In the following example, is the BLOCK statement used correctly?

```
START
BLOCK A
BLOCK B
        B1, A6
PROC
```

ANSWER: BLOCK A is an invalid statement because a BLOCK statement must be followed by one or more data statements. BLOCK A is followed by another BLOCK statement, BLOCK B, which is correct.

The only reason a block of data requires a name is if the information is to be referenced by an XMIT (data transfer) statement. A record may be read and stored in this area, or the contents of this area may be written (output). There is no punctuation between a BLOCK statement and its name (if there is to be a name).

***

QUESTION: What can be deduced about the second block statement following?

```
START
BLOCK A
        A1, A6
        A2, D1
BLOCK
        B1, D3
PROC
```

ANSWER: Since the second BLOCK statement does not have a name, it is not intended to be used as an input/output buffer. It is used only for temporary storage of program data.

***

The data statement is used to define all data elements with respect to type and size. The DIBOL compiler assigns storage for the data on the basis of these statements. Any data statement that follows a block name is assigned to the contiguous memory locations in the order that the element occurs. If a block name is missing, the succeeding data statements are assigned to contiguous locations but not associated with any block name for input/output. If such data statements are referenced, they are done so individually.

The general format of the data statement is:

```
data name     data specification,     initialization-specif.
optional           required                  optional
```

The data name is optional, that is, a "," may be used without a data name if the program does not reference that individual data element but only references the entire BLOCK. This is convenient when formatting an output line for the printer, so that intercolumn spaces do not require a data name but merely a comma followed by the type and size; e.g., (,A5). Normally, the data name is used, followed by the data specification (type and size), with an optional initialization field.

The initialization-specification would normally be used in the temporary storage block but could also be used in an output block. If a specific data element is to be referenced, it must have a data name.

Following are examples of valid data statements:

```
A, A10
A, A7, 'DIGITAL'
A, D6, 123456
FISH, A4, 'FISH'
     , A5
COST, 5D6
```

NOTE

The data element COST consists of 5D6, which means there is an array of five fields, six digits long. This could have been written as:

```
COST1, D6
COST2, D6
COST3, D6
COST4, D6
COST5, D6
```

(The referencing of this COST ,5D6, is done with subscripts which will be defined in the procedure section)

QUESTION: In the following statement, what are the fields?

```
TOT,D6,000000
```

ANSWER: TOT is the name by means of which the data can be referred; D6 is the data specification (in this case, six decimal digits) and 000000 is the initialization-specification (setting the six decimal digits to an initial value of zero).

The data specification field (which follows the data name) consists of a data type (Alpha or Decimal), and the data size in characters. If the data size is omitted, 1 is assumed. If the initialization specification is present, the data specification is followed by a comma, then an alphanumeric or decimal constant. The alphanumeric may contain any legal character enclosed in apostrophe marks except a question mark (?). The decimal constant is a string of digits, optionally preceded by a plus or minus sign. The plus sign is implied, and the minus sign does not require a character for storage; i.e., NUM,D5,-12345. Data types cannot be mixed. For example, an alphanumeric constant may not be assigned to a decimal variable. The data element is assigned to the value of the initialization specification at the beginning of program execution.

***

QUESTION: Describe the data specified for the following four items:

a)	A,	A10
b)	TOT,	D6
c)	NUMS,	10D3
d)	HDRS,	10A12

ANSWER:

a) A is an alphanumeric element with ten characters
b) TOT is a decimal element with six digits
c) NUMS is an array of ten decimal numbers, each with three digits
d) HDRS is an array of ten alphanumeric elements, each with 12 characters

***

BLOCK C

There are three ways data items can be initialized in the Data Section:

1) A BLOCK statement containing a C
2) A data statement containing a P
3) A data statement containing an initialization specification value (previously discussed)

To set all alphanumeric elements to spaces and all decimal elements to zero, at the beginning of a program run, specify a C following the word BLOCK. In the case of temporary storage, instead of initializing each individual data element, the entire BLOCK can be cleared with BLOCK C. Following are valid uses of this feature:

a) BLOCK OUTBUF, C
b) BLOCK, C

QUESTION: What would be the value to have example b) initialized - cleared?

ANSWER: In the case of a temporary storage BLOCK, instead of individually clearing each storage location, before program execution, the entire BLOCK can be cleared with one statement. This will ensure that memory locations do not contain information from the previous run.

***

Another way of initializing a data item is by putting a P immediately after the data specification field of a data statement. Upon loading the program, the computer will ask the operator (via the teletypewriter) to type in the value he wants to give that data item. A common use of this feature is to obtain the date. For example, the data statement might be described as:

```
DATE,A8,P
```

Upon loading the program, the following message would type out on the console:

```
ENTER DATE      (Carriage Return)
```

At this point the program would halt, waiting for the operator to type in eight characters and type carriage return. The operator might type in 07/07/70 and type carriage return.

QUESTION: Suppose the programmer wanted a three-digit customer-number to be supplied by the operator at the beginning of the run. Write a data statement to initialize a field named CUST.

ANSWER: CUST,D3,P Ordinarily, input from the teletype must be described in alpha format. This restriction does not hold true for initialization data (P).

***

INITIALIZATION-SPECIFICATION

A third way to set the value of a data item is by using the initialization option, i.e., specifying an alphanumeric or decimal constant. The alphanumeric may contain any legal characters enclosed in apostrophe marks. The decimal constant is a string of digits, optionally preceded by a plus or minus sign but without apostrophies. Data types cannot be mixed, in that an alphanumeric constant (i.e., 'ABC') cannot be assigned to a decimal variable (i.e., DEC,D3). But '004' can be assigned to an A3 alphanumeric field.

QUESTION: Are any of the following statements incorrect?

a) A,A8,'ABCDEF98'
b) TOT,D3,'123'
c) NUM,A3,123
d) B,D6,222334
e) C,D3,23A

ANSWER: Statement b) is incorrect because an alphanumeric notation (an apostrophe) was used to enclose a decimal item; c) is incorrect because a decimal notation was used to describe an alphanumeric constant; e) is incorrect because 23A is not a valid decimal number. Statements a) and d) are correct.

***

When an initialized value is specified in a data statement, its length must correspond to the length of its respective data statement, for example, NUM,D4,0070. It would be illegal to initialize NUM to 70 since NUM was defined as a D4 field. The initialization specification does not insert leading or trailing blanks (zeros). The initialized information is moved from left to right. This is the only case in the DIBOL language in which information is moved in this direction. If the initialized value is shorter than the data field, the unused characters would contain data left in memory from the previous computer run. If the initialized value is longer than the data field, i.e., NUM,D4,55555, characters would run over into the following data fields.

QUESTION: Which data statements are incorrect?

a) A1,A8,'ABCDEF641'
b) COST,D4,7779
c) QTY,D5,'10000'
d) NUM,D7,59796
e) B1,A4,'1987'

ANSWER: Statement a) is incorrect because the initialized value is longer than the data field. Statement c) is incorrect because apostrophe marks are used to enclose a decimal item (apostrophe marks are an alphanumeric notation). Statement d) is incorrect because the initialized value is too short, NUM is defined as a 7-digit decimal field. Statements b) and e) are correct usage of the initialization specification.

***

QUESTION: We want NUMS to be an array of two decimal numbers, of three characters each. The two numbers are to have constant values of 333 and 61 respectively. What is the appropriate data specification?

ANSWER: NUMS,2D3,333061

***

NOTE

If the BLOCK C (clear) option is used, and if a data statement in that BLOCK has an initialized value, that value will not be cleared. The C option has no effect on an initialized data statement value. For example:

```
        BLOCK A,C
                NUM,D6
                TOT,D7
                COST,D4,4999
                B1,A7
```

The initialized value of COST would not be cleared, but NUM,TOT, and B1 memory locations would be cleared.

QUESTION: What information is generated by this BLOCK?

```
        BLOCK A,C
              A,A10,'ABC'
```

ANSWER: ABC___________. The initialized value is too short for the data field (A10). The other seven character positions would contain information that was left in memory from the previous run. Remember, an initialized data statement is not cleared with a BLOCK C statement.

***

QUESTION: What information is generated by this data specification?

```
        B,D6,000013
```

ANSWER: 000013. The initialized value must be the same length as the data size. In this case, B is defined as a six-decimal digit.

***

QUESTION: What information is generated by this data specification?

```
        TOT,D1,C8
```

ANSWER: None. The specification is incorrect. An alphanumeric description (apostrophe marks) cannot be used for a decimally defined item.

***

QUESTION: Consider the following operation.

```
DATE,A8,P
```

When the program is loaded, the computer types

```
ENTER DATE
```

and the operator types 06/06/1970. What happens?

ANSWER: Since too many characters were typed, it is an error.

***

CAUTION

The DIBOL System will accept too few or too many characters during operator-initialization. Too few characters would be left-justified, where too many characters would run over to the following fields.

BLOCK X

The concept of the overlay is a valuable tool in the preservation of computer memory. By means of the overlay, two BLOCK statements can describe exactly the same area of computer memory. Whenever there is an X there must be a previously defined BLOCK statement without an X. There can be one or more overlays defining the same area. Note the use of X below.

```
BLOCK A
        A1,A5,'DIBOL'
        A2,A9,'SOFTWARE'
BLOCK B,X
        B1,A5
        B2,A9
        B3,A7 'SYSTEM'
```

In this example, the first fourteen characters of BLOCK B occupy the same area of computer memory as BLOCK A.

QUESTION: What is the value of data labeled B1?

ANSWER: B1= 'DIBOL'

***

QUESTION: What is the value of B2?

ANSWER: B2= 'SOFTWARE'.

***

Be careful to make sure data specifications in overlays are consistent. Gross abnormalities arise if an alphanumeric item is redefined as decimal in an overlay specification. As a general rule, initialized values should not be used in overlays.

QUESTION: Is the following a correct use of the overlay?

```
BLOCK A
        A1,A3,'FUN'
        A2,A5,'LOVER'
BLOCK B,X
        B1,D8
```

ANSWER: It is incorrect. B1 is decimal, and the data it redefines is alphanumeric.

***

QUESTION: Suppose you want to reserve, in computer memory, a place to store an input record with four fields to be described as follows: FLD1 has three alpha characters, FLD2 has six decimal digits, FLD3 has four decimal digits, and FLD4 has 20 alpha characters and will be used to store a company name. Write the appropriate BLOCK and data specifications for this input buffer.

ANSWER: BLOCK IN (any name will do)

```
FLD1, A3
FLD2,D6
FLD3,D4
FLD4,A20
```

***

QUESTION: Is it legal to name a BLOCK of records C?

ANSWER: Yes. BLOCK C. It is also legal to have BLOCK C followed by C (clear upon program loading — BLOCK C,C).

***

QUESTION: Is it legal to name a BLOCK X ?

ANSWER: Yes, for the same reason as in the preceding answer.

***

NOTE

X (overlay) and C (clear) can be used simultaneously, i.e., BLOCK A,C,X or BLOCK A,X,C. An example of this would be:

```
BLOCK A
        A1,A50
        A2,A25
        A3,D10
        A4,D18
        A5,A20
BLOCK B,X,C
        B1,A75
        B2,D28
```

This would overlay blanks for the first 75 alpha characters and zeroes for the next 28 decimal digits in BLOCK A. Note that A5 was not changed, i.e., the entire BLOCK A is not cleared.

DATA SECTION RESTRICTIONS

An important restriction on data names and block names is that they contain no more than six characters. Also, constants specified in initialization statements can be no longer than 18 characters. The maximum length of a record is 256 characters.

QUESTION: In the following record specification, determine errors in name length, block, or record length.

```
BLOCK INBUFFR
          NAME,A20
          TOWN,A20,'ELK MOUNTAIN WYOMING'
          POP,A6
          CODE,A10,'ABD-XXM-YY'
          Q1,   A1
          Q2,   A1
          WORKAREA,A200
```

ANSWER: The name of the block (INBUFFR) is too long (seven characters); TOWN cannot be initialized with a 20-character constant (18 is the maximum); the data-name

ANSWER: (Cont) WORKAREA is too long (eight characters); finally, the record is too long (258 characters — 256 is the maximum allowed). If these errors were made, the DIBOL system would flag them as errors.

***

Many times a programmer will make comments, so someone else reading his program will know what he is doing. A semi-colon (;) tells the compiler-program that all information following is not to be interpreted as program text, but rather as comments by the programmer. Thus, comments can appear on a program listing but will not affect the operation of the program. Here is an example of a comment:

```
START       ;THIS PROGRAM READS INDIVIDUAL TRANSACTIONS
BLOCK A     ;THIS IS THE INPUT RECORD BUFFER
     A1,A16;CUSTOMER'S NAME IS STORED HERE
```

The comment is terminated by a carriage-return line feed.

QUESTION: What are the functions of these three computer-defined symbols?

(a) ,C
(b) ,X
(c) ;

ANSWER: The C causes the block of data to be cleared when the program is first loaded into the computer; the X indicates one block of data elements will overlay (use the same space in computer memory that the previous block was using); when using X, it must follow a block statement without an X; the ; indicates the beginning of a comment.

***

If the C, X, or both options are used in a BLOCK statement without a block name, then a comma must follow the word BLOCK, i.e., BLOCK,X or BLOCK,X,C.

QUESTION: What is the difference between BLOCK C and BLOCK,C?

ANSWER: BLOCK C denotes the beginning of a group of data statements referenced as BLOCK C. BLOCK,C would clear the data statements following this unnamed block (C option).

***

DATA SECTION SUMMARY

The data section describes all data used in a program and causes allocation of memory storage. It consists of one or more data blocks. Each data block section is made up of a BLOCK statement followed by one or more data statements.

I. BLOCK STATEMENT

a) Normal Form — BLOCK block name, e.g., BLOCK INBUF

b) Unnamed Form — A block name may be omitted, e.g., BLOCK

c) Data Overlay — Overlay a preceding storage area, e.g., BLOCK,X or BLOCK B,X

d) BLOCK Clear — Initialize a block with zeroes and/or blanks, e.g., BLOCK A,C or BLOCK,C

e) BLOCK overlay and Block Clear Combined — e.g., BLOCK A,X,C

II. DATA STATEMENT

a) Normal Form —

data name,	data specification	initialization specif.
(optional)	(required)	(optional)

For example:

```
A1,D4,1234
A2,A4,'ABC4'
```

b) Operator Initialization — Specified by a P and causes entry of data from teletype before program execution, e.g., DATE,A8,P

III. Three ways to initialize data elements in the Data Section:

Block Initialization —	a BLOCK with a ,C — set BLOCK to blanks and/or zeroes
Data Statement Initialization —	*A data statement with an initialization specification*
Operator Initialization —	A data statement with a ,P which will allow entry of data from console

PROCEDURE SECTION

QUESTION: What is the compiler statement that separates the data and procedure sections?

ANSWER: The PROC statement separates the data and procedure sections.

***

QUESTION: What is the difference between a procedural statement and a data statement?

ANSWER: Procedural statements are executable.

***

In a computer program, procedural statements are executed sequentially; the sequential execution of instructions can be changed by a branching instruction.

The first procedural statement discussed is the file-initialization statement. The general form is:

```
INIT      file number,   file type,   direction
           (required)    (required)   (required)
```

Examples of the file initialization statements are:

```
INIT (2,F,OUT)
INIT (3,F,IN)
INIT (4,V,OUT)
```

NOTE

F means fixed; i.e., all records in the file are the same length. V means variable; i.e., all records in this file are not necessarily the same length. The direction can be either IN (for input file) or OUT (for output file). At present, the F file type is not implemented in the DIBOL language. File types must be specified as V, even if all records in the file are of the same length; i.e.,

```
BLOCK A
        A1,A7
        A2,D5
        A3,A7
        A4,D10  (14 alpha, 15 decimal)
BLOCK,X
        B1,A5
        B2,A2
        B3,D10
        B4,A7
        B5,D5   (14 alpha, 15 decimal)
```

QUESTION: In the following initialization, what is each field?

INIT (4,V,OUT)

ANSWER: File number is 4; File type is V (variable); Direction of Data Transmission is OUT.

***

The file number in an INIT statement refers to a peripheral device. When any device is attached to a computer, the computer must be told how to refer to that device. As devices are added to the system, they are given a physical number, or file number. Thus, whenever the programmer wants to access a peripheral device, he must tell the computer the name of that device.

QUESTION: If the device is a teletype for input (let's say 8) what would the file initialization statement (if any) look like?

ANSWER: The teletype is not a file-oriented device and, as such, does not require file-initialization. Only file-oriented devices (magnetic tapes) need file initialization.

***

QUESTION: What does the following initialization statement do?

INIT (2,V,IN)

ANSWER: It says "Initialize device 2 (DECtape 2) as an input file with variable length records."

***

QUESTION: What does the following initialization statement do:

INIT (3,V,OUT)

ANSWER: It says "Initialize device 3 (DECtape 3) to output variable length records."

***

XMIT

To read or write a record, the transmit data statement is used. Its general form is:

XMIT	file-number, (required)	block name, (required)	end of file label (only for input file)

Examples of the transmit-data statement are below:

```
a)  XMIT(1,OUTBUF)
b)  XMIT(2,INBUF,EOF)
```

a. Assuming device 1 has been previously initialized for output, statement a) would transfer the contents of BLOCK OUTBUF to device 1.

b. Assuming that device 2 has been previously initialized for input, statement b) would transfer data into BLOCK INBUF from device 2.

QUESTION: What is accomplished by the following DIBOL program?

```
START
BLOCK INBUF
        INA,A10
        INB,A6
        INC,A6
BLOCK
        DATE,A8,P
PROC
        INIT(2,V,IN)
BEGIN,XMIT(2,INBUF,EOF)
        GO TO BEGIN
EOF, FINI  (2)
STOP
END
 $
```

ANSWER: After the program has been loaded into the computer memory, the Teletype will output ENTER DATE and wait for the operator to input an eight-character date (note the P option on data-item DATE). File 2 will be initialized and all records will be read from device 2 into the area assigned to BLOCK INBUF. After all records are read, the program will transfer to End of File routine (EOF) in which device 2 is rewound, and then the program will stop.

***

FINI

QUESTION: What is the function of the FINI statement?

ANSWER: The FINI is a finalize-file statement and must refer to a previously initialized file. For output files, an end-of-file marker is written onto the file (if the device is magnetic tape, the file is rewound); for an input file, reading is discontinued (and if the device is a tape file, a rewind begins). The form is:

FINI (device number)

NOTE

A FINI statement must be used for each device initialized. (File-oriented devices, i.e., Magnetic tape-DECtape is the only device that needs initialization.)

***

In Summary:

a. An INIT statement has the following format:

INIT (file number, file type, direction)

For example:

INIT(4,V,OUT) -	Device 4 is initialized as an output device.

b. XMIT statement has the following format:

XMIT file number, block name, end-of-file (only for input files)

For example:

XMIT(1,INBUF,EOF) -	Assuming device 1 was initialized as an input file, this XMIT statement would read data into BLOCK INBUF, till the end-of-file, and then branch to EOF routine.

c. FINI statement has the following format:

FINI (file number)

For example:

FINI(1) -	The file number refers to an initialized device; for an output file device, EOF is marked at end of file, and file is rewound. For an input file the file is rewound.

CLEAR DATA

Next comes a general class of commands known as data-manipulation statements. They take the general form:

destination data element=source data

The destination data element is always an alphanumeric or decimal element. The source data can be either an alphanumeric or decimal element; an arithmetic expression such as A+1, or A+B/C.A data-manipulation statement appears as follows:

A=A+1

A will now contain the contents of A+1. The only exception to this format is the clear-data statement.

The general format for the clear-data statement is:

destination data element=

The destination data element may be a single element or an element of an array whose type may be alphanumeric or decimal. If it is alphanumeric, it is set to blanks. If decimal, it is set to zeroes.

QUESTION: What is accomplished by statements S1 through S3 in the following example?

```
START
BLOCK A,A10
      NUMS,D25
      BUF,A100
PROC
S1  A=
S2  NUMS=
S3  BUF(56,70)=
```

ANSWER: Statement S1 sets the ten-character element A to blanks. S2 sets the 25-character element NUMS to zeroes. S3 sets the characters 56 through 70 of element BUF to blanks.

***

An attractive feature of the DIBOL language is the ability to reference characters within an element. The notation BUF (56,70) allows the programmer to reference characters within a data element without assigning a specific data name. The general format to accomplish this is:

Data name (starting character position, end character position)

QUESTION: What would be accomplished by statements S1 and S2 in the following example:

```
START
BLOCK B
    NUMS,10D2
    B1,5A6
PROC
S1  NUMS (5)=
S2  B1 (4)=
```

ANSWER: S1 zeroes the fifth element of the array NUMS. S2 blanks the fourth element in the array B1.

***

This notation is called subscripting. It allows the programmer to reference a specific data element of an array. This form of subscripting can be either a positive numeric number or a data name. The data name option is called variable subscripting. For example:

```
START
BLOCK
    NUMS,10D2
    A,D1
PROC
A=5
NUM(A)=
```

This will accomplish the same as NUM(5)= which is in the previous example.

NOTE

An entire array cannot be referenced, only a single element within an array. However, it is possible to reference an entire array by redefining the array, using BLOCK,X (overlay). For example:

```
BLOCK
    NUMS,10D2
BLOCK X
    NUMS1,D20
PROC
    S1,NUM1=
```

This will set the entire array of NUMS to zero.

***

In summary, the Clear Data Statements have the following formats:

```
Destination data element= e.g., A=   ,
Destination data element (subscript)= e.g., A(4)=
                                      or    A(B)=
                                      or    A(51,71)=
```

ALPHA=ALPHA

The second type of data manipulation is the move-alphanumeric-variable statement. It takes the general form:

```
alpha data element=alpha data element
   (destination)          (source)
```

This allows one alpha field to be moved to another alpha field. If the source is shorter than the destination, the result is right-justified with the left-most characters set to blanks. If the source is longer than the destination, the right-most characters of the source are truncated (chopped off) in the move.

QUESTION: What is the value of A in the following example, after the move has been executed?

```
START
BLOCK
        A,A5,'ABCDE'
        B,A3,'FGH'
PROC
        A=B
```

ANSWER: Variable A now has the value bbFGH (b signifies blanks). Source is shorter than destination. Left-most characters are set to blanks.

***

QUESTION: What is the value of NAME in the following example?

```
START
BLOCK A
        NAME,A4,'FRED'
        NAME1,A7,'JOHNSON'
PROC
NAME=NAME1
```

ANSWER: NAME now has the value of JOHN.

NOTE

While the receiving field is changed (destination), the sending (source) field remains unchanged, so NAME1 still has the value JOHNSON.

In review, the general format of move alpha to alpha data element is:

```
Alpha data element = alpha data element
   (destination)           (source)
```

***

COMPUTE DECIMAL Statement

The third form of data manipulation is the compute decimal value statement. The general format for this expression is:

```
decimal data element=arithmetic expression
    (destination)           (source)
```

The arithmetic expression may be any expression with decimal elements, subscripted data elements constants, and the prorators plus (+), minus (-), multiply (*) and divide (/). The order of operation is, /, *, +, -, with the contents of parentheses being performed first. The destination data element would be right-justified after the move. Below is an example:

```
START
BLOCK
        QORDER,D4,0002
        UCOST,D4,0200
        ECOST, D8
        X,     D2,04
        Y,     5D3,000007100025023
PROC
  1     ECOST=UCOST * QORDER
  2     X=X+1
  3     Y(1)=Y(X)+(25*Y(2) + Y(3))/Y(4)
```

***

Statement 1) ECOST is calculated by multiplying UCOST and QORDER. In the above example, the answer would be: ECOST=00000400

Statement 2) The new value of X shall be X+1 (answer, X=05)

Statement 3) The first element in array Y will be equal to the fifth element in array Y (X=5), plus the quantity; second element in array Y plus third element in array Y divided by fourth element in array Y. The answer would be:

```
Y (1) = Y (05) + (25 * Y(2) + Y(3))/Y(4)
Y (1) = Y (023 + (25 * 007 +100))/025
      =023 + (275)/25
      =023 + 011
Y (1) =034
```

QUESTION: In the following expression, select the defined items; i.e., decimal data element, subscripted data element, constants, and the operators, plus, minus, multiply, and divide.

```
X=Y(3)+Y(2)+66*(13-Z)/2
```

ANSWER: Subscripted variables are Y(3),Y(2); decimal variable is Z; constants are 66,13,2; the arithmetic operators used are +,-,*, and /.

***

QUESTION: Is the expression X=Y(2) equal to X=Y*2?

ANSWER: No. Y(2) is a subscripted data element denoting the second element of an array with the name Y. The expression X=Y*2 is the equivalent of multiplying Y times 2 and storing it in X.

***

QUESTION: What is the expression which would accomplish the following?

a) Take a number X and add it to the second element in an array named K
b) Take the result of that operation and divide it by 145 and store it in M.

ANSWER: M=(X + K (2))/145

***

In summary, the compute decimal data element has the general format of:

decimal data element = arithmetic expression

A = A + B/C

CONVERT
Decimal-Alpha
Alpha-Decimal

Since all input/output information to the line printer (or teletype) must be in alphanumeric form, there must be a way to convert decimal information from one form to another.

The two forms of converting from one data type to another are:

a) Decimal data element=alpha data element
b) Alpha data element=decimal data element.

QUESTION: In the following example, data fields are described in both alphanumeric and decimal formats. Convert TOT from decimal to an alpha format of corresponding length and convert NUM to its decimal format of corresponding length.

```
START
BLOCK A
        NUM,A6
        A1,A6
BLOCK B
        TOT,D6
        B1,D6
PROC
```

ANSWER: A1=TOT (converts decimal to alpha)
B1=NUM (converts alpha to decimal)

***

The result of the conversion is always stored in the destination data element (the expression located to the left of the equals sign). The decimal-to-alpha conversion is always right-justified with leading blanks, if required. The alpha-to-decimal conversion is also right-justified with leading zeroes, if required.

QUESTION: What would be the contents of B1 and A1 after the following conversions?

```
START
BLOCK A
        COST,D4,9999
        A1,D5
BLOCK B
        NUM,A6,'678912'
        B1,A6
PROC
    B1=COST
    A1=NUM
```

ANSWER: B1=COST (converts decimal to alpha). B1 would contain bb9999, where b signifies blanks. A1=NUM (converts an alphanumeric number to decimal). A1 would contain 78912; the result is right-justified.

***

CONVERT
Alpha-Decimal
(formatted)

In business data processing, it is frequently desirable to output decimal information with imbedded commas and (if needed) a minus sign. For example, -34,259.00 reads easier than -3425900. It is troublesome however, for a programmer to worry about the mechanics of editing decimal information. DIBOL makes it possible to accomplish the formatting of decimal information during the conversion from decimal-to-alpha format. The general form of conversion is:

alpha data element=decimal element, format

For example, if B=125677700 (decimally formatted), the expression, A=B, '-X,XXX,XXX.XX' will move B to A and cause A to look like this:

1,256,777.00 (with no minus sign, since the number is greater than zero).

QUESTION: For B=4432567- (assume two decimal places); what would the conversion instruction look like?

ANSWER:
```
A=B'-XX,XXX.XX' or 'XX,XXX.XX-'
    (-44,325.67)  (44,325.67-)
```

The minus sign in the edit format can be either on the left or on the right. If the decimal value is positive, the sign will appear as a blank.

***

QUESTION: Since commas are inserted only if the corresponding comma has a nonzero digit to the left, if B=311 what would be the value of A after the following?

```
A=B,'-X,XXX,XXX.XX'
```

ANSWER: Where b signifies a space, A would be equal to bbbbbbbbb3.11. When decimal data element is converted, it is right-justified.

***

QUESTION: Device 8 is for Teletype output; what is typed out by the following program?

```
START
BLOCK A,C
        A1,A7
        A2,A4
        A3,A11
BLOCK
        NUM,D6,100000
        B1,A7,'CREDIT'
        B2,A11,'TO DIGITAL'
PROC
        A1=B1
        A3=B2
        A2=NUM,'X,XXX.XX'
        XMIT (8,A)
```

ANSWER: CREDIT 1,000.00 TO DIGITAL

***

The Alpha=Decimal (formatted) statement can use compiled-defined symbols for formatting the alpha data element resultant. The compiler-defined symbols are:

X) represents a digit
-) inserts a minus sign if number is negative
.) inserts period

,) will suppress leading zeroes unless there is an integer other than zero to the left.
Z) will suppress a digit position and right-justify it, if required. That is:

```
NUM,D3,987
A1=NUM,'XXZ'
(A1=b98 where b signifies a blank)
```

Any other symbol, except for those mentioned above, may be used for formatting. That is:

```
DATE=102370
A1=DATE,'XX/XX/XX
(A1-10/23/70)
          OR
A1=NUM'XXX0'
(A1=9870)
```

When using a comma, period, slash, minus sign, or any other notation, it must be counted as a character position. In the above example, B1 must be defined as an eight-character alphanumeric field.

QUESTION: What would the contents of statements a) through d) be in the following example:

```
    START
    BLOCK A
            A1,A8
            A2,A4
            A3,A4
            A4,A11
    BLOCK B
            DATE,D6,103070
            NUM,D3,123
            COST,D3,999
            TOT,D12,   7894211
    PROC
a)      A1=DATE,'XX/XX/XX'
b)      A2=NUM,'XXZ'
c)      A3=COST,'XXX0'
d)      A4=TOT,'-XXX,XXX.XX*'
```

ANSWER:
Statement a) A1=10/30/70
Statement b) A2=bb12 (b specifies blanks)
Statement c) A3=9990
Statement d) A4-b78,942.11* (b specifies blank)

***

In summary, the data manipulation statements have the following formats:

Operation	Format	Example
CLEAR	Clear Data Element=	(A=)
CLEAR	Clear Data Element (Subscripted)	(.A4=) or (A(B)=)
MOVE	Alpha Data Element=Alpha Data Element	(A=B)
COMPUTE	Decimal Data Element=Arithmetic Expression	(A=B*C/D)
CONVERT	Decimal=Alpha	(A1=NUM)
CONVERT	Alpha=Decimal	(B1=TOT)
CONVERT	Alpha=Decimal (formatted)	(A=B,'-XX,XXX.XX')

In most of the examples in which subscripting was used, it was done by referencing specific elements of an array, i.e., NUMS (2)=. In certain cases it is desirable to vary the value of the subscript or to have a variable subscript. This is done by using a data name for the subscript. For example:

```
START
        BLOCK C
                C1,10A5
        BLOCK
                A,D2
                B,A5,'DIBOL'
        PROC
                A=1
                C1(A)=B
```

This places the value of DIBOL in the first element of the array C1. If all elements of the array were to be set to the value DIBOL, the procedure section would look like:

```
         PROC
             A=1
BEGIN,       C1(A)=B
             A= A+1
             IF (A.LT.11)  GO TO BEGIN
             STOP
         END
```

NOTE

An important restriction for the data manipulation statements is that block names cannot be used. Block names can only be used in the XMIT statement. For example:

```
START
BLOCK AAA
        A1,A80
BLOCK BBB
        B1,A80
PROC
        AAA=BBB
        STOP
END
```

Statement AAA=BBB is not legal. A block name cannot be moved to a block name. A1=B1 is legal.

The next type of statement is the GO TO statement.

QUESTION: From the previous section (and using Foldout #2), what is the purpose of the basic GO TO statement?

ANSWER: This statement causes the program control to branch to the executable statement in the procedure section with the specified label, and has the form:

GO-TO statement-label

***

The statement label must be a label (tag). It cannot be a data name. A data name refers to an element which has been defined in the data section. A statement label is a tag associated with a statement in the procedure section.

QUESTION: Is the following use of GO TO correct?

```
START
BLOCK A
        A1,A90
PROC
        INIT (2,V,OUT)
        XMIT (2,A)
        GO TO START
        FINI    (2)
        STOP
END
```

ANSWER: No. START is not an executable statement. Executable statements are found only in the processing section of the program.

***

QUESTION: Is the following use of GO TO correct?

```
START
BLOCK B
        B1,A50
PROC
        INIT (2,V,IN)
LOOP,   XMIT (2,B,EOF)
        XMIT (8,B)
        GO TO LOOP
EOF,    FINI    (2)
END
$
```

ANSWER: Yes. LOOP is a tag associated with a statement in the procedure section. LOOP is not a data name.

***

Another type of GO TO statement is the Multi-way-Go To. It has the form:

```
GO TO (label-1,label-2,. . . . . label-10), decimal expression
```

For example:

```
GO TO (LOOP,RUN,STOPS),KEY
```

This statement reads "If decimal variable named KEY is equal to 1, then go to LOOP; if it is equal to 2, then go to RUN; and if it is equal to 3, go to STOPS. If KEY is equal to none of these (1,2, or 3) then go to the next sequential statement.

Label 1 value must begin with 1, label 2 value with 2, label 3 value with 3, etc. A GO TO statement can have up to ten 6-character labels with label values of 1 through 10. The label can only be a statement label (tag). The decimal expression may be a constant data element, subscripted data element or an arithmetic expression.

NOTE

If a DIBOL program statement exceeds a teletype line, it can be continued to the next line with a $ carriage return.

QUESTION: If NUM is equal to 2, what does the following accomplish?

```
GO TO (X1,X2,X3),NUM
```

ANSWER: The program branches to the statement labeled X2.

***

QUESTION: In the above example, if NUM is equal to 6, what happens?

ANSWER: The program continues to execute statements sequentially.

***

The last form of the GO TO is the conditional Go To statement. It takes the form:

```
IF (Item 1 . relation . item 2) GO TO label
```

The data items for comparison may be a data element, a subscripted data element, a constant, or an arithmetic expression. They must be both alphanumeric or both decimal. The relations are:

.EQ.	for equal
.NE.	for not equal
.LT.	for less than
.LE.	for less than or equal
.GT.	for greater than
.GE.	for greater than or equal

NOTE

The format requires a period before and after the two character relation codes.

If item 2 is an alphanumeric constant, it must be enclosed in apostrophies.

QUESTION: Write an equivalent DIBOL statement for the following. If NUMB is less than or equal to 46, then go to the statement labelled LOOP.

ANSWER: If (NUMB.LE.46) GO TO LOOP. If NUMB was defined as a D4 field, item 2 must be 0046.

***

QUESTION: Write an equivalent DIBOL statement for the following. If DESC is equal to HAPPY, go to the statement labelled FUN.

ANSWER: IF (DESC.EQ.'HAPPY') GO TO FUN. This assumes that DESC is defined as A5.

***

NOTE

Compare field (data elements) must be of equal length. If they are of unequal length you will always fail the comparison.

QUESTION: Is the following use of the conditional Go To statement correct?

```
START
BLOCK
     NUMB,D3,223
     ALPH,A3,'ZAP'
     TOTL,D3,999
PROC
BEGN,NUMB=NUMB+1
      IF(NUMB.EQ.ALPH)GO TO BEGN
END,STOP
END
$
```

ANSWER: Use of the conditional Go To is incorrect. You cannot compare NUMB (which is a decimal item) with ALPH (which is an alpha item).

***

CALL
RETURN

Also of importance are the CALL and RETURN statements. There is a method to use the same routine several places in a program (i.e., a complicated mathematical calculation). Instead of writing the same routine several places in the program, write it as a subroutine. To use the routine write:

CALL name of first statement

The CALL statement does two things. It puts the memory address of the following statement in the RETURN statement of the subroutine and then performs an unconditional branch to a subroutine. You cannot use a subroutine name in a GO TO statement. You can only CALL subroutines. (GO TO statement transfers program control to a statement label and does not automatically return). The return from a subroutine is to the next statement after the CALL statement. This is accomplished by the RETURN statement. For example:

To return from the subroutine to the place where you were, the subroutine uses the RETURN statement

```
PROC
      .
      .
      CALL LIST
      .
      .
LIST, operation
      operation
      RETURN
      .
      .
THEN, STOP
```

The RETURN statement causes the program control to return to the statement after the last CALL. A subroutine may be called by a subroutine (this is called nesting.) The maximum number of nested subroutines is ten.

QUESTION: Is the following correct use of the subroutine?

```
START .
      .
      .
PROC  .
      .
      .
      .
      CALL SUB1
      .
SUB1,X=X+1
      IF (X.NE.3) GO TO EXIT
      CALL SUB2
EXIT  RETURN
SUB2, X=X*2
      RETURN
      .
      .
```

ANSWER: The example is correct. SUB2 is an example of a nested subroutine, called by SUB1. SUB1 calls SUB2, which is multiplied decimal-data element X by 2 and returns to SUB1, which returns to the instruction following the original CALL statement.

***

STOP

The final DIBOL statement, which occurs frequently, is the STOP statement. It causes the program to terminate its execution and to return control to the DIBOL monitor. For example:

```
START
BLOCK A
        A1,A10
        A2,A2
BLOCK B,X
        B1,A12
        B2,D1,0
PROC
        INIT (2,V,IN)
LOOP,   XMIT (2,A,EOF)
        XMIT (8,B)
        GO TO LOOP
EOF,    FINI (2)
        STOP
END
$
```

This example would print BLOCK B from DECtape 2 onto the teletype and continue this until EOF whereby DECtape 2 would rewind. Then the program would terminate transferring control to the DIBOL monitor.

On Foldout #2 is a listing of a complete DIBOL-coded program. Examine it, and answer the following questions.

QUESTION: From statements 14 and 16, what is the function of INBUF?

ANSWER: INBUF is the input buffer, into which all data from device 1 is stored.

***

QUESTION: From statements 15 and 19, what is the function of the block named OUTBUF?

ANSWER: It is the output buffer, from which all records are written by device 2.

***

QUESTION: What is the purpose of the X in describing OUTBUF?

ANSWER: The first four fields of OUTBUF occupy the same area of computer memory as the four fields of INBUF.

***

QUESTION: Which statements separate the data section from the processing section?

ANSWER: The PROC statement (13).

***

QUESTION: Why should the input record occupy the same area of computer memory as the output record?

ANSWER: With the exception of the field named ECOST, the output records contain the same information as the input record. Thus, not only is computer memory saved, but many move instructions from one buffer to another are eliminated.

***

QUESTION: In our input and output buffers data is described as being in decimal format. Is this valid, and why?

ANSWER: This is valid. Only information coming from or going to the line printer (or teletype) must be described in alpha-format. We know devices 1 and 2 are not of this class, since they have been initialized (which makes them file-oriented devices).

***

QUESTION: Put statement number 17 into your own words.

ANSWER: "If data named STOCKN is less than 1000, then go to the instruction labeled LOOP. Otherwise, execute the next sequential instructions."

***

QUESTION: In the example program, STOCKN refers to a stock number, DESC refers to an item description, UCOST refers to unit cost of the item, QORDER is the quantity ordered, and ECOST denotes the extended cost. Describe in your own words the operation of this program (the logic).

ANSWER: The program reads fixed-length records containing a stock number, item description, unit cost, and quantity ordered. It skips records which have a stock number less than 1000. Output records are generated with the same information as the input with an additional item — an extended cost which is the product of the unit cost and the quantity ordered.

***

QUESTION: Put statement 15 into your own words.

ANSWER: "Initialize device 2 as an output device which will write variable length records, the size of which is determined by the size of the block of data named OUTBUF."

***

QUESTION: There doesn't seem to be any way for the program to execute statements beyond statement 20 (an unconditional branch). How is the statement labeled EOF executed?

ANSWER: Statement 16 carries the solution. It says "Read a record from device 1 and store the information in the block labelled INBUF. If there are no more records, go to the instruction labelled EOF."

***

QUESTION: How do we know statement 16 is a read statement? (two reasons)

ANSWER: First, statement 14 initializes device 1 as an input device; second, only read uses of the XMIT statement have three parameters (device, buffer, and end-of-file routine name); write statements have only two parameters (device-selection and name of output buffer).

***

You have completed an in-depth discussion of the DIBOL language. This is not a long section, so if you doubt you understand DIBOL clearly, by all means study the section a second time.

In summary, the procedure section has the following instructions:

1) Initialize File Device statement (as input or output)

General form:

```
INIT (file number,file type,direction)
```

Example:

```
INIT (2,V,IN)
```

2) Transmit statement (Read-from or write-into file)

General form:

```
XMIT (File number,block name,EOF for input files)
```

Example:

```
XMIT (2,INBUF,EOF) Read from
XMIT (1,OUTBUF) Write onto
```

3) Finalize File statement (rewind file)

General form:

```
FINI (file number)
```

Example:

```
FINI (2)
```

4) Data Manipulation Statement

General form:

destination data element=source data element or expression

a) Clear data (destination data element=)
b) Move alphanumeric data (alpha data = alpha data)

c) Compute decimal data (decimal data = decimal expression
d) Convert alpha to decimal (decimal data = alpha data)
e) Convert decimal to alpha (alpha data = decimal data)
f) Convert decimal to alpha, formatted (alpha data = decimal data, formatted)

5) GO TO statement (program control branches to statement label)

General form:

```
GO TO  statement label
```

Example:

```
GO TO LOOP
```

6) Multi-way GO TO statement (program control branches to label 1 if the decimal data element is 1, etc.)

General form:

```
GO TO (label 1,label 2 . . . . label 10),decimal data element
```

Example:

```
GO TO (TAX,COST,PRICE),A2
```

7) Conditional GO TO statement. (If statement is true, control goes to statement label.)

General form:

```
IF (data element 1 . relation . data element 2) GO TO LABEL
```

Example:

```
IF (A.EQ.B) GO TO C
```

8) Subroutine Call Statement (Control goes to statement label.)

General form:

```
CALL statement label
```

Example:

```
CALL COST
```

9) Return Statements (Program control returns to the statement after the last CALL.)

General form:

```
RETURN
```

10) STOP Statement (Causes program to terminate and transfers control to the monitor.)

General form:

```
STOP
```

SECTION III

THIS SECTION CONTAINS A REVIEW OF DIBOL AND A PROGRAMMING EXERCISE

FOLDOUT #2

SAMPLE PROBLEM #2

```
START                                                   ;1
BLOCK  INBUF                                            ;2
       STOCKN,    D4                                    ;3
       DESC,      A25                                   ;4
       UCOST,     D5                                    ;5
       QORDER,    D4                                    ;6
BLOCK  OUTBUF,X                                         ;7
       A1,        D4                                    ;8
       A2,        A25                                   ;9
       A3,        D5                                    ;10
       A4,        D4                                    ;11
       ECOST,     D9                                    ;12
PROC                                                    ;13
       INIT (1,V,IN)                                    ;14
       INIT (2,V,OUT)                                   ;15
LOOP,  XMIT (1,INBUF,EOF)                               ;16
       IF (STOCKN.LT.1000) GO TO LOOP                   ;17
       ECOST=UCOST*QORDER                               ;18
       XMIT (2,OUTBUF)                                  ;19
       GO TO LOOP                                       ;20
EOF,   FINI (2)                                         ;21
       FINI (1)                                         ;22
       STOP                                             ;23
END
$
```

The program reads variable length records containing a stock number, item description, unit cost and quantity ordered. It skips records which have a stock number less than 1000. Output records are generated with the same information as the input with an additional item — an extended cost which is the product of the unit cost and the quantity ordered.

The following is an outline of the elements which make up a DIBOL program:

I. The DATA SECTION contains:

  A. Data Statements:

    1. The BLOCK statement denotes the beginning of a specific group of data statements.

    2. The DATA STATEMENT describes all program variables.

II. The PROCEDURE SECTION contains the following elements:

  A. Initialize File Statement (INIT);

  B. Transmit Data Statement (XMIT);

  C. Finalize File Statement (FINI);

  D. The following set of DATA MANIPULATION statements:

    1. Clear Data Statement

    2. Move Alphanumeric Variable Statement

    3. Compute Decimal Variable Statement

    4. Convert Alphanumeric to Decimal Statement

    5. Convert from Decimal to Alphanumeric Statement

    6. Convert from Decimal to Alphanumeric; Formatted Statement

  E. The following forms of GO TO statements:

    1. The Conditional GO TO

    2. The Multi-way GO TO

    3. The Conditional GO TO

  F. The Subroutine Call Statement

  G. The Subroutine Return Statement

  H. The STOP

START and END statements are required at each respective end of a DIBOL-coded program. The PROC statement must separate the Data Section from the Procedure Section.

Since DIBOL is syntax oriented spacing of source statements is irrelevant. For example, the data statement NUMB,D6,123456 can be correctly rewritten as NUMB, D6, 123456. The one restriction is that imbedded spaces (in data names or procedural verbs) cannot be used. Examples or incorrect usage of spaces are:

```
a)  NU MB,          D 6,          123 4 5 6
b)  XM IT (1 0, INBU F, EOF)
```

The arrows in the examples above point to erroneous usage of spaces.

EXPLANATION OF FOLDOUT #2
INVENTORY PROBLEM

```
START
BLOCK    INBUF                                  ;DECtape 1, containing stock list
                                                  (Master File)
             STOCKN,D4                          ;Stock number, six digits
             DESC,A25                           ;Description, 25 characters
                                                  Alpha word
             UCOST ,D5                          ;Unit Cost, five digits
             QORDER,D4                          ;Quantity ordered four digits
BLOCK    OUTBUF,X                               ;DECtape 2, will contain new file
                                                  after Program Execution (X refers to
                                                  BLOCK overlays)
             ,D4
             ,A25
             ,D5
             ,D4
             ECOST,D9                           ;Extended Cost, seven digits
PROC                                            ;Begin Procedure Section
             INIT (1,V,IN)                      ;Initialize file 1, as an Input Device
             INIT (2,V,OUT)                     ;Initialize file 2, as an Output Device
LOOP,        XMIT (1,INBUF,EOF)                 ;Read a record from file 1, and store it
                                                  in area assigned to INBUF until end
                                                  of file
             IF (STOCKN.LT.1000) GO TO LOOP     ;If stock number is less than 1000, go
                                                  to read another record
             ECOST=UCOST*QORDER                 ;Extended cost would be calculated for
                                                  STOCKN 1000 or over
             XMIT (2,OUTBUF)                    ;Write the record onto file 2 from area
                                                  assigned to OUTBUF (overlay of
                                                  INBUF plus ECOST)
             GO TO LOOP                         ;Read another record
EOF,         FINI (2)                           ;OUTBUF file is rewound and EOF
                                                  mark is put at end of file
             FINI (1)                           ;INBUF file is rewound
             STOP
END                                             ;OUTBUF file will contain all stock
                                                  items with a stock number of 1000 or
                                                  over, with DESC,UCOST,QORDER,
                                                  and ECOST for each item.
$                                               ;A DIBOL program must be terminated
                                                  with a $
```

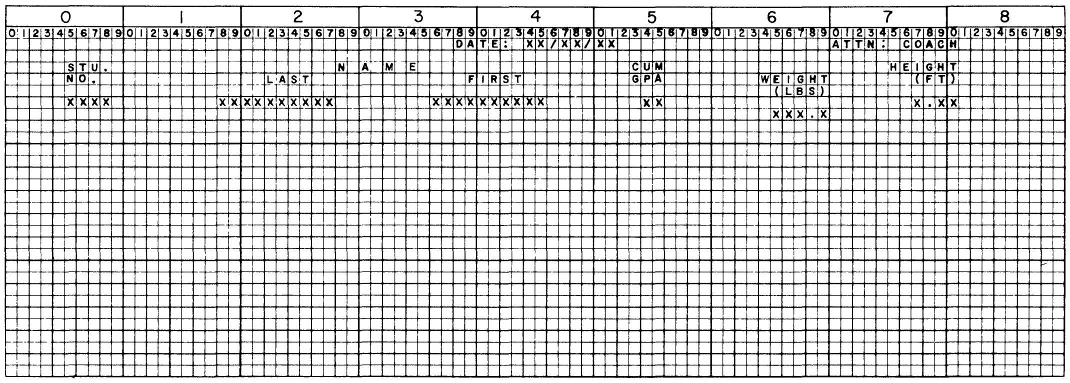
QUESTION: On this page is the definition of a program you are to write. It is imperative you complete writing the program before you look at this author's solution. It is also important that the student write the program during the same sitting in which he studied the previous two sections, for the simple reason that prompt reinforcement (through application) is the only way he will retain the thorough knowledge of DIBOL. Feel free to use sections I and II as a reference.

YOU ARE TO WRITE A PROGRAM FOR THE ATHLETICS DEPARTMENT OF A COLLEGE. STUDENT RECORDS ARE STORED ON MAGNETIC TAPE IN THE FOLLOWING FORMAT:

STUDENT NUMBER	LAST NAME	FIRST NAME	CUMULATIVE G.P.A.	SEX	WEIGHT (LBS)	HEIGHT (FEET)
4 digits	10 char.	10 char.	2 digits	1 char. M/F	4 digits XXX.X	3 digits X.XX

54-CHARACTER TAPE RECORD

THE COACH WANTS A LIST OF ALL MEN ON CAMPUS WHO HAVE A GRADE POINT AVERAGE ABOVE 85, WHO WEIGH OVER 170.0 POUNDS, AND WHO ARE OVER 5.75 FEET TALL. THE REPORT IS TO HAVE A HEADING AND LOOK LIKE THIS:



```
                                      DATE: XX/XX/XX                  ATTN: COACH

     STU.                   N A M E                  CUM                   HEIGHT
     NO.              LAST             FIRST         GPA        WEIGHT       (FT)
                                                                 (LBS)
     XXXX         XXXXXXXXXX        XXXXXXXXXX        XX                     X.XX
                                                                 XXX.X
```

DON'T LOOK AT OUR SOLUTION UNTIL YOU HAVE COMPLETED ALL WORK ON YOURS.

```
START
BLOCK    TAPEIN                              ;INPUT BUFFER FOR TAPE RECORDS
         STUNO,  D4                          ;Student Number
         LNAME,  A10                         ;Last Name
         FNAME,  A10                         ;First Name
         GPA,    D2                          ;Cumulative Grade Point Average
         SEX,    A1                          ;Sex (M or F)
         LBS,    D4                          ;Weight (XXX.X)
         FEET,   D3                          ;Height (X.XX)
BLOCK    TTYBUF                              ;Teletype Output Buffer
         FLD,    A80                         ;Print Line
         ENDL,   D1,0                        ;Causes Carriage-Return/Line-Feed
BLOCK    HEAD,C                              ;Heading line of Report
         ,       A37                         ;Filler
         H1,     A5, 'DATE'
         ,       A1                          ;Filler
         DATE,   A8,P                        ;Request date when program is loaded
                                              XX/XX/XX
         ,       A18                         ;Filler
         H2,     A11, 'ATTN:COACH'
         ,       D1,0                        ;Causes Carriage-Return/Line-Feed
BLOCK    COL1,C                              ;First line of column heading
         ,       A4                          ;Filler
         C1,     A4, 'STU'
         ,       A19
         C2,     A7, 'N A M E'
         ,       A18
         C3,     A3, 'CUM'
         ,       A8
         C4,     A6, 'WEIGHT'
         ,       A5
         C5,     A6, 'HEIGHT'
         ,       D1,0                        ;Causes Carriage-Return/Line-Feed
BLOCK    COL2,C                              ;Second line of column heading
         ,       A4                          ;Filler
         C10,    A3, 'NO.'
         ,       A14
         C11,    A4, 'LAST'
         ,       A13
         C12,    A5, 'FIRST'
         ,       A9
         C13,    A3, 'GPA'
         ,       A9
         C14,    A5, '(LBS)'
         ,       A7
         C15,    A4, '(FT)'
         ,       D1,0                        ;Causes Carriage-Return/Line-Feed
BLOCK
         LINECT, D2,50
PROC                                         ;BEGINNING OF PROCEDURE
                                              SECTION
         INTIT (1,F,IN)                      ;Initialize the input tape
         CALL HEADER                         ;PRINT REPORT HEADINGS
```

```
REPT,       XMIT (1, TAPEIN,EOF)                 ;READ INPUT TAPE
            IF (SEX.NE. 'M') GO TO REPT          ;**TEST TO Determine
            IF (GPA.LE.85) GO TO REPT            ;**IF record is to be
            IF (LBS.LE.170) GO TO REPT           ;**Select. If a test fails
            IF (FEET.LE.575) GO TO REPT          ;**Read another record
            FLD (5,8) = STUNO                    ;Fornett print
            FLD (18,27) = LNAME                  ;line by moving
            FLD (36,45) = FNAME                  ;all field to
            FLD (54,55) = GPA                    ;the appropriate print
            FLD (65,69) = LBS, 'XXX.X'           ;position. Edit feet
            FLD (77,80) = FEET, 'X.XX'           ;and lbs.
            CALL PRINT                           ;Print the line
            GO TO REPT                           ;Read another record

PRINT,      XMIT (8,TTYBUF)                      ;Print the line
            FLD=                                 ;Clear print line
            LINECT=LINECT +1                     ;Add one to line count
            CALL HEADER                          ;Test if header is to be printed
            RETURN                               ;Return to instruction after last call

HEADER,     IF (LINECT.LT.50) GO TO EXIT         ;Print header after every 50 lines
            LINECT=                              ;Set line count to zero
            XMIT (8,HEAD)                        ;Print header line
            FLD=                                 ;Clear Print Buffer
            XMIT (8,TTYBUF)                      ;Print blank line
            XMIT (8,COL1)                        ;Print first column header line
            XMIT (8,COL2)                        ;Print second header line
            XMIT (8,TTYBUF)                      ;Print blank line
EXIT,       RETURN                               ;Return to instruction after last call
            EOF,      FINI (1)                   ;Rewind input file
                      STOP                       ;Return control to DIBOL monitor
            END
```

APPENDIX

Coding Tips and Conventions

A) Printer

```
        Block                   HDF
                                , D2, 70            Head of Form
                                , A1
                                , D1, 0

        Block                   BLK
                                , D2, 00            Blank Line
                                , A1
                                , D1, 0
```

B) TTY Input

```
        Block                   TBUF

                                TP, A6              TP can be any size depending on
                                   , A1             amount of data expected.  Make
                                                    buffer at least one character larger
                                                    (for CR 10 is recommended)
```

C) TTY Output

```
        Block                   MSG

                                , A6,     'ENDJOB'
                                , D1,     0
```

D) You cannot use a "?" even in a literal.

E) Tape device assignment can be assigned at run time by:

```
        BLOCK       TMSG
                    , A7,          'TAPE NO'
                    , D1,          0

        BLOCK       ATAPE

                    TT, A1
                      , A1

        BLOCK       T1, D1
        PROC
        LOOP,       XMIT (8,TMSG)                   ;Types      TAPE NO
                    XMIT (7,ATAPE)                  awaits response from TTY.
                    T1=TT                           Move response 'convert' to decimal
        IF (T1.EQ.0) GO TO E1.                      If no more tapes are to be mounted,
                                                      Branch to Ending Routine
                    INIT (T1,V,IN)                  Initialize input tape
                    XMIT (T1,BUF,EOF)               Read
                           or

                         T1,D1,P                    This allows only one device assignment
```

F) Use of data names when referencing characters of an element

```
Block    P

           , D2,00
         PP, A80
           , D1,0

Block

         A, D1
         B, D1

Proc

         PP (1,4 = 'DATE')

               or

            A = 1
            B = 4
           PP (A,B) = 'DATE'
```

G) FORMAT EDIT

```
1)  DATE,D6, 031370
    PP (10,17) = DATE, 'XX/XX/XX'           Result = 03/13/70

2)  AMT, D10, 7942576301
    PP (10,22) = AMT, 'XX,XXX,XXX.XX'       Result = 79,425,763.01

3)  AMT,D5, 12345
    PP (1,3) = AMT, 'XXXZZ'                 Result = bb123

    PP (1,6) = AMT, 'XXXXX0'                Result = 123450

4)  AMT, D5, 12345
    PP (1,7) = AMT 'XXXXX**'                Result = 12345**
```

The format edit summary

```
Special symbols   X
                  .
                  ,
                  Z
                  0 (zero)
```

X	represents a digit
.	inserts the period, replaces zeroes to the left with blanks, prints zeroes to the right
,	The comma is not printed unless there is an integer other than zero to the left
Z	The Z suppresses the digit position
0	Zero is inserted in the position indicated

Symbols — Symbols other than those mentioned above, such as a / are inserted in the position indicated.

NITE — If the value of the sending field is zero, the result is leading blanks and one zero, except for the period in which case it is a period followed by all zeroes to the right

```
i.e.,     AMT, D5, 0
          DATE, D6, 0
          PP (1,6) = AMT'XX.XXX'      Result = bb.000
          PP (1,8) = DATE 'XX/XX/XX' Result = bbbbbbb0
```

INDEX

HOW TO OBTAIN SOFTWARE INFORMATION

Announcements for new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters.

Digital Software News for the PDP-8 Family
Digital Software News for the PDP-9/15 Family
PDP-6/PDP-10 Software Bulletin

These newsletters contain information applicable to software available from Digital's Program Library.

Please complete the card below to place your name on the newsletter mailing list.

Questions or problems concerning DEC Software should be reported to the Software Specialist at your nearest DEC regional or district sales office. In cases where no Software Specialist is available, please send a Software Trouble Report form with details of the problem to:

Software Information Service
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

These forms, which are available without charge from the Program Library, should be fully filled out and accompanied by teletype output as well as listings or tapes of the user program to facilitiate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

New and revised software and manuals, Software Trouble Report forms, and cumulative Software Manual Updates are available from the Program Library. When ordering, include the document number and a brief description of the program or manual requested. Revisions of programs and documents will be announced in the newsletters and a price list will be included twice yearly. Direct all inquiries and requests to:

Program Library
Digital Equipment Corporation
146 Main Street, Bldg. 1-2
Maynard, Massachusetts 01754

Digital Equipment Computer Users Society (DECUS) maintains a user Library and publishes a catalog of programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information please write to:

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

Send Digital's software newsletters to:

Name ______________________________

Company Name ______________________________

Address ______________________________

(zip code) __________

My computer is a	**PDP-8/I** ☐	**PDP-8/L** ☐
	LINC-8 ☐	**PDP-12** ☐
	PDP-9 ☐	**PDP-15** ☐
	PDP-10 ☐	**Other** ☐ __________ **Please specify**

My system serial number is ____________________ (if known)

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

PDP-8/I
DIBOL PROGRAMMING
DEC-08-WDRA-D

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback – your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability.

____

____

____

____

Did you find errors in this manual? ______________________________

____

____

____

____

How can this manual be improved? ______________________________

____

____

____

____

DEC also strives to keep its customers informed of current DEC software and publications. Thus, the following periodically distributed publications are available upon request. Please check the appropriate boxes for a current issue of the publication(s) desired.

☐ **Software Manual Update**, a quarterly collection of revisions to current software manuals.

☐ **User's Bookshelf**, a bibliography of current software manuals.

☐ **Program Library Price List**, a list of currently available software programs and manuals.

Please describe your position. ______________________________

Name ______________ Organization ______________

Street ______________ Department ______________

City ____________ State ____________ Zip or Country ____________

- Fold Here -

- - - - - - - - - - - - - - - Do Not Tear - Fold Here and Staple - - - - - - - - - - - - - - -

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

Digital Equipment Corporation
Maynard, Massachusetts

digital

printed in U.S.A.