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(2,0, '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.

  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.

CCP also allows several similar (but not identical) programs to be combined into one source program.

If the control variable used in a CCP statement is undefined, the compiler will automatically set aside space for it; this is wasteful of space. For CCP, use variables that are already being used for some other purpose.

The CCP value of a variable (on or off) is independent of the variable's ordinary DIBOL value.

If a CCP variable is used in the middle of a record definition (in the Data Division of a DIBOL program) the variable must have been previously defined, otherwise the Compiler will allocate additional space for it in the middle of a record.

CCP sections can be nested to any depth. Any CCP section that is turned off will be ignored by the Compiler. To indicate that certain statements are not being used, the Compiler listing will not print the line number for that statement. There must be a matching > for each < used. If this condition is not met, the Compiler generates a CCP ERROR message. This error is fatal if angle brackets do not match by the end of the program.

5.3 SIZE OF THE BINARY PROGRAM

Each variable uses as many bytes of memory as specified in its data definition statement. For example: a variable defined as 6D3 requires 18 bytes of storage. This is 9 words since a computer word consists of 2 bytes.

Variables defined in an overlay record share memory with the variables in the record being overlaid.

Each RECORD statement requires one additional word of memory. This word is reserved for storing the COS-310 word count during I/O operations.

Each record begins on a word boundary (an even-numbered byte address). If the record length is odd (in bytes), one byte of memory is wasted.

Each literal used in the Procedure Division of a DIBOL program requires storage (in bytes) equal to the length of the literal. The length of a numeric literal is equal to the number of digits in the number, including leading zeros.

Each distinct literal with a length of four or fewer characters appears only once in the space reserved for literals. Thus, if the literal 32 appears three times in the program, it will appear only once in the reserved data area. However, literals larger than four characters require space each time they appear in the program.

Each statement requires an overhead of one word. Each statement label requires one word. Unlabeled statements with line numbers 1000 more than the previous line number require one additional word each.

The number of words of memory generated by an expression can be determined by the following formula:

Add together the number of variables and literals used.

Add in the number of binary operators which appear. The binary operators include +, -, /, *, #.

Add one for each subscript reference.

The following table shows how many words of code are required by various DIBOL statements.

Table 5-1
DIBOL Statement Words of Code Requirements

Statement	No. of Words of Code Generated
ACCEPT (y,x)	y+x+1
CALL label	1
CHAIN chnum	chnum+1
DISPLAY (line,column,expr)	line+column+expr+1
END [/list control]	1
FINI (channel)	channel+1
FORMS (channel,skipcode)	channel+skipcode+1
GOTO label	1
GOTO (label1,...,labeln),key	key+n+2
IF (expr1.rel.expr2)stmnt	expr1+expr2+3
INCR var	var+1
INIT (channel, dev)	channel+2
INIT (channel, dev,filnam[,unit])	channel+3+filnam+unit
ON ERROR label	1
PROC [n] [/list control]	0
READ (channel,record,number)	channel+number+record+1
RETURN	1
START [/list control]	0
STOP	1
[NO] TRACE	1
TRAP	2
var=	var+1
var=expr	var+expr+1*
var=expr1,expr2	var+expr1+expr2+1
WRITE (channel,record,number)	channel+number+record+1
XMIT (channel,record[,label])	channel+record+2

For the statement marked with an asterisk (*) in the previous table, subtract 1 if the principal operator of expr is binary + or -, and if both types are numeric.

Example:

```
D = 3+5       takes 4 words of storage, while
D = 3*5       takes 5 words. Similarly,
D = 3+4+5     takes 6 words while
D = 3*(4+5)   takes 7 words.
```

Additional space is also required by the internal symbol table. This table consists of two words for each distinct variable, statement label, or literal used.

5.4 COMPILER ERROR MESSAGES

Most Compiler error messages are printed on the source listing directly after the line on which the error occurs. A caret (^) in the error message points to the approximate location of the error. Other errors are listed in the storage map listing.

Message	**Explanation**
BAD ALPHA VALUE	Initial value in an alphanumeric data definition statement did not begin or end with a single quotation mark. Insert single quotes.
BAD NUMERIC VALUE	The initial value for a numeric field was incorrectly formed. Check and form correctly.
BAD PROC #	The number in a PROC statement was not a digit from 0 to 7. Enter 0 through 7.
BAD RELATIONAL	An illegal relational occurs in an IF statement. For example, a .GX. instead of a .GT. Retype correctly.
CCP ERROR	Matching angle bracket (< or >) missing. Insert brackets.
COMMA MISSING	No comma appeared where one was expected. Insert comma.
DATA INITIALIZATION MISSING	No data initialization followed a comma in a data definition statement. Remove comma or input initial value.

Message	Explanation
EXPECTED LABEL IS MISSING	A required label is missing. Enter label.
EXPRESSION NOT ALLOWED	A complex expression or bad character occurs to the left of an = or where only a variable is allowed. Find and correct.
EXTRA CHARS AT STMNT END	Extra characters occur at the end of a legal statement. Eliminate extra characters.
FIELD TOO LARGE OR 0	In a data description statement, the dimension was 0 or more than 3 digits long, or the field size was 0 or larger than 511. Bring size dimensions within limits.
ILLEGAL OPERATOR	A bad character was encountered in an expression where an operator would be expected. Check and replace with correct character.
ILLEGAL STMNT	The statement was not a data manipulation statement (it had no =) nor did it start with a recognizable keyword. Use appropriate keyword and use = sign.
INITIAL VALUE WRONG SIZE	The initial value in a data specification statement had a length different from the field size specified. Make initial value agree with defined size.
LABEL NOT ALLOWED	A label in an expression was the wrong type or a record or a label which had been redefined was used. Use unique label of the correct type.
MISSING CLOSE PAREN	No close parenthesis occurred where one was expected. Add parenthesis.
MISSING OPEN PAREN	No open parenthesis occurred where one was expected. Add parenthesis.
MISSING OPERAND	A binary operator occurs in an expression with no operand following it; or no expression at all occurs where one is expected. Insert operand and/or appropriate expression.

Message	Explanation
MISSING OR BAD MODE	The mode designation in an INIT statement was missing or began with an illegal character. Insert correct mode designation.
MISSING QUOTE	The statement contained an odd number of quotes ('). Delete or add quote when appropriate.
MISSING RELATIONAL	No relational appeared in an IF statement. Enter legal relational.
NAME PREVIOUSLY DEFINED	An attempt was made to redefine a previously defined name. Use unique name.
NOT A OR D	A character other than A or D occurred in a data specification statement where A or D was expected. Replace character with A or D.
NOT LABEL	A symbol which was not a 'label' occurred where a label was required. Enter proper label.
PROGRAM TOO BIG	Binary output too big for the binary scratch area. Enlarge scratch area with PIP OPT- E.
RECORD TOO BIG	A named record exceeded 510 characters in size. Either use unnamed record or reduce the size of record.
STMNT TOO COMPLEX	The statement is too complex or is nested too deep. Simplify the statement.
SUBSCRIPT ERROR	No comma or close parenthesis occurred after a subscript. Enter appropriate punctuation.
SUBSCRIPT NOT NUMERIC	The type of a subscript was not numeric. Use numeric subscript.
TOO MANY ITEMS	More elements were initialized in an array than are specified in the field dimension. Eliminate excess elements.

Message	Explanation
TOO MANY SYMBOLS!	A fatal error message. Only 365 symbols allowed in symbol table in 16K-byte system, and only 511 symbols allowed in larger systems. The compiler stops compiling; no storage map can be produced. Rewrite and shorten program.
TOO MUCH DATA	Data Division exceeds 24K bytes. Rewrite program.
UNDEFINED NAME	A name is used which was not defined in the Data Division. Define this name or use a name already defined.
WRONG DATA TYPE	Mixed data types occurred in an expression, an argument which was supposed to be numeric was not, or one of the arguments in a data manipulation statement was of the wrong type. Replace the data.

CHAPTER 6

DIBOL DEBUGGING TECHNIQUE (DDT)

The DIBOL Debugging Technique (DDT) is used to debug binary programs. If a program is compiled with the DDT option (/D), the compiled binary program automatically branches to DDT upon execution. The features of DDT include breakpoint, variable examination, subroutine call trace-back, and iteration.

6.1 DDT OPERATING PROCEDURES

To execute a binary program with DDT, type:

```
RUN [ { pronam                    } ][,filnam1...,filnam7]/D
    { chain0+chain1...+chain7     }
```

where:

pronam is the name of the binary program to be debugged.

If the program name is omitted, the Monitor loads and executes the DIBOL program in the binary scratch area.

chain0+chain1...
are binary programs which constitute one large program broken up into several chained programs. These are the programs to be debugged.

filnam1...,filnam7
are names of source files on the system device.

/D is the option switch that requests DDT.

An additional 768 words of memory plus 3 words for each label in the Data Division are required because of the /D option.

During execution of the program, control is passed to DDT. The DDT program displays an appropriate DDT version number followed by a hyphen (-) to indicate that it is ready to accept commands.

6.2 DDT COMMANDS

Command	Explanation
variable=	Display the contents of variable (a label from the Data Division). Variable can have single or double subscripts.
variable=v	Set variable equal to v (v is any legal alphanumeric string). If v has more characters than defined for variable, ERR IN CMD is displayed.
=	Display the contents of the last variable examined.
=v	Set the last variable examined equal to v.
$nnnn	Set a breakpoint at line nnnn. One breakpoint is active at a given time. A breakpoint set at line 0 is meaningless because the program never executes line 0.
>n	Execute the breakpoint at the nth occurrence of line nnnn. For example: -$300 ->4 When the program starts to execute line 300 for the fourth time, the breakpoint is executed and control is transferred to DDT.
CTRL/Z	Start execution of DIBOL program. If a breakpoint, $, is set at line number nnnn, control reverts to DDT when nnnn is reached and the following message is printed: BREAK! Type additional commands in response to the hyphen (-).

↑ Display the lines from which calls (CALL or TRAP commands) were made (pushdown stack) during execution of the DIBOL program. This command is generally used to trace the execution of the program after a breakpoint or system error has occurred. The is usually the shift/six key.

While a DDT breakpoint is pending, if a DIBOL program error causes a message such as ILLEGAL SUBSCRIPT or NUMBER TOO LONG to appear, control is transferred to DDT; DDT commands can be used for program examination. If an error is fatal, the DIBOL program cannot be restarted by the CTRL/Z command.

Once a DIBOL program is running under DDT, DDT cannot be restarted unless a breakpoint occurs or an error occurs with a breakpoint pending. Therefore, if you do not require a breakpoint but want to return to DDT for program examination if an error occurs, set a breakpoint at a line number which will not be executed.

6.3 DDT ERROR MESSAGES

Message	Explanation
ERR IN CMD	Entered an invalid DDT command. Correct the command and retry.

CHAPTER 7

CROSS REFERENCE PROGRAM (CREF)

The Cross Reference Program (CREF) is primarily an aid to program development. It provides a table showing an alphabetical listing of all labels used in a DIBOL source program, the line number where each label is defined, and the line numbers where each label is used.

7.1 CREF OPERATING PROCEDURES

To execute CREF, type:

```
RUN CREF[,filnam1...,filnam7]
```

where:

filnam1...,filnam7
: are the parts of a DIBOL source program (maximum 7). If no files are specified, the program in the edit buffer is used.

The CREF program reads the DIBOL program, lists the cross-reference table on the line printer, and returns control to the Monitor.

CREF requires 16K bytes of memory and can handle any 16K-byte program that does not have an excessive number of symbols and labels. If 24K bytes or more is available, CREF expands its cross-reference table to make use of the available space.

A minimal amount of error checking is performed by CREF; no attempt should be made to cross reference programs having compilation errors. If CREF finds a line it cannot work with, it prints:

```
nnnn IS BEING IGNORED
```

where:

nnnn
: is the number of the line CREF cannot work with.

Following is the cross-reference table for the DIBOL program in Figure 1-1 of Chapter 1.

```
COS-310    CREF          V 8.00    24-MAY-78    WED          PAGE  1

LABEL      DEF       REFERENCES

DESC       130
ECOST      220       340
EOF        390       310
INBUF      110       310 320
LOOP       310       330 380
OUTBUF     170       320 360
QORDER     150       340
STOCKN     120       330
TITLE      260       280
UCOST      140       340

LABELS DEFINED BUT NEVER REFERENCED:  01
```

where:

LABEL is the name of the label used in the program.

DEF is the line number in the program where the label is defined.

REFERENCES are the line numbers where each label is referenced.

7.2 CREF ERROR MESSAGES

Message	Explanation
nnnn IS BEING IGNORED	CREF detected a line it cannot work with. Usually means an invalid DIBOL statement. Check line number for valid statement and retry.

CHAPTER 8

PERIPHERAL INTERCHANGE PROGRAM (PIP)

The Peripheral Interchange Program (PIP) moves files between two logical units, copies the contents of one device onto another, and consolidates files to remove free blocks. PIP is also used to allocate more space to the binary scratch area.

8.1 PIP OPERATING PROCEDURES

To execute PIP, type:

```
RUN PIP[,cmndfl][/n]
```

where:

,cmndfl is a previously stored file containing PIP commands. Each command is on a separate line; no blank lines or comments are used. When the command file is specified, PIP reads a line from the file each time one of the following prompts is displayed:

```
OPT-
IN-
OUT-
MORE?
TYPES OF FILES TO BE SKIPPED (S,B,V):
```

An end-of-file mark terminates the command file and requires all responses to come from the keyboard.

The command file is ignored on machines with less than 24K bytes of memory capabilities.

Example:

If the cmndfl EXAMP contains:

```
0100  C
0110  RX0
0120  RX1
0130  X
```

Then:

```
.RUN PIP,EXAMP     ;will copy RX0 to RX1.
```

/n indicates the number (0-9) of segments to allocate to the binary scratch area. The /n switch is used in conjunction with OPT- E, but is entered at the time that the RUN PIP command is typed.

PIP responds to the RUN PIP command with:

```
PIP V 8.00 (or current version number)
OPT-
```

Respond with one of the following options:

Option	Explanation
B	transfer a binary file
C	copy device
D	transfer a data file
E	consolidate directory space
I	copy and verify data
R	perform a read/check
S	transfer source file
V	transfer system file
X	return to Monitor

After you respond, PIP displays IN and OUT questions requesting option-dependent information.

Following is a summary of the PIP options and the information being requested by the IN and OUT questions.

OPT	IN	OUT
B	filnam[,dev]	filnam[,dev]
C	dev	dev
D	filnam[/logical unit #] /K	filnam[/logical unit #] /L /T
E	dev	dev
I	dev	dev
R	dev	
S	filnam[,dev]	filnam[,dev]
V	filnam[,dev]	filnam[,dev]

8.1.1 Transfer Binary File (OPT- B)

Type B in response to OPT- to move a binary program between two file-oriented devices.

Answer the IN question with the name of the binary program to be moved and, optionally, a comma and an input device designation. If no device is designated, the system device is assumed.

Answer the OUT question with the name to be assigned to the output file and, optionally, a comma and an output device designation. If no device is designated, the system device is assumed.

If you attempt to move data to or from a non-file-oriented device, the IN or OUT message is repeated.

Example:

```
.RUN PIP
PIP V8.00
OPT- B          ;Request move of binary file.
IN- TEST,DK0    ;File named Test from DK0.
OUT- TEST,DK1   ;File named Test to DK1.
OPT- X
```

8.1.2 Copy Device (OPT- C)

Type C in response to OPT- to copy the contents of one device onto a similar device.

Answer the IN question with a device designation.

Answer the OUT question with a device designation.

Example:

```
.RUN PIP
PIP V8.00
OPT- C          ;Request a copy between devices.
IN- DKO         ;Input device is DK0.
OUT- DK3        ;Output device is DK3.
OPT- X
```

8.1.3 Transfer Data Files (OPT- D)

Type D in response to OPT- to transfer data files between devices.

Answer the IN question with a file name and optionally, a logical unit number (1-15) preceded by a slash, or answer the IN question with /K.

Answer the OUT question with a file name and optionally, a logical unit number preceded by a slash, or answer the OUT question with a device switch. Output device switches are:

```
/L   printer
/T   screen
```

When the end of the input file is reached, PIP asks:

```
MORE?
```

Type N and the RETURN key if there is no more input or Y and the RETURN key to specify more input.

PIP transfers alphanumeric data only. A negative number is treated as the letter which has the equivalent code.

Examples:

```
.RUN PIP
PIP V8.00
OPT- D
IN- EMPNAM/1    ;Dump EMPNAM from logical unit 1 onto the printer.
OUT- /L
MORE? N
OPT- X
```

```
.RUN PIP
PIP V8.00
OPT- D
IN- HRPAY,2     ;Combines two data files into one output file.
OUT- PAYFIL/1
MORE? Y
IN- SALPAY,3
MORE? N
OPT- X
```

8.1.4 Consolidate Space in Directory (OPT- E)

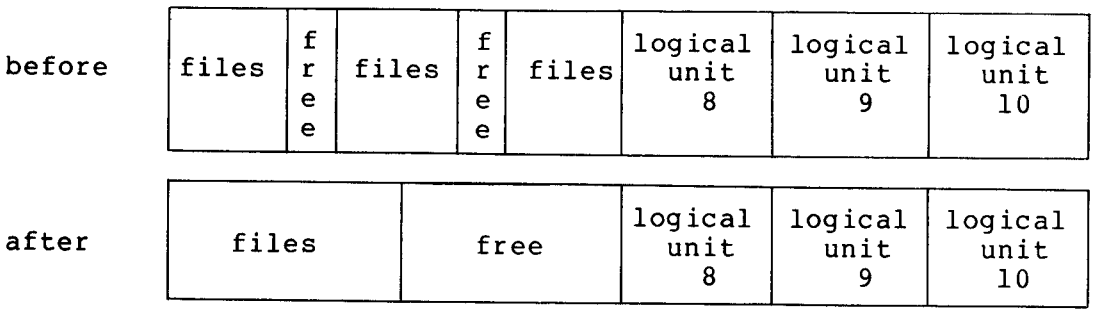
Type E in response to OPT- to consolidate the free blocks on the input device and store the files on the output device. It is possible to erase one or two of the kinds of files (source, binary, or system) during the consolidation. Free blocks are shown in the file directory and are created when a file is deleted from the directory. The bootstrap, Monitor, and logical units are not copied by OPT- E.

Answer the IN question with a device designation.

Answer the OUT question with a device designation.

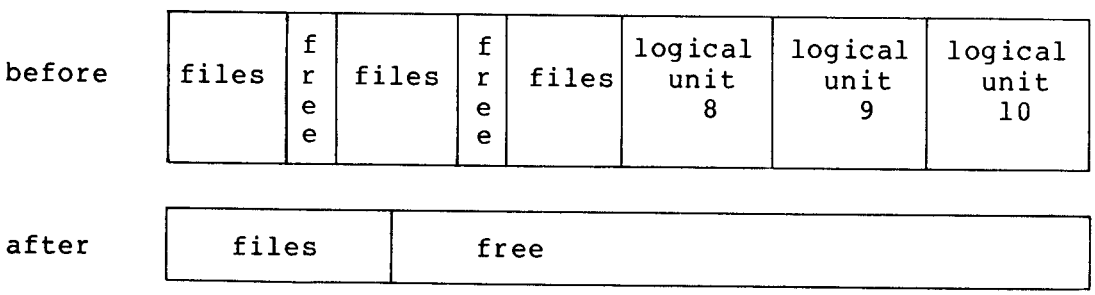
When consolidating the system device onto itself, PIP OPT- E eliminates the free space as shown below:

SYSTEM DEVICE (SYS)



When consolidating a device other than the system device onto another device, PIP OPT- E consolidates the free space but does not copy logical units.

NONSYSTEM DEVICE



The following message asks which files you do not want copied, consolidated, and stored. The files you choose to skip will be erased.

```
TYPES OF FILES TO BE SKIPPED (S,B,V):
```

If all files are to be copied, consolidated, and stored, type the RETURN key. If one or two of the types of files are to be skipped (erased), type one or a combination of two of the characters S, V, B separated by a comma and followed by the RETURN key.

The following example will consolidate free space but will not copy source files from DK0 to DK1.

Example:

```
.RUN PIP
PIP V 8.00
OPT- E
IN- DK0
OUT- DK1
TYPES OF FILES TO BE SKIPPED (S,V,B): S
```

All CTRL keys are ignored until the PIP OPT- E consolidation operation is completed.

8.1.5 Allocate Space to Binary Scratch Area (OPT- E)

The PIP OPT- E is used in conjunction with /n to change the size of the binary scratch area. The /n is typed along with the RUN PIP command.

Extremely large DIBOL programs may need more space than available in the two segments (32 blocks) usually allocated to the binary scratch area. Up to nine segments can be allocated with PIP OPT- E.

The following PIP operation will allocate two additional segments (32 blocks) to the binary scratch area on DK0. This particular operation uses /2 as the /n option typed in the RUN PIP command.

```
.RUN PIP/2
PIP V 8.00
OPT- E
IN- DK1
OUT- DK0
TYPES OF FILES TO BE SKIPPED (S,V,B):
```

The following information is helpful when using OPT- E to change the size of the binary scratch area.

- If the output device is not the input device, the size of the binary scratch area on the output device equals the sum of the two segments normally in the binary scratch area plus the number of segments stipulated by the /n in the RUN PIP command.

- If a device is consolidated onto itself, the binary scratch area is set either to the size of the current area or to 2+n, whichever is less. Compressing a device onto itself can shrink the binary scratch area. The binary scratch area cannot be expanded if a device is being compressed onto itself because that would require writing over existing files.

- If /n is not specified in the RUN PIP command, the binary scratch area is assumed to be the same size as the binary scratch area of the input device.

8.1.6 Copy and Verify (OPT- I)

Type I in response to OPT- to copy an entire device onto a similar device and verify the copy.

Answer the IN question with a device designation.

Answer the OUT question with a device designation.

Example:

```
.RUN PIP
PIP V8.00
OPT- I          ;Request copy and verify.
IN- DK1         ;Input device is DK1.
OUT- DK2        ;Output device is DK2.
OPT- X
```

If your machine configuration includes an LQP printer, PIP OPT- I will require 32K bytes of memory.

8.1.7 Perform a Read/Check (OPT- R)

Type R in reply to OPT- to verify the readability of a device.

Answer the IN question with the designation of the device to be read.

No OUT question is displayed.

Example:

```
.RUN PIP
PIP V8.00
OPT- R          ;Request Read/Check.
IN- DK1         ;Read contents of DK1.
OPT- X
```

8.1.8 Transfer Source Files (OPT- S)

Type S in response to OPT- to transfer source files between two file-oriented devices.

Answer the IN question with the name of the source file to be transferred and, optionally, a comma and the input device designation. If no device is specified, the system device is assumed.

Answer the OUT question with the name to be assigned to the output file and, optionally, a comma, and the output device designation. If no device is specified, the system device is assumed.

Example:

```
.RUN PIP
PIP V8.00
OPT- S          ;Request transfer of source file.
IN- TEST,DK0    ;Transfer TEST from DK0.
OUT- TEST,DK1   ;Receive TEST into DK1.
OPT- X
```

If you attempt to transfer to or from a non-file-oriented device, the IN or OUT question is repeated.

8.1.9 Transfer System Program (OPT- V)

Type V in response to OPT- to move a system program between two file-oriented devices.

Answer the IN question with the name of the system program to be transferred and, optionally, a comma and the designation for the input device. If no device is specified, the system device is assumed.

Answer the OUT question with the name to be assigned to the output file, and, optionally, a comma and the output device designation. If no device is specified, the system device is assumed.

Example:

```
.RUN PIP
PIP V8.00
OPT- V          ;Request transfer of system program.
IN- SORT,DK1    ;Transfer SORT from DK1.
OUT- SORT,DK3   ;Transfer SORT to DK3.
OPT- X
```

If you attempt to transfer to or from a non-file-oriented device, the IN or OUT question is repeated.

8.1.10 Return to Monitor (OPT- X)

Type X in response to OPT- to terminate PIP and return to the Monitor.

This OPT- X feature is useful when PIP is included in a string of Monitor commands in a BATCH program. The OPT- X signals the end of the PIP program and the next Monitor command in the BATCH program is executed.

8.2 PIP ERROR MESSAGES

Message	Explanation
BAD DIRECTORY	Attempted to reference or store a file on a device with a damaged or nonexistent directory. Only files with directories can be used. If the directory is damaged, call your Software Specialist.
COMPARISON ERROR	The verification part of OPT- I found a discrepancy of information between the original text and its copy. Retry the operation. If discrepancies continue, you have a media problem or a hardware problem.
ILLEGAL DEVICE SWITCH	A switch was specified for OPT- D that was not /K for input or /L or /T for output. Use one of the allowable switches.
NO ROOM	Attempted to store a file on a full device. Stipulate another device.

CHAPTER 9

SORT PROGRAM (SORT)

SORT is a utility program that arranges the records within COS-310 data files according to your needs. The data files must contain fixed-length records.

Before you can execute SORT you must write a SORT command file that defines the records to be sorted, specifies labels for input and output files, and designates the arrangement (key) to be used in the sort.

SORT uses a command file to rearrange one data file at a time. There must be a separate SORT command file for each file sorted.

The SORT command file sorts each volume of a multivolume file separately. A merge pass must then be done to combine the volumes into one file.

9.1 SORT OPERATING PROCEDURES

To execute SORT, type:

```
RUN SORT,cmndfl1...,cmndfl7[/L]
```

where:

cmndfl1...,cmndfl7
: is the SORT command file which can be stored as more than one file. This file defines the records to be sorted, specifies the labels for input and output files, and designates the arrangement (key) to be used. If no files are specified, the command file in the edit buffer is used.

/L
: lists the SORT command file on the printer.

9.2 SORT COMMAND FILE

The SORT command file is created with editor commands and written on a mass storage device. The command file consists of a Record Descriptor Division and an INPUT/OUTPUT Division.

9.2.1 Record Descriptor Division

The Record Descriptor Division defines the fields within the records to be stored. This division has the form:

```
DEFINE
Fs, xn
.
.
.
```

where:

DEFINE — is the division heading (must be DEFINE) and is the first statement in the file.

Fs — are the fields in the record (must be F). All fields must be defined in the Record Descriptor Division and numbered in the order they appear in the record. These numbers (s) begin with 1, are nonskipped sequential, and cannot exceed 511. Total record size cannot exceed 510 characters.

xn — represents the field type (alphanumeric or numeric), and the number of characters (1-510) in the field.

Each field descriptor statement (Fs) is entered on consecutive lines and is terminated with the RETURN key.

9.2.2 INPUT/OUTPUT Division

The INPUT/OUTPUT Division specifies the names of input and output files and how many logical units are to be used for work areas during the SORT operation. This division has the following format:

```
INPUT [filnam][/logical unit #][,filnam][/logical unit #]
[SORT n /logical unit #,...logical unit #]
KEY Fs[(m,n)][-],...
OUTPUT [filnam][/logical unit #]
END
```

where:

INPUT [filnam][/logical unit #][,filnam][/logical unit #]

is the name of the file containing the records to be sorted. If no name is specified, SORTIN is assumed. If the command file is used for a separate merge operation, the second file name is the name of a file to be merged with the first. The logical units identify the storage location of the file.

[SORT n /logical unit #...,logical unit #]

is the number (n) of logical units (3 to 7) to be used as work areas during the sort. These work areas are labeled $WORK1, $WORK2...,etc. If the SORT statement is not present, 4 units are assumed. The size of the work units should be as large as one volume of the input file. The logical unit numbers are default work units. Using default units bypasses the MOUNT message.

KEY Fs[(m,n)][-],...

Fs is the field name (F) and number(s) of the field to be used as the SORT key. The [(m,n)] delimit the part of the field to be used as a SORT key. If no characters are specified, the entire field is used as a SORT key. The - requests a SORT in descending order. Up to eight fields or parts of fields can be specified for the SORT key. The total size of the fields which comprise the key cannot be larger than 510 characters. The SORT is done left to right. The leftmost key is most significant, and the leftmost character in each key field is most significant for sorting purposes.

OUTPUT [filnam][/logical unit #]

is the file name to be given to the sorted records. If this statement is missing, SORT assigns the name SRTOUT to the output. For multivolume files, the names $TMPnn (nn can be any two-character numeric from 00 to 99) are used. The /logical unit number is the default unit for the output file.

END terminates control program.

Following is an example of a SORT command file:

```
0010   DEFINE
0020   F1,D6             ;Part number.
0030   F2,A30
0040   F3,D7
0050   F4,D6             ;Date.
0060   F5,D6
0070   INPUT PRTFIL/1    ;Data file name.
0080   SORT 4/2,3,4,5    ;Work units.
0090   KEY F1-,F4        ;Sort part numbers in descending order,
                         ;Sort date in ascending order.
0100   OUTPUT PRTFIL/1   ;Sorted data file name.
0110   END
```

9.3 MERGE OPERATING PROCEDURE

To execute SORT as a merge operation, type:

```
RUN SORT,cmndfl1...,cmndfl7/x[L]
```

where:

cmndfl1...,cmndfl7
: is the command file (possibly stored in two or more files).

/x
: is one of the following switches:

  /A
  : names of files to be merged are entered from the keyboard in answer to the message INPUT FILE LABELS:. The output data file and default unit name are specified in the OUTPUT line of the command file.

  /M
  : names of files to be merged are listed in the INPUT line of the SORT command file. This option bypasses the message INPUT FILE LABELS:.

  /n
  : name of files to be merged (all files must have the same name) is listed in the INPUT line of the SORT command file. This checks for the number of files with the same name on the number of default units as specified. If the number of units specified is more than the number of units shown on the SORT control INPUT line, a MOUNT message is displayed for those files not on the INPUT line.

/L
: can optionally be used with any of the above switches and lists the SORT control program on the printer.

9.3.1 Merge Using SORT and the /A Option

To use the SORT program with the /A option to merge data files, first write a SORT command file.

Following is a sample command file named PAYKEY:

```
0100   DEFINE
0110   F1,A6
0120   F2,D5
0130   F3,D11
0140   INPUT
0150   SORT 3/1,2,3
0160   KEY F2
0170   OUTPUT PAYROL/6
0180   END
```

To execute this sample command file with the SORT program and the /A option, type:

```
·RUN SORT,PAYKEY/A
```

The program displays the following message to request the names of the data files to be merged:

```
INPUT FILE LABELS:
```

Enter up to a maximum of six data file names and default units. Enter at least two names or the error message NO INPUT is displayed.

After you enter the file names and default units, the program is executed. There are three SORT work units: logical units 1, 2, and 3. The output data file is PAYROL; the sorted file is stored on unit 6.

9.3.2 Merge Using SORT and the /M Option

To use the SORT program with the /M option to merge data files, first write a command file.

Following is a sample command file named PAYKEY:

```
0100   DEFINE
0110   F1,A6
0120   F2,D5
0130   F3,D11
0140   INPUT PAYROL/4, PAY1/2
0150   SORT 3/1,2,3
0160   KEY F2
0170   OUTPUT PAYROL/6
0180   END
```

To execute this sample command file with the SORT program and the /M option, type:

```
.RUN SORT,PAYKEY/M
```

The input data file names to be merged are found in the INPUT line of the control file:

```
PAYROL on logical unit 4
PAY1 on logical unit 2
```

The output data file, PAYROL, is put on logical unit 6.

9.3.3 Merge Using SORT and the /n Option

To use the SORT program with the /n option to merge data files with the same name, first write a command file. The INPUT line contains the name common to the files and the default units where the files are found.

Following is a sample command file named PAYKEY:

```
0100   DEFINE
0110   F1,A6
0120   F2,D5
0130   F3,D11
0140   INPUT PAYROL/4,2
0150   SORT 3/1,2,3
0160   KEY F2
0170   OUTPUT PAYROL/6
0180   END
```

To execute this sample command file with the SORT program and the /n option, type:

```
.RUN SORT, PAYKEY/2
```

The input data files to be merged are:

```
PAYROL on logical unit 4
PAYROL on logical unit 2
```

9.4 SORT ERROR MESSAGES

Message	Explanation
BAD DIGIT IN NUMERIC INITIAL VALUE	Alphanumeric character in a numeric initial value. Use only numeric characters in initial numeric values.
BAD RECORD SIZE	File contains records of variable length. All records to be stored must be the same length. Redefine record length.
BAD WORK UNIT COUNT	Number of work units not in range 3-7. Specify work units within allowable range.
EXTRA CHARS AT STMNT END	Characters not relating to statement appear on the statement line. Delete any nonessential characters from statement lines.
FIELD NUMBER MISSING OR 0	Field number is missing, is 0, or is greater than or equal to 512. Enter the missing number or enter a number between 1 and 511.
ILLEGAL SORT KEY	Bad syntax on KEY statement, KEY too complex, or KEY statement missing. Review key information and correct command file.
ILLEGAL UNIT	Default unit is 0 or greater than 15. Correct command file.
INITIAL ALPHA VALUE DOESN'T BEGIN WITH QUOTE	Beginning quotation mark missing for initial alphanumeric value. Put single quotation at the beginning of the initial value.
INITIAL VALUE TOO BIG	The initial value specified is larger than the field size. Either define a larger field size or reduce the size of the initial value.
INITIAL VALUE TOO SMALL	The initial value specified is smaller than the field size. Either redefine the field size or increase the size of the initial value.

Message	Explanation
MISSING CLOSE QUOTE ON ALPHA INITIAL VALUE	Quotation mark not specified at the end of an alphanumeric initial value. Add the missing close quotation marks.
MISSING INITIAL VALUE	Comma was inserted after type and size but initial value was not specified. Either delete comma or insert initial value.
NO COMMA AFTER FIELD NAME	No comma or a character other than comma was specified after the field name. Enter the missing comma or delete incorrect character and then enter the comma.
NO INPUT	Input file is null or not enough input files specified for a merge. Two files are needed to execute a merge. Use two nonempty files.
NOT A OR D	A character other than A or D occured in a data specification statement where A or D was expected. Correct the command file.
NOTHING AFTER FIELD NAME	Field type and size are not specified after field name and comma. Correct the command file.
NUMBER REPEATED OR OUT OF ORDER	A field sequence number is used more than once or is out of ascending order sequence. Correct the command file.
OUTPUT ERROR	Indicates an I/O error. Start SORT over. If it continues, check for media or hardware problem.
TOO MANY FILES	Merge only, more than 6 input files specified. Specify no more than six files for merge command file.
UNIT xx IS FREE	This is not an error. It is an informative message showing the logical units that are free. xx is a COS logical unit number.
UNRECOGNIZABLE LINE	Parameter line did not start with a good keyword. Correct the command file.

CHAPTER 10

FILE EXCHANGE PROGRAM (FILEX)

The File Exchange Program (FILEX) transfers files between diskettes in universal format and any COS-310 file storage device or any OS/8 file on an RK05 disk (OS/8 files cannot be transferred to or from diskettes).

Files are transferred in one of three formats: ASCII, IMAGE, or EBCDIC (universal format). The EBCDIC format is compatible with diskettes produced by an IBM 3741 except when using multivolume universal interchange files or when mapping bad sectors.

10.1 UNIVERSAL DISKETTE

A universal diskette contains 77 tracks (some of which cannot be used for data). Each track has 26 sectors numbered from 1 to 26. A sector contains one record of 128 characters or less. (See Figure 10-1 for a visual representation of a universal diskette.)

A record in a COS-310 file is assumed to be a string of characters preceded by a word count and independent of sector boundaries. A record on a universal diskette in EBCDIC format must begin on a sector boundary and only one record is allowed per sector. If the record does not fill a sector, the remainder of the sector is filled with blanks. Since these restrictions make EBCDIC format inefficient and wasteful of space, only use EBCDIC when you must read or write diskettes compatible with IBM 3741 format.

Track 0 of the universal diskette contains the information which describes the files on the diskette. Each of the 26 sectors on a diskette has a specific function.

Sector	Function
1-6	Reserved.
7	Identifies the diskette format. If bytes 0-3 contain VOL1 in EBCDIC characters, the diskette is assumed to have a universal interchange format directory. The remainder of sector 7 contains other information which FILEX does not use.
8-26	Contain the labels or the directory entries. These sectors contain information such as record length (up to 128 characters) and creation date. For further details on these sectors refer to the IBM manual, form number GA21-9128-0. Each byte in sectors 8-26 has a special function.

Bytes	Function
0-3	Are for label identification and contain HDR1 (DDR1 if the file has been deleted).
6-13	Contain the file name.
23-27	Specify the record length.
29-30	Contain two EBCDIC characters which identify the track number at the beginning of the data.
31	Must be EBCDIC 0 (360 octal).
32-33	Contain two EBCDIC characters which identify the sector number at the beginning of data.
35-36	Contain the number of the last track reserved for this file. Byte 37 must be EBCDIC 0.
38-39	Contain two EBCDIC characters which identify the number of the last sector reserved for this file.
48-53	Contain creation year, month, and day.

Bytes	Function
75-76	Contain the track number.
77	Must be EBCDIC 0.
78-79	Contain the number of the next unused sector.

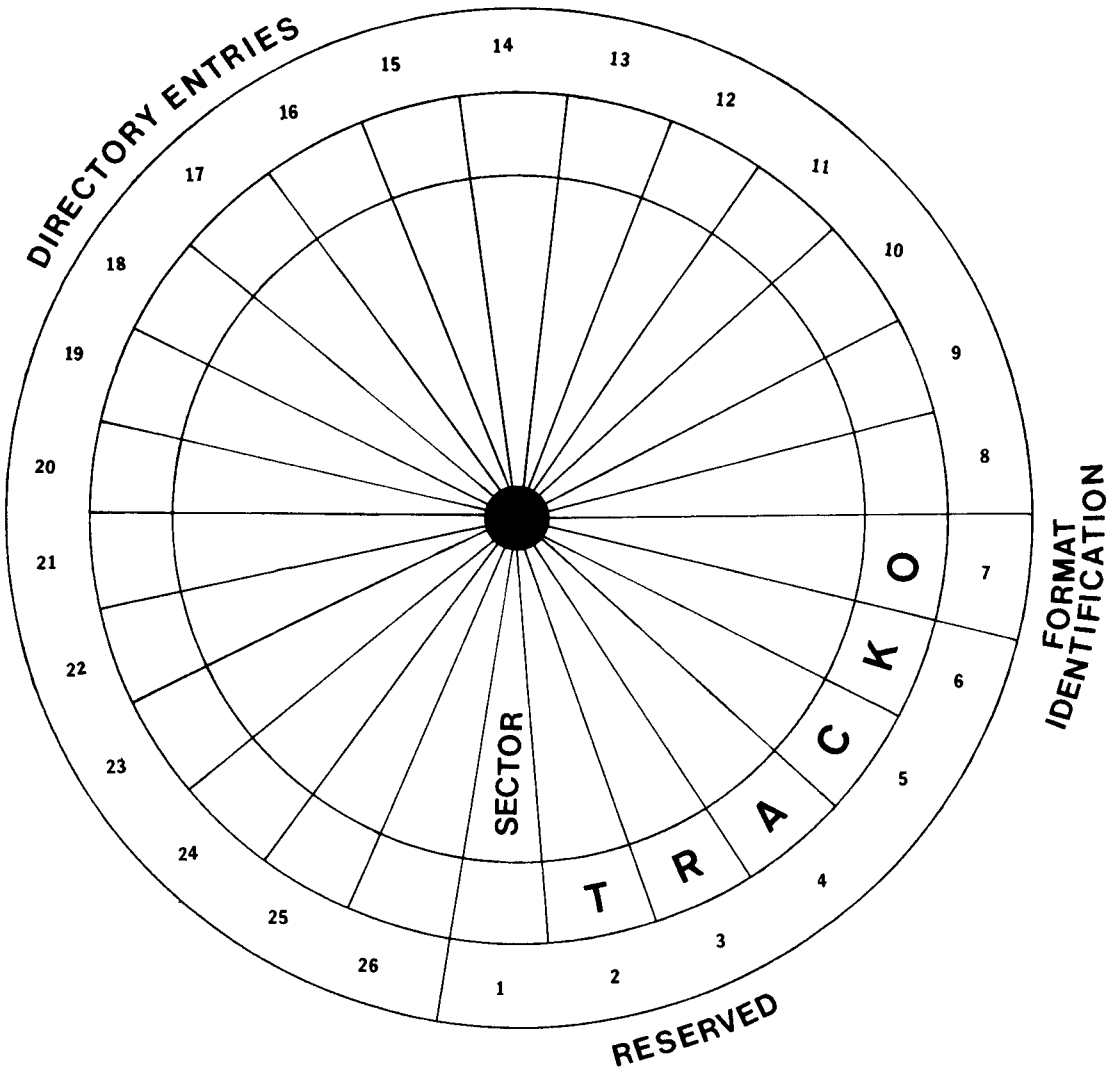


Figure 10-1 Universal Diskette

COS-310 character codes are never used in a universal interchange file. All data on a universal diskette is stored in either 7-bit ASCII or 8-bit EBCDIC. ASCII format is equivalent to data written as a continuous string of bytes ignoring sector boundaries; records are terminated by a carriage return/line feed. The first record of a file must begin on a sector boundary. All character code translation and record blocking is done implicitly by FILEX and need not be explicitly specified.

File name extensions are not normally recognized in universal interchange format; instead, a single eight-character file name is used. In order to provide some degree of compatibility with OS/8, FILEX has been designed to accept a six-character file name with a two-character extension. If a file name on a universal diskette has more than six characters, it must be entered in the format of filnam.ex. File names must not include spaces anywhere within the file name or between it and the extension.

10.2 FILEX OPERATING PROCEDURES

To execute FILEX, type:

```
RUN FILEX [,cmndfl]
```

where:

,cmndfl — is the name of a previously created file containing a table of desired logical unit assignments.

If the command file option is not used, FILEX uses the logical unit assignments already in the system.

If the command file is used, FILEX uses a special RX02 handler that reads and writes RX01 compatible diskettes and assigns logical unit numbers on RX01 and RX02 diskettes in the same system, provided the system is configured for RX02s. Assignments of this kind usually cannot exist on the same system.

These RX02 assignments remain in effect. When FILEX is completed, however, all logical units assigned to RX01s become undefined.

Following the RUN FILEX command, the program displays:

```
FILEX V8.00 (or current version number)
OPT (C, D, L, X, Z):
```

Enter one of the options; C, Copy; D, Delete; L, List; X, eXit; or Z, Zero (clear).

10.3 COPY (OPT:C)

OPT:C copies the contents of one file onto another. If you select option C, the system requests the input mode (the directory structure and the file format) of the file to be copied.

```
INPUT MODE (A, D, U):
```

Type the letter corresponding to the input mode to be used: OS/8 ASCII (A), COS-310 Data (D), or Universal (U).

10.3.1 OS/8 ASCII Input (Mode A)

ASCII format is that used by OS/8. If you select input mode A, the program displays:

```
FILE NAME:
```

Type the file name and the device designation in the following form:

```
filnam[.ex][,dev]
```

where:

filnam[.ex]
: is a six-character or less file name plus an optional two-character extension identifying the file to be input.

,dev
: is a three-character device designation. OS/8 RX01 and COS-310 RX01 diskettes are incompatible so do not specify an RXn device designation.

  If the device is not specified, the system device is assumed.

If the file name given already exists, FILEX displays:

```
REPLACE?
```

Type Y for YES, N or any other character for NO.

Having established the input file name, the program displays:

```
OUTPUT MODE (A, D, S, U):
```

Sections 10.3.4 through 10.3.4.4 explain the OUTPUT MODE.

10.3.2 COS-310 Data Input (Mode D)

If you select input mode D, the program displays:

```
FILE NAME:
```

Type the file name and the logical unit # in the following form:

```
filnam [/logical unit #]
```

where:

filnam is a six-character or less name identifying the file to be input.

[/logical unit #] identifies the logical unit where the file is found.

Having been given the input file name, the program displays:

```
OUTPUT MODE (A, D, S, U)
```

Sections 10.3.4 through 10.3.4.4 explain the OUTPUT MODE.

10.3.3 Universal Input (Mode U)

If you select input mode U, the program displays:

```
DISKETTE DATA MODE (A, I, U):
```

Type the letter corresponding to the diskette data mode to be used: A, ASCII; I, Image; U, Universal.

If you select any one of the diskette data modes A, I, or U, the program displays:

```
FILE NAME:
```

Type the input file name in the following form:

```
filnam[.ex][,RXn]
```

where:

filnam[.ex] is a six-character or less name plus an optional two-character extension. This name identifies the input file.

,RXn is a three-character device designation. Must be RX; n represents the drive on which it is mounted.

After you type the file name, FILEX displays:

```
OUTPUT MODE (A, D, S, U):
```

Sections 10.3.4 through 10.3.4.4 explain the OUTPUT MODE.

10.3.4 Output Modes (A, D, S, U)

The four output modes are: A, OS/8 ASCII; D, COS-310 data file; S, COS-310 source file; and U, Universal diskette.

10.3.4.1 OS/8 ASCII Output (Mode A)

If you select output mode A, the program displays:

```
FILE NAME:
```

Type the file name and the device designation in the following form:

```
filnam[.ex][,dev]
```

where:

filnam[.ex]
is a six-character or less file name plus an optional two-character extension. This file name identifies the file where output is to go.

,dev is a three-character designation of the device where the output is to go. OS/8 RX01 and COS-310 RX01 diskettes are incompatible so do not specify an RXn device designation.

If the device is not specified, the system device is assumed.

Type the file name; FILEX executes the transfer and returns to:

```
OPT (C, D, L, X, Z):
```

OS/8 files are always multiples of 16 blocks long. For this and other reasons, the resulting OS/8 output files may be longer than necessary. Use the OS/8 PIP program /A to recover the unnecessary space.

10.3.4.2 COS-310 Data File Output (Mode D)

If you select output mode D, the program displays:

```
FILE NAME:
```

Type the file name and logical unit # in the following form:

```
filnam[/logical unit #]
```

where:

filnam — is a six-character or less name identifying the file where output is to go.

/logical unit # — identifies the logical unit where the output file is found.

Type the file name; FILEX executes the transfer and returns to:

```
OPT (C, D, L, X, Z)):
```

10.3.4.3 COS-310 Source File Output (Mode S)

If you select output mode S, the program displays:

```
FILE NAME:
```

Type the file name in the following form:

```
filnam
```

where:

filnam — is a six-character or less name to be assigned to the COS-310 output file.

The output file is generated 16 blocks long.

To correct the directory entry to reflect the actual length of the file, do a FETCH and a WRITE as follows:

```
FE filnam       ;Fetch the file you have just created.

WR filnam/Y     ;The WRITE command enters the correct file
                ;length into the directory.  The /Y switch
                ;bypasses the REPLACE?  message response when
                ;a duplicate file name is encountered.
```

Type the file name; FILEX executes the transfer and returns to:

```
OPT (C, D, L, X, Z):
```

10.3.4.4 Universal Diskette Output (Mode U)

If you select output mode U, the program displays:

```
DISKETTE DATA MODE (A, I, U):
```

The three diskette data modes are: A, ASCII; I, Image; U, Universal.

Select diskette data mode A (ASCII), I (Image), or U (Universal), and the program displays:

```
FILE NAME:
```

Type the file name in the following form:

```
filnam[.ex][RXn]
```

where:

filnam[.ex]
: is a six-character or less name plus an optional two-character extension to be assigned to the output file.

,RXn
: is a three-character device designation. Must be RX; n represents the drive on which it is mounted.

If you selected diskette data mode A or I, FILEX performs the transfer and returns to:

```
OPT (C, D, L, X, Z):
```

In diskette data mode A or I, sector boundaries are ignored. An ASCII transfer (A) deletes nulls and rubouts, removes parity, and terminates each record with a RETURN. An Image transfer (I) reads and writes each byte exactly. The net effect of an Image transfer is similar to, and, in most cases, indistinguishable from an ASCII transfer.

If you selected diskette data mode U, the program displays:

```
OUTPUT RECORD SIZE (DEFAULT=80):
```

Type a number (1-128) representing the size of the output record. If you respond with RETURN, the record size defaults to 80. In this universal diskette data mode, one sector is equal to one record which is equal to one line.

Type the output record size; FILEX performs the transfer and returns to:

```
OPT (C, D, L, X, Z):
```

Figure 10-2 is a visual representation of the execution of the C option in FILEX.

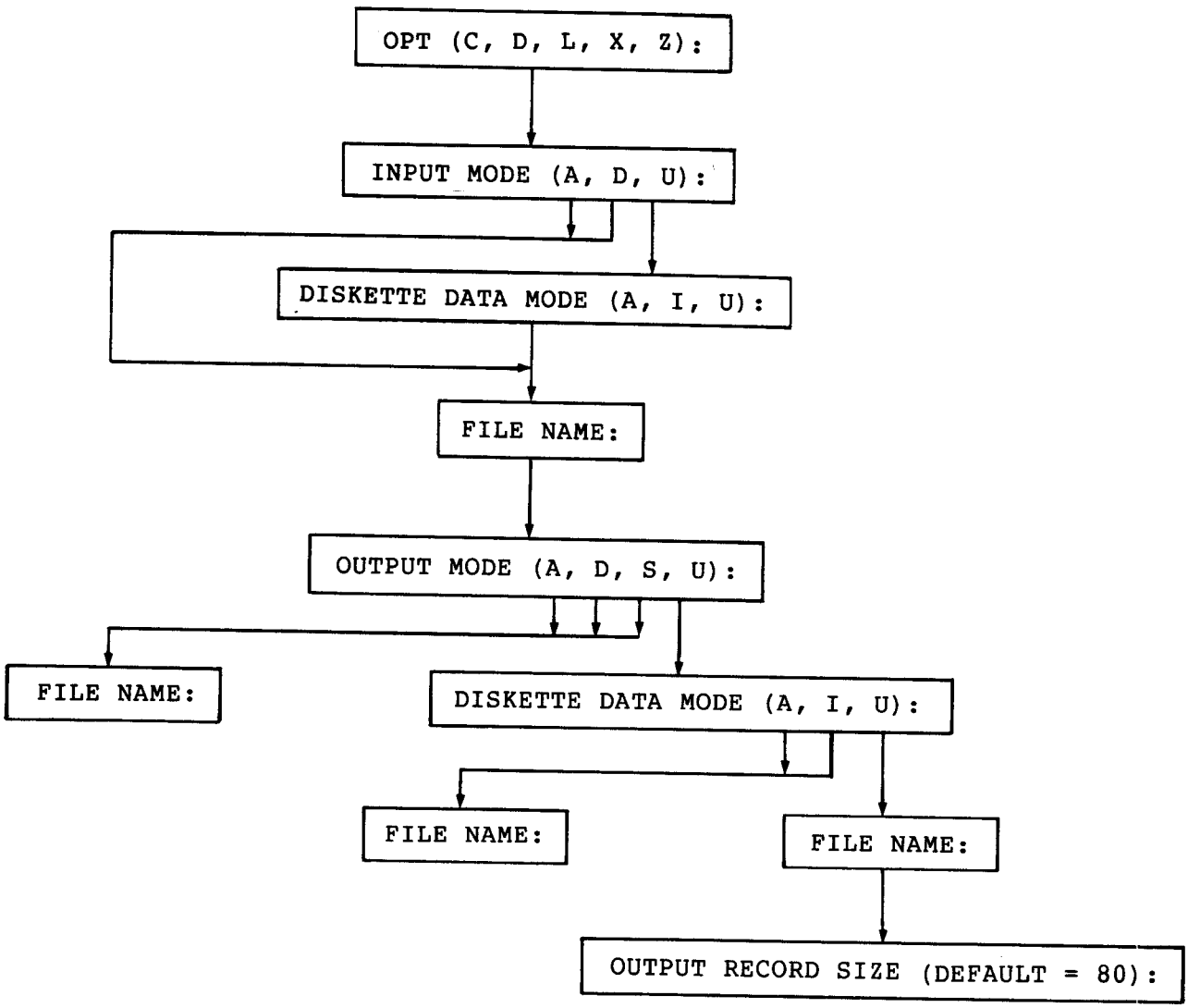


Figure 10-2 Flowchart of FILEX OPT:C

10.4 DELETE (OPT:D)

OPT:D deletes a single file from the universal diskette directory. If you select option D, the program displays:

```
FILE NAME:
```

Type the file name in the following form:

```
filnam[.ex],RXn
```

where:

filnam[.ex]
: is a six-character or less file name plus an optional two-character extension which identifies the file to be deleted.

,RXn
: is the three-character designation of the device on which the file is found. Must be RX; n identifies the drive on which the device is mounted.

Type the file name; FILEX deletes the file and returns to:

```
OPT (C, D, L, X, Z):
```

10.5 LIST (OPT:L)

OPT:L displays a listing of all the files in the universal diskette directory. If you select option L, the program displays:

```
DISKETTE DRIVE NUMBER:
```

Type the number corresponding to the drive on which the diskette is mounted. After you type the number, FILEX displays a table similar to the following:

NAME	RESERVED	USED	DATE
IMAGE	73	73	03-JUN-76
ASCII	73	73	03-JUN-76
IBM	576	576	03-JUN-76
ASC2	73	73	03-JUN-76
IBM2	193	193	03-JUN-76
<EMPTY>	910	0	
<EMPTY>	0	0	
<EMPTY>	0	0	
<EMPTY>	0	0	
<EMPTY>	0	0	
<EMPTY>	0	0	
<EMPTY>	0	0	

```
<EMPTY>          0       0
<EMPTY>          0       0
<EMPTY>          0       0
<EMPTY>          0       0
<EMPTY>          0       0
<EMPTY>          0       0
<EMPTY>          0       0
```

where:

NAME — is either the file name or a designation for an empty area.

RESERVED — is either the number of sectors reserved for a file or the number of sectors available for additional files. There is space for a total of 19 files using a total of 1898 sectors.

USED — is the number of sectors actually used by the file.

DATE — is the creation date of the file.

FILEX then returns to:

```
OPT (C, D, L, X, Z):
```

10.6 EXIT (OPT:X)

OPT:X returns control to the COS-310 Monitor.

10.7 ZERO (OPT:Z)

OPT:Z zeros (clears) an entire universal diskette and makes it ready for new files. If you select option Z, the program displays:

```
DISKETTE DRIVE NUMBER:
```

Type the number corresponding to the drive on which the diskette to be zeroed is mounted. When zeroing is completed, FILEX returns to:

```
OPT (C, D, L, X, Z):
```

A zeroed universal diskette has one file name entry (DATA) in the directory. This reserves 1898 sectors (the entire diskette). Before any files can be transferred to this diskette, DATA must be deleted using OPT:D of FILEX.

10.8 FILEX ERROR MESSAGES

The most common error message is a return to the options. This is caused by inputting an answer, usually a file name, in the wrong format. Check the format of your answer and retry.

Message	Explanation
BAD DIRECTORY	Attempted to reference or store a file on a device without a directory or on a device where the directory has been destroyed. Only devices with directories can be used. If the directory is damaged, call your Software Specialist.
DEVICE ERROR	The system failed in an attempt to read from or write to a device. Retry the operation. Check for media problem (use PIP OPT- R), or check for hardware problem.
FILE ALREADY EXISTS	A file name already on the universal diskette was entered in response to the output FILE NAME: message. The system returns to OPT (C, D, L, X, Z): Start the option sequence again and use a unique file name.
FULL	COS-310 source output file exceeds 16 blocks; IBM output file too large for device. The system outputs as much as it can and then displays the error message. Use larger device, reduce the size of output, or determine if the loss is worth the change.
ILLEGAL DEVICE	The file name information contains a device that is not in agreement with the logical unit information. Stipulate new device information.
INSUFFICIENT SPACE ON DEVICE	Attempted to allocate more segments than are available on a device. Either allocate fewer segments, make more segments available, or use a larger device.
NO END	No end-of-file mark in the OS/8 input file. Correct the input file.

Message	Explanation
NO ROOM	No room is available for the file in the output device directory (OS/8, IBM). Delete to make space or use a device with directory space.
NOT ENOUGH ROOM FOR SYSTEM AND FILES	Designated a device that is too small to accomodate both the system program and the files. Use PIP OPT- E to put system program on one device and files on another.
NOT FOUND	The input file name was not found. The system displays FILE NAME: Check the directory for the name of a file on the system. Enter a name found in the directory.
NOT UNIVERSAL DISKETTE?	Requested a device that does not contain universal floppy. Request a device with universal floppy.
# TOO LARGE	The number of segments in a logical unit exceeded 4095.
SYNTAX ERROR	The file containing the logical unit assignments is formatted incorrectly. Reformat the file.
TOO BIG	The record is too large. It exceeds 120 characters for source file output or 510 characters for data file output. Reduce the size of the record.

CHAPTER 11

PATCH PROGRAM (PATCH)

PATCH is used to fix (patch) either a system program or the Monitor on a COS-310 system. All input information for the PATCH operation is distributed as official patches from Digital Equipment Corporation. The patch information is a line-by-line dialogue. No other information should be used.

System programs (files) and the Monitor consist of blocks of numerical information coded into machine language instructions and stored on the system device. These machine language instructions are numbered from 0 to 377 octal. PATCH reads one of these blocks, allows you to examine and/or change individual words within the block, and writes the block back out to the system device.

11.1 PATCH OPERATING PROCEDURES

To execute PATCH, type:

```
RUN PATCH[,cmndfl][/C]
```

where:

,cmndfl is a previously stored file of PATCH commands. Each command is on a separate line; there can be no blank lines or comments. When the command file option is specified, PATCH reads a line from the command file each time one of the following prompts is displayed:

```
FILE NAME:
BLOCK:
LOCATION:
NEW VALUE:
RELATIVE CHECKSUM:
```

After the last line of the command file has been used or an error is encountered, all responses must come from the keyboard.

/C changes the blocks on the system device. Without /C, PATCH simulates the patching operation but does not change the file on the system device. When run without the /C option, PATCH displays:

CHECKSUM CORRECT--USE OPTION C TO UPDATE

After the RUN PATCH command, the program displays:

```
PATCH V8.00 (or current version number)
FILE NAME:
```

Respond with the following information as provided by DIGITAL:

- The name of the file to be patched.
- /N to indicate a patch for the Monitor. The system responds PATCHING MONITOR.
- /X to indicate the end of the PATCH operation. The message EXIT is printed and control returns to the Monitor.

If you enter a file name or /N, PATCH displays:

```
BLOCK:
```

Answer with either the number corresponding to a block within a file, or END to indicate that no more blocks are to be patched.

If the block number is typed, the program displays:

```
LOCATION:
```

Respond with either the number corresponding to a location to be patched, or END to indicate that no more locations are to be patched.

If a location number is typed, the program displays:

```
OLD VALUE: nnnn
```

where:

nnnn is the old (current) value at the location.

This display requires no input and is followed by:

```
NEW VALUE:
```

Enter the new value as indicated in the information supplied by DIGITAL. The program displays:

```
LOCATION:
```

Answer with either the number corresponding to a location to be patched, or END to indicate that no more locations are to be patched.

If END is typed, the program displays:

```
RELATIVE CHECKSUM:
```

Enter the checksum from the information supplied by DIGITAL. If this checksum is correct, the program displays either:

```
NEW BLOCK PATCHED OK
        or
CHECKSUM CORRECT--USE OPTION C TO UPDATE
```

The patch has been accurately entered. Use option C to update the program. Once the program is updated it cannot be changed except by another PATCH routine. Following the OK statement, the program asks for another block number where patching is to be done:

```
BLOCK:
```

If further patching information is available, enter it. If no more patching information is provided, type END. Following END the program displays the number (nn) of blocks patched within the file:

```
nn BLOCK(S) PATCHED IN THIS FILE
```

With this statement the program requests the name of another file to be patched.

```
FILE NAME:
```

Enter the file name as supplied by DIGITAL. If patching is complete, type /X.

Following /X, the program displays:

```
EXIT

COS MONITOR V 8.00 (or current version number)
```

11.2 ERROR CORRECTION

Much of the seriousness of errors while patching can be eliminated with the use of the command file option.

11.2.1 CTRL/U or R (Restart)

If at any time prior to the end of the checksum statement an error is discovered, type R (for Restart) and PATCH will return to the FILE NAME question. During the PATCH operation, the DELETE key is inoperable. If you make an error on a line, type CTRL/U and the correct information.

11.2.2 Wrong Old Value

The old value displayed by the program must be the same as the old value supplied in the PATCH information. If it is not, go through the following procedure.

Step 1 Be sure that everything previously typed is letter perfect. If the wrong BLOCK number was typed, type R and restart at FILE NAME. If the wrong LOCATION was typed, type RETURN in answer to NEW VALUE. This makes no change to the location specified. Type the correct location number in answer to LOCATION.

Step 2 If everything typed was correct, check the version number of the Monitor or the system program in question.

Step 3 If everything seems in order but the dialogue doesn't agree, save all output and consult your Software Specialist.

11.2.3 Bad Checksum

If an error in the checksum is detected, the following message is displayed:

```
BAD CHECKSUM
LOCATION:
```

The faulty block is not written to the system device.

The newly changed block is still in memory. Review the numbers and the locations to see if they are correct. If the error is found, fix it and then type END to the LOCATION: message. If an error is not found, type R to restart the program and patch the entire block again.

11.3 PATCH ERROR MESSAGES

Most error messages result from incorrect entries. Check each entry for accuracy. All entries must be exactly as supplied by DIGITAL.

Message	Explanation
BAD CHECKSUM	An attempt was made to write a block which was incorrectly patched. Type R and restart the program.
BAD DIRECTORY	Attempted to reference or store a file on a device with a damaged or nonexistent directory. Only files with directories can be used. If the directory is damaged, call your Software Specialist.
BAD NUMBER	A number with either more than 4 digits, a nondigit, or the digits 8 or 9 was typed. Enter number correctly.
BLOCK TOO BIG	An incorrect block number was typed. It cannot be larger than the length of the file being patched. Enter the correct block number.
FILE NOT FOUND	The file was not found on the system device. Check the directory for the file name. If the file name is not found, check for correct version number.
LOCATION TOO BIG	A location greater than 377 was typed. Retype location number.
NO CHANGE IN BLOCK	An attempt was made to write a block with no changes in it. Make proper changes using patch information.

CHAPTER 12

BOOT PROGRAM (BOOT)

BOOT is used to bootstrap the system from one device to another, i.e., if the system has been moved from one type of device to another, boot is run to start the system on the new device.

12.1 BOOT OPERATING PROCEDURES

To execute BOOT, type:

```
RUN BOOT/xx
```

where:

/xx is the two-character designation for the device which you want to get into operation.

/DK is the RK05 disk unit 0.

/RX is the RX01 diskette unit 0.

/DY is the RX02 diskette unit 0.

An attempt to bootstrap a device which is not ready or does not exist will produce unpredictable results.

12.2 BOOT ERROR MESSAGES

Message	Explanation
NO	No device or an illegal device designation was specified. Control returns to the Monitor. Specify a legal device designation.

CHAPTER 13

LINE CHANGE PROGRAM (LINCHG)

The Line Change Program (LINCHG) is a utility program which temporarily changes the lines-per-page configuration of printed programs without affecting the SYSGEN lines-per-page default value of 66. Its use is limited to printers without forms hardware.

13.1 LINCHG OPERATING PROCEDURES

To execute LINCHG, type:

```
RUN LINCHG[/n]
```

where:

/n is the number of lines you want on a page.

If you do not use the /n option, the program displays:

```
HOW MANY LINES PER PAGE?
```

Type the number (1-99) of lines you want. LINCHG installs this specified number as the number of lines-per-page. The system defaults to 66 lines-per-page.

The LINCHG number will remain in effect unless a further call to LINCHG is made, the system is rebooted, or the system is closed down.

Example:

Following is a batch program with a SYSGEN default of 66 lines-per-page. Line change commands change the lines-per-page of various programs.

```
.RUN JOB1       ;66 lines-per-page.
.RUN LINCHG/10
.RUN JOB2       ;10 lines-per-page.
.RUN LINCHG/33
.RUN JOB3       ;33 lines-per-page.
.RUN LINCHG/99
.RUN JOB4       ;99 lines-per-page.
.RUN LINCHG
HOW MANY LINES PER PAGE?
50
.RUN JOB5       ;50 lines-per-page.
.RUN LINCHG/66
```

The above example shows how a series of programs may be run starting with the normal default for the first program, then incorporating a variety of changes for the programs which follow, and finishing with the default number. The 66 lines-per-page can also be reestablished by rebooting the system.

13.2 LINCHG ERROR MESSAGES

Message	**Explanation**
INVALID OPERATION	Attempted to change lines-per-page on a printer with forms hardware. Such a change cannot be made.

CHAPTER 14

FORMAT PROGRAMS (DKFMT, DYFMT)

Before an RK05 disk or an RX02 diskette can be used on COS-310, it must be initialized. Initialization consists of formatting the disks. Do not initialize a disk or diskette containing any important information such as the Monitor or other such files. Initialization destroys the data on the disk.

Formatting the RK05 and RX02 means writing the necessary timing and sense marks onto the disk or diskette and erasing any other information.

An RX01 diskette can be formatted to become an RX02 diskette. This procedure cannot be reversed. The RX01 diskette does not need to be formatted to be used on an RX01 drive.

14.1 FORMATTING RK05 DISKS

To format an RK05 disk, type:

```
RUN DKFMT
```

The program displays:

```
DKMFT  V 8.00
DRIVE?
```

Respond with the number (0-3) of the drive where the disk is mounted. After you type this number the following message is displayed:

```
ARE YOU SURE?
```

Any response other than Y (Yes) brings back the DRIVE? question. A Y (Yes) response causes the program to display:

```
WRITE PASS
READ PASS
```

These two phrases indicate that the program is in operation; they require no response from the keyboard. Some time (a matter of seconds) elapses after each phrase appears while the program completes that particular phase of operation.

When the formatting operation is complete, the program displays:

```
DRIVE?
```

This is a cue to begin formatting another disk. Time is allowed for the physical changing of disks. If the formatting is complete, type CTRL/C to return to the Monitor.

14.2 FORMATTING RX02 DISKETTES

To format an RX02 diskette, type:

```
RUN DYFMT
```

The program displays:

```
DYFMT  V 8.00
DRIVE?
```

Respond with the number (0-1) of the drive where the disk is mounted. The program displays:

```
ARE YOU SURE?
```

Any response other than Y (Yes) brings back the DRIVE question. A Y (Yes) response causes the program to display:

```
FORMATTING DRIVE n
```

where:

n is the number (0-1) of the drive previously indicated.

This statement remains on the screen until formatting is completed. When the formatting is completed, the program displays:

```
DRIVE?
```

This is a cue to begin formatting another diskette. Time is allowed for the physical changing of the diskettes. If the formatting is completed, type CTRL/C to return to the Monitor.

CHAPTER 15

DUMP AND FIX TECHNIQUE (DAFT)

The Dump and Fix Technique (DAFT) program is similar in function to an editor, but it is used for data records. DAFT allows you to search for, examine, and change records, and to list records or parts of records on the printer or on the screen.

DAFT allows one input and one output file to be open at the same time. These two files can be the same file when in UPDATE mode. Memory always contains a record from the input file known as the current record. The current record can be modified by the CHANGE command before being written on the output file. An output file is not needed if records from the input file are only being examined.

15.1 DAFT COMPILING PROCEDURE

Because the DAFT program is distributed as two DIBOL source files, these two source files must be compiled into one binary file before DAFT can be executed. To compile DAFT, type:

```
.RUN COMP,DAFTA,DAFTB
.SAVE DAFT
```

15.2 DAFT OPERATING PROCEDURES

To execute DAFT, type:

```
RUN DAFT[,cmndfl1...,cmndfl7]
```

where:

cmndfl1...,cmndfl7
: are previously stored files which contain DAFT commands to be used to dump or fix a data file. If the optional command files are not present, commands are entered via the keyboard. After the last command in the last file

is executed, additional commands can be entered through the keyboard. An asterisk (*) is displayed to indicate that the DAFT program is ready for a command.

15.3 DAFT COMMAND FILE

The command file is created with the COS editor and contains DAFT commands. The entries in the command file are ordered according to a sequence of needs within individual records. To create an effective command file you must know the contents of the record and the possible areas needing correction.

15.4 DAFT COMMANDS

The first word in a DAFT command is a keyword consisting of any number of nonblank characters, only the first of which is significant. Some commands involve both a keyword and arguments. These arguments are separated from each other and from the keyword by one or more spaces.

15.4.1 Symbols Used in DAFT Commands

n — represents an unsigned nonzero positive integer. If it is optional in a command and is omitted, n=1 is assumed.

<a,b> — represents a key field of character positions a through b inclusive. Both a and b are unsigned nonzero positive integers and b must not be smaller than a. If this is optional in a command and is omitted, the subscripted area specified in the KEY command is used.

\+ — indicates that before a record is read from the input file, the current record in memory (if there is one) is written on the output file. The + sign does not have a space before it unless it is the only argument.

data — represents a piece of data. Alphanumeric data has the form:

```
'characters...'
```

Numeric data has the form:

```
[-] digits...
```

Before being used in executing a command, data is adjusted to the same length as the key field <a,b>. If the data is smaller and alphanumeric, it is left-justified in the field and filled with spaces on the right. If data is smaller and numeric, it is right-justified and filled with zeros on the left. If data is larger and alphanumeric, excess characters on the right are ignored. If it is larger and numeric, excess characters on the left are ignored.

15.4.2 DAFT Command Summary

Commands are entered after DAFT displays an asterisk (*).

Command	Function
Advance [n][+]	Advances the input file n records.
Backspace [n]	Backspaces n records if the input file was opened with the UPDATE command.
Change [<a,b>]data	Replaces the data in the current key field of the record currently in memory with the data specified. If <a,b> is used, it temporarily overrides the key field specified in the key statement.
Display [n]	Sets the width of the line of the listing device (screen or printer) to n characters (maximum 130). If n is omitted, this command turns the grid on if it is off and turns it off if it is on.
Exit	Returns control to the COS-310 Monitor if no output file is open.
Fini [+]	Closes the output file. If + is specified, the current record and the remainder of the input file are first copied to the output file. To write data to a file opened for UPDATE, the + must be specified.
Goto n[+]	Makes record n the current record.
Help	Displays a summary of DAFT commands.

Command	Function
Input filnam[/logical unit #]	Opens the specified file for input. The first record is read and becomes the current record.
Key a,b	Sets the key to character positions a through b inclusive.
List [n][<a,b>][+]	Prints n consecutive records beginning with the current record. The subscript <a,b> represents the consecutive characters that are to be considered.
Output filnam[/logical unit #]	Opens the specified file for output.
Put [n]	Writes n copies of the record currently in memory onto the output file.
Query	Displays the names of the input and output files, the units where the files are located, the record currently in memory, and the version number of the DAFT program.
Rewind	Reopens the input file. The first record becomes the current record.
Search [<a,b>]data[+]	Searches the current record and then succeeding records for an occurrence of the specified data appearing in the key field.
Type [n][<a,b>][+]	Same as List except output is displayed on the screen.
Update filnam[/logical unit #]	Opens the specified file for updating. This command can only be specified for a file with fixed-length records since direct access I/O is used to move records.
Version	Displays the version number of DAFT.
Write [n]	Performs the same function as Advance [n][+]. The nth record after the current record becomes the new current record.
X	Outputs the record number and size of the current record on the output device (either screen or printer depending on whether the last record was output by a Type or List DAFT command). The printer is the initial output device.

15.5 DAFT OUTPUT

Records can be listed with a grid above them. The grid has two lines of numbers which show the character positions. The lower of the two lines represents the ones digits of the column counts. The upper line represents the tens digits. The tens digits are printed for the first and last column in the record (or part of the record) or whenever the tens digit increments. If there is a hundreds digit, it is printed in column 1 or whenever it increments.

Following is an example of a DAFT program in operation.

```
.R DAFT
*HELP
ADVANCE N+
BACKSPACE N
CHANGE <A,B> DATA
DISPLAY N
EXIT
FINI +
GOTO N+
HELP
INPUT LABEL/UNIT
KEY A,B
LIST N KEY+
OUTPUT LABEL/UNIT
PUT N
QUERY
REWIND
SEARCH <A,B> DATA +
TYPE N <A,B>+
UPDATE LABEL/UNIT
VERSION
WRITE N
X
*VERSION
DAFT VERSION 8.00
*INPUT MAILING/1
*DISPLAY 70
*T 1

RECORD 000001 OF FILE MAILNG,   RECORD LENGTH=140 CHARACTERS

 DIGITAL EQUIPMENT CORP.  D. F. PAVLOCK            12-3

      146 MAIN ST.             MAYNARD       MA017540S/8-1  0012345A11

*D
*T 2

RECORD 000001 OF FILE MAILNG,   RECORD LENGTH=140 CHARACTERS
```

```
 0        1         2         3         4         5         6         7
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
 DIGITAL EQUIPMENT CORP.  D. F. PAVLOCK            12-3

 7        8         9        10         1         2         3         4
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
      146 MAIN
 ST.              MAYNARD         MA017540S/8-1  0012345A11


RECORD 000002 OF FILE MAILNG,   RECORD LENGTH=140 CHARACTERS

 0        1         2         3         4         5         6         7

 12345678901234567890123456789012345678901234567890123456789012345678901234567890
 DIGITAL EQUIPMENT CORP.  K. RICHER                12-3

 7        8         9        10         1         2         3         4

 12345678901234567890123456789012345678901234567890123456789012345678901234567890
      146 MAIN ST.              MAYNARD
 MA01754COS 300 0001972T 3

*T 2<25,50>

RECORD 000002 OF FILE MAILNG,   RECORD LENGTH=140 CHARACTERS

 2    3         4         5
 56789012345678901234567890
  K. RICHER


RECORD 000003 OF FILE MAILNG,   RECORD LENGTH=140 CHARACTERS

 2    3         4         5
 56789012345678901234567890
  S. RABINOWITZ

*A 1
*KEY 1,50

*T 2

RECORD 000004 OF FILE MAILNG,   RECORD LENGTH=140 CHARACTERS

 0        1         2         3         4         5
 12345678901234567890123456789012345678901234567890
 DIGITAL                    R. LARY


RECORD 000005 OF FILE MAILNG,   RECORD LENGTH=140 CHARACTERS

 0        1         2         3         4         5
 12345678901234567890123456789012345678901234567890
 DEC                       S. G. WELCOME
```

```
*Q
INPUT FILE: MAILNG    OPEN
UNIT: 01
OUTPUT FILE:  /NONE/
UNIT: 00
KEY=<001,050>

RECORD 000005 OF FILE MAILNG,   RECORD LENGTH=140 CHARACTERS

DAFT VERSION 8.00
```

15.6 DAFT ERROR MESSAGES

Message	Explanation
BAD DIGIT IN DATA	In a Change or Search command, a character other than digits or a minus sign is contained in a numeric data field. Remove bad characters.
CANT BACKSPACE PAST BEGIN OF FILE	Attempted to backspace past the beginning of file. The first record in the file becomes the current record.
CANT BACKSPACE WITH SEQUENTIAL INPUT	Attempted to backspace with sequential input. Backspace only possible when file is in update mode.
END OF INPUT FILE AT RECORD nnnn	Attempted to read past the end-of-file mark on the input file. This is not necessarily an error. nnnn was the last record read. The input file is closed. Reopen the file.
EXCESSIVE GRID SIZE	The grid (printer width) may not be greater than 130 characters. Reduce the grid size.
EXTRA CHARS	Extra characters were found after the end of a command. Remove extra characters.
ILLEGAL RECORD - CLOSING FILE	The file being updated contains a bad record (one not the same size as record 1). Only fixed-length records are permitted on such files. The file is closed. Reopen the file.

Message	Explanation
KEY ENTIRELY PAST END OF RECORD	The key specified in a List or Type DAFT command began with a character greater than the record size. Reduce the size of the key.
KEY EXTENDS PAST RECORD END	Attempted a change with a key that extends past the end of a record. However, a list with such a key is possible. In such a case, the list is terminated at the end of the record.
KEY TOO BIG	The key exceeded 100 characters. Reduce the size of key.
NO DATA	Data was not specified in a CHANGE or SEARCH command. Specify required data.
NO INPUT FILE	The command does not have an input file. Open an input file.
NO LABEL NAME	The file name was omitted in an INPUT, OUTPUT, or UPDATE command. Implement a name.
NO OUTPUT FILE	The command requires an output file but one is not open. The command is terminated at the point just prior to writing the current record on the output file. Open an output file.
OUTPUT FILE ALREADY OPEN	A request was made to open an output file while one was already open. Only one output file can be open at a time. The request is ignored. Close the current output file before opening a new file.
OUTPUT FILE STILL OPEN	An EXIT cannot be made when the output file is open. The output file can be closed with the FINI command or CTRL/C.
PUSHDOWN OVERFLOW	The program will abort with this message when too many errors are made. Restart DAFT.
0 NOT ALLOWED	The 0 (zero) is not a permissible argument. Don't use 0.

CHAPTER 16

REPORT PROGRAM GENERATOR (PRINT)

PRINT eases the creation of report programs. Using a command file which describes the report, PRINT generates a DIBOL program which produces the report.

PRINT is two programs chained together. The first program reads and validates the command file while creating memory table entries. If no command file errors are detected, the second program produces the DIBOL report program. The two programs which generate the report program require a total of 16K bytes of memory.

16.1 PRINT COMPILING PROCEDURE

PRINT is distributed as several source files. The files must be compiled into two programs before PRINT can be executed.

The PRINT source files contain the following information:

- PRINT1, PRINT2 are the data sections.
- PRINT3, PRINT4, PRINT5, PRINT6 are the procedure sections of the parsing (reading and validating) phase.
- PRINT7, PRINT8, PRINT9, PRINT0 are the procedure sections of the generation phase.

Use the following procedure to compile the distributed PRINT source files into binary programs.

```
.RUN COMP,PRINT1,PRINT2,PRINT3,PRINT4,PRINT5,PRINT6
.SAVE PRINTA
.RUN COMP,PRINT1,PRINT2,PRINT7,PRINT8,PRINT9,PRINT0
.SAVE PRINTB
```

16.2 PRINT OPERATING PROCEDURES

To execute PRINT, type:

```
RUN PRINTA+PRINTB,cmndfl[/xy]
```

where:

cmndfl is the name of a previously stored command file.

PRINTA+PRINTB
are the compiled PRINT DIBOL programs.

x is a switch which determines whether the command file is to be listed on the printer; N means no list, L means list.

y is a switch which determines whether the DIBOL program (data file) created by executing PRINT is to be listed on the printer; N means no list, L means list.

If x and y are both N, the switches and their preceding slash can be omitted.

The output from the RUN PRINT command is a data file which must be converted to a source file with the use of FILEX.

16.2.1 FILEX - Creation of Source File

Use the following FILEX command sequence to create a source file from the data file created by PRINT. (See Chapter 10 for FILEX information.)

```
.RUN FILEX
FILEX V 8.00
OPT (C, D, L, X, Z): C
INPUT MODE (A, D, U): D
FILE NAME: $RPG/logical unit #
OUTPUT MODE (A, D, S, U): S
FILE NAME:  pronam
OPT (C, D, L, X, Z): X
```

Pronam is any name desired, but it is usually the same name as is used in the IDENT line of the command file.

16.2.2 Compilation

To compile the DIBOL source program created by running the PRINT output through FILEX, type:

```
.RUN COMP,pronam
.SAVE pronam
```

16.2.3 Program Execution

To execute the compiled DIBOL program, type:

```
.RUN pronam
```

where:

pronam — is the name of the source program created by FILEX and compiled in Section 16.2.2

The execution of this program produces the report.

16.3 PRINT COMMAND FILE

The Print Command File has six sections:

IDENT	identifies the program and author
HEAD1,HEAD2	provides page headings for the report
INPUT	describes the input file
COMPUTE	describes any computation to be done
PRINT	describes the report column headings
END	is an optional directive at the end of the Print Command File

16.3.1 IDENT Section

The form of the IDENT section is:

```
IDENT pronam[/logical unit #][,author]
```

where:

pronam — is the name of the DIBOL source program to be generated.

/logical unit# — is the number referencing the storage location of the program.

,author any text from 1 to 24 characters in length.

Example:

```
IDENT TEST41/14, JOHN DOE ;Program named TEST41, on logical  unit
                          ;14, written by John Doe.
```

16.3.2 HEAD1 and HEAD2 Section

HEAD1 is the first heading line on each page of the report. HEAD2 is the second line. HEAD1 and HEAD2 are both optional. The only difference between HEAD1 and HEAD2 is that HEAD1 information will be expanded (if space permits) by inserting a space between each character. HEAD2 has no such expansion capability.

The form is:

```
[HEAD1 'text']

[HEAD2 'text']
```

where:

text is a string of up to 132 characters from the COS-310 character set, exclusive of single quotes.

There can be more than one HEAD1 or HEAD2 line. If this is the case, the individual texts are linked together.

Example:

```
HEAD1 'COMPUTATION AND SUMMARY RESULTS'

HEAD1 'FOR AUGUST, 1973'
```

16.3.3 INPUT Section

The INPUT section consists of the INPUT statement on one line followed by field description lines describing the fields of the input record.

The form of the input statement is:

```
INPUT [filnam[/logical unit #]][,S]
```

where:

filnam is the name of the input file.

/logical unit #
: is the logical unit on which the input file resides.

,S
: summarizes rather than describes the report.

If the file name is omitted, the generated program will request it when the report is run.

The form of each field description line is:

$$[\text{fldnam}],\left\{\begin{matrix}A\\D\end{matrix}\right\}n[.m][,Lr[P]]$$

where:

fldnam
: is the name of the field.

A
: is an alphanumeric field.

D
: is a numeric field.

n
: is the size of the field, expressed in characters (510 maximum for A field, 15 maximum for D field).

.m
: is the number of decimal places in the field and is valid only for numeric fields.

,Lr
: is used only for the break fields (a break field is used in conjunction with ACCUMULATE to print totals). The r is a single digit expressing the relative importance of the field; 1 indicates least important and 9 indicates most important. When totals are printed for a more important break, totals for all lesser breaks are also printed.

P
: starts a new report page after the totals for this break are printed.

The maximum number of fields is 20.

16.3.4 COMPUTE Section

This optional section starts with a line containing only one word:

```
COMPUTE
```

DIBOL statements appear on succeeding lines in the following form:

```
fldnam = expression
```

where:

fldnam is the name of a destination field and must not duplicate any input field name nor any previous computation result name.

expression
is any valid DIBOL expression. It may be a single alphanumeric field or literal, or any numeric expression.

Unlike DIBOL, PRINT uses decimal places. It will not allow addition and subtraction of expressions with different numbers of decimal places. The result of such an expression will have the same number of decimal places as the elements of the expression.

The number of decimal places in the result of a multiplication expression is the sum of the decimal places in the two expressions.

The number of decimal places in the result of a division expression is the difference in decimal places between the number being divided and the divisor.

To adjust the number of decimal places for any expression, multiply by the constant 1.00 with the number of decimal places equal to the longest number of decimal places in the other values. For example, expr 1 has 2 places, and expr 2 no places, then

```
expr 1 * expr 2 * 1.00
```

has four decimal places.

PRINT allows decimal places in numeric constants.

16.3.5 PRINT Section

The PRINT section begins with a line containing only one word:

```
PRINT
```

The next 12 lines describe the fields to be printed (not all 12 lines must be used). The form of these field descriptions is:

```
fldnam, 'text'[,A][,format]
```

where:

fldnam is the name of the field and must be either an input field or the result of a computation.

'text' is an alphanumeric string of COS-310 characters delimited by single quotes. This text is used as the heading of the report columns and on the total lines for break fields.

,A is present only if the field is to be accumulated and the sum is to be printed on total lines. The field must be numeric.

format is a string of text showing the format for the numeric fields. If the format is not included, PRINT will create one using the description of the field (if this is an accumulated field, two extra places will be assumed). That created format will use an appropriate number of decimal and integer places in the following form:

```
XX,XXX.XX-
```

The text used for titles may be several words separated by asterisks. This centers each word over a column in separate lines. For example:

```
GPAY,  'GROSS*PAY'
```

will cause

```
GROSS
PAY
```

to be placed over the GPAY column. If GPAY is also a break field, then:

```
GROSS PAY TOTAL
```

will be used on the total line instead of GROSS*PAY TOTAL.

The field descriptor lines may also be of the form:

```
,An
```

Where n is the size of the field. This will produce n blank columns in the report.

If any two field descriptors are not separated by a filler descriptor, then two blank columns will separate the fields in the report.

Example:

```
PRINT
NAME, 'EMPLOYEE NAME'
DEPT, 'DEPARTMENT'
GPAY, 'GROSS*PAY',A,XX,XXX.XX
```

16.3.6 END Section

The END section is optional and consists of a line containing only the word:

```
END
```

16.4 PRINT ERROR MESSAGES

Most errors in PRINT result from incorrect information in the command file. These are correctable with the editor and Monitor commands.

PRINT evaluates each statement in the command file for correctness. Whenever errors occur, the entire line where the error occured and a message are printed. The number printed near the error message indicates the character position where the error occurred. The following error messages are used by PRINT.

Message	Explanation
ALPHA LITERAL REQUIRED	Expected alphanumeric literal is missing. Insert where appropriate.
ALREADY DEFINED	Attempt to name a field in the INPUT or COMPUTE section with a name that was previously used. Correct the command file.
HEADER IS TOO LONG	The header line exceeds 132 characters. Correct the command file.
IMPROPER DEFINITION	Filler item in the PRINT section is used incorrectly. Correct the command file.
IMPROPER LITERAL	Literal too long. Shorten the literal.
IMPROPER USE OF DECIMAL PLACES	The number of decimal places exceeds the size of the field being defined. Reduce the number of decimal places.
INTEGER FROM 1-15 REQUIRED	The size of a numeric field specified in the INPUT section must be between 1-15. Correct the command file.

Message	Explanation
INTEGER FROM 1-132 REQUIRED	The expected numeric fields out of range. Reduce the field size to fewer than 132 characters.
INTEGER REQUIRED	Integer missing where expected. Check and insert as needed.
LITERAL TOO LONG	Field description exceeds 30 characters. Reduce to fewer than 30 characters.
MUST BE IDENT	The first section in the command file must be IDENT. Correct the command file.
MUST BE NUMERIC ITEM	An item expected to be numeric is defined incorrectly. Redefine.
MUST BE S	S is the only legal option in the INPUT statement that follows the comma. Insert S.
NEED FILE NAME	File name missing from IDENT statement. Correct the command file.
NO ENDING QUOTE	No closing quote for a HEAD1, HEAD2, or PRINT statement. Insert closing quote.
NO INPUT DIRECTIVE	The INPUT statement is missing. Correct the command file.
NO PRINT ITEMS	No fields are specified following the PRINT statement. Correct the command file.
NOT DEFINED	Attempted to print a field that has not been defined. Correct the command file.
NOT ENOUGH RIGHT PARENTHESES	A statement in the COMPUTE section has too few right parentheses. Correct the command file.
PICTURE TOO LONG	The picture or edit mask for printing exceeds 22 characters. Reduce to fewer than 22 characters.

Message	Explanation
SYNTAX ERROR	Statement contains illegal characters or options. Correct the command file.
TOO MANY COLUMNS IN REPORT	More than 132 columns under HEAD1, HEAD2, or PRINT sections. Correct the command file.
TOO MANY COMPUTE STATEMENTS	More than eight COMPUTE statements were specified. Correct the command file.
TOO MANY DATA ITEMS	More than 20 data items in the INPUT section. Correct the command file.
TOO MANY LEFT PARENTHESES	A statement in the COMPUTE section is too complicated to be deciphered by PRINT. Simplify the command file.
TOO MANY LIST ITEMS	More than 20 list items in the INPUT section. Correct the command file.
TOO MANY RIGHT PARENTHESES	A statement in the COMPUTE section has too many right parentheses. Correct the command file.
UNKNOWN DIRECTIVE	An invalid section statement. Only IDENT, HEAD1, HEAD2, INPUT, COMPUTE, PRINT, and END are legal. Check your statements for incorrect statements; correct the command file.

CHAPTER 17

FLOWCHART GENERATOR PROGRAM (FLOW)

The flowchart generator program (FLOW) produces a flowchart from a set of input commands.

The flowchart is always written to a file named $PASS1 located on logical unit 1. A printed flowchart can also be produced by using the appropriate option switches. The flowchart generator programs are distributed as source programs and must be compiled before use.

17.1 FLOW COMPILING PROCEDURE

FLOW consists of several DIBOL programs.

To compile the FLOW programs, type:

```
.RUN COMP,FLOW1,FLOW2,FLOW3,FLOW4
.SAVE FLOW
.RUN COMP,KREF
.SAVE KREF
```

17.2 FLOW OPERATING PROCEDURES

The commands to execute FLOW have the form:

```
RUN FLOW,cmndfl1...,cmndfl7[/xx]
RUN SORT,KRFSRT
RUN KREF
```

where:

cmndfl1...,cmndfl7
: are previously stored source files containing the FLOW commands to be used in generating a flowchart.

/xx
: is one of the following option switches:

/L lists the flowchart on the printer. If /L is omitted, the flowchart will only be placed in file $PASS1 on logical unit 1.

/P indicates that the input files are DIBOL programs with the FLOW commands imbedded in the program. When the /P option is used, only input lines beginning with a semicolon followed by any number of periods followed by a space or a tab are treated as FLOW commands. No semicolon - period - space configuration is needed if the FLOW commands are not part of a DIBOL program.

KRFSRT is a special sort command file that will sort the cross-reference scratch file. KRFSRT is distributed as part of the COS-310 software.

KREF is a cross-reference DIBOL program that works specifically with FLOW. This produces a cross-reference table containing an alphabetical listing of all labels used in the flowchart, the page number where each label is defined, and the page numbers where each label is used. KREF is distributed as part of the COS-310 software.

FLOW uses logical units 1, 2, 3, 4, and 5. Logical unit 1 must be large enough to contain the flowchart print image (usually 10 segments is sufficient). Logical units 2, 3, 4, and 5 must each be large enough to contain the KREF scratch file (usually 5 segments in each logical unit).

17.3 FLOW COMMANDS

Although some FLOW commands look like DIBOL statements, they are defined and used differently. FLOW commands have the following general format:

```
[;.. ][label][,] command
```

where:

;.. is the special indicator used with the /P option to distinguish between FLOW commands and DIBOL statements.

label is the FLOW statement label.

command is one of the following FLOW commands:

```
PROC [;][text]
DISK [;][text]
IF {YES} :label [;][text]
   {NO }
```

```
CALL label [;][text]
START [;][text]
STOP [;][text]
GOTO label
CGOTO label1,label2,...
I/O [;][text]
TITLE [;][text]
SBTTL [;][text]
PAGE
```

text is the information to be placed in the flowchart block. This text is usually centered within the blocks. Some commands require the text in a specific format.

Spaces and/or tabs may be inserted for legibility.

17.3.1 PROC Command

The PROC (process) command allows you to put up to 65 characters inside a process block. The following process block will be generated by the command line:

```
PROC   ;BUILD A TAB CHARACTER
```

```
***************
*             *
*             *
* BUILD A TAB *
*  CHARACTER  *
*             *
***************
```

17.3.2 DISK Command

The DISK command allows you to put up to 55 characters inside a disk block. The following disk block will be generated by the command line:

```
DISK   ;OPEN SYS FILE FOR INPUT
```

```
  *************
 *             *
* OPEN SYS     *
* FILE FOR     *
*   INPUT      *
 *             *
  *************
```

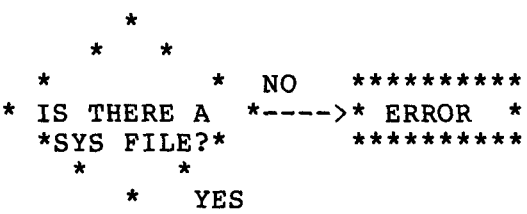
17.3.3 IF Command

The IF command allows you to put up to 37 characters inside a decision block. The IF command requires that the text field be preceded by the following field:

```
YES
     :label to branch to
NO
```

The following decision block will be generated by the command line:

```
IF NO:ERROR ;IS THERE A SYS FILE?
```



17.3.4 CALL Command

The CALL command allows you to put up to 33 characters inside a subroutine block. The CALL command requires that the text field be preceded by the subroutine name.

The following subroutine block will be generated by the command line:

```
CALL    HOF    ;OUTPUT PAGE MARKER
```

17.3.5 START Command

The START command allows you to put up to 13 characters inside a start block. The following start block will be generated by the command line:

```
START     ;HOF ROUTINE

          *************
         * HOF ROUTINE *
          *************
```

17.3.6 STOP Command

The STOP command allows you to put up to 13 characters inside a stop block. The following stop block will be generated by the command line:

```
STOP      ;RETURN

          ************
         *   RETURN   *
          ************
```

17.3.7 GOTO Command

The GOTO command allows you to put up to six characters inside a GOTO block. The following GOTO block will be generated by the command line:

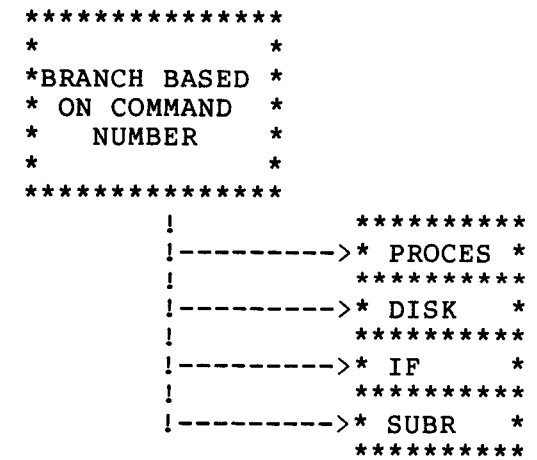
```
GOTO      NEXT

                    **********
          --------->* NEXT   *
                    **********
```

17.3.8 CGOTO Command

The CGOTO (computed GOTO) command allows you to flowchart multiway branches. The text field consists of labels separated by commas without imbedded spaces. The following blocks will be generated by the command lines:

```
PROC     ;BRANCH BASED ON COMMAND NUMBER
CGOTO     PROCES,DISK,IF,SUBR
```



17.3.9 I/O Command

The I/O command allows you to put up to 47 characters inside an I/O block. The following I/O block will be generated by the command line:

```
I/O      ;DISPLAY 'ERROR'
```

17.3.10 TITLE Command

The TITLE command allows you to specify up to 40 characters as a flowchart title. The title will appear at the top of all subsequent pages.

17.3.11 SBTTL Command

The SBTTL command allows you to specify up to 40 characters as a subtitle. The subtitle is printed on the line below the title. The SBTTL command implies a top-of-page command.

17.3.12 PAGE Command

The PAGE command advances the listing to the top of the next page. FLOW automatically generates new pages when necessary, making the PAGE command unnecessary in most instances.

17.4 FLOW EXAMPLE

The best example of the use of the flowchart generator is FLOW itself. FLOW commands have been inserted into the FLOW source files (FLOW1, FLOW2, FLOW3, FLOW4). To produce a flowchart of FLOW, use the following procedure:

- Assign the necessary logical units using DFU.
- Compile the flowchart programs.
- Enter the following commands:

```
.RUN FLOW,FLOW1,FLOW2,FLOW3,FLOW4/PL
.RUN SORT, KRFSRT
.RUN KREF
```

17.5 FLOW ERROR MESSAGES

Message	Explanation
NO INPUT	FLOW has no information to build a flowchart. Build a command file.
ERROR	An error has occurred. The line of text where the error occurred will be displayed on the next line. Check the line and correct the error.

CHAPTER 18

MENU PROGRAM (MENU)

The MENU program allows you to select and execute commands from a previously created command file. MENU permits more orderly execution of commands.

18.1 MENU OPERATING PROCEDURES

To run MENU, type:

```
RUN MENU,cmndfl
```

where:

cmndfl — is the name of the MENU command file stored on the system device. If none is specified, the file in the edit buffer is used.

MENU displays the text found in the Display Section of the command file and will accept a six-character operator response at the screen location specified in the Accept Section. The operator response is compared to the list of valid responses specified in the Command Section. If a match is found, the corresponding Monitor or editor commands are then executed by COS-310.

MENU can be included in a batch command file only as the last command in the file.

18.2 MENU COMMAND FILE

The MENU command file is created using COS-310 editor commands and is stored on the system device. It consists of three sections: Display, Command, and Accept. The order of these sections within the file is vital.

Example:

```
     DISPLAY
COPY COPY RX0 TO RX1
DIR  PRINT DIRECTORY OF RX0
     COMMAND
COPY =ER
     =1 C
     =2 RX0
     =3 RX1
     =4 X
     =WR $COPY/Y
     =PLEASE MOUNT DISKETTE ON DRIVE 1
     =R PIP,$COPY
     =DE $COPY/S
     =R MENU,cmndf1
DIR  =DI,RX0
     =R MENU,cmndf1
     ACCEPT (24,10)
```

18.2.1 Display Section

The Display Section has the form:

```
DISPLAY [/N]
text
.
.
.
```

where:

DISPLAY — is the first statement in the command file (must be DISPLAY).

/N — is an optional switch to suppress clearing the screen prior to displaying the text. Without /N the screen is cleared.

text — is text to display on the screen beginning on line one. Each line of text begins with a new line number. If a line of text contains more characters than can be displayed on a screen line, the extra characters are lost. Any line beginning with a semicolon is assumed to be a comment and is not displayed.

18.2.2 Command Section

The Command Section has the form:

```
COMMAND
code=command
.
.
.
```

where:

COMMAND is the first line in the section (must be COMMAND).

code is an operator response that contains a maximum of six characters. The Command Section may contain up to sixty codes.

command is a COS-310 Monitor or editor command that is executed when its corresponding code is entered as the operator response. There can be no spaces or tabs between the equal sign and the command.

A series of commands may be executed by listing the commands on subsequent lines with no code to the left of the equal sign. The series of commands is combined to produce a batch command file. This batch command file cannot be longer than one block.

There may be as many as 3995 characters in the Command Section.

18.2.3 Accept Section

The Accept Section has the form:

```
ACCEPT (y,x) [/N]
```

where:

ACCEPT is the first line in the section (must be ACCEPT).

y is a decimal number (cannot be an expression) designating the screen line number where the operator response is to be entered. If y is greater than the number of lines on the screen the results are unpredictable.

x — is a decimal number (cannot be an expression) representing the screen column number where the operator response is to be entered. If x plus the operator response (maximum of six) is greater than the screen width, the results are unpredictable.

/N — is an optional switch to suppress clearing the screen prior to executing the selected command from the Command Section.

The location (y,x) may fall within the text displayed by the Display Section.

18.3 MENU ERROR MESSAGES

Message	Explanation
ACCEPT SECTION NOT FOUND	No Accept Section in the command file. Correct command file.
COMMAND SECTION NOT FOUND	No Command Section in the command file. Correct command file.
DISPLAY SECTION NOT FOUND	No Display Section in the command file. Correct command file.
ILLEGAL CURSOR POSITION	An illegal cursor position (or none) was requested in the Accept Section. Correct command file.
ILLEGAL STATEMENT	Command file contains a meaningless statement. Correct command file.
TOO MANY COMMANDS	The Command Section is too large. Reduce size of Command Section.
TOO MANY COMMANDS FOR 1 CODE	The series of commands under one code exceeds 1 block in length. Correct command file.