DECUS
PROGRAM LIBRARY

DECUS NO.	12-45
TITLE	FOCALP - FOCALPE
AUTHOR	Judson Gilbert
COMPANY	Florida State University Tallahassee, Florida
DATE	June 10, 1971
SOURCE LANGUAGE	DIAL

FOCALP - FOCALPE

DECUS Program Library Write-up — DECUS NO. 12-45

GENERAL

This is a new version of FOCAL 5/69 which has been tailored to the 4K PDP-12A with an incremental plotter.

This program exists as symbolic and binary programs on a DIAL V2 tape. In this way it can be readily modified/reassembled/and loaded.

There are two versions of this program available; one with extended functions and one without.

Many of the commands and features have been changed in this program. The specific changes are listed in the latter sections.

LOADING

From the DIAL V2 tape:

a) Call the DIAL monitor
b) Load the version of FOCAL desired using the format: LO N, U
   Where N = "FOCALPE" for extended functions, and "FOCAL" without them

ASK

This command now operates in two modes, Interpretive and Character. It is initially set to Interpretive mode and may be changed by the OPTION commands.

In Interpretive mode FOCAL accepts either numbers or valid numerical expressions (composed of numbers, functions and defined variables). The following are correct responses:

```
:3.4576
:23*4.68E-5
:3/FSQT(77)
:3+3*B-2          (assuming A and B are defined)
```

RUBOUT may now be used to correct mistakes (as in command input), and the back-arrow works normally. Leading spaces are ignored. The number or expression must be terminated by a carriage return, comma or CTRL/L character. Note that SPACE and ALTMODE are no longer terminators. If there is an error in the input expression, FOCAL will type an error message and stop, giving the step number of the ASK command as the location of the error.

There is also a provision for attaching comments to the input; anything typed after a semicolon will be considered a comment and ignored.

The ASK command types a colon for each variable whose value it is asking. The colon printout may be suppressed, if desired, by the OPTION X command. All other features of ASK work normally (as in TYPE); see TYPE for the change to % and the addition of &.

In Character mode, which is selected by the OPTION C command, FOCAL accepts a single ASCII character for each variable and converts the character to its decimal code (see the list of decimal ASCII codes). Any printing or nonprinting character may be input by this method, and there is, of course, no need to terminate the input. Example:

```
<--- OPTION C; FOR I=1, 6; ASK X (I), " "
     :A :B :X :# : @:3 <---
<--- FOR I=1,6; TYPE %3.Ø, X (I)
     ABX#@ 3 <---
<--- OPTION i; FOR 1=1,6; TYPE X (1)
     193 194 216 163 192 179 <---
```

(Note: In FOCAL8/68, the only valid input is a number, and RUBOUT cannot be used.)

COMMENT

No change. Notice that a blank line is also a comment, as is step ØØ.ØØ (the heading C-FOCAL, 5/69, which is typed without its step number).

DO

The DO command works normally. Note that DO 3 is equivalent to DO 3.ØØ.

There are two new features available in the DO command. First, more than one line number may be given, separated by commas; FOCAL will execute them in order, as if they had been given in separate DO commands. For example,

```
DO 3.12, 5.54, 4, 14, 1.56;
```

is equivalent to

```
DO 3.12; DO 5.54; DO 4; DO 14; DO 1.56;
```

This feature is also implemented in WRITE.

Second, a line number may be any valid numerical expression if it is preceded by a sign (+ or -). Thus,

```
DO + N-.1
```

will DO step 3.2 if N is equal to 3.3. The expression is evaluated and truncated to the next lower hundredth to arrive at a line number; if it is negative or greater than 15.99, an error message will result.

WARNING: In some cases, due to errors in approximating a decimal number in binary, a number like 4.3Ø will be stored as 4.299999, and will be truncated to the step number 4.29; if this occurs, simply add a factor of .ØØ5 to the line number expression.

This feature, which is also available in ERASE, GOTO, IF, MODIFY, and WRITE, is very useful within a program for allowing variables to determine what steps are to be executed.

Note that a line number must be in standa d form (XX.XX, X.XX, XX.X, X.X, XX or X) if it is not preceded by a sign. Note also that FOCAL uses over five times as much time evaluating a line number in floating point as it does in fixed point.

ERASE

Erase works normally, with the exception that the command ERASE Ø will no longer destroy FOCAL as it did in FOCAL 8/68.

ERASE without a line number or ALL erases only the variable storage; the command does not terminate the line as it did before. When a line number of ALL is given, however, FOCAL returns to command mode immediately after executing the command; it also erases the variables (this was true of FOCAL 8/68 as well, although not made clear by the manual). This is because the variable pointers are upset by erasing, modifying or adding steps (program and variables share the same storage).

<u>Summary</u>

ERASE ERASE Ø	Erase the variables and continue running
ERASE 12.34	Erase step 12.34 and variables
ERASE 3	Erase group 3 and variables
ERASE ALL	Erase program and variables

Note: Calculated line numbers are legal as in DO.

FOR

This command has been improved. The format is the same as it was before (either FOR I=n1, n2, n3; ... or FOR I-r1, n2; with an assumed increment of 1), but there are, in addition, two new features:

1) A negative increment may be given for backward counting. For example. FOR I=4, -1,-3 will count: 4,3,2,1,Ø,-1,-2-3.

2) If the initial value s already beyond the final value, the object statements (the rest of the line after the FOR command) will not be executed at all. (FOCAL 8/68 would always execute the object statements at least once). For example:

```
<— FOR 1=1,3; TYPE "X"
   XXX
<— FOR 1=1,Ø; Type "X"
<— FOR 1=5,-2,1; TYPE "X"
   XXX
<— FOR 1=5, -2,6; TYPE 'X"
<— SET N=77; FOR 1=5,-1,N; TYPE "X"
<—
```

GO

The GO command works normally. Note that GO is equivalent to GOTO Ø.

GOTO

GOTO allows calculated line numbers as in DO.

HELLO

This is a new command which quickly and easily initializes FOCAL to its original conditions. HELLO performs the following commands:

```
TYPE %8.4; OPTION I, E, :, ERASE ALL
```

Use of this command will prevent confusion, especially to beginners, resulting from output being typed in unexpected formats or a previous program not being ercsed.

IF

IF command operates as before. Calculated line numbers are allowed as in DO. The space between IF and the left parenthesis is now no longer necessary.

Note: The expression must be enclosed by curved parentheses; square or angle brackets are not allowed.

LIBRARY

Non existant.

MODIFY

MODIFY works as usual. Calculated line numbers are accepted. Remember that the use of MODIFY erases the variables and returns to command mode.

CAUTION: It is possible, using MODIFY, to make a line more than 72 characters long (including line number). FOCAL will not object to this. However, if the program is punched and later loaded back in via the reader, the line will be too long to read and will cause an error message.

Symbol	Example	
		2) Rounding is now done correctly. In %2 format, the number 7.5 will be rounded to 8, and 7.499 will be rounded to 7.
		3) Output fields are no longer restricted to 31 characters.
	%	This is the signal for E format output.
	%Ø.Ø	Numbers are printed with six decimal places (but no leading zero), followed by a sign and tens exponent.
	%Ø.4	This means type in E format, but with only four digits.
		Note: The format is initially set (and set by HELLO) to %8.4.
$	$	This is the signal for FOCAL to type the values of all defined variables. The $ does not terminate the command line as it did in FOCAL 8/68. The values are typed in numerical format even if the I/O mode is Character. Also, the full, signed subscript is typed; FOCAL 8/68 could only print positive subscripts up to 99. Legal subscripts are -2Ø48 to +2Ø47.

Note: The values are typed in whatever format is in effect at the moment.

Example:

```
←— ERASE; SET X=2.354; SET A (44)=2345.6
←— SET VW(-554)=X+A(44); TYPE $; ERASE ALL
    X@(+ØØ)=        2.354Ø
    A@(+44)=  =2345.6ØØØ
    VW(-554)= -2343.24ØØ
←—
```

Symbol	Example	
&	&22 &N & B+3	This new symbol is a tabulation controller. For tabulation purposes, the teletype columns are numbered from 1 to 72 from left to right, and the specification & N, where N is an expression, will cause the teletype to tabulate to the Nth column (if it is already at or past the Nth column, the command will be ignored). This feature is useful for aligning input data, providing more concise output, and plotting. Note that nonprinting characters will not be counted and, therefore, will not throw off the alignment.

The symbols ", !, and # will work normally. Note that it is not necessary to put a comma after a !, #, or right-hand quote.

WRITE

WRITE works normally. Several line or group numbers may be given after a single WRITE command, separated by commas, and the numbers may be calculated (see DO). A WRITE command no longer terminates the command line.

Note: WRITE without an argument is equivalent to WRITE ALL.

A convenient way to WRITE groups of steps is as follows: to WRITE all the steps from 3.42 to 3.77, give this command:

```
            ←— F 1=3.42, .Ø1, 3.77; W + 1
```

This will work even if there are only a few steps.

COMMAND FORMATS

a) The character @ is no longer ignored by FOCAL on input, and is treated like any other character. Thus, it is not suitable for leader. For this reason, tapes prepared on FOCAL 8/68 with @@@@@ for leader, will cause an error message at the start of a load.

b) Blank characters, generated by CTRL/SHIFT/P or by pressing HERE-IS (if the answer-back drum has not been coded), are now suitable for leader, and for this purpose they will echo on-line. They are ignored by FOCAL at all times. To prepare a tape of a program:

1) Type WRITE ALL, but do not press RETURN.
2) Turn on the tape punch.
3) Generate leader, using HERE-IS.
4) Press RETURN, and the program will be typed and punched.
5) When it is finished, generate more leader, turn off the punch, and remove the tape.

Blank tape may also be generated by an operating program by the use of OPTION C; leader has codes of Ø and 128.

c) FOCAL maintains an input buffer and an output buffer. However, whereas the output buffer is 16 characters long, allowing FOCAL to continue processing while it is still typing previous output, the input buffer is only one character long. Input buffer overflow occurs when this buffer is filled faster than it can be used, and causes FOCAL to stop and print the message ?11.18.

Due to the construction of the keyboard control, when the ow speed reader is on and contains tape, the tape is read continuously, whether or not FOCAL is ready for the next character. When reading in long tapes, the reader can easily get more than 16 characters ahead of the output and cause an input buffer overflow. There are two ways to prevent this:

1) Every ten or fifteen lines, stop the reader momentarily, until the printing has caught up.

2) If a listing of the program is not needed, give an OPTION N command before reading in the tape. The program will not be typed as it is read in. To restore the echo at the end, type OPTION E

Note that this problem does not occur with high speed reader; reading takes place only as fast as FOCAL can use it.

d) FOCAL has a Command Buffer, which is 72 characters long, for holding direct commands as they are being typed in; there is also an ASK Buffer, which is 46 characters long, for holding input typed in response to the ASK command. If either buffer is overflowed, FOCAL will type an error message and then read in the rest of the line, which will also probably cause an error. Therefore it is important that 1) program steps do not exceed 72 characters, including the line number in its standard form (XX.XX), and 2) data input expressions do not exceed 46 characters (including comment, if any).

e) FOCAL recognizes commands by their first letter and ignores the rest of the command word. Thus FOCAL commands must be properly terminated so that FOCAL can see where they end. In FOCAL 8/68, the only terminator is space, and it is necessary to type a space after the command word. In FOCAL 5/69, the following characters will terminate a command word:

space ! " . + - ([, ;

Thus, in a long program where space is at a premium, some storage may be saved in some cases by omitting spaces. In the following example, the commands on the left, formerly illegal, are now equivalent to the commands on the right:

```
T ! !              |  T ! !                 TYPE ! !
A"WA="WA, ! !      |  A "WA="WA, ! !        ASK "WA="WA, ! !
I(WA-1.2)3.21;     |  I (WA-1.2) 3.21;      IF (WA-1.2) 3.21;
T-1.1*WA           |  T -1.1*WA             TYPE -1.1*WA
```

Note: GOTO must be one word, since it is a single command. If GO TO is typed (as two words), FOCAL will only skip to the first (separating) space, then, finding no line number, will interpret the command as "GOTO Ø."

f) In FOCAL 8/68, there was a restriction of about 1Ø consecutive digits in a number, which, if exceeded, caused an error message "literal number too large." This restriction has been removed. Any number of digits may be typed; however, digits beyond about the first eight will have no significance and will be processed only as place markers.

MISCELLANEOUS CHANGES AND RESTRICTIONS

a) The arithmetic priorities have been changed to conform to standard algebra and to FORTRAN. Multiplication and division and addition and subtraction are on the same priority level. In the absence of parentheses, FOCAL will perform multiplication and division together in left-to-right order, followed by addition and subtraction in left-to-right order. The priority list is now as follows:

↑	Exponentiation
* and /	Multiplication and Division
+ and -	Addition and Subtraction

b) It is now permissible to raise a number to a negative power. Legal exponents are integers from -2Ø48 to + 2Ø47. For exponents outside this range, or which are not integers, ↑ cannot be used. Instead, use:

$$N^X = e^{X \ln N} = \text{FEXP(X * FLOG(N))}$$

c) Changes to functions:

1) FIRT(X) returns to value of the greatest i iteger less than or equal to X. FOCAL 8/68 worked improperly for numbers between zero and -1.

```
FITR(#.1)=3
FIRT(2.9)=2
FIRT(1.5)=1
FIRT(-.1)=-1
FIRT(-.9)=-1
FIRT(-1)= -1
FIRT(-5.4)=-6
```

2) FSGN(X), the sign function, is equal to 1 if X is positive and non-zero Ø if X is zero, and -1 if X is negative. Previously, FSGN(Ø) was given as 1.

```
FSGN(-27.9)= -1
FSGN(13.4E7)= 1
FSGN(Ø)=      Ø
FSGN(2E-555)= 1
```

3) FRAN( ), the random number function, is equal to a random fraction between zero and one. Unlike FRAN( ) in FOCAL 8/68, this random number generator is suitable for statistical use of any kind. The numbers are well-distributed and, in fact, are not expected to repeat for over four billion numbers.

(d) The * (asterisk) command, which controlled the high speed reader in FOCAL 8/68, is no longer legal. It is for this reason that FOCAL's "Ready" character has been changed from asterisk to back arrow, so that the "ready" character at the end of a punched tape will not result in an error message when the tape is read back in (a back arrow at the beginning of a line is simply ignored). Thus, when FOCAL types a back arrow, it is in command mode, and this is a signal, "Your turn to type."

e) The FOCAL 8/68 provision for "Alphabetic numbers" is not implemented in FOCAL 5/69. Numbers are composed only of signs, digits, a decimal point and "E;" and a number such as ØABC is not permitted. Alphabetic information in response to the ASK command is now interpreted as a variable name rather than as an "alphabetic number." To program for user responses such as "YES' and "NO," simply define the variables YE and NO before giving the ASK statement. More sophisticated alphabetic input (and also output) may be handled in character mode.

ALT-MODE is no longer a special ASK character. In FOCAL 8/68, it was used to skip over the ASK command. The same result may be achieved in FOCAL 5/69 by just typing the name of the variable that is being ASKed.

f) When FOCAL 8/68 encountered a variable which had not yet been defined, it did not recognize this fact and took that variable's value to be zero In practice this caused confusion because users were usually not aware that they were dealing with undefined variables.

This has been fixed in FOCAL 5/69. When an undefined variable name is found in an expression which is being evaluated, an error message is printed. This reminds the user to initialize his variables before he uses them in expressions.

g) PLOT: Plot on the incremental plotter using one of the following formats:

1) To establish a zero reference for the pen position:

a: Plot I or P I

2) To move the pen to a new location:

a: Plot M XARG, YARG or
b: P M XARG, YARG or
c: P M, YARG* or
d: P M XARG,*

3) To draw a line from current pen location to new location:

a: Plot D XARG, YARG or
b: P D XARG, YARG or
c: P D, YARG* or
d: P D XARG,*

* Where one ARG is missing it is assumed to be zero.

USAGE

The arguments may be any legal FOCAL expression excluding Plotter commands ("Plot" is not recursive).

NOTE

The increments on the plotter are .005 inches each. It is usually necessary to scale up arguments to present a good picture. Arguments must be in the range of + 2047 to -2047.

Plotter commands may be strung together:

EXAMPLE 1:

P I, M 100, 100, D -100, -100, M 100, -100, D -100, 100, M 0, 0

Example 1 will draw an "X" on the paper and return the pen to zero.

EXAMPLE 2:

P I, M 100, 100, I, M 100, 100, I

Example 2 will draw a line from X (current), Y (current) to X + 200, Y + 200 and establish zero at this point.

<u>EXAMPLE 3:</u>

```
F A = 0, .1, 6. 28; P D FSIN(A)* 100, FCOS(A)* 100
```

Example 3 will draw a circle.

USER AREA INFORMATION

The user area is the area of core from 3432 to 5112, or from 3432 to 5577 if the extended functions have been deleted. In decimal, these figures are 816 and 1126 words, respectively. The user area is used for storing the user program ("text"), the names and values of all defined variables and the pushdown list which is used by FOCAL for keeping track of operations. The text is stored in the user area starting at the bottom. Variables are stored starting at the end of the text (this is why they are erased every time the text is changed). The pushdown list is filled from the top of the user area backwards. Eleven words are reserved for the pushdown list when execution starts, but usually more is needed. When the pushdown list and the variable storage run into each other, the message ?Ø2.81 is printed and execution stops.

The following is a discussion of storage requirements:

a) Every defined variable requires 5 words of storage.

b) Every line of text requires three words of storage plus one word for every two printing characters on the line, not including the line number and the space immediately after it. Other spaces, however, take half a word each like printing characters. Nonprinting characters take a full word each.

c) The length of the pushdown list varies during the running of a program, and tends to take up more space the more complex the program is. Here are some guides:

1) Each DO requires 7 words for the duration of the DO. FOCAL uses these locations to remember where it was before the DO was executed, so that it can return to it later.

2) Each FOR command requires 12 words, which are used, until the FOR is exited, for storing the loop values.

3) Extra space is also needed while FOCAL evaluates an expression; the amount needed depends on the complexity of the expression.

d) A useful rule-of-thumb is as follows:

Space required (in words) = $3n = \frac{c}{2} + 5v$

where	n = the number of lines in the program
	c = the number of characters in the program, not including line numbers
	v = the number of variables defined within the program. Note that the separate subscripted elements of an array count the same as single variables

Subtract the resulting figure from the total space available (316 or 1126) to find the amount of remaining space (which must also serve the needs of the pushdown list).

DECIMAL ASCII CODES FOR FOCAL, 5/69

Code	Character	Code	Character	Code	Character
128	CTRL/SHFT/P	172	,	216	X
	(leader)	173	-	217	Y
129	CTRL/A	174	.	218	Z
13Ø	CTRL/B	175	/	219	[
132	CTRL/D	176	Ø	22Ø	
133	CTRL/E	177	1	221	]
134	CTRL/F	178	2	222	
135	CTRL/G(bell)	179	3	223	
136	CTRL/H	18Ø	4		
137	CTRL/I	181	5	255	rubout
138	line feed	182	6		
139	CTRL/K	183	7		
14Ø	CTRL/L	184	8		
141	return	185	9		
142	CTRL/N	186	:		
143	CTRL/O	187	;		
144	CTRL/P	188			
145	CTRL/Q	189	=		
146	CTRL/R	19Ø			
147	CTRL/S	191	?		
148	CTRL/T	192	@		
149	CTRL/U	193	A		
15Ø	CTRL/V	194	B		
151	CTRL/W	195	C		
152	CTRL/X	196	D		
153	CTRL/Y	197	E		
154	CTRL/Z	198	F		
155	CTRL/SHFT/K	199	G		
156	CTRL/SHFT/L	2ØØ	H		
157	CTRL/SHFT/M	2Ø1	I		
158	CTRL/SHFT/N	2Ø2	J		
159	CTRL/SHFT/O	2Ø3	K		
16Ø	space	2Ø4	L		
161	!	2Ø5	M		
162	"	2Ø6	N		
163	#	2Ø7	O		
164	$	2Ø8	P		
165	%	2Ø9	Q		
166	&	21Ø	R		
167	.	211	S		
168	(	212	T		
169	)	213	U		
17Ø	*	214	V		
171	+	215	W		

Codes 224 to 25Ø are for lower case letters on some teletypes. On most terminals, printing these characters will result in characters identical to 192 through 218.

Codes 251 to 254 vary on different teletypes.

FOCAL considers the printing characters to be codes 16Ø through 223.

Code 131 (CTRL/C) is FOCAL's break character.

The characters [, and] are generated by SHFT/K, SHFT/L and SHFT/M respectively.

ERROR DIAGNOSTICS OF FOCAL, 5/69

Messages in these forms:

?nn.nn	Error in direct command
?nn.nn@ nn.nn	Error number@ step number

?ØØ.ØØ	-	Manual restart, or restart from monitor
?Ø1.ØØ	-	Interrupted by CTRL/C
?Ø1.34	-	Illegal group zero usage
?Ø1.42	-	Line number with zero fractional part
?Ø1.113	-	Error in step number
?Ø2.45	-	Nonexistent step referenced by DO
?Ø2.63	-	Nonexistent group referenced by DO
?Ø2.81	-	Pushdown list overflowed. May be too many variables, expression too complex, or DO's nested too deeply (something DOing itself?)
?Ø3.1Ø	-	Nonexistent line referenced by GOTO or IF
?Ø3.32	-	Illegal command
?Ø4.Ø1	-	IF not followed by parenthesized expression
?Ø4.26	-	Left of = in error in FOR or SET
?Ø4.35	-	Excess right parentheses in FOR or SET
?Ø4.42	-	Bad format in FOR
?Ø5.26	-	Excess right parentheses or bad format in TYPE
?Ø5.61	-	Illegal or nonexistent step number after MODIFY
?Ø6.Ø7	-	Illegal use of function or number
?Ø6.56	-	Undefined variable used in expression
?Ø6.62	-	Variable storage exceeded
?Ø7.12	-	Comma or operator missing in an expression
?Ø7.32	-	Operator missing before left parenthesis
?Ø7.1Ø8	-	Double operators in an expression
?Ø7.123	-	Function name not immediately followed by parenthesized expression
?Ø8.Ø2	-	Illegal function name
?Ø8.59	-	Mismatched parentheses
?11.18	-	Input buffer overflowed. Something typed or keyboard or read from teletype reader when FOCAL was not ready for it
?12.44	-	Input line too long, or storage filled by program
?13.57	-	OPTION M attempted on teletype 1
?22.46	-	Zero or negative argument for FLOG(X)
?23.11	-	Double periods in a number
?24.22	-	Fixed-point number to large
?26.58	-	Format error in plot command
?26.72	-	Argument or comma missing in plot command
?26.42	-	Illegal option name
?26.113	-	Illegal LIBRARY command
?28.123	-	Division by zero or zero to a negative power
?3Ø.21	-	Raising to a power greater than 2Ø47 or less than -2Ø48
?3Ø.52	-	Square root of a negative number
?31.127	-	Unavailable command or function, illegal character, or miscellaneous error