FOCAL XVM
LANGUAGE MANUAL

DEC-XV-LFLGA-A-D

XVM Systems

digital

FOCAL XVM
LANGUAGE MANUAL

DEC-XV-LFLGA-A-D

digital equipment corporation · maynard. massachusetts

First Printing, January 1976

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM		TYPESET-11

CONTENTS

CONTENTS (Cont'd)

LIST OF ALL XVM MANUALS

The following is a list of all XVM manuals and their DEC numbers, including the latest version available. Within this manual, other XVM manuals are referenced by title only. Refer to this list for the DEC numbers of these referenced manuals.

BOSS XVM USER'S MANUAL	DEC-XV-OBUAA-A-D
CHAIN XVM/EXECUTE XVM UTILITY MANUAL	DEC-XV-UCHNA-A-D
DDT XVM UTILITY MANUAL	DEC-XV-UDDTA-A-D
EDIT/EDITVP/EDITVT XVM UTILITY MANUAL	DEC-XV-UETUA-A-D
8TRAN XVM UTILITY MANUAL	DEC-XV-UTRNA-A-D
FOCAL XVM LANGUAGE MANUAL	DEC-XV-LFLGA-A-D
FORTRAN IV XVM LANGUAGE MANUAL	DEC-XV-LF4MA-A-D
FORTRAN IV XVM OPERATING ENVIRONMENT MANUAL	DEC-XV-LF4EA-A-D
LINKING LOADER XVM UTILITY MANUAL	DEC-XV-ULLUA-A-D
MAC11 XVM ASSEMBLER LANGUAGE MANUAL	DEC-XV-LMLAA-A-D
MACRO XVM ASSEMBLER LANGUAGE MANUAL	DEC-XV-LMALA-A-D
MTDUMP XVM UTILITY MANUAL	DEC-XV-UMTUA-A-D
PATCH XVM UTILITY MANUAL	DEC-XV-UPUMA-A-D
PIP XVM UTILITY MANUAL	DEC-XV-UPPUA-A-D
SGEN XVM UTILITY MANUAL	DEC-XV-USUTA-A-D
SRCCOM XVM UTILITY MANUAL	DEC-XV-USRCA-A-D
UPDATE XVM UTILITY MANUAL	DEC-XV-UUPDA-A-D
VP15A XVM GRAPHICS SOFTWARE MANUAL	DEC-XV-GVPAA-A-D
VT15 XVM GRAPHICS SOFTWARE MANUAL	DEC-XV-GVTAA-A-D
XVM/DOS KEYBOARD COMMAND GUIDE	DEC-XV-ODKBA-A-D
XVM/DOS READERS GUIDE AND MASTER INDEX	DEC-XV-ODGIA-A-D
XVM/DOS SYSTEM MANUAL	DEC-XV-ODSAA-A-D
XVM/DOS USERS MANUAL	DEC-XV-ODMAA-A-D
XVM/DOS V1A SYSTEM INSTALLATION GUIDE	DEC-XV-ODSIA-A-D
XVM/RSX SYSTEM MANUAL	DEC-XV-IRSMA-A-D
XVM UNICHANNEL SOFTWARE MANUAL	DEC-XV-XUSMA-A-D

PREFACE

FOCAL (Formula CALculator) XVM (FOCAL) is an interactive utility program designed to solve numerical problems of any complexity. FOCAL is a component of the XVM/DOS software system.

This manual is designed to allow the reader to master and apply the FOCAL language within hours.

Chapters 1 through 3 of this manual describe the structure and use of the FOCAL language (particularly in the formulation and solution of numeric problems).

Chapter 4 contains demonstration programs which illustrate the many features and applications of FOCAL. The reader, by running these programs using different variables, can more fully realize the power and flexibility of FOCAL.

Chapters 5 and 6 describe advanced user-library storage and retrieval functions and user-defined FOCAL functions. The FOCAL library functions permit the storage of lengthy programs by the use of "chaining." User defined functions enable frequently used operations to be called (requested and performed) by a single command.

Chapter 7 describes FOCAL data functions which permit the user to store and then retrieve data on auxiliary Input/Output devices other than the console terminal.

CHAPTER 1
INTRODUCTION TO FOCAL

FOCAL is an on-line, interpretive service program designed to assist scientists, engineers, and students in solving complex numerical problems. The language consists of concise, imperative statements; mathematical expressions are typed in standard notation.

FOCAL puts the full calculating power and speed of the computer at the user's fingertips. With FOCAL, the user can easily generate mathematical models, plot curves, solve sets of simultaneous equations in n-dimensional arrays, and much more. Examples of various problems that FOCAL is capable of solving are described in Chapter 4.

1.1 HARDWARE REQUIREMENTS

FOCAL can be run on any XVM or PDP-15 computer which runs XVM/DOS software.

1.2 LOADING PROCEDURE

FOCAL may be loaded with the Linking Loader after XVM/DOS has been loaded.

After the bootstrap is loaded XVM/DOS types

```
XVM/DOS Vnxnnn
$
```

at the left margin of the console terminal.

The Linking Loader requires assignment of .DAT (Device Assignment) Table) slots -1 and -4. FOCAL requires assignment of .DAT slots +3, +7 (input) and +5, +10 (output). (.DAT slot assignments for FOCAL are summarized in Appendix E.) An example of the required ASSIGN command is:

```
$A DPO -1,-4, <SCR> 3,5,7,10↵
```

On the device assigned to .DAT -1, the Linking Loader, expects to find the System Library.[1] On the device assigned to .DAT slot -4,the Linking Loader expects to find the relocatable binary program, FOCAL BIN and its external function file, FNEW BIN.

[1] The user should be careful to use the non-floating point FORTRAN library. If FOCAL is to be used on a system with floating point hardware, the user should rename the non-floating point library to .LIBR5 BIN and assign .DAT -5 to the UIC containing .LIBR5 BIN.

FOCAL uses .DAT slot +3 for the library input function and .DAT slot +5 for the library output function (see Chapter 5 for FOCAL library commands). FOCAL uses .DAT slot +7 for DATA input function and .DAT slot +10 for DATA output function. (Refer to Chapter 7 for detailed information about the DATA commands.)

An example for loading FOCAL follows.

After the .DAT slots are assigned as above, XVM/DOS types another $. Now type

```
$GLOAD
```

and depress the RETURN key. The Loader types

```
LOADER XVM  Vnxnnn or BLOADER XVM  Vnxnnn
```

Now type a P, a back arrow (P←), FOCAL, a comma and FNEW after the Loader's >,

```
>P ←FOCAL, FNEW
```

and depress the ALT MODE key.

Teleprinter output format is as follows:

```
XVM/DOS V1A000
$A RK <SCR> -4,-1,3,5,7,10

$GLOAD

BLOADER XVM V1A000
>P_FOCAL,FNEW
P FOCAL   028 071742
P FNEW    004 071527
P .BH     005 071473
P DSQRT   007 071402
P DSIN    001 071367
P DCOS    002 071346
P DATAN   001 071333
P DEXP    001 071320
P DLOG    004 071272
P .DD     006 071124
P .DB     004 071004
P .DE     003 070703
P .DF     001 070544
P .DC     001 070475
P .DA     012 070420
P .CAT1   000 070353
P DOUBLE  004 070150
P RELEAE  10P 067051
P .CB     004 067027

BFOCAL XVM V1A000
```

1.3 RESTART PROCEDURE

Restart is accomplished by the use of CTRL P (echoes ↑P). However this may not work if it is typed while output is underway to the teleprinter.

1.4 SAVING FOCAL PROGRAMS (Refer to Chapter 5 for full description)

To save the current FOCAL program, type the following sequence of commands; where necessary, wait for FOCAL to type an * on the next line.

```
*
*LIBRARY OUT NAME
*LIBRARY WRITE *ERASE ALL
*LIBRARY WRITE ALL
*LIBRARY WRITE "*
*LIBRARY CLOSE
*
```

This sequence does not destroy the current program. Execute an ERASE ALL before starting the program to clear all variables and prevent placing previous programs in the library along with the current program during current library storage (refer to Sections 3.5 and 5.4.2). When a program is to be saved, Loader assignment must be to the proper output device. The assignment described in Section 1.2 will output the program on DECtape.

To load a saved FOCAL program, type:

```
*LIBRARY IN NAME
```

1.5 DATA INPUT/OUTPUT (Refer to Chapter 7 for full description)

To use auxiliary I/O devices for data storage and retrieval, type the following sequence of commands:

```
*DATA OUT NAME
*DATA CLOSE
*DATA IN NAME
```

This sequence will initialize and enter the named file for the data on a mass storage device, close the named file on that device, and then initialize a device under the given filename for data retrieval.

CHAPTER 2
FOCAL LANGUAGE

After FOCAL has been loaded, the program types out

```
FOCAL XVM Vnxnnn
*
```

to indicate that it is ready to accept commands from the user. Each time the user terminates a typed line by depressing the RETURN key, or after FOCAL has performed a command, an * (asterisk) is typed to tell the user that FOCAL is ready for another command.

2.1 ELEMENTARY COMMANDS

One of the most useful commands in the FOCAL language is TYPE[1] which instructs FOCAL to "type out the result of the following expression." Then, the user types an expression after TYPE (following the asterisk which FOCAL typed) such as

```
*TYPE 123.456+9.8765
```

and presses the RETURN key; FOCAL types the answer.

```
133.3325*
```

SET is another useful command, which instructs FOCAL to "store this symbol and its numerical value; then when this symbol is used in an expression, insert the numerical value." Thus, the user may type

```
*SET A=3.141592;  SET B=23.572;  SET C=485.5
```

and then use these symbols to identify the values defined in the SET command.

```
*TYPE A+B+C
  512.2136*
```

Symbols may consist of one, two, or three alphanumeric characters. The first character must be a letter, but must not be the letter F which refers to function names (Refer to Section 3.16).

---

[1] Any number appearing in a TYPE command must have its magnitude represented in 35 bits of mantissa, otherwise, FOCAL will type the ?27 error message and ignore your request. This error message will occur with an 11 or 12 digit or longer number, depending on the magnitude of the number. The same is applicable for the ASK command.

FOCAL always checks user input for syntax errors (e.g., invalid commands, illegal formats, etc.). When an error is detected, FOCAL types an error message in the form of a question mark and code number to indicate the type of error. In the following example,

```
*
*
*HELP
?10
*TYPE 2++4
?21
```

HELP is not a valid command and two plus signs (double operators) is an illegal operation. The complete list of error messages and meanings is given in Appendix B.

2.2 OUTPUT FORMAT

The FOCAL program is originally set to produce results showing up to eight digits, four to the left of the decimal point (the integer part) and four to the right of the decimal point (the fractional part). Leading zeros are suppressed, and spaces are shown instead. Trailing zeroes are included in the output to the limits of the format, as shown in the examples below.

```
*
*SET A=77.77; SET B=111111.1111; SET C=39
*TYPE A,B,C
   77.7700 111111.11    39.0000*
*
```

The output format may be changed if the user types

```
*
*TYPE %x.yz
```

where the percent sign (%) is the format operator symbol,x is the total number of digits to be output and yz is the number of digits to the right of the decimal point. The values x and yz are positive integers, and the value of x cannot exceed 63 digits. The value yz is always written as a 2 digit number, (e.g., 03). For example, if the desired output format is 2 places to the left of the decimal point and five to the right, the user types

```
*
*TYPE %7.05, 12.222222+2.37184
```

and FOCAL types

```
*
*
*14.59406*
```

Notice that the format operator (%x.yz) must be followed by a comma, and that until the user changes the output format all results will be typed in the last specified format, i.e. %7.05.

The results are calculated to nine digits. In some circumstances since rounding may place some uncertainty on the 9th place, the user may need to account for the rounding. If the user types

```
*
*
*TYPE %9.04, 123456.789
```

FOCAL types

```
 123456.789*
*
*
```

Of the 9 available digits, priority is given to those to the left of the decimal point.

In the following examples, the number 2848.5363 is typed out in several formats.

```
*
*SET A=2848.5363
*TYPE %7.03, A
 2848.536*
*TYPE %8.04, A
 2848.5363*
*TYPE %9.05, A
 2848.53630*
*
```

If the user does not indicate the number of places in the fractional part of the number, only the integer part is printed.

If the specified output format is too small to contain the integer portion of the number, FOCAL converts the number to floating point form, 0.LE+mn, where E+mn indicates the mn^{th} power of 10 of the number L printed as a number between 0.0 and 1.0 (refer to Section 2.3).

```
*
*TYPE %3, A
 0.285E+04*
*
```

If the specified format is larger than the number, FOCAL inserts leading spaces up to, but not including, the asterisk column.

```
*
*TYPE %11, A
          2849*
*
```

Leading blanks and zeros in integers are always ignored by FOCAL, except for numbers between 0.0 and 1.0, where a zero precedes the decimal point..

```
*
*
*TYPE %8.04, 0016, 0.016, ., 00700
   16.0000    0.0160    0.0000  700.0000*
```

2.3 FLOATING-POINT FORMAT

The user may request output in exponential form which is called floating-point or E format. This notation is frequently used in scientific computations and is the format in which FOCAL performs its internal computations. The user requests floating-point format by including a % followed by a comma in a TYPE command. FOCAL will print out 0, a decimal point, a 9-digit number, the letter E, and the number of places to move the decimal point for standard notation. Until the user specifies another output format, all results are typed out in floating-point format.

For example,

```
*
*TYPE %,1111
 0.111100000E+04*
*
```

is interpreted as .1111 times 10^4 or 1111. Exponents can be used to ±999. The largest number that FOCAL can handle is +0.999999983 times 10^{998}, and the smallest is -0.999999983 times 10^{999}.

If the absolute value of the exponent is 1000 or greater a colon (:) will replace the higher order digits of the exponent. (An exponent of 1021 comes out as :21). For example, see the description of FEXP in Section 3.16.

The user should furthermore note that for systems having EAE only eight places of accuracy can be guaranteed for numbers larger than 1.0E43. (In fact, for numbers near 1.0E+998 or 1.0E-999 you only get 7-place accuracy.)

To demonstrate the ability of FOCAL to compute large numbers, find the value of 449 factorial by typing the following commands:

```
*SET A=1
*FOR I=1,449; SET A=A*I
*TYPE %,A
 0.385193052E+998*
```

The FOR statement, which will be explained later, is used to set I equal to each integer from 1 to 449.

2.4 ARITHMETIC OPERATIONS AND SYMBOLS

FOCAL performs the usual arithmetic operations (addition, subtraction, multiplication, division, and exponentiation). These operations are written by using the following symbols:

SYMBOL		MATH NOTATION	FOCAL
↑ Exponentiation		3^3	3↑3 (Power must be a positive integer)
* Multiplication		3·3	3*3
/ Division		3÷3	3/3
+ Addition	equal	3+3	3+3
- Subtraction	priority	3-3	3-3

These operations may be combined into expressions. When FOCAL evaluates an expression comprising several arithmetic operations, the priority follows the above list.

Note that addition and subtraction have equal priority. Expressions with these two operators are evaluated from left to right.

A+B*C+D is A+(B*C)+D not (A+B)*(C+D) nor (A+B)*C+D
A*B+C*D is (A*B)+(C*D) not A*(B+C)*D nor (A*B+C)*D

X/2*Y is $\frac{X}{2Y}$ not $\left(\frac{X}{2}\right)Y$

2 ↑2 ↑3 is 4^3 not 2^8

To perform exponentiation to a negative power, X^{-A}, use FEXP(A*FLOG<X>).

Expressions (except IF) to be evaluated by FOCAL may be enclosed in any properly paired parentheses, square brackets, or angle brackets. The IF statements, however, must be enclosed in parentheses.

For expressions without IF statements:

```
*SET A1 = (A+B)*<M+N>*[X+Y]
```

The left bracket ([) and the right bracket (]) enclosures which do not appear on certain teletypewriter keys are typed using the SHIFT and K keys and the SHIFT and M keys, respectively.

For expressions that are nested, FOCAL computes the value of the innermost expression first and then works outward.

```
*TYPE %,  (2+<3-[1*4]+5 > -2)
 0.400000000F+01*
```

Note that this number is expressed in floating-point format, as specified by the unmodified % symbol.

2.5 ADDITIONAL SYMBOL INFORMATION

The value of a symbolic name or identifier is not changed until the expression to the right of the equal sign is evaluated by FOCAL. Therefore, before it is evaluated, the value of a symbolic name or

identifier can be changed by retyping the identifier and assigning it a new value.

```
*SET A1=3↑2;  SET A1=A1+1
*TYPE  %2, A1
 10*
```

Symbolic names or identifiers must not begin with the letter F. (Refer to Section 3.16)

The user can request FOCAL to type out all user defined identifiers, in the order of definition, by typing a dollar sign ($) after a TYPE command. (Refer to Section 3.1)

```
*TYPE  %7.2,$
```

The user's symbol table is typed out in the following manner:

```
A@@(00)=  0.3851931E+998
B@@(00)=  111111.1
C@@(00)=  39.00000
K@@(00)=  0.000000
I@@(00)=  450.0000
A1@(00)=  10.00000
M@@(00)=  0.000000
N@@(00)=  0.000000
X@@(00)=  0.000000
Y@@(00)=  0.000000
```

NOTE

"A" and "I" defined in a previous example (on page 2-3) were not erased before going on to the present example.

If an identifier consists of less than three letters, an @ is inserted as the second/third character in the symbol table printout, as shown in the example above. An identifier may be longer than three characters, but only the first three are recognized by FOCAL and stored in the symbol table.

2.6 SUBSCRIPTED VARIABLES

FOCAL always allows identifiers, or variable symbols, to be further identified by single subscripts in the range ± 131071 (2^{17}-1), which are enclosed in parentheses immediately following the identifier. For example, the following identifiers are subscripted:

```
A(I)  B(3)
```

A subscript may also be an expression:

```
*SET A1(I+3*J)=2.33
*SET X2(5+3*J)=8.20
```

The ability of FOCAL to compute subscripts is especially useful in generating arrays for complex programming problems. A convenient way to generate linear subscripts is shown in Section 4.5.

2.7 THE ERASE COMMAND

To delete all of the symbolic names which are defined in the symbol table, except those in the COMMON area (refer to Section 5.4), type ERASE. As FOCAL does not clear the user's symbol table area in core memory when it is first loaded, it is good programming practice to type an ERASE command before defining any symbols.

2.8 HANDLING TEXT OUTPUT

Text strings are enclosed in quotation marks ("...") and may include most teletypewriter printing characters and spaces. The carriage return, line feed, and leader-trailer characters are not allowed in text strings. To instruct FOCAL to type an automatic carriage return line feed at the end of a text string, the user inserts an exclamation mark (!).

```
*TYPE"ALPHA"!"BETA"!"GAMMA"!
ALPHA
BETA
GAMMA
*
```

If only a carriage return without a line feed is desired at the end of a text typeout, the user inserts a number sign (#).

```
*TYPE  !"  X Y Z"#"   +   ="#"   /"!
   X+Y = Z
*
```

The number sign operator is useful in formatting output and in plotting another variable along the same coordinate (Refer to Section 4.7).

2.9 INDIRECT COMMANDS

Up to this point, only direct commands, executed immediately by FOCAL, have been discussed. In contrast, commands may be delayed to alter sequences, assign all variables or generate a lengthy program. These delayed execution statements are called indirect commands which are prefixed by a line number and are stored by FOCAL for later execution, usually as part of a sequence of commands. Line numbers must be in the range 1.01 to 99.99. The number to the left of the point is called the group number; the number to the right is called the step number. (The numbers 1.00, 2.00, etc., are illegal line numbers; they are used to indicate an entire group of lines.) For example,

```
*
*
*ERASE ALL
*1.1 SET A=3
*1.2 SET B=8
*1.3 TYPE %2, A+B
*
```

To execute indirect commands the user types one of the direct commands GO, GOTO, and DO.

The GO command causes FOCAL to go the lowest numbered line to begin executing the program. If the user types a direct GO command after the indirect commands above, FOCAL will start executing at line 1.1.

```
.GO
 11*
```

The GOTO command causes FOCAL to start the program by executing the command at a specified line number.

```
*GOTO 1.2
 11*
```

FOCAL started executing the program at line, 1.2 SET B=8, in the above example, and then continued to line 1.3.

The DO command is used to transfer control to a specified step, or group of steps, and then return automatically to the command following the DO command.

```
*
*ERASE ALL
*1.1 SET A=1;SET B=2
*1.2 TYPE " STARTING "
*1.3 DO 3.2
*2.1 TYPE " FINISHED "
*3.1 SET A=3; SET B=4
*3.2 TYPE %1, A+B
*GO
 STARTING  3 FINISHED  7*
*
```

When the DO command at line 1.3 was reached, the command TYPE %1, A+B was performed and then the program returned to line 2.1 and continued from there.

The DO command can also cause FOCAL to jump to a group of commands and then return automatically to the normal sequence.

```
*
*ERASE ALL
*1.1 TYPE "A"
*1.2 TYPE "B"
*1.3 TYPE "C"
*1.4 DO 5.0
*1.5 TYPE " END "; GOTO 6.1
*5.1 TYPE "D"
*5.2 TYPE "E"
*6.1 TYPE "."
*GO
ABCDE END .*
*
```

When the DO command at line 1.4 was reached, FOCAL executed the group 5 lines and then returned to line 1.5. An indirect command, with the proper sequential line number, can be inserted in a program at any time before the direct execute command. For example,

```
*
*ERASE ALL
*4.8 SET A=1; SET B=2
*6.3 TYPE %8.3, B/C+A
*4.9 SET C=3.4581
*GO
 1.5783523*
```

where line 4.9 will be executed before line 6.3 and after line 4.8. FOCAL arranges and executes indirect commands in numerical sequence by line number.

2.10 ERROR DETECTION

FOCAL checks all input commands for a variety of errors. If an error is detected, FOCAL types a question mark, followed by an error code and the appropriate line number if the error is in an indirect command. A complete list of these error codes is shown in Appendix B.

The WRITE command without an argument causes FOCAL to print out the entire indirect program so that the user may check it for errors.

The trace feature of FOCAL is valuable in program debugging. Any part of an indirect statement or program can be enclosed in question marks, and when that part of the program is executed, the portion enclosed in question marks will be printed out. If only one question mark is inserted the program is printed out from that point until completion. The trace feature is also used to follow program control and to create special formats (Refer to Section 3.15).

2.11 CORRECTIONS

If the user types the wrong character, or several wrong characters, the RUBOUT key, which echoes a backslash (\) for each RUBOUT typed, is used to delete one character to the left each time the RUBOUT key is depressed.

```
*
*ERASE ALL
*1.1 RYPE\\\\TYPE X-Y
*1.2 SET X=12\3
*WRITE

C FOCAL XVM V1A000
01.10 TYPE X-Y
01.20 SET X=13
```

Typing CTRL U (echoes an @) deletes everything which appears to its left on the same line.

```
*1.3 TYPE A,B,C@
*WRITE

C FOCAL XVM V1A000
01.10 TYPE X-Y
01.20 SET X=13
```

A line can be overwritten. Repeat the same line number and type the new command. For example, the second instance of line 14.99 replaces the first:

```
*
*
*
*14.99 SET C9(N+3)=15
*
*

*
*
*14.99 TYPE C9/Z5-2
*WRITE 14.99
14.99 TYPE C9/Z5-2
```

When WRITE is typed after corrections are made, FOCAL will print the indirect program as altered. With this feature, commands can be checked and a "clean" program printout can be obtained. Remember that all indirect input is printed when WRITE is typed. Therefore, it is useful to type ERASE ALL at the start of a new sequence. (Refer to Chapter 5 for storing programs.) The ERASE command with an argument will delete a line or group of lines. For example, to delete line 2.21, the user types

```
*
*ERASE  2.21
*
```

To delete all of the lines in group 2, the user types

```
*
*ERASE 2.0
*
```

Used alone, without an argument, the ERASE command causes FOCAL to erase the user's entire symbol table. FOCAL does not zero memory when loaded; consequently, it is good practice to type ERASE before defining symbols. The command ERASE ALL erases all user input, except COMMON variables.

The MODIFY command is another valuable feature. It may be used to change any number of characters in a particular line, as explained in Section 3.14.

2.12 ABBREVIATIONS

All FOCAL commands (except COMMON and DATA) may be abbreviated to the first letter of the command. Thus,

```
*TYPE 10,!
 0.1E+02
*
```

is equivalent to

```
*
*T 10,!
 0.1E+02
*
```

2.13 ALPHANUMERIC NUMBERS (Using Letters as Numbers)

Numbers must start with a numeral but may contain letters. FOCAL interprets as a number any character string beginning with a numeral (0 through 9). An alphanumeric number is a string of alphanumeric characters (excluding symbols) which starts with a numeral. For example,

```
*
*0ABC     23BAT     2836A1
?02
```

Each letter in an alphanumeric number is taken as a number (A through Z correspond to 1 through 26, respectively) except for E (which denotes exponentiation).

NOTE

E denotes exponentiation; consequently, the number 5 cannot be represented in alphanumeric form.

A=1	J=10	S=19
B=2	K=11	T=20
C=3	L=12	U=21
D=4	M=13	V=22
E=(exponentiation)	N=14	W=23
F=6	O=15	X=24
G=7	P=16	Y=25
H=8	Q=17	Z=26
I=9	R=18	

An easy way to give FOCAL numerical valued letters is to start the number with 0, as in the following example.

```
*
*TYPE %,0AB
 0.120000000E+02*
*
```

After 0, A=1 and B=2; thus, 0AB=12. Alphanumeric characters may be used in arithmetic operations.

```
*
*TYPE %, 0AB+0C
 0.150000000E+02*
*
```

The letter E denotes exponentiation to base 10 when used in a number. Alphanumerics after the letter E are taken as the exponent of the preceding alphanumerics.

Only one E is allowed in any one alphanumeric number.

```
*
*
*TYPE %8,0AED
    10000*
*TYPE %8,0SEC
    19000*
*
*
*
```

Alphabetic characters may be used when assigning numerical values to identifiers or variables in response to an ASK statement (Refer to Section 3.9 for a use of this feature and lines 3.20 and 3.30 of "Intercept and Plot of Two Functions" in Section 4.7 for an application).

CHAPTER 3
FOCAL COMMANDS

3.1 TYPE

The TYPE command is used to compute and type out a text string, the result of an expression, or the value of an identifier. For example,

```
*
*
*4.14 TYPE 3.2*6-(36.2*65)/2.348
*4.15 TYPE 3^6+(7.23/4.2753)*73.4
*
*
```

Several expressions can be computed by a single TYPE command; commas are used to separate each expression.

```
*1.1 TYPE %6.03, A1*2, 2^12, 2.28*83.636
*DO 1.1
   0.000 4096.00 190.690*
*
*
*
```

The output format (%) can be included in the TYPE statement as shown in the example above and as explained in Section 2.2.

The user may request a typeout of all identifiers which he has defined by typing TYPE $ and pressing the RETURN key. This causes FOCAL to type out the identifiers with their values, in the order in which they were defined. The $ can follow other statements in a TYPE command, but must always be the last operation on the line.

```
*
*
*ERASE ALL
*SET L=33; SET B=22; SET Q=385
*SET A3=94.3; SET A7T=2.485
*TYPE %5.03,$
-
L@@(00)= 33.000
B@@(00)= 22.000
Q@@(00)= 385.00
A3@(00)= 94.300
A7T(00)=  2.485
*
```

A text string enclosed in quotation marks can be included in a TYPE command, and a carriage return can replace the closing quotation mark:

```
*
*
*TYPE "X SQUARED
X SQUARED*
*
```

A text string or any FOCAL command or group of commands cannot exceed the capacity of a teletype line (72 characters for KSR33 Teletype[1]). A command cannot be continued on the following line. To print out extended text, each line must start with a TYPE command.

FOCAL does not automatically perform a carriage return after executing a TYPE command. To insert carriage return-line feed characters type an exclamation mark (!). To insert a carriage return without a line feed, type a number sign (#). To insert spaces, enclose them in quotation marks. These operations are useful for format output.

3.2 ASK

The ASK command is normally used in indirect commands to allow the user to input data at specific points during the execution of the program. The ASK command is written in the form:

```
*
*11.99 ASK X,Y,Z,
*
```

When step 11.99 is encountered by FOCAL, it types a colon (:). Then, the user types a value in any format for the first identifier, followed by a carriage return or ALT MODE. The ALT MODE key continues the text on the same line. FOCAL then types another colon, and the user types a value for the second identifier. This continues until all the identifiers or variables in the ASK statement have been given values.

```
*11.99 ASK X,Y,Z
*DO 11.99
:4:4:8*
*
*
*
```

In the above example, the user typed 4,4 and 8 as the values, respectively, for X,Y,Z.

FOCAL recognizes each value when its terminator (i.e., carriage return or ALT MODE) is typed. Therefore a value can only be changed before its terminator is typed. This is done by using RUBOUT or CTRL U.

---

[1] Teletype is a registered trademark of the Teletype Corporation.

A text string can be included in an ASK statement if the string is enclosed in quotation marks.

```
*
*
*ERASE ALL
*1.1 ASK "HOW MANY APPLES DO YOU HAVE?" APPLES
*DO 1.1
HOW MANY APPLES DO YOU HAVE?:25
*TYPE APP
 25.000*
*
```

The identifier APP (FOCAL recognized only the first three characters of the identifier APPLES.) now has the value 25. When APP is used, it will equal 25. Its value may be reassigned if it is asked for again.

```
*
*
*ERASE ALL
*ASK APP
:30
*TYPE APP
 30.000*
*
```

Alphabetic characters can be used if numerical values are assigned to identifiers or variables:

```
*1.1 ASK A; TYPE %4,A
*DO 1.1
:ABCD
 1234*
*
*
*
*
```

When the user typed ABCD and RETURN, FOCAL typed the numerical value of ABCD (Refer to "Alphanumeric Numbers", Section 2.13).

Alphabetic responses are especially useful for keyboard responses to FOCAL statements. A YES or NO answer can be typed by the user during program execution in response to a program question, as explained in Section 3.9.

3.3 WRITE

A WRITE command without an argument causes FOCAL to write out all indirect statements which the user has typed. Indirect statements are those preceded by a line number.

A group of line numbers, or a specific line, can be typed out with the WRITE command using arguments, as shown below.

```
*7.97 WRITE 2.0
*7.98 WRITE 2.1
*7.99 WRITE
*
```

3.4 SET

The SET command is used to define identifiers. When FOCAL executes a SET command, the identifier and its value, are stored in the user's symbol table. When the identifier is encountered in the program, the value is substituted for the identifier.

```
*
*ERASE ALL
*4.1 SET A=394.83; SET B=4.373
*4.2 TYPE %,A+B
*GO
 0.399203000E+03*
*
*
*
```

An identifier can be set equal to previously defined identifiers, which, can be used in arithmetic expressions.

```
*
*3.7 SET G=(A+B)*2^6
*
```

3.5 ERASE

An ERASE command without an argument is used to delete all identifiers and their values except those in COMMON (Refer to Section 5.4 for the ERASE COMMON command.) from the symbol table.

If the ERASE command is followed by a group number or a specific line number, a group of lines or a specific line is deleted from the program.

```
*
*ERASE 2.0
*ERASE 7.11
*
```

The ERASE ALL command erases all the user's input. In the following example, an ERASE command is used to delete line 1.50.

```
*
*
*ERASE ALL
*1.2 SET B=2
*1.3 SET C=34
*1.4 TYPE B+C
*1.5 TYPE C-B
*ERASE 1.5
*WRITE

C FOCAL V3A000
01.20 SET B=2
01.30 SET C=34
01.40 TYPE B+C
```

The ERASE ALL command is generally used only in immediate mode because it returns to command mode upon completion.

3.6 GO

The GO command is used to execute the program which starts with the lowest numbered line. The remainder of the program is executed in line number sequence. Line numbers must be in the range 1.01 to 99.99.

3.7 GOTO

The GOTO command causes FOCAL to transfer control to a specific line in an indirect program. It must be followed by a specific line number. After executing the command at the specified line, FOCAL continues to the next higher line number, executing the program sequentially.

```
*
*
*
*ERASE ALL
*1.1 TYPE "A"
*1.2 TYPE "B"
*1.3 TYPE "C"
*1.4 TYPE "D"
*GOTO 1.2
BCD*
*
*
```

3.8 DO

The DO command transfers control momentarily to a single line, a group of lines, or an entire indirect program. If transfer is made to a single line, the statements on that line are executed, and control is transferred back to the statement following the DO command. Thus, the DO command makes a subroutine of the lines to which control is transferred, as shown in the following example:

```
*
*
*ERASE ALL
*1.1 TYPE "F"
*1.2 DO 2.3; TYPE "C"
*1.3 TYPE "A"
*1.4 TYPE "L"
*1.5 QUIT
*2.3 TYPE "O"
*GO
FOCAL*
*
*
*
```

If a DO command transfers control to a group of lines, FOCAL executes the group sequentially and returns control to the statement following the DO command.

If DO is written without an argument, FOCAL executes the entire indirect program in the same manner as a GO command.

DO commands cause specified portions of the indirect program to be executed as closed subroutines. These subroutines can also be terminated by a RETURN command.

A GOTO or an IF statement within a DO subroutine modifies the program execution sequence.

3.9 IF

To transfer control after a comparison, FOCAL contains a conditional IF statement in the form IF (m)x,y,z; m is an expression or variable, and x,y,z,are three line numbers. The expression is evaluated, and the program transfers control to the first number, x, if the expression is less than zero; to the second line number, y, if the expression equals zero; or to the third line number, z, if the value of the expression is greater than zero.

```
*
*
*2.1 TYPE "LESS THAN ZERO"; QUIT
*2.2 TYPE "EQUAL TO ZERO"; QUIT
*2.3 TYPE "GREATER THAN ZERO"; QUIT
*IF (25-25) 2.1,2.2,2.3
EQUAL TO ZERO*
*
*
```

In the above example, the parenthetical expression equals zero; consequently, line 2.2 is executed. Note that an IF statement must be enclosed in parentheses.

The IF statement can be shortened by terminating it with a semicolon or carriage return after the first or second line number. If a semicolon follows the first line number, the expression is tested, and control is transferred to that line if the expression is less than zero. If the expression is not less than zero, the program continues with the next statement.

```
*
*2.20 IF(X)1.8;TYPE "C"
*
```

In the above example, when line 2.20 is executed, if X is less than zero, control is transferred to line 1.8. If not, C is typed out.

```
*
*
*3.19 IF(B)1.8,1.9
*3.20 TYPE B
*
*
```

In the above example, if B is less than zero, control goes to line 1.8, if B is equal to zero, control goes to line 1.9. If B is greater than zero, control goes to the next statement (in this case, line 3.20), and the value of B is typed.

In programs that require a keyboard response (as in Section 4.7 line 3.2), it is useful to determine if the answer by the user to an ASK question is YES or NO. Alphabetic responses used with an IF statement permit one of two possible commands to be executed, depending on the user's answer. For example:

```
IF (answer-0Yes)1.1,2.1,1.1
```

where answer is YES or NO, as typed by the user. The next command depends on whether answer is YES (in which case, answer-0YES equals 0, and line 2.1 is executed) or NO (producing a nonzero result and moving program execution to line 1.1).

For example,

```
*1.1 TYPE "DO YOU WANT A LINE?",!
*1.2 ASK "TYPE YES OR NO",ANS,!
*1.3 IF (ANS-0YES) 2.1,2.2,2.1
*2.1 QUIT
*2.2 TYPE "--------------------",!
*2.3 GOTO 1.1
*GO
DO YOU WANT A LINE?
TYPE YES OR NO:YES

--------------------
DO YOU WANT A LINE?
TYPE YES OR NO:NO
```

If a GOTO or an IF command is executed within a DO subroutine, two actions are possible:

a. If a GOTO or IF command causes transfer to a line inside the DO group, the remaining commands in that group are executed as in any subroutine before returning to the command following the DO command.

b. If transfer is to a line outside the DO group, that line is executed and control is returned to the command following the DO command unless that line contains another GOTO or IF.

```
*
*
*
*ERASE ALL
*1.1 TYPE "A"; SET X=-1; DO 3.1; TYPE "D"; DO 2
*1.2 DO 2.0
*2.1 TYPE "G"
*2.2 IF (X) 2.5,2.6,2.7
*2.5 TYPE "H"
*2.6 TYPE "I"
*2.7 TYPE "J"
*2.8 TYPE "K"
*2.9 TYPE %2.01,X; TYPE " "; SET X=X+1
*3.1 TYPE "B"; GOTO 5.1; TYPE "F"
*5.1 TYPE "C"
*5.2 TYPE "W"
*GO
ABCDGHIJK-1.0 GIJK 0.0 GJK 1.0 BCW*
*
*
*
*
*
```

3.10 RETURN

The RETURN command is used to exit from a DO subroutine. When a RETURN command is encountered during execution of a DO subroutine, the program exits from its subroutine status and returns to the command following the DO command that initiated the subroutine status.

3.11 QUIT

A QUIT command causes the program to halt and return control to the user. FOCAL types an asterisk and the user can type another command.

3.12 COMMENT

Beginning a command string with the letter C (except for COMMON) will cause the remainder of that line to be ignored to allow insertion of comments into the program. Such lines are skipped over when the program is executed, but are typed out by a WRITE command. A program that is well documented with comments is more meaningful and easier to understand than one without comments.

3.13 FOR

This command is used for convenience in setting up program loops and iterations. The general format is

```
*FOR A=B,C,D;(COMMANDS)
```

The identifier A is initialized to the value B. Then, the commands following the semicolon up to the line terminator are executed. When the commands have been executed, the value of A is incremented by C and compared to the value of D. If A is less than or equal to D, the commands after the semicolon are executed again. This process is repeated until A is greater than D; then, FOCAL goes to the next sequential line.

The identifier A must be a single variable. B,C, and D can be either expressions, variables, or numbers. If a comma and the value C are omitted, it is assumed that the increment is one. If C,D is omitted, it is handled like a SET statement and no iteration is performed.

The computations involved in the FOR statement are done in floating-point arithmetic, and it may be necessary, in some circumstances to account for this type of arithmetic computation.

Example 1 below is a simple example of how FOCAL executes a FOR command. Example 2 shows the FOR command combined with a DO command.

Example 1:

```
*
*ERASE ALL
*3.11 SET A=383.383
*3.12 FOR B=20,10,70; TYPE %7.03, "B IS " B+A,!
*GO
B IS   403.383
B IS   413.383
B IS   423.383
B IS   433.383
B IS   443.383
B IS   453.383
*
*
```

Example 2

```
*1.1 FOR X=1,2,9; DO 2.0
*1.2 GOTO 3.1
*2.1 TYPE ! "   "%5, "X" X
*2.2 SET A=X+100.00
*2.3 TYPE ! "   "%5, "A" A
*3.1 QUIT
*GO

        X     1
        A   101
        X     3
        A   103
        X     5
        A   105
        X     7
        A   107
        X     9
        A   109*
```

If two FOR statements are put on one line, the second FOR statement is performed in full for each incrementation in the first FOR statement. Any number of FOR statements may appear on one line.

3.14 MODIFY

Frequently, only a few characters in a particular line require changes. To facilitate this job, and to eliminate the need to replace the entire line, FOCAL has a MODIFY command. For example, to modify the characters in line 5.41, the user types MODIFY 5.41 and then depresses the RETURN key. The program then waits for the user to type the character he wishes to modify. After the user has typed the search character, the program types out the contents of that line until the search character is typed.

At this point, the user has seven options:

a. Type in new characters in addition to the ones that have already been typed out.

b. Type ALT MODE to continue the search to the next occurrence, if any, of the same search character.

c. To change the search character, type CTRL BELL and the new search character as at the beginning of the MODIFY command.

d. Use the RUBOUT key to delete one character to the left each time RUBOUT is depressed.

e. Type a CTRL U to delete the line over to the left margin, but not the line number.

f. Type carriage return to terminate the line at that point, removing the text to the right.

g. Type a LINE FEED to save the remainder of the line.

The MODIFY command is generally used only in immediate mode because it returns to command mode upon completion.

During command input, CTRL U deletes the line numbers as well as the text if the CTRL U is the right-most character on the line. However, when using the MODIFY command the line number is not deleted by the use of CTRL U. Note the error in line 7.01.

```
*
*
*7.01 JACK AND HILL WRNT UP THE GILL
*MODIFY 7.01
H JACK AND H\JRILL WR\EGNT UP THE G\H
                                        ILL
*WRITE 7.01
07.01 JACK AND JILL WENT UP THE HILL

*ERASE ALL
*
*
```

To modify line 7.01, the user typed an H to indicate the character to be changed. FOCAL stopped typing when it encountered the search character H. The user typed the RUBOUT key to delete the H, and then typed the correct letter, J. The user then typed the CTRL BELL key followed by the R, the next character to be changed. The RUBOUT deleted the R and the user typed E. Again a search was made (this time for the G), and the G was changed to H. The user typed a line feed to save the remainder of the line.

When the MODIFY command (or another command which alters the stored indirect program) is used, the values in the user's symbol table (except those defined as COMMON) are reset to zero. Therefore, if the user defines his symbols in direct statements and then uses a MODIFY command, the values of his symbols are erased and must be re-defined.

However, if the user defines his symbols by indirect statements prior to using a MODIFY command, the values are not erased because these symbols are not entered in the symbol table until the statements defining them are executed.

In the following example, notice that the values of Y and Z were set using direct statements. The use of the MODIFY command resets their values to zero and lists them after the defined symbols:

```
*ERASE ALL
*SET Z=9
*SET Y=8
*1.1 SET X=3
*1.2 SET W=4
*1.3 TYPE W+X+Y+Z; TYPE !; TYPE $
*MODIFY 1.1
S SET X=5

*
*GO
  9
-
X@@(00)=  5
W@@(00)=  4
Y@@(00)=  0
Z@@(00)=  0
*
*
```

3.15 USING THE TRACE FEATURE

As noted in Section 2.10, the trace feature is useful in checking an operating program. Those parts of the program which are enclosed in question marks are printed out as they are executed.

In the following example, parts of three lines are printed.

```
*
*
*ERASE ALL
*1.1 SET A=2
*1.2 SET B=5
*1.3 SET C=3
*1.4 TYPE %2, ?A+B-C?,!
*1.5 TYPE ?B+A/C?,!
*1.6 TYPE ?B-C?
*1.6 TYPE ?B-C/A?
*GO
A+B-C  4
B+A/C  6
B-C/A  4*
*
```

Also, GO? will trace the program starting with the lowest numbered line, provided no other question marks are present in the program.

3.16 INTERNAL FUNCTIONS

The internal functions provide extended arithmetic capabilities. User defined external functions are described in Chapter 6. A standard function call consists of four letters, beginning with the letter F, and followed by a parenthetical expression.

The following are the internal functions:

a. The square root function (FSQT) computes the square root of the expression within parentheses.

```
*
*
*TYPE %,FSQT(43.489)
 0.659461902E+01*
*TYPE FSQT(2.333)
 0.152741612E+01*
*TYPE FSQT(3718)
 0.609754049E+02*
*
*
```

b, The absolute value function (FABS) outputs the absolute or positive value of the number in parentheses.

```
*
*
*TYPE %, FABS(-394)
 0.394000000E+03*
*TYPE FABS(-.93)
 0.930000000E+00*
*TYPE FABS(73)
 0.730000000E+02*
*
*
```

c. The sign part function (FSGN) outputs the sign part (+ or -) of a number and the integer part becomes a 1. Zero is considered a positive number.

```
*
*TYPE %, FSGN(-283.3)
-0.100000000E+01*
*TYPE FSGN(0.00)
 0.100000000E+01*
*TYPE FSGN(-0.38)
-0.100000000E+01*
*
*
*
*
```

d. The integer part function (FITR) outputs the integer part of a number up to $\pm 131071(2^{17}-1)$.

```
*
*
*TYPE %,FITR(-34.8)
-0.340000000E+02*
*TYPE FITR(0.73)
 0.000000000E+00*
*TYPE FITR(374.92)
 0.374000000E+03*
*
*
*
```

e. The random number generator function (FRAN) computes a non-statistical pseudo-random number between -1 and +1 (most numbers fall in the range 0 to +1.). Another random number generator function (FRNO) is provided in the external function file FNEW and is described in Section 6.2.

```
*
*
*TYPE %, FRAN( )
 0.719269147E-02*
*TYPE FRAN( )
 0.549454402E-01*
*
*
*
```

f. The exponential function (FEXP) computes e (e=2.718281) to the power within parentheses.

```
*
*
*
*TYPE %, FEXP(27)
 0.532048241E+12*
*TYPE FEXP(2.348)
 0.104646196E+02*
*TYPE FEXP(0.374)
 0.145353715E+01*
*
*
*
```

In floating-point format (%) you only get correct results for FEXP(X) if $2300 \geq X \geq -2302$. If X is not between these limits the exponent of the result will not be between ± 999, and a colon will replace its high order digits as described in Section 2.3.

Example:

```
*T FEXP (2300)0.7538907IE+999*

*T FEXP (2301)0.20492876E+:00
```

The ?27 error message does not appear as it applies only to mantissas with more than 35 binary bits (11 or 12 decimal digits) while the above limitation relates to exponents, not mantissas.

g. The sine function (FSIN) calculates the sine of an angle expressed in radians.

```
*
*
*
*TYPE %, FSIN(3.10)
 0.415806618E-01*
*TYPE FSIN(0.278)
 0.274432986E+00*
*TYPE FSIN(1.272)
 0.955691507E+00*
*
*
```

FOCAL requires that angles be expressed in radians; thus, to find a function of an angle in degrees, the conversion factor $\pi/180$, must be used. To find the sine of 10 degrees:

```
*
*SET PI=3.14159; TYPE FSIN(10*PI/180)
 0.173648033E+00*
```

h. The cosine function (FCOS) calculates the cosine of an angle expressed in radians.

```
*
*TYPE  %,  FCOS(2*PI)
 0.100000000E+01*
*TYPE FCOS(.3628)
 0.934906789E+00*
*TYPE FCOS(1.37)
 0.199449721E+00*
*
```

i. The arctangent function (FATN) calculates the angle in radians the tangent of which is the argument within parentheses.

```
*TYPE %, FATN(1.000)
 0.785398164E+00*
*TYPE FATN(23.44)
 0.152816007E+01*
*TYPE FATN(0.728)
 0.629271798E+00*
*
```

j. The logarithm function (FLOG) computes the natural logarithm ($\log_e$) of the number within parentheses.

```
*TYPE %, FLOG(238.48467)
 0.547430503E+01*
*TYPE FLOG(0.2876)
-0.124618465E+01*
*TYPE FLOG(1.23)
 0.207014169E+00*
```

CHAPTER 4
EXAMPLES OF FOCAL PROGRAMS

4.1 TABLE GENERATION USING FUNCTIONS

The ability to evaluate simple arithmetic expressions and to generate values with the aid of internal functions is one of the first benefits to be derived from learning the FOCAL language. In the example that follows, a table of sine, natural logarithm, and exponential values is generated for a series of arguments. As the user becomes more familiar with these functions, he can easily combine them with standard arithmetic operations and evaluate any given formula for a single value or for a range of values.

In this example, line *1.01 outputs the desired column headings. Line *1.10 is the loop to generate values for I, beginning with the value 1.00000000 and continuing in increments of .00000010 through the value 1.00000100; the DO 2.05 command at the end of this second line causes the various functions to be executed for the I arguments. The output format %9.08 in line 2.05 specifies that all output results up to the next % symbol are to appear in fixed-point format with one digit position to the left of the decimal point and eight digit positions to the right; the second % symbol reverts the output mode back to floating point for the remaining values FLOG and FEXP.Line 01.20 (optional) returns control to the user.

The following techniques are apparent in line *2.05 of this example:

a. FOCAL commands can be abbreviated to the first letter of the command followed by a space, as shown by the use of T instead of TYPE. This technique can be used to shorten command strings.

b. Arguments can be enclosed in various ways. This feature is useful in matching correctly when a number of enclosures appear in a command.

c. Spaces can be inserted in an output format by enclosing the appropriate number of spaces within quotation marks. This procedure is recommended to improve the readability of the output results.

d. The use of very small loop increments (in this example .0000001) eliminates the need to interpolate between table values of trigonometric functions. FOCAL is usually accurate to eight significant digits but rounding in certain cases may place some uncertainty on the 8th place. Thus, the user, in some circumstances, may need to account for the rounding.

```
*1.01 T "     I             SINE                LOG                    E"!
*1.10 FOR I=1,.0000001,1.000001; DO 2.05
*1.20 QUIT
*2.05 T %9.08,I," ",FSIN(I)," ",%,FLOG<I>," ",FEXP(I),!
*GO
     I             SINE                LOG                    E
 1.00000000   0.84147099   0.806929521E-10   0.271828183E+01
 1.00000010   0.84147104   0.999785677E-07   0.271828210E+01
 1.00000020   0.84147109   0.199876442E-06   0.271828237E+01
 1.00000030   0.84147115   0.299733971E-06   0.271828264E+01
 1.00000040   0.84147120   0.399631845E-06   0.271828292E+01
 1.00000050   0.84147126   0.499529720E-06   0.271827319E+01
 1.00000060   0.84147131   0.599427595E-06   0.271828346E+01
 1.00000070   0.84147136   0.699285123E-06   0.271828373E+01
 1.00000080   0.84147142   0.799182998E-06   0.271828400E+01
 1.00000090   0.84147147   0.899040526E-06   0.271828427E+01
 1.00000100   0.84147153   0.998978747E-06   0.271828454E+01
*
```

4.2 FORMULA EVALUATION FOR CIRCLES AND SPHERES

In this example, FOCAL is used to calculate, label, and output geometric values for an indefinite number of radii typed in by the user.

Given a radius, R, FOCAL can calculate such values as:

a. circle diameter: $2R$

b. circle area: πR^2

c. circle circumference: $2\pi R$

d. sphere volume: $4\pi R^3/3$

e. sphere surface area: $4\pi R^2$

Although inches are used in this example, conversions to other systems (metric, for example) could be easily incorporated into the program, without the need for hand-calculated conversions.

The program is very straightforward. ASK is used to allow the user to type in the radius value to be used in the calculations. SET is used to supply the value of π. TYPE is used for all calculations and output. If a value (e.g., π in this example) is to be entered once and then used in repeated calculations, it should be entered by a SET command which is outside the calculation loop; otherwise, the variable must be set at the beginning of each pass through the loop. If the value of the variable changes during each iteration, however, then it must be calculated either by a SET or TYPE command within the loop.

The use of the GOTO command (line *1.50) results in an infinite loop of lines *1.10 through *1.50. This technique is used when the number of desired repetitions is not known. The looping process can be ter-

minated at any time by typing CTRL P. If, however, the number of desired repetitions is known (e.g., 10), the following method can be used.

```
*SET PI=3.14159
*1.1 ASK ...
*
*
*1.6 TYPE !!!!!
*FOR I = 1,10; DO 1
```

The ability to choose between these methods provides great flexibility in actually running FOCAL programs.

```
*C FOCAL XVM V1A000
*1.01 SET PI=3.14159
*1.10 ASK " A RADIUS OF ",R, "INCHES"
*1.20 TYPE %8.04, !,"GENERATES A CIRCLE OF: ",!
*1.21 TYPE "          DIAMETER", 2*R," INCHES",!
*1.30 TYPE "          AREA", PI*R^2," SQUARE INCHES",!
*1.31 TYPE "          CIRCUMFERENCE", 2*PI*R," INCHES",!
*1.40 TYPE !, "AND A SPHERE OF:",!
*1.47 TYPE "          VOLUME", (4/3)*PI*R^3," CUBIC INCHES",!
*1.50 TYPE !!!!!; GOTO 1.1
*GO
 A RADIUS OF :26.39INCHES
GENERATES A CIRCLE OF:
          DIAMETER   52.7800 INCHES
          AREA 2187.9041 SQUARE INCHES
          CIRCUMFERENCE  165.8131 INCHES

AND A SPHERE OF:
          VOLUME 76985.053 CUBIC INCHES




 A RADIUS OF :0.73INCHES
GENERATES A CIRCLE OF:
          DIAMETER    1.4600 INCHES
          AREA    1.6742 SQUARE INCHES
          CIRCUMFERENCE    4.5867 INCHES

AND A SPHERE OF:
          VOLUME    1.6295 CUBIC INCHES
```

4.3 ONE-LINE FUNCTION PLOTTING

This example demonstrates the use of FOCAL to present, in graphic form, some given function over a range of values. In this example, the function used is

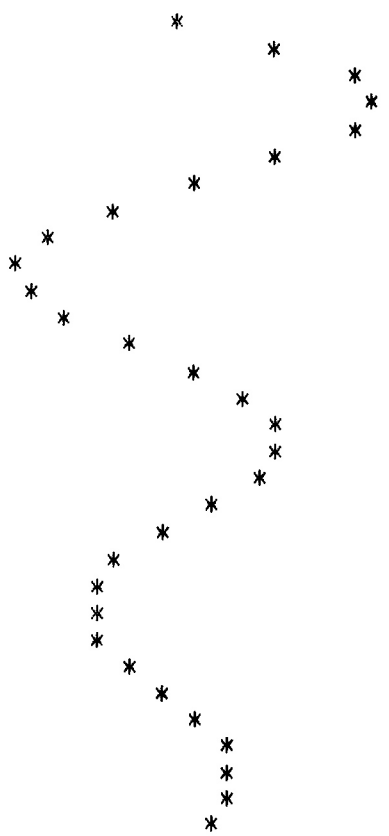
$$y=30+15[\sin(x)]e^{-0.1x}$$

with x ranging from 0 to 15 in increments of .5. This damped sine wave has many physical applications, especially in electronics and mechanics (for example, in designing automobile shock absorbers).

In the actual coding of the example, the variables I and J were used in place of x and y, respectively; any two variables could have been used. The single line 1.10 contains a set of nested loops for I and J. The J loop types spaces horizontally for the y coordinate of the function; the I loop prints the * symbol and the carriage return and line feeds for the x coordinate. The function itself is used as the upper limit of the J loop, again showing the power of FOCAL commands.

The technique illustrated by this example can be used to plot any desired function. Although the * symbol was used here, any legal FOCAL character is acceptable.

```
*1.1 F I=0,.5,15; T "*",!; F J=0,30+15*FSIN(I)*FEXP(-.1*I); T " "
*DO 1.1
*
```



4.4 DEMONSTRATION DICE GAME

Occasionally, the computer user will apply the computer to tasks solely for his own enjoyment. Because such pastimes are usually keyboard oriented, FOCAL lends itself nicely to these ends. The following example uses the random number generator, FRAN ( ), to produce dice combinations, as well as IF logic to check bets and winning combinations.

Note again the use of initials to abbreviate commands throughout the example (remember that each such abbreviation must be followed by a space).

The random number generator must be modified for use with statistical or simulation programs to achieve true randomness. However, it is sufficiently random for most applications in its present form.

NOTE

DEC does not assume any responsibility for the use of this routine or any similar routines.

```
C FOCAL XVM V1A000
01.10 S B=0$T !!"DICE GAME"!,"HOUSE LIMIT IS $1000"
01.13 T ". MIN, BET IS $1.00"!!
01.20 ASK "YOUR BET IS"A$I (1000-A) 3.10
01.22 I (A-1)3.40,1.26,1.26
01.26 IF (A-FITR(A))3.50,1.30,3.50
01.30 ASK M$DO 2$SET D=C$DO 2$T " "$SET D=D+C
01.32 I (D-7)1.42,3.20,1.42
01.40 I (D-2)1.50,3.30,1.50
01.42 I (D-11)1.40,3.20,1.40
01.50 I (D-3) 1.60,3.30,1.60
01.60 ASK M$DO 2$S E=C$DO 2$T " "$S E=E+C
01.72 I (E-7) 1.74,3.30,1.74
01.74 I (E-D) 1.60,3.20,1.60
02.10 SET C=FITR(10*FABS(FRAN()))$IF (C-6)2.20,2.20,2.10
02.20 I (C-1)2.10$T %1," "C$ RETURN
03.10 T "HOUSE LIMITS ARE $1000"!!$ G 1.20
03.20 S B=B+A$T %6.0, !"YOU WIN. YOUR WINNINGS ARE   ",B,!!$G 1.2
03.30 S B=B-A$T %6.0, !"SORRY YOU LOSE.  YOUR WINNINGS ARE  ",B,!!$G 1.2
03.40 T "MIN. BET IS $1"!!$G 1.2
03.50 T "NO PENNIES, PLEASE"!!$GOTO 1.2
```

```
*GO

DICE GAME
HOUSE LIMIT IS $1000. MIN. BET IS $1.00

YOUR BET IS:.50
MIN. BET IS $1

YOUR BET IS:15
:
  5  5 :
  5  5
YOU WIN. YOUR WINNINGS ARE        15

YOUR BET IS:5
:
  5  5 :
  5  5
YOU WIN. YOUR WINNINGS ARE        20

YOUR BET IS:3
:
  5  2
YOU WIN. YOUR WINNINGS ARE        23

YOUR BET IS:I'LL QUIT WHILE I'M AHEAD. THANKS!
```

4.5 SIMULTANEOUS EQUATIONS AND MATRICES

Many disciplines use subscripted variables for vectors in one, two, or more dimensions to store and manipulate data. A common use is the 2-dimensional array or matrix for handling sets of simultaneous equations. For example,

Given $1X_1 + 2X_2 + 3X_3 = 4$

$4X_1 + 3X_2 + 2X_3 = 1$

$1X_1 + 4X_2 + 3X_3 = 2$

Find: The values of X_1, X_2, and X_3 to satisfy all three equations simultaneously.

The solution can be reduced to simple mathematics between the various elements of the rows and columns until correct values of X are found.

Each individual quantity in an array is referred to in terms of its position within the array. This identifier is a subscript. The notation A(I) refers to element I of array A.

FOCAL uses only a single subscript. Thus, the handling of two or more dimensions requires the generation of a linear subscript which represents the correct position if it were stored in normal order, i.e., leftmost subscript moving fastest.

In one dimension:

For example:

ARRAY	(0)	A
	(1)	B
	(2)	C
	(3)	D
	(4)	E

Element D could be represented as ARRAY (3); any element in this array can be represented by a subscript in the range 0 through 4. The first element in an array always has a subscript of 0.

In two dimensions:

ARRAY (row,column) or A(I,J)

This must be reduced to the form A(G). Because subscripts are linear, G is a function of I and J; that is, A(I,J)=A(G). Consider the diagram

	J= 0	1	2
I = 0	0	5	10
1	1	6	11
2	2	7	12
3	3	8	13
4	4	9	14

This array has five rows and three columns; thus, two values can be defined:

```
IMAX = 5
JMAX = 3
```

To generate the number (G) in any box, using the corresponding values of I and J, the formula

G=I+IMAX*J or A(G)=A(I+IMAX*J)

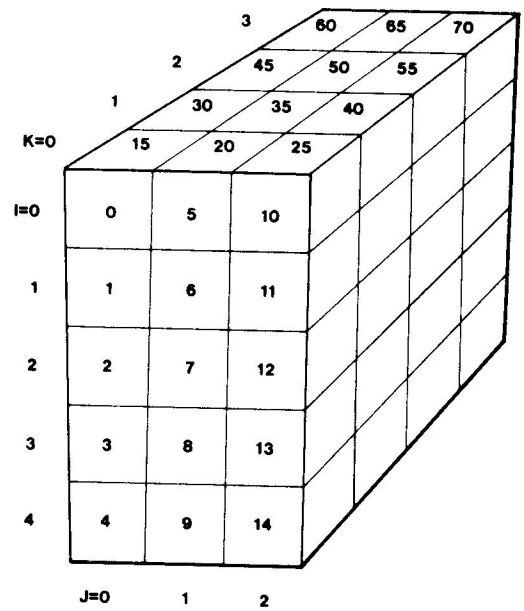
can be used. Each element in a 2-dimensional array represents an area. The example for solving simultaneous equations, above, uses this algorithm for subscripts, merely by replacing I, IMAX, and J with J, L, and K, respectively, to form the equation

A(J+L*K)

In three dimensions

ARRAY (row,column,plane)=A(I,J,K)=A(G)

Three dimensions can be illustrated as a rectangular solid.



This rectangular solid has dimensions of five rows, three columns, and four planes; thus, IMAX=5,JMAX=3, and KMAX=4. Each plane is numbered exactly as in the 2-dimensional example, except 15 times K (with K = the number of planes back from the first) is added to each subscript in the first plane.

Example:

Upper lefthand square, back one plane from the first = 15

I=0,J=0,K=1;I+(IMAX*J)+(IMAX*JMAX*K)=15=G

or

A(0,0,1)=A(15)

In four dimensions:

ARRAY (row,column,plane,cube)=A(I,J,K,L)=A(G)

Assign the values for IMAX, JMAX, KMAX; a method similar to the one above yields

G=I+(IMAX*J)+(IMAX*JMAX*K)+(IMAX*JMAX*KMAX*L)

This process can theoretically be extended indefinitely to n-dimensions.

```
C FOCAL XVM V1A000
01.02 TYPE !"ROUTINE TO SOLVE MATRIX EQ. AX=B FOR X"!
01.04 ASK "ENTER DIMENSION OF A, THEN
01.05 TYPE !"ENTER COEFF'S A(J,K)...A(J,N) AND B(J)"!
01.10 ASK L,!;SET N=L-1; SET I=-1
01.11 FOR K=0,N; SET R(K)=K+1
01.12 FOR J=0,N; TYPE !; FOR K=0,L; ASK A(J+L*K)
01.14 SET M=1E-6
01.16 FOR J=0,N; FOR K=0,N; DO 4
01.17 SET R[P]=0.
01.18 FOR K=0,L; SET A[P+L*K]=A[P+L*K]/M
01.20 FOR J=0,N; DO 5
01.22 SET I=I+1
01.23 IF (I-N) 1.14, 1.26 , 1.14
01.26 FOR J=0,N; FOR K=0,N; DO 7
01.28 FOR K=0,N;TYPE !%2,"X("K,")  ",%8.05,X(K)
01.29 TYPE !!; GOTO 1.02
04.05 IF (R<J>) 0, 4.3, 4.1
04.10 IF (FABS(A(J+L*K)) - FABS[M]) 4.3;
04.20 SET M=A(J+L*K)
04.22 SET P=J; SET Q=K
04.30 RETURN
05.10 IF (J-P) 5.2,5.4,5.2
05.20 SET D=A(J+L*Q)
05.30 FOR K=0,L; SET A<J+L*K>=A<J+L*K>-A<P+L*K>*D
05.40 RETURN
07.10 IF (1E-6-FABS[A(J+L*K)]) 7.2; RETURN
07.20 SET X(K)=A(J+L*L)

*GO

ROUTINE TO SOLVE MATRIX EQ. AX=B FOR X
ENTER DIMENSION OF A, THEN
ENTER COEFF'S A(J,K)...A(J,N) AND B(J)
:3


:1:3:2:4

:5:3:6:2

:3:1:2:1

X(  0)      0.25000
X(  1)      1.75000
X(  2)  -   0.75000


ROUTINE TO SOLVE MATRIX EQ. AX=B FOR X
ENTER DIMENSION OF A, THEN
ENTER COEFF'S A(J,K)...A(J,N) AND B(J)
:3


:4:5:2:3

:1:3:8:5

::2:6:1

X(  0)      5.00000
X(  1)  -   4.00000
X(  2)      1.50000
```

4.6 INTEREST PAYMENT PROGRAM

This is an example of a business-oriented FOCAL program. It is designed to completely describe the payments to be made on a loan, with interest, on an installment plan basis.

Under program control, the computer requests as input the amount of a loan, the percentage of interest on that loan, and the length of time over which the loan is to be paid. The computer than calculates and types the amount of monthly payments to be paid, the total amount of interest to be paid, and a table showing interest paid, amount applied to principal, and balance due after each payment.

```
C FOCAL XVM V1A000
01.02 TYPE !!, %7.02
01.20 TYPE " THIS PROGRAM WILL COMPUTE MONTHLY PAYMENTS AND THE"
01.21 TYPE " CONTRIBUTION OF EACH"
01.22 TYPE ! "PAYMENT TO INTEREST AND PRINCIPAL. PLEASE ANSWER THE"
01.23 TYPE " FOLLOWING:",!!
01.30 ASK "WHAT IS THE AMOUNT OF THE PRINCIPAL?" PRINCIPAL
01.31 ASK "WHAT IS THE RATE OF INTEREST?" INTEREST
01.32 ASK "WHAT IS THE TERM OF THE LOAN IN MONTHS?" TERM
01.33 TYPE !!
01.40 SET D=1+(<INTEREST/12>*.01)
01.41 FOR A=1,1,TERM; DO 15.99
01.42 SET C=PRINCIPAL/B
05.01 SET BALANCE=PRINCIPAL
05.02 TYPE "PRINCIPAL"
05.03 TYPE PRINCIPAL;  TYPE "           "
05.04 TYPE "PAYMENTS"; TYPE C; TYPE !!!
05.05 TYPE "     PAYMENT"; TYPE "          "
05.06 TYPE "INTEREST"; TYPE "         "
05.07 TYPE "PRINCIPAL"; TYPE "         "
05.08 TYPE "BALANCE"; TYPE "    "; TYPE !
06.01 FOR M=1,1,TERM; DO 7.00
06.02 TYPE !!!; TYPE "             ";
06.03 TYPE "TOTAL INTEREST",%6.02
06.04 TYPE TOTINTEREST
06.05 TYPE !!!
06.06 GOTO 1.02
07.01 SET CINTEREST=BALANCE*(INTEREST/12)*.01
07.02 SET TOTINTEREST=TOTINTEREST + CINTEREST
07.03 SET CPRINCIPAL=C-CINTEREST
07.04 SET BALANCE=BALANCE-CPRINCIPAL
07.05 TYPE %7.02,M,%14.02
07.07 TYPE CINTEREST,CPRINCIPAL,BALANCE,!
15.99 SET B=B+(1/<D>^A)
```

```
 THIS PROGRAM WILL COMPUTE MONTHLY PAYMENTS AND THE CONTRIBUTION OF
EACH PAYMENT TO INTEREST AND PRINCIPAL. PLEASE ANSWER THE FOLLOWING:

WHAT IS THE AMOUNT OF THE PRINCIPAL?:2000
WHAT IS THE RATE OF INTEREST?:6.4
WHAT IS THE TERM OF THE LOAN IN MONTHS?:12


PRINCIPAL  2000.00          PAYMENTS   172.50
```

PAYMENT	INTEREST	PRINCIPAL	BALANCE
1.00	10.67	161.84	1838.17
2.00	9.80	162.70	1675.47
3.00	8.94	163.57	1511.90
4.00	8.06	164.44	1347.47
5.00	7.19	165.32	1182.15
6.00	6.31	166.20	1015.96
7.00	5.42	167.08	848.87
8.00	4.53	167.97	680.90
9.00	3.63	168.87	512.03
10.00	2.73	169.77	342.26
11.00	1.83	170.68	171.59
12.00	0.92	171.59	0.00
	TOTAL INTEREST	70.01	

4.7 INTERCEPT AND PLOT OF TWO FUNCTIONS

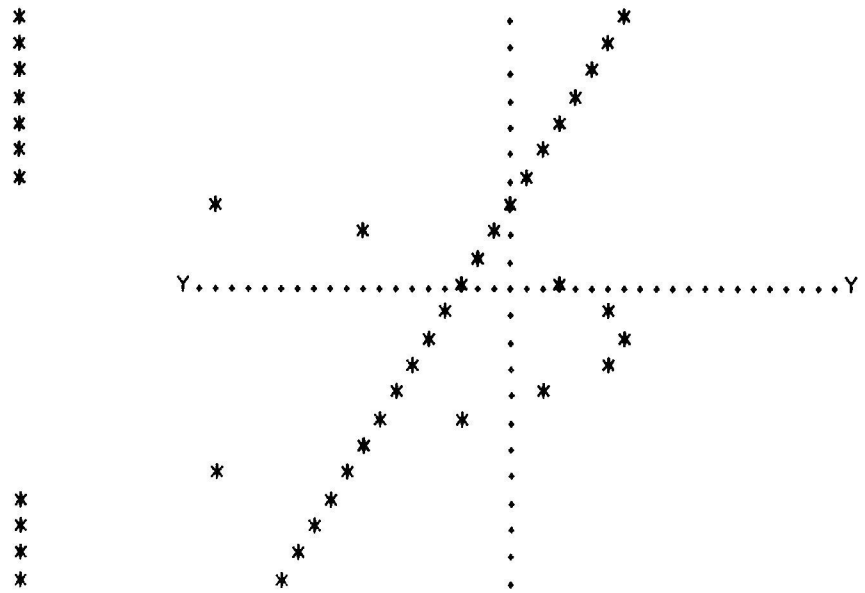
Values are first computed and printed for two monotonic functions. Then these curves are plotted within specified limits. Non-monotonic functions must be plotted using the method of residuals.

```
C FOCAL XVM V1A000
01.01 T %8.04
01.02 ASK "LOWER LIMIT",LL,!"UPPER LIMIT",UL,!"INCREMENT",IN,!
01.10 SET Y1=0; SET Y2=0;
01.20 FOR X=LL,IN,UL; SET Y1=-X-3; SET Y2=3+4*X-X^2; DO 2.0
02.10 IF (Y2-Y1) 2.3,2.2,2.3
02.20 TYPE "THE POINT OF INTERSECTION IS ",!
02.30 TYPE "X1=",X," ","Y1=",Y1,!,"X2=",X," ","Y2=",Y2,!!
03.10 TYPE "DO YOU WANT A PLOT?"
03.20 ASK "(TYPE Y FOR YES, TYPE N FOR NO) ",AN,!!
03.30 IF (AN-0Y)9.1,4.1,9.1
04.10 FOR X=LL,IN,UL; DO 5.0
05.01 IF (X) 5.1,5.02,5.1
05.02 TYPE "           Y.....................................Y",#
05.10 FOR Y=0,30; TYPE " "
05.20 TYPE ".",#
05.30 FOR Y=0,30+(-X-3); TYPE " "
05.40 TYPE "*",#
05.50 FOR Y=0,30+(3+4*X-X^2); TYPE " "
05.60 TYPE "*",!
09.10 QUIT
```

```
*GO
LOWER LIMIT:-10

UPPER LIMIT:10

INCREMENT:1

X1=-   10.0000  Y1=     7.0000
X2=-   10.0000  Y2=-  137.0000

X1=-    9.0000  Y1=     6.0000
X2=-    9.0000  Y2=-  114.0000

X1=-    8.0000  Y1=     5.0000
X2=-    8.0000  Y2=-   93.0000

X1=-    7.0000  Y1=     4.0000
X2=-    7.0000  Y2=-   74.0000

X1=-    6.0000  Y1=     3.0000
X2=-    6.0000  Y2=-   57.0000

X1=-    5.0000  Y1=     2.0000
X2=-    5.0000  Y2=-   42.0000

X1=-    4.0000  Y1=     1.0000
X2=-    4.0000  Y2=-   29.0000

X1=-    3.0000  Y1=     0.0000
X2=-    3.0000  Y2=-   18.0000

X1=-    2.0000  Y1=-    1.0000
X2=-    2.0000  Y2=-    9.0000

THE POINT OF INTERSECTION IS
X1=-    1.0000  Y1=-    2.0000
X2=-    1.0000  Y2=-    2.0000

X1=     0.0000  Y1=-    3.0000
X2=     0.0000  Y2=     3.0000

X1=     1.0000  Y1=-    4.0000
X2=     1.0000  Y2=     6.0000

X1=     2.0000  Y1=-    5.0000
X2=     2.0000  Y2=     7.0000

X1=     3.0000  Y1=-    6.0000
X2=     3.0000  Y2=     6.0000

X1=     4.0000  Y1=-    7.0000
X2=     4.0000  Y2=     3.0000

X1=     5.0000  Y1=-    8.0000
X2=     5.0000  Y2=-    2.0000

X1=     6.0000  Y1=-    9.0000
X2=     6.0000  Y2=-    9.0000

X1=     7.0000  Y1=-   10.0000
X2=     7.0000  Y2=-   18.0000
```

```
X1=     8.0000  Y1=-   11.0000
X2=     8.0000  Y2=-   29.0000

X1=     9.0000  Y1=-   12.0000
X2=     9.0000  Y2=-   42.0000

X1=    10.0000  Y1=-   13.0000
X2=    10.0000  Y2=-   57.0000

X1=    11.0000  Y1=-   13.0000
X2=    11.0000  Y2=-   57.0000
```

```
DO YOU WANT A PLOT?(TYPE Y FOR YES. TYPE N FOR NO) :Y
```

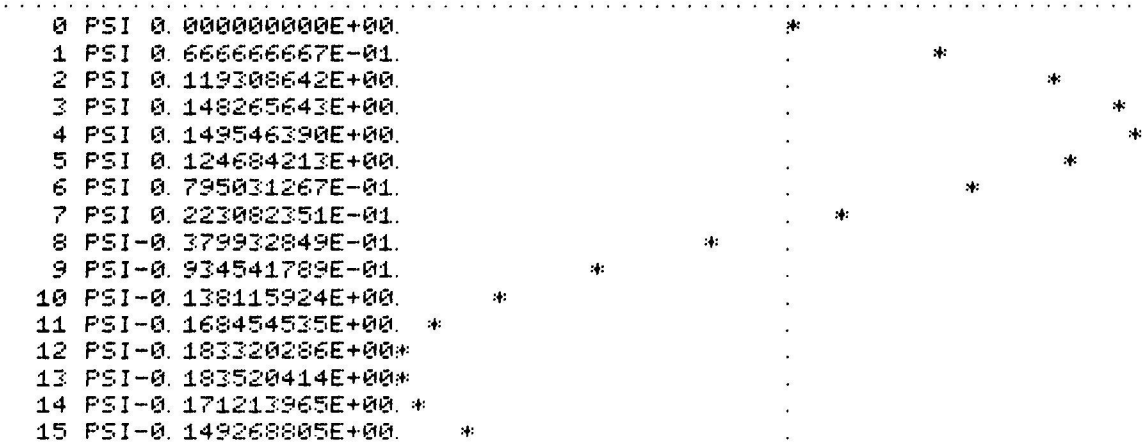


4.8 SCHROEDINGER EQUATION SOLVER

This program is designed to aid the user in searching for possible energy-states of an electron in a potential well. This is one of the most compex equations yet written in FOCAL. It calculates and plots the energy levels of an electron within specified boundary conditions.

```
C FOCAL XVM V1A000
01.01 T !,"SCHROEDINGER EQUATION SOLVER -",!
01.02 T !,"           -DELSQUARED PSI + AX * PSI = E * PSI",!!
01.03 A "TILTED SQUARE WELL PROBLEM WITH WIDTH",X0,!
01.08 A "WELL TILT SLOPE A",A1,!,"TRIAL ENERGY E",B1,!
01.09 A "NUMBER OF STEPS",NT,!
01.11 S VF=0; S SL=1
01.70 S P(0)=0; S DX=X0/NT; S P(1)=SL*DX; S R0=0
01.75 S VF=0
01.80 S P0=0
01.90 F N=0,1,NT-2; D 6
01.93 T !,"PSI ZEROS"%2.0, P0
01.95 GOTO 7.02
```

```
05.10 T !,%3.0, PX," PSI",%,P(PX),","
05.20 S PZ=FITR(PM*SC); S PE=FITR<(P[PX]+PM)*SC>
05.30 F X=1,1,PZ-1; T " "
05.40 T ".",#; F X=1,1,PE+24; T " "
05.50 T "*"; R
06.10 S P(N+2)=<(-B1+A1*DX*[N+1])*DX^2+2>*P(N+1)-P(N)
06.20 I (NT-N-2) 12.90,6.9,6.3
06.30 S RB=P(N+2)*P(N+1); I (RB) 6.4,6.4,6.9
06.40 S P0=P0+1; R
06.90 CONTINUE
07.02 S CF=(P<NT>/P<1>)^2; T "  CONV IND"%, CF
07.05 A "  NEW E?"NY
07.07 I (NY-9) 7.9,7.08,7.9
07.08 I (VF) 7.09,7.8,7.09
07.09 I (CF-100) 7.1,7.1,7.8
07.10 S R2=P(NT)*VF; I (R2) 7.73,7.80,7.85
07.73 S DB=-0.5*DB; GOTO 7.85
07.80 S DB=0.1
07.85 S B1=B1*(1+DB); T B1; S VF(NT); G 1.80
07.90 DO 14; GOTO 12.01
12.01 T !,!,"EIGEN E"B1; S HP=B1/(A1*X0)
12.20 T "  EN/MAX POT"HP,!
12.90 QUIT
14.10 S PM=0; S PP=0; F PX=1,1,NT; D 15
14.20 S PS=PM+PP; S SC=45/PS
14.30 T !!!; F PX=1,1,70; T "."
14.40 F PX=0,1,NT; D 5
14.50 T !;F PX=1,1,70; T "."
14.60 T !!; R
15.10 I (P[PX]) 15.2,15.9,15.5
15.20 I (PM+P<PX>) 15.3,15.4,15.4
15.30 S PM=FABS(P[PX])
15.40 RETURN
15.50 I (P<PX>-PP) 15.9,15.9,15.6
15.60 S PP=P(PX)
15.90 RETURN

*GO

SCHROEDINGER EQUATION SOLVER -

          -DELSQUARED PSI + AX * PSI = E * PSI

TILTED SQUARE WELL PROBLEM WITH WIDTH:1

WELL TILT SLOPE A:40

TRIAL ENERGY E.50

NUMBER OF STEPS:15


PSI ZEROS  1  CONV IND 0.501326462E+01  NEW E?:Y
```



EIGEN E 0.500000000E+02 EN/MAX POT 0.125000000E+01

CHAPTER 5

LIBRARY COMMANDS

FOCAL LIBRARY commands allow the user to save and then call programs by name. These commands cause files consistent with the XVM/DOS file format to be produced and accepted. These files, which use IOPS ASCII data mode, can be manipulated by other XVM/DOS programs such as PIP and EDITOR. In addition to the library commands, a COMMON command and an ERASE COMMON command are available. These commands allow the effective segmentation (chaining) of FOCAL programs, with the COMMON area defining those variables which are to be used by all segments.

5.1 LIBRARY OUTPUT COMMANDS

Three operations are required to produce a file with the FOCAL library commands:

a. File initialization
b. File output
c. File termination

5.1.1 Library File Initialization

The command

```
*
*LIBRARY OUT NAME
*
```

initializes a file on the output device associated with .DAT slot 5. If the device is directoried (i.e., has named files), then the file name NAME is used. NAME can be up to six alphanumeric characters and is terminated by a carriage return. The extension FCL is supplied by the system.

5.1.2 Library File Output Operations

Commands of the form

```
*
*LIBRARY WRITE nnn
*
```

cause information to be entered into the library file. The character string nnn can take four forms which are explained below.

5.1.2.1 Direct Command Output

If the character string nnn, in the example above, begins with quotation marks ("), the command indicates that the character string following the quotation marks is to be inserted into the file. This character string may be any FOCAL command.

For example, the command here is an *, a FOCAL symbol typed by the user to signal termination of input from the device associated with .DAT slot 3.

```
*
*LIBRARY WRITE "*; GO
*
```

causes the command

```
*
*
**;GO
```

to be inserted into the library file as a direct command. This will start the program when the file is later called for execution by a library input command.

5.1.2.2 Single Line Output

If the character string nnn is a legal line number which is present in the program in core, this command causes a single line to be inserted into the file. For example, the command

```
*
*LIBRARY WRITE 10.02
*
```

causes line 10.02 to be inserted into the currently opened output file.

5.1.2.3 Group Output

If the character string nnn is a legal group number, this command causes the entire group of lines to be inserted into the file. For example, the command

```
*
*
*LIBRARY WRITE 2.00
```

causes all group 2 lines to be inserted into the current output file.

5.1.2.4 Program Output

If the character string nnn is ALL or A, then the entire indirect program is inserted into the current output file.

5.1.3 Library File Termination

After using the appropriate library output commands, it is necessary to issue the command

```
*
*
*LIBRARY CLOSE
```

to complete file output and enter the file name into the directory

of the mass storage unit of the system. The LIBRARY CLOSE command allows an input or output file to be closed. An error message "?35" will be printed if a file has not be opened. If the LIBRARY CLOSE command is not issued, the user remains in library mode and all other commands are illegal. However, to leave library mode without actually finishing the output file, the command

```
*
*
*LIBRARY KILL
```

is used. After using this command, the user is in command mode, and the file which had been started by the library output commands is lost.

5.2 LIBRARY INPUT COMMANDS

To load a library file which has been output from FOCAL or which has been prepared off-line, the command

```
*
*
*LIBRARY IN NAME
```

is used where NAME follows the conventions used for library output. The library input from the device associated with .DAT slot 3 is terminated by an end of file or end of tape condition on the input file. It is also terminated by the presence of a direct command of asterisk (*), supplied during a LIBRARY WRITE command within the library file. If none of these three conditions occurs, FOCAL assumes that subsequent input (e.g., for ASK command) will come from the device associated with .DAT slot 3. This direct command can be a multiple command which can provide automatic program starting. For example, a direct command to terminate input and to start a program at line number 8.21 would be

```
*
*
*; GOTO 8.21
```

This command can be inserted at the end of the library output by the command

```
*
*
*LIBRARY WRITE "*; GOTO 8.21
```

5.3 LIBRARY .DAT SLOT USAGE

The FOCAL LIBRARY commands assume input on .DAT slot 3 and output on .DAT slot 5, and the FOCAL DATA commands assume input on .DAT slot 7 and output on .DAT slot 10. The recommended assignment to the Linking Loader is the System Library located on system disk unit 0.

The following table shows a typical set of device assignments where the system software is on disk pack unit 0, input is from DECtape unit 1 and output is to DECtape unit 2.

.DAT Slot	Contents	Sample Assignment
.DAT -1	System Library	DP0
.DAT -4	FOCAL binary program plus FNEW binary	DP0
.DAT 3	FOCAL library input	DT1
.DAT 5	FOCAL library output	DT2
.DAT 7	FOCAL data input	DT1
.DAT 10	FOCAL data output	DT2

FOCAL data commands are described in Chapter 7, and .DAT slot assignments are again summarized in Appendix E.

5.4 COMMON VARIABLES AND ARRAYS

The COMMON command allows the user to define permanent FOCAL variables and arrays which are saved with their current values when the user modifies the stored indirect program. Thus, the caution at the MODIFY command (refer to Section 3.14) does not apply to COMMON variables. The COMMON command is legal only if no other variables have been defined in the symbol table. Thus, it is good programming practice to precede a COMMON definition by an ERASE command to clear the symbol table. If any non-COMMON variables have been defined when the COMMON command is executed, it will be treated as a COMMENT and ignored.

5.4.1 COMMON Format

Three types of variables can be defined in a COMMON command as the following example shows.

```
*
*
*COMMON A,B(5),(C,1,1,3)
```

The first variable, A, defines a single non-subscripted variable. The second variable, B(5), defines a single array element to be COMMON. All other elements of the array B are non-COMMON. The third variable (C.1.1.3) defines a series of array elements as COMMON by using a

notation similar to the FOR statement. Thus, the command

```
*
*COMMON (C,1,1,5)
*
```

is equivalent to the command

```
*
*
*COMMON C(1),C(2),C(3),C(4),C(5)
```

but much shorter. Note that, because COMMON and COMMENT both have the initial letter C, the COMMON command must not be abbreviated.

5.4.2 ERASE COMMON Command

The ERASE COMMON command must be used to clear the COMMON area if a user wishes to define a completely new COMMON area. It will clear both the COMMON area and all variables in the symbol table, but not the program itself. If, however, the user simply wants to add to the current COMMON area, it is only necessary to erase any non-COMMON variables by using ERASE. The ERASE ALL command has no effect on the COMMON area variables and does not change their values.

5.5 CHAINING OF FOCAL PROGRAMS

For FOCAL programs that exceed the capacity of user's core memory, it is possible to segment the program. By combining the library input and COMMON commands, one segment can call another by name. ALL COMMON variables are retained in core memory for access to them from all segments of the program. New COMMON variables can be added to the permanent table from any segment. The command LIBRARY IN NAME brings in the next segment to core memory and provides access to the COMMON table for the operations in this segment of the program.

NOTE

> Ensure that a segment does not exceed its alloted memory location or the overflow will begin to erase the compiler core.

The following example shows three FOCAL segments and the operations required to provide the segments with linkage capability. The first segment, named CH1, defines a COMMON area and initializes the variables. If CH1 is called again, it checks for the values in the second part of the COMMON area which were defined in CH2 and QUITS. The second-segment, CH2 defines more COMMON variables, checks the original COMMON values created in CH1, and initializes the additional COMMON variables. The third segment checks all the COMMON values and calls the first segment again.

```
*WRITE

C FOCAL XVM V1A000
01.05 TYPE "CH1 READY",!
01.10 COMMON A,B,C,(ABC,1,5)
01.20 IF (A) 20.1,1.3,20.1
01.30 SET A=1
01.40 SET B=2; SET C=3
01.50 FOR X=1,5; SET ABC(X)=X*10
01.55 TYPE "CH1 DONE  -  CALLING CH2",!
01.60 LIBRARY IN CH2
20.10 IF (0-10) 20.2,20.3,20.2
20.20 TYPE "COMMON ERROR ON 2ND CALL OF CH1",!
20.30 TYPE "ALL DONE",!; QUIT
```

Listing of CH1

```
*LIBRARY OUT CH1
*LIBRARY WRITE "ERASE ALL
*LIBRARY WRITE ALL
*LIBRARY WRITE "*;  GO
*LIBRARY CLOSE
*
```

Library Output of CH1

```
*WRITE ALL

C FOCAL XVM V1A000
01.10 TYPE "CH2 READY",!
01.20 COMMON Q,(ABC,5,1,10)
01.30 FOR X=6,10; S ABC(X)=X*10
01.40 FOR X=1,10; DO 25.0
01.50 IF (A+B+C-6) 1.6,1.7,1.6
01.60 TYPE "CH2 COMMON ERROR -  SUM = ",A+B+C,!
01.70 TYPE "CH2 DONE  -  CALLING CH3",!
01.80 LIBRARY IN CH3
25.10 IF (ABC(X)-10*X) 25.2,25.3,25.2
25.20 TYPE "ERROR AT X = ",X," ABC ARRAY = ",ABC(X),!
25.30 RETURN
```

Listing of CH2

```
*LIBRARY OUT CH2
*LIBRARY WRITE "ERASE ALL
*LIBRARY WRITE ALL
*LIBRARY WRITE "*; GO
*LIBRARY CLOSE
*
```

Library Output of CH2

```
*WRITE ALL

C FOCAL XVM V1A000
01.10 TYPE "CH3 READY",!
01.20 FOR X=1,10; DO 25.0
01.30 SET Q=10
01.40 TYPE "CH3 DONE  -  CALLING CH1 AGAIN",!
01.50 LIBRARY IN CH1
25.10 IF (ABC(X)-10*X) 25.2,25.3,25.2
25.20 TYPE "ERROR AT X = ",X," ABC ARRAY = ",ABC(X),!
25.30 RETURN
```

Listing of CH3

```
*LIBRARY OUT CH3
*LIBRARY WRITE "ERASE 1.00
*LIBRARY WRITE 1.00
*LIBRARY WRITE "*; GO
*LIBRARY CLOSE
*
```

Library Output of CH3

```
BFOCAL XVM V1A000
*ERASE COMMON
*LIBRARY IN CH1

CH1 READY
CH1 DONE  -  CALLING CH2
CH2 READY
CH2 DONE  -  CALLING CH3
CH3 READY
CH3 DONE  -  CALLING CH1 AGAIN
CH1 READY
ALL DONE
*TYPE $
A@@(00)=     1.0000
B@@(00)=     2.0000
C@@(00)=     3.0000
ABC(01)=    10.0000
ABC(02)=    20.0000
ABC(03)=    30.0000
ABC(04)=    40.0000
ABC(05)=    50.0000
Q@@@(00)=    10.0000
ABC(06)=    60.0000
ABC(07)=    70.0000
ABC(08)=    80.0000
ABC(09)=    90.0000
ABC(10)=   100.0000
-
*
```

Library Input and Execution of Chained Programs CH1, CH2 and CH3

Dump of COMMON Symbols

CHAPTER 6
USER DEFINED FOCAL FUNCTIONS

The ability to write FOCAL functions in MACRO assembly language and subsequently interface these functions with the FOCAL interpreter is an important feature which allows real-time use of FOCAL. These functions are processed in the same way as the normal internal functions supplied with the interpreter (i.e., FSIN,FITR,etc.). Some external functions are provided in the FNEW file (Refer to Section 6.2). User-defined functions can be incorporated into the source file of FNEW and then reassembled by MACRO. The process is described in the XVM/DOS System Installation Guide. The interface of external functions is accomplished by:

a. Use of a function table which contains the three letter function name in .SIXBT (6-bit ASCII) and a jump to the function processor.

b. Use of .GLOBL definitions which allow the function processors to use character processing and expression evaluation routines which are in the interpreter.

The following detailed example shows the operations necessary to write an external FOCAL function in MACRO assembler language and to merge it into the external function file, FNEW, described in Section 6.2.

6.1 EXAMPLE

A scope routine has been generated to display characters at a given point on a scope. This routine is called from FOCAL as a function by XYC (X,Y,SHOW). Here, X and Y are expressions to be used as display coordinates for the start of SHOW.

First, the function name and transfer instruction must be added to the .NEWF function table within FNEW. Refer to page 6-6.

```
        .SIXBT      /XYC/
        JMP         SETXYC
```

When control arrives at SETXYC, the X has already been evaluated.

```
SETXYC  JMS*        .AX         /make 18 bits
        DXL                     /set X coordinate
```

Now, check for the second argument and give an error if no second argument is given.

```
        JMS*            XSPNOR      /skip spaces
        SAD             (254        /is it a comma?
        JMP             +3          /yes
        .DEC
        LAW             1           /?01 error
        .OCT
        JMP*            FUNERR      /function error
```

Move past the separating comma.

```
        JMS*            UTRA
```

Evaluate the second argument.

```
        LAC             EVAL        /address of evaluation
                                       routine
        DAC             +2
        JMS*            XPUSHJ
        XX
```

The second argument must be made into an 18-bit quantity and the y coordinate set:

```
        JMS*            .AX
        DYL
```

Test for a comma; if present, bypass it to get to the character string.

```
        JMS*            XSPNOR      /skip spaces
        SAD             (254        /is it comma?
        JMP             +3          /yes
        .DEC
        LAW             1           /?01 error
        .OCT
        JMP*            FUNERR      /error return
        JMS*            UTRA        /skip comma
```

Now, pick up the single ASCII characters and display them. (This example assumes the character display routine is called DYCHAR.):

```
DCLOOP  LAC*            CHAR        /get character
        JMS             DYCHAR      /display it
        JMS*            UTRA        /get next char
        SAD             (251        /is it end?
        JMP*            EFUN3       /yes-return
        JMP             DCLOOP      /no-go display next
                                       character
```

6.2 FILE FNEW

The following functions are supplied in FNEW (where N cannot be another function):

FUNCTION	MEANING
FDXS(N)	Set the x coordinate of the 34H display
FDIS(M)	Set the y coordinate of the 34H display and itensify the point
FDXY(N,M)	Set the x and y coordinates and intensify.
FADC(N)	Set the multiplexer of the A/D converter and perform one conversion
FDAC(N)	Set the D/A converter to the specified value.
FRLB(N,M)	If N=-1, clear the relay buffer (M should not be specified) If N=0 to 17, set the appropriate bit of relay buffer according to the value of M(0 or 1).
FRND(N)	Start a sequence of random numbers in the range of -1.0 less than or equal N less than or equal +1.0 based on the value of N. The value N may be any valid arithmetic expression. FRND always generates the same number for the same value of N. If N is either 0 or ␣ (a space), a random number will be generated based upon the previously generated number. The first time FRND is called, if with a 0 or ␣ argument, it produces a value of 1.0000.

The following points should be noted:

1. Other names for the 34H display are the RM503 Scope or the VP15B Scope.

2. The proper way to use these functions is as follows:

   SET J = FDXY (N,M)

   (What J will be set to is immaterial, and other letters may be used in place of J.) You cannot use:

   FDXY (N,M)

3. The 34H display is a refresh-type scope. This fact, combined with the time it takes to interpret and execute the graphics functions, limits them to a maximum of 75-100 points on the screen, before the first point put out begins to diappear. Hence, they are suitable for putting out graphs consisting of a set of points. They are not suitable for putting out "continuous line" drawings, in which many points spaced close together look like a continuous line.

User Defined Focal Functions

The functions are supplied to provide the user with additional examples in the method of coding external functions.

The listing that follows is representative of the source code in the file FNEW is supplied by DEC but may not exactly match what is available.

```
        .TITLE  FNEW
/
/FOCAL EXTERNAL FUNCTION PROGRAM
/
/COPYRIGHT 1969,1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS. 01754
/
/DAVE LENEY
/2-7-69
/
/EDIT 004
/
        .GLOBL  .NEWT           /FUNCTION TABLE
        .GLOBL  XPUSHJ          /PUSH JUMP
        .GLOBL  XPUSHA          /PUSH AC
        .GLOBL  PD2             /PUSH FP
        .GLOBL  PD3             /POP FP
        .GLOBL  UTRA            /GET CHAR
        .GLOBL  XSPNOR          /SKIP SPACE & ZERO
        .GLOBL  FUNERR          /FUNCTION ERROR
        .GLOBL  EFUN3           /FUNCTION RETURN
        .GLOBL  FINT            /FLOATING AA:=N
        .GLOBL  CHAR            /CHAR STORAGE
        .GLOBL  EVAL            /EXPRESSION EVALUATOR
        .GLOBL  .AA             /EXPONENT (2'S COMP)
        .GLOBL  .AB             /SIGN AND HIGH ORDER MANTISSA
        .GLOBL  .AC             /LOW ORDER MANTISSA
        .GLOBL  .BA             /ALGATE
        .GLOBL  .AX             /FIX
        .GLOBL  .AW             /F.OAT
FFOR=000000
FADD=100000
FSUB=200000
FMUL=300000
FDIV=400000
FSET=500000
FPUT=600000
FNOR=700000
FXIT=000000
/
        .EJECT
/
/FOCAL OPERATIONS (FOCAL SOURCE EQUIVALENT IS PAGE..)
/
/PUSH JUMP TO EXPRESSION EVALUATOR (PUSHJ EVAL)
/       LAC     EVAL
/       DAC     .+2
/       JMS*    XPUSHJ
/       XX
/
/PUSH THE HARDWARE ACCUMULATOR (PUSHA)
/       JMS*    XPUSHA
/
/ POP THE HARDWARE ACCUMULATOR (POPA)
/       LAC*    .3
```

```
/
/PUSH FLOATING ACCUMULATOR (PUSHF.AA)
/       LAC     .AA
/       DAC     .+2
/       JMS*    FD2
/       XX
/POP FLOATING ACCUMULATOR (POPF .AA)
/       LAC     .AA
/       DAC     .+2
/       JMS*    FD3
/       XX
/
/PUSH FLOATING VARIABLE (PUSHF VAR)
/       JMS*    FD2
/       .DSA    VAR             /WHERE VAR IS THE FIRST OF
/                               /THREE(3) REGISTER BLOCK
/
/POP FLOATING VARIABE (POPF VAR)
/       JMS*    FD3
/       .DSA    VAR             /SAME AS FOR PUSHF VAR
/
/FETCH CURRENT CHARACTER
/       LAC*    CHAR
/
/FETCH NEXT CHARACTER (GETC)
/       JMS*    UTRA            /CHARACTER IN BOTH CHAR AND AC
/
/IGNORE LEADING SPACES AND ZEROS (SPNOR)
/       JMS*    XSPNOR          /NEXT CHARACTER IN BOTH CHAR AND AC
/
/INDICATE A FUNCTION ERROR (ERROR NN)
/       .DEC
/       LAW     NN              /NN=TWO DIGIT (DECIMAL) ERROR CODE
/       .OCT
/       JMP*    FUNERR          /ERROR WILL HAVE DOUBLE QUESTION MARK
/
/RETURN FROM FUNCTION (RETURN)
/       JMP*    EFUN3           /VALUE OF FUNCTION IN FLOATING AC
/
/ENTER FLOATING POINT INTERPRETER (JMS FINT)
/       JMS*    FINT            /FROM HERE ON ALL INSTRUCTIONS
/       FNN     A               /ARE ASSUMED TO BE FLOATING POINT
/       FNN*    B               /UNTIL FXIT IS REACHED - FNN
/       FNN     C               /REFERS TO FLOATING POINT OPERATIONS
/       FXIT                    /LISTED NEXT.
/
/FLOATING POINT OPERATIONS (USE * FOR INDIRECT)
/       FPOW    VAR             /RAISE F.P. AC TO VAR/CONSTANT
/       FADD    VAR             /ADD F.P. VARIABLE/CONSTANT TO F.P. AC
/       FSUB    VAR             /SUBTRACT FROM F.P. AC
/       FMUL    VAR             /MULTIPLY THE F.P. AC
/       FDIV    VAR             /DIVIDE INTO THE F.P. AC
/       FGET    VAR             /LOAD INTO THE F.P. AC
/       FPUT    VAR             /STORE THE F.P. AC
/       FNOR                    /NORMALIZE THE F.P. AC
/       FXIT                    /EXIT FROM FLOATING POINT MODE
/
/NEGATE THE FLOATING ACCUMULATOR
/       JMS*    .BA
/
/FIX THE FLOATING ACCUMULATOR INTO THE HARDWARE ACCUMULATOR
/       JMS*    .AX
```

```
/
/FLOAT THE HARDWARE ACCUMULATOR INTO THE FLOATING ACCUMULATOR
/       JMS*    .AW
/
/
/EXTERNAL FUNCTION TABLE FORMAT
/       WORD 0: TWO'S COMPLEMENT COUNT OF NUMBER OF ENTRIES
/       WORDS 1 TO 2N: N TWO WORD ENTRIES
/               EACH ENTRY:     WORD0: .SIXBT THREE LETTER FUNCTION NAME
/                               WORD1: JMP TO FUNCTION ADDRESS
/
/
        .EJECT
/
/TABLE OF NAMES OF EXTERNAL FUNCTIONS
/
.NEWF   .NEWFE-.-1/2\777777+1   /TWO'S COMP COUNT OF NAMES
        .SIXBT  /DXS/
        JMP     SETX            /SET X COORDINATE
        .SIXBT  /DIS/
        JMP     SETYI           /SET Y COORDINATE AND INTENSIFY
        .SIXBT  /DXY/
        JMP     SETXYI          /SET X AND Y AND INTENSIFY
        .SIXBT  /ADC/
        JMP     SETADC          /SET MULTIPLEXER AND CONVERT
        .SIXBT  /DAC/
        JMP     SETDAC          /LOAD D/A CONVERTER
        .SIXBT  /RLB/
        JMP     SETRLB          /SET RELAY BUFFER
        .SIXBT  /RND/
        JMP     RAND            /RANDOM NUMBER ROUTINE
.NEWFE=.
/
/IOT DEFINITIONS
/
DYL=700606                      /LOAD Y COORDINATE
DXS=700546                      /LOAD X COORDINATE AND INTENSIFY
ADSF=701301                     /SKIP ON A/D FLAG
ADSC=701304                     /SELECT AND CONVERT
ADRB=701312                     /READ A/D BUFFER
ADSM=701103                     /SET MULTIPLEXER
DAL1=705501                     /LOAD D/A CHANNEL ONE
ORC=702101                      /CLEAR RELAY BUFFER
ORS=702104                      /SET RELAY BUFFER
/
        .EJECT
SETX    JMS*    .AX             /SET X COORDINATE
        DAC     XCOORD          /SAVE
        JMP*    EFUN3           /RETURN
SETYI   JMS*    .AX             /SET Y COORDINATE
        DYL                     /LOAD REG WITH Y
        LAC     XCOORD
        DXS                     /LOAD WITH X AND INTENSIFY
        JMP*    EFUN3           /RETURN
SETXYI  JMS*    .AX             /SET X COORDINATE
        DAC     XCOORD          /SAVE
        JMS     GETARG
        JMP     SETYI           /NOW SAME AS FDIS
SETADC  JMS*    .AX             /GET MULTIPLEXER CHANNEL
        ADSM                    /NOTE: WILL USE LOW ORDER
                                /SIX BITS AS CHANNEL NUMBER
        IOF                     /TURN OFF INTERRUPT SYSTEM
        ISA+10
        ADSC                    /SELECT AND CONVERT
        ADSF                    /WAIT FOR FLAG
```

```
          JMP     .-1
          ADRB                        /READ A/D BUFFER
////CODE REMOVED WHICH PROPAGATED SIGN BIT ASSUM. 12 BIT A/D
///WAD AUGUST 1969...
          JMS*    .AW                 /FLOAT RESULT
          LAC     (400000             /TURN ON INTEVVUPT SYSTEM
          ISA
          ION
          JMP*    EFUN3               /RETURN
SETDAC    JMS*    .AX                 /GET VALUE AS INTEGER
          DAL1
          JMP*    EFUN3
SETRLB    JMS*    .AX                 /GET VALUE
          SMA                         /IS IT MINUS
          JMP     .+4                 /NO - GO FIND BIT POSITION
          DZM     RELAYB
          ORC                         /YES - CLEAR WHOLE RELAY BUFFER
          JMP*    EFUN3               /RETURN
          CMA                         /FORM ONE'S COMP
          DAC     TEMP                /SAVE AS COUNT
          .DEC
          TAD     (18                 /MUST BE RELAY 0 TO 17
          .OCT
          SMA                         /IS IT LEGAL POSITION
          JMP     .+3                 /YES
          .DEC
          LAW     2                   /NO
          .OCT
          JMP*    FUNERR              /??02 ERROR
          CLA!CLL!CML                         /INITIALIZE AC
          RAR
          ISZ     TEMP                /PUT BIT IN RIGHT
          JMP     .-2                 /POSITION
          DAC     TEMP                /SAVE BIT POSITION
          JMS     GETARG
          LAC     TEMP                /CLEAR OLD RELAY VALUE
          CMA
          AND     RELAYB
          DAC     RELAYB              /SAVE NEW BUFFER
          JMS*    .AX
          SZA!CLA                     /CLEAR OR SET RELAY
          LAC     TEMP                /IF NON ZERO SET
          XOR     RELAYB
          DAC     RELAYB
          ORS                         /LOAD BUFFER
          JMP*    EFUN3               /RETURN
/
GETARG    0                           /GET ARGUMENT
          JMS*    XSPNOR              /SKIP SPACES
          SAD     (254                /IS IT A COMMA?
          JMP     .+3                 /YES
          .DEC
          LAW     1                   /NO - ??01 ERROR
          .OCT
          JMP*    FUNERR              /EXTERNAL FUNCTION ERROR
          JMS*    UTRA                /SKIP COMMA
          LAC     EVAL
          DAC     .+2
          JMS*    XPUSHJ              /CALL EXPRESSION EVALUATOR
          XX
          JMP*    GETARG              /RETURN
/
XCOORD    0                           /X COORDINATE STORAGE
RELAYB    0                           /RELAY BUFFER STORAGE
TEMP      0                           /TEMPORARY STORAGE
/
```

```
/ROUTINE TO GENERATE A RANDOM NUMBER     //DEC. 1970
/NUMBER SEQUENCE CAN BE INITIALIZED
/BY HAVING A NON-ZERO FUNCTION ARGUMENT
/
/USES A SIMULATED 18BIT SHIFT REGESTER
/WHICH IS SHIFTED 1 BIT RIGHT WITH FEEDBACK
/FROM THE 7'TH&18'TH STAGES TO THE 1'ST STAGE
/THROUGH AN EXCLUSIVE OR GATE
/
RAND    JMS*    .AX             /GET ARGUMENT IN AC
        SZA                     /IS IT 0
        JMP     STGEN           /NO  GENERATE RANDOM NUMBER
        LAC     RANPT           /YES  IS LAST RANPT 0
        SNA
        LAW     -1                      /YES  INITIALIZE RANPT
STGEN   DAC     RANPT           /START TO GENERATE
        RCR                     /NEW RANDOM NUMBER
        DAC     RANPT           /BY SHIFTING ONE BIT RIGHT
        GLK                     /GET BIT17 AND STORE (STAGE18)
        DAC     BITSAV
        LAC     RANPT
        RTR                     /GET BIT6 (STAGE 7)
        RTR                     /SETUP TO FEEDBACK TO BIT0
        RTR                     /(STAGE 1) THROUGH XOR GATE
        RTR
        RTR
        AND     (1
        XOR     BITSAV
        SZA
        TAD     (377777         /FEEDBACK A ONE
        TAD     RANPT           /MAKEUP NEW RANDOM NUMBER
        DAC     RANPT           /STORE FOR FUTURE USE
        DAC*    .AB             /SET SIGN AND HIGH ORDER MANTISSA
        DZM*    .AA             /SET EXPONENT=0
        DZM*    .AC             /SET LOW ORDER MANTISSA=0
        JMP*    EFUN3           /EXIT TO FUNCTION RETURN
BITSAV  0
RANPT   0
/
        .END
```

CHAPTER 7
DATA COMMANDS

FOCAL data commands allow the user to store and then retrieve substantial amounts of data through the use of auxiliary Input/Output devices other than the Teletype. The steps for processing the data need not be incorporated in the FOCAL program itself. The commands cause files consistent with the XVM/DOS format to be produced and accepted; and as with the library commands, the files can be manipulated by other XVM/DOS programs such as PIP and EDITOR, as may be noted from the examples at the end of this chapter.

Data commands are used with other FOCAL commands and follow the same conventions with only minor exceptions. The DATA command, because of the conflict with the DO command, cannot be abbreviated. Also, under some conditions, library commands are illegal as with DATA OUT or DATA CLOSE (see 7.1.1 and 7.1.2).

7.1 DATA COMMANDS

Three operations are required to produce a file with the FOCAL data commands:

a. file initialization
b. file output
c. file termination

7.1.1 DATA File Initialization and Output

The command

```
*DATA OUT NAME
```

initializes and enters a filename on the device associated with .DAT slot 10. The name can be up to six alphanumeric characters and is terminated by a carriage return. The extension FCL is supplied by the system.

Every TYPE or WRITE command issued after DATA OUT NAME will output data to the device assigned to .DAT slot 10. For example with the following .DAT slots assigned:

```
$A DPO -1,-4/DT1 3,5,7,10
```

data is output to .DAT 10 on DECtape unit 1, until a DATA file termination command is given.

If a LIBRARY command is issued while in the DATA OUT mode, error message "?31" will be output to the teletype. What has already been output to the DATA file can then only be saved if a DATA CLOSE command is issued.

7.1.2 DATA File Termination

After the appropriate DATA output commands are used, it is necessary to issue the command

```
*DATA CLOSE
```

to complete file output and enter the filename and data into the device associated with .DAT slot 10. DATA CLOSE commands allow input or output files to be closed. If a file has not been opened, FOCAL will output the "?35" error message on the teletype. The DATA CLOSE command also returns the user to the teletype mode.

If the command is not issued, the user remains in the data mode. However, to leave the data mode without finishing the output file the user may type

```
*DATA KILL
```

which aborts the output file and returns the user to the teletype mode. The file started by the DATA output command is lost when DATA KILL is issued.

7.1.3 DATA Input

The command

```
*DATA IN NAME
```

function is related to its use in indirect programs. When DATA IN NAME is used in an indirect program (e.g., 1.10 DATA IN filename), it initializes the device associated with .DAT slot 7 for data input from an ASK command. Recall that the ASK command is normally used in indirect commands and that its use is to input data at specific points during the execution of an indirect program. Thus, when a line number and a DATA IN command such as

```
*1.10 DATA IN filename
```

is inserted in a program, .DAT slot 7 is initialized for data input when an ASK command such as

```
*2.10 ASK X,Y,Z
```

is encountered during program execution.

7.2 DATA .DAT SLOT USAGE

Data commands, as previously stated, assume input on .DAT slot 7 and output on .DAT slot 10. The recommended assignment to the Linking Loader is the system library located on unit 0. Recommended FOCAL

program and user input/output assignments for DECtape and Disk are:

.DAT Slot	Contents	Sample Assignment
.DAT -1	System Library	DP0
.DAT -4	FOCAL binary program plus FNEW binary	DP0
.DAT 3	FOCAL library input	DT0
.DAT 5	FOCAL library output	DT0
.DAT 7	FOCAL DATA input	DT0
.DAT 10	FOCAL DATA output	DT0

7.3 DATA COMMAND USE

Some of the data commands so far described are used in the following examples. Also demonstrated are the commands for loading FOCAL with the Linking Loader and for use of PIP for a Directory listing and output of data on the teletype.

```
XVM/DOS Vnxnnn
$A DP1 -4,3,5,7,10          /Prior..DAT slot assignment
$GLOAD
LOADER XVM  Vnxnnn
  FOCAL
FOCAL XVM Vnxnnn
*1.10 DATA IN INDISK        /Type indirect program
*1.20 ASK A,B,C,D,E         /File INDISK contains the input
                                Data.
*1.30T A+B+C+D+E,!          /See PIP listing of INDISK below.
*GO
 15,0000
*LO SHOW                    /Save indirect program.
*LWA
*LC
*LI Show                    /Recall indirect program
*WA

C FOCAL XVM Vnxnnn
01.10 DATA IN INDISK        /List program on the teletype.
01.20 ASK,A,B,C,D,E
01.30 T A+B+C+D+E,!

*GO
 15,0000

*TA
 1.0000*
*TB,!
 2,0000
```

```
*DATA OUT OUTDT              /Output data onto DECtape using
*TA                          /the TYPE command
*TB
*TC
*TD
*TE,!
*DATA CLOSE                  /Close the output file.
* C
XVM/DOS Vnxnnn

SPIP
XVM Vnxnnn                   /Examination of files created
                                 and input

PIP XVM Vnxnnn
 LTT  DT1

DIRECTORY LISTING
 1042 FREE BLKS
    4 USER FILES
   10      SYSTEM BLKS
INDISK     FCL   1   1
FOCAL      BIN   2  23
SHOW       FCL   3   1
OUTDT      FCL   4   1

 T TT  DT1 INDISK FCL(A)

 1.0
 2.0
 3.0
 4.0
 5.0

C FOCAL XVM Vnxnnn
01.10 DATA IN INDISK
01.20 ASK A,B,C,D,E
01.30 T A+B+C+D+E.!

 T TT  DT1 OUTDT FCL(A)

     1.0000      2.0000      3.0000      4.0000      5.0000

   C

XVM/DOS Vnxnnn

$
```

APPENDIX A

FOCAL COMMAND SUMMARY

Command	Abbreviation	Example of Form	Explanation
ASK	A	ASK X,Y,Z	FOCAL types a colon for each variable, user then types a value to define each variable.
COMMENT	C	COMMENT	If a line begins with the letter C, but not COMMON, the remainder of the line is ignored during program execution.
COMMON	none	COMMON A,B, (C,1,2,20)	Assigns COMMON variables to be stored in indirect program.
CONTINUE	C	C	Dummy lines.
DATA	none	DATA OUT NAME	Initializes the device assigned to .DAT 10 and, if file-oriented, enters the filename in the file directory.
		DATA CLOSE NAME	Closes the output file and returns the user to the teletype mode.
		DATA KILL	Aborts the output file and returns the user to the teletype mode.
		DATA IN NAME	Initializes the device assigned to .DAT 7 and reads in the file named.
DO	D	DO 4.1	Execute line 4.1, return to command following DO command.
		DO 4.0	Execute all group 4 lines, return to command following DO command, or when a RETURN is encountered.
		DO ALL	Execute entire indirect program
ERASE	E	ERASE	Erases user's entire symbol table [1]
		ERASE 2.0	Erases all lines in group 2 [1]
		ERASE 2.1	Erases line 2.1 [1]
		ERASE ALL	Deletes all user input [1]
		ERASE COMMON	Deletes all COMMON variables

[1] Has no effect

Command	Abbre-viation	Example of Form	Explanation
FOR	F	FOR I=X,Y,X, (commands)	Where the command following is executed at each new value. X=initial value of I. Y = value added to I until I is greater than Z. Y = 1, if not defined.
GO	G	GO	Starts indirect program at lowest numbered line number.
GO?	G?	GO?	Starts at lowest numbered line number and traces entire indirect program until another question mark (?) or an error is encountered, or until completion of program.
GOTO	G	GOTO 3.4	Starts indirect program (transfers control to line 3.4); must have argument.

IF	I	IF (X)Ln,Ln,Ln
		
		IF (X)Ln,Ln; (commands)
		
		IF (X)Ln; (commands)
			Where X is a defined identifier, a value or an expression, followed by three numbers/commands.
			
			If X is less than zero, control is transferred to the first line number.
			
			If X is equal to zero, control is transferred to the second line number or command.
			
			If X is greater than zero, control is transferred to the third line number or command.

LIBRARY	L	LIBRARY OUT NAME	Initializes a file on the output device.
		LIBRARY WRITE NNN	Inserts NNN in library output file.
		LIBRARY WRITE 2.01	Inserts line 2.01 in output file.
		LIBRARY WRITE 2.00	Inserts group 2 lines in library output file.
		LIBRARY WRITE ALL	Inserts entire indirect program in library output file.

Command	Abbreviation	Example of Form	Explanation
		LIBRARY CLOSE	Causes file name to be entered in directory.
		LIBRARY KILL	Returns user to command mode & file is deleted.
		LIBRARY IN NAME	Loads library file NAME.
MODIFY	M	MODIFY 1.15	Enables editing of any character on line 1.15 (see below).
QUIT	Q	QUIT	Returns control to the user.
RETURN	R	RETURN	Terminates DO subroutines, returning to the original sequence.
SET	S	SET A=5/B*C	Defines identifiers in the symbol table. Each occurrence of A is replaced by the value of the expression.
TYPE	T	TYPE A+B - C	Evaluates expression and types out result in current output format.
		TYPE A-B,C/E	Computes and types each expression separated by commas.
		TYPE "TEXT STRING"	Types text, can be followed by exclamation point (!) to generate carriage return-line feed or by # to generate carriage return only.
WRITE	W	WRITE WRITE ALL	FOCAL types out the entire indirect program.
		WRITE 1.0	FOCAL types out all group 1 lines.
		WRITE 1.1	FOCAL types out line 1.1

FOCAL Operations

Operation	Command	Explanation
To set output format	TYPE% x.yz	Where x is the total number of digits, and yz is the number of digits to the right of the decimal point.
	TYPE% 6.3, 123.456	FOCAL types: 123.456
	TYPE %	Resets output format to floating point.

Operation	Command	Explanation
To type symbol table	TYPE $	Other statements may not follow on this line.
To produce carriage return-line feed	!	
Carriage return only	#	

Modify Operations

After a MODIFY command, the user types a search character, and FOCAL types out the contents of that line until the search character is typed. The user can then perform any of the following optional operations.

a. Type in new characters. FOCAL adds these to the line at the point of insertion.

b. Type an ALT MODE. FOCAL proceeds to the next occurrence of the search character.

c. Type a CTRL BELL. After this, the user can change the search character.

d. Type RUBOUT. This deletes characters to the left; one character for each time the user strikes the RUBOUT key.

e. Type CTRL U. Deletes the line over to the left margin, but not the line number.

f. Type RETURN. Terminates the line, deleting characters over to the right margin.

g. Type LINE FEED. Saves the remainder of the line from the point at which LINE FEED is typed over to the right margin.

Summary of Internal Functions

Function	FOCAL Representation	Operation
Square Root	FSQT(x)	Focal computes $\sqrt{X}$ where x is a positive number or expression greater than zero.
Absolute Value	FABS(x)	FOCAL ignores the sign of x.
Sign Part	FSGN(x)	FOCAL evaluates the sign part only with 1 as integer.
Integer Part	FITR(x)	FOCAL operates on the integer part of x, ignoring any fractional part.
Random Number Generator	FRAN ( )	FOCAL generates a random number.
Exponential Function (e^X)	FEXP(x)	FOCAL generates e to the power x. (2.718281828^X).

Function	FOCAL Representation	Operation
Sine	FSIN(x)	FOCAL generates the sine of angle x expressed in radians.
Cosin	FCOS(x)	FOCAL generates the cosine of angle x expressed in radians.
Arc Tangent	FATN(x)	FOCAL generates the arc tangent of angle x expressed in radians.
Logarithm	FLOG(x)	FOCAL generates $\log_e(x)$.

APPENDIX B

ERROR DIAGNOSTICS

CODE	MEANING
?00	Function not implemented
?01	Illegal character at beginning of line
?02	Group number illegal as line number
?03	Group number too large
?04	Illegal type/ask format
?05	Too many periods
?06	Line number too large
?07	Line number missing
?08	Illegal group number
?09	Push-down list overflow
?10	Illegal command
?11	Illegal IF format
?12	Left of equals in error on FOR or SET
?13	Excess right parenthesis
?14	Illegal FOR format
?15	Illegal variable name
?16	Text/variable buffer overflow
?17	Illegal expression format
?18	Operator missing before parenthesis
?19	Missing left parenthesis
?20	Illegal function name
?21	Double operator
?22	Parenthesis error
?23	ERASE or WRITE argument error
?24	Negative line number
?25	Zero argument for log
?26	Input overflow
?27	Number too large
?28	Negative power illegal
?29	Division by zero illegal
?30	Square root of a negative number
?31	Illegal command during library or data output
?32	Illegal library command
?33	Illegal file name
?34	File not found
?35	No file open
?36	.OTS error from arithmetic package
?37	COMMON statement format error
??nn	User defined function error

APPENDIX C
ESTIMATING THE LENGTH OF USER PROGRAMS

FOCAL requires five words for each identifier stored in the symbol table, and one word for each three characters of stored program. This may be calculated by

$$5s+\left(\frac{c}{3}\cdot 1.01\right) = \text{length of user's program}$$

where s = Number of identifiers defined

c = Number of characters in indirect program

If the total program area or symbol table area becomes too large, FOCAL types the error message

?16

The following routine allows the user to find out how many core locations remain for his use.

```
*FOR I=1,5000;  SET A(I)=I
?16
*TYPE %4, I*5,  "LOCATIONS LEFT"
 8160LOCATIONS LEFT*
```

(Wait for FOCAL to type the error message.)

At the end of this routine, use ERASE to clear all the variables A(I) from the symbol table.

NOTE

The upper limit on I varies with the amount of core memory in the user's system.

APPENDIX D

CALCULATING TRIGONOMETRIC FUNCTIONS

Function	FOCAL Representation	Argument Range	Function Range
Sine	FSIN(A)	$0\leq\|A\|<10\uparrow 4$	$0\leq\|F\|\leq 1$
Cosine	FCOS(A)	$0\leq\|A\|<10\uparrow 4$	$0\leq\|F\|\leq 1$
Tangent	FSIN(A)/FCOS(A)	$0\leq\|A\|<10\uparrow 4$ $\|A\|\neq(2N+1)\pi/2$	$0\leq\|F\|<10\uparrow 6$
Secant	1/FCOS(A)	$0\leq\|A\|<10\uparrow 4$ $\|A\|\neq(2N+1)\pi/2$	$1\leq\|F\|<10\uparrow 6$
Cosecent	1/FSIN(A)	$0\leq\|A\|<10\uparrow 4$ $\|A\|\neq 2N\pi$	$1\leq\|F\|<10\uparrow 6$
Cotangent	FCOS(A)/FSIN(A)	$0\leq\|A\|<10\uparrow 4$ $\|A\|\neq 2N\pi$	$0\leq\|F\|<10\uparrow 440$
Arc sine	FATN(A/FSQT(1-A↑2)	$0\leq\|A\|<1$	$0\leq\|F\|\leq\pi/2$
Arc cosine	FATN(FSQT(1-A↑2)/A)	$0<\|A\|\leq 1$	$0\leq\|F\|\leq\pi/2$
Arc tangent	FATN(A)	$0\leq A\leq 10\uparrow 6$	$0\leq F<\pi/2$
Arc secant	FATN(FSQT(A↑2-1))	$1\leq A<10\uparrow 6$	$0\leq F<\pi/2$
Arc cosecant	FATN(1/FSQT(A↑2-1))	$1<A<10\uparrow 300$	$0<F<\pi/2$
Arc cotangent	FATN(1/A)	$0<A<10\uparrow 615$	$0<F<\pi/2$
Hyperbolic sine	(FEXP(A)-FEXP(-A))/2	$0\leq\|A\|<700$	$0\leq\|F\|\leq 5*10\uparrow 300$
Hyperbolic cosine	(FEXP(A)+FEXP(-A))/2	$0\leq\|A\|<700$	$1\leq F<5*10\uparrow 300$
Hyperbolic tangent	(FEXP(A)-FEXP(-A))/ (FEXP(A)+FEXP(-A))	$0\leq\|A\|<700$	$0\leq\|F\|\leq 1$
Hyperbolic secant	2/(FEXP(A)+FEXP(-A))	$0\leq\|A\|<700$	$0<F\leq 1$
Hyperbolic cosecant	2/(FEXP(A)-FEXP(-A))	$0<\|A\|<700$	$0<\|F\|<10\uparrow 7$
Hyperbolic cotangent	(FEXP(A)+FEXP(-A))/ (FEXP(A)-FEXP(-A))	$0<\|A\|<700$	$1\leq\|F\|<10\uparrow 7$
Arc hyperbolic sine	FLOG(A+FSQT(A↑2+1))	$-10\uparrow 5<A<10\uparrow 600$	$-12<F<1300$
Arc hyperbolic cosine	FLOG(A+FSQT(A↑2-1))	$1\leq A<10\uparrow 300$	$0\leq F\leq 700$
Arc hyberbolic tangent	(FLOG(1+A)-FLOG(1-A))/2	$0\leq\|A\|<1$	$0\leq\|F\|<8.31777$
Arc hyperbolic secant	FLOG((1/A)+FSQT((1/A↑2)-1))	$0<\|A\|\leq 1$	$0\leq F<700$
Arc hyperbolic cosecant	FLOG((1/A)+FSQT((1/A↑2)+1))	$0<\|A\|<10\uparrow 300$	$0\leq\|F\|<1400$
Arc hyperbolic cotangent	(FLOG(X+1)-FLOG(X-1))/2	$1\leq A<10\uparrow 616$	$0\leq F<8$

APPENDIX E

.DAT SLOT AND HANDLER ASSIGNMENTS

SYSTEM IN XVM/DOS

The .DAT slots to be assigned with FOCAL are

.DAT	Function	Typical Device Assignments*
-1	System Library	DPO <SYS>
-4	FOCAL binary program plus FNEW binary	DPO <SYS>
3	Library input	DPO <UIC>
5	Library output	DPO <UIC>
7	Data input	DPO <UIC>
10	Data output	DPO <UIC>

*Here the system disk is assumed to be disk pack. If it is cartridge disk, substitute RK for DP in the assignments. Similarly, for fixed-head disk, substitute DK for DP.

INDEX

Please cut along this line.

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form.

Did you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

_____

_____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you require a written reply, please check here. ☐

---------- Fold Here ----------

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

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

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

Postage will be paid by:

digital

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

digital

digital equipment corporation

PRINTED IN U.S.A.