FOCAL-11, PRELIM

RICHARD M. MERRILL

FEBRUARY 16, 1971

PREFACE

FOCAL-11 is identical to FOCAL-8 for the user with the exception of the error diagnostics (Appendix A), the symbol table typeout (Appendix C), and additional features (Appendix B). The extended functions are not all implemented, but may be generated as done in the program of Appendix E, using the special FNEW(A,G) function. Notes on hardware connected problems are given in Appendix D. For an introduction to FOCAL programming see "FOCAL-8" in Programming Languages.

Several features have been added to this version. All field test versions should be discarded. This is a preliminary release; a final release is scheduled for late '71.

This is a stand-alone version of the program for use on the PDP-11/20 computer.

The following is a concise description of the FOCAL language to serve as a review for users of FOCAL-11.

Revised 3-11-71
From Paul Hudson

FOCAL-11 SUMMARY

CONTENTS

APPENDICES

FOCAL-11 SUMMARY

-INPUT/OUTPUT-

TYPE ASK

The *TYPE* and *ASK* Statements are used for output and input of alphanumeric calculations. Formatting of input/output is done within the statement itself. Several expressions may be in the same statement if the expressions are ended by commas. In the ASK statement only single variable names may be used between commas, and the user types in the values.

Text may be placed in quotation marks. A carriage return automatically closes quotation marks. The bell may only be inserted during initial input. If the user wishes an expression typed out to the left of the result from the evaluation process, he may bracket the expression by question marks (e.g. ?A*5.2?). This is a special use of the trace feature. *

To supply carriage returns within a TYPE or ASK statement, the exclamation mark (!) is used. A number sign (#) returns the print mechanism to the left-hand margin but does not feed the paper forward. Both the TYPE and the ASK statements may be followed by ";" and other commands unless a "$" is in the string.

There is a symbol (%) to change the output format within a TYPE statement: "%X.0Y", where X and Y are positive integers less than or equal to 31. X is the total number of digits to be output and Y is the number of digits preferred to the right of the decimal point. On output, leading 0's are typed as spaces. If the number is larger than the field width, E format will be used. E format is also specified by % alone: +0.XXXXXXXE+Y where E+Y means "10 to the Yth power". The current output format is retained until explicitly changed. Y is reduced when necessary. (see Appendix B.4.)

ASK

In the *ASK* statement, input may be terminated by certain characters: space, comma, ";", return, or alt-mode (a special nonprinting terminator used to synchronize the program with external events). If alt-mode is the first character to be input after the colon, then the value of the variable is unchanged. The left arrow erases input; rubout is ignored (see Appendix B.0); and line feed is ignored. Input words like "YES", "NO", "INCHES" are given the same numerical value as constants 0YES, 0NO, 0INCHES respectively.

The *OPERATE* command is used to select the input and/or output device. The command OPERATE R(or "O<space>R") causes the high speed reader to be used as the input device for all subsequent input. Other arguments to the OPERATE command are "P" for high speed punch, "L" for line printer, "K" for keyboard, and "T" to reselect the teletype printer. The command OPERATE RP selects both the high speed devices.

-CONTROL-

The *DO* command forms subroutines out of single lines, groups or of the entire text buffer. For a single line subroutine, control will be returned when the end of the line is encountered or when the line is otherwise terminated (such as by a RETURN statement, or in the case of TYPE with the $). An "IF" command or "GOTO line" delays return until the end of "line".

The statement "DO 5" calls all of group 5 as a subroutine, beginning with the first group 5 line number. Control will then proceed through the group numbers going from smaller to larger. A return is generated from this type of subroutine (1) by using the word RETURN, or (2) by encountering the end of that group, or (3) after transferring control out of that group via a GOTO or IF command. Similarly, the entire text buffer may be used as a recursive subroutine by simply using DO or DO ALL.

The DO statement may be combined with other legitimate commands by terminating it with a semicolon. Several forms of subroutine groupings may be tested from the console.

The number of DO commands which may be nested linearly or recursively is limited by the amount of core storage remaining in the push-down list. (see Appendix B.3)

When a GOTO or IF statement is executed within a DO group subroutine, control is transferred to the object of the GOTO or IF command. That line will be executed before return is made to the DO processor; if a line number outside the group is now about to be executed, then a return will be made from the DO subroutine call.

The *GOTO* Command will cause control of the program to be transferred to the indicated line number after the GOTO command. If control is initially handed to the program by means of an immediately executed GO, control will proceed from low numbered lines to higher numbered lines as is usual in a computer program. Control will be returned to Command Mode upon encountering a QUIT command, the end of the text, or a RETURN at the top level.

The *CONTINUE* Statement is used to indicate dummy lines. For example; it might be used to replace a line referenced elsewhere without changing those references to that line number.

The *RETURN* Command is used to exit from DO subroutines.

The *QUIT* Command causes the program to return immediately to command/input mode, type "*", and wait. It does not reset the I/O devices, however.

The *COMMENT* Statement will cause the remainder of that line to be ignored so that comments may be inserted into the program.

The *EXECUTE* Command is used to call functions without setting up a dummy SET statement. Any expression may follow the XECUTE command. For example "XECUTE FCHR(48)" will output the character "0".

The Conditional *IF* Statement is used to transfer control after a comparison. The normal form of the statement contains the word IF, a space, a parenthesized expression, and three line numbers separated from each other by commas. The program will GOTO the first line number if the expression is less than zero, the second line number if the statement has a value of zero, and the third line number if the value of the expression is greater than zero: IF (A-3) 1.1, 1.2, 1.3.

Alternative forms of the IF command are obtained by replacing extra commas and line numbers by a semicolon: IF (A-3) 1.1; TYPE "HIGH".

-CALCULATIONS-

The *SET* Statement is used for setting the value of a variable equal to the result of an expression. An expression may contain function calls, variable names, and numerical values. All of the usual arithmetic operations plus exponentiation, may be used with these operands. Any pairs of parentheses may be used: SET A=B*(1.23+4/FSQR<DEL>) SET

The *FOR* Statement: This command is used for convenience in setting up program loops and interations. The general format is: "FOR A = B,C,D;---". The index A is initialized to the value B, then the command string following the semicolon is executed. When the carriage return is encountered, the value of A is incremented by C and compared to the value of D. If A is less than or equal to D, then the command string after the semicolon is executed again. This process is repeated until A is greater than D. Note: The computations involved in the FOR statement are done in FOR

floating point arithmetic.

"A" must be a single variable; B, C, and D may all be expressions, variables, functions, or numbers. If the comma and the value of C are omnitted, then it is assumed that the increment is one.

-EDITING-

EDITING

Editing commands may not be followed by other commands on the same line.

The *ERASE* Command erases a group of command lines or a single line. For example, ERASE 2.2 will cause line 2.2 to be deleted. No error comment will be given if that line number does not exist. The command ERASE 3 or 3.0 will cause all of group 3 to be erased. To delete all of both the text and the variables, one must type the words ERASE ALL. This is to insure that erasure is not accidental. ERASE

ERASE, used alone, has the function of merely removing the variables without affecting the text. This may also be thought of as initializing the values of the variable to zero. Modifications to the text may or may not cause the variables to be deleted as if an ERASE command has been given. It is good practice to use an ERASE at the beginning of a program.

The ERASE TEXT command leaves the variables intact but removes all program text in preparation for using another program of the same or smaller size with the same variables.

The *MODIFY* command facilitates correction of individual characters within a given line. Thus, to modify the text in line 5.4, the user types "MODIFY 5.4". The command is terminated by a carriage return, whereupon the program waits for the user to type a character that is in the position at which he wishes to make changes. This character is not printed immediately, but the program prints out the contents of the line until an occurance of the search character is found and that character is printed. MODIFY

At this point the program waits for the user to exercise one of seven options by typing

1. new characters;
2. form-feed (CNTRL/L); this will cause the search to proceed to the next occurance, if any, of the search character; L^C
3. CNTRL/G(bell); this allows the user to change the search character; G^C
4. rubout to delete single characters to the left;
5. left arrow to delete all of the characters to the left; ←

6. carriage return to delete all characters to the right; (R)
   or
7. line-feed to save the remainder of the line. <-!

The *WRITE* command is used to examine the FOCAL text: WRITE
WRITE ALL (or W A) for all text;
WRITE 7 to list all of group seven;
WRITE 5.1 to print a single line.

-FUNCTIONS-

The trigonometric functions available are FCOS(arg) and FSIN(arg) for the cosine and sine of an argument in radians. Further functions may be used as indicated in Appendix E by using the FNEW(,) function.

The user programmed function FNEW(arg, group) is used to transmit an argument to the variable & (ampersand) and then call the indicated group (or line number) as a subroutine. When the subroutine returns, the last value of & (ampersand) is taken as the function return (i.e. the final value of the function FNEW(,)). See Appendix E for examples. FNEW &

The UNIBUS control function FX(func,UNIBUS-address,data) is used to control additional device options or non-standard peripherals or to reference core storage. The first argument "func" can have a value that is negative (-1), zero (0), or positive (+1) to select the function that is to be performed. The functions are respectively read (-1), logical "AND" (0), and load (+1) onto the UNISUS. The second argument "UNIBUS-address" must be either an octal number or a variable name. The function selected will be performed on this address with the data given, if any, in the third argument ("data"). FX -1 = READ 0 = AND +1 = LOAD

The character I/O function FCHR(arg) is used to manipulate the value of a single character from the currently selected input or output devices (see the OPERATE command for how to select devices). If the argument is negative then the function will acquire the next available eight bit character from the input device. If the argument is zero or positive then that (decimal) value will be converted to an eight bit integer and transmitted to the currently selected output device (The value of the function is the integer value of the argument.). The function may also be used recursively: SET Z = FCHR(FCHR(-1)) will accept a character from the input device, output the same 8 bits, and leave the value in the variable Z. FCHR

The random number function FRAN( ) has a pseudo-random value between 0 and 1 (no argument). FRAN

The analog to digital converter function FADC(GAIN,CHANNEL) allows easily programmed access to as many as 32 A/D channels on the ADØ1-A. The first argument is a gain control with a value of Ø,1,2, or 3 for a multiplier (on the input voltage) of 1,2,4,8 respectively. The value of the function is in units of volts times the multiplier factor. For other types of A/D converters, use the FX(,,) function. FADC

The general purpose functions are FSQT(arg) to find the square root of the argument; FABS(arg) to take the absolute value; FSGN(arg) to give the sign (+1 or -1) of the argument; and FITR(arg) to find the integer part of an expression. FSQT FABS FSGN FITR

The clock function FCLK( ) has a value of the time elapsed in 60ths (or 50ths) of a second since the clock was started (by FX(1,177546,64)). No argument is used with the FCLK( ) function. The clock can be stopped by FX(1,177546,Ø), power-fail, or manual-restart but not by CNTRL/C. The overhead per clock "tick" is 30.3 microseconds (see Appendix E.4). FCLK

-OPERATION-

Loading FOCAL-11 is done with the absolute loader: the program will start itself and automatically use all core from zero to the binary loader. The program may be restarted at any time by typing CNTRL/C or by pressing "STOP" and restarting at location zero. C

Upon power-fail the program will save all registers and halt. Upon auto-restart the program will resume operation from wherever it stopped. A few characters may be lost on the I/O devices.

-POLICY-

The preliminary release is not not budgeted for software support nor advertising. We shall have to depend upon word of mouth to spread the good word.

The internal documentation (listings, sources, etc.) will be available in March on an all-or-nothing basis for $2900.00. This policy gives the sophisticated user all he needs to create new systems. It makes costly the addition of small dialectic changes outside of DEC and thus keeps us in charge of the language specifications. Finally, it means that software support people will be spared large numbers of questions about the internal workings. Such questions may be written to Maynard.

Appendix A

FOCAL-11 ERROR DIAGNOSTICS

?00 - manual restart made from location 0 or by CNTRL/C. (r)

?01 - illegal line number designated.

?02 - illegal variable or function name used.

?03 - parenthesis do not match.

?04 - illegal command used.

?05 - nonexistent line number requested.

?06 - nonexistent group or line number requested by *DO*.

?07 - illegal format found in *SET* or *FOR*.

?08 - operator missing in expression.

?09 - stack overflow or nonexistent device selected.

?10 - core filled by text. (o)

?11 - core filled by variables or no room for variables. (o)

?12 - exponent range is outside of E+38. (o)

?13 - disallowed buss address in "FX". (o)

?14 - division by zero attempted. (r)

?15 - negative power tried or power too large. (r)

?16 - too many characters in input data. (r)

?17 - square root of negative number tried (r)

?18 - input buffer overflow

(o) - operational error
(r) - a run-time error
other - language or format error

Appendix B

FOCAL-11 DIFFERENCES FROM FOCAL-8

1. Subscripts

a. Double subscripts are modulo 255 (see also Appendix C.)

b. Single subscripts are modulo 2↑16-1.

c. The Symbol table typeout ("$") is not in chronological order and variables with single subscripts are printed in the symbol table printout as double subscripted variables.

2. Line Numbers

a. Line numbers larger than 99.99 are NOT error checked and some of them are accepted. Line numbers with greater significance than .01 are chopped off without notice: 1.995 becomes 1.99.

b. Variables not starting with the letters "A" or "F" may be used wherever a line number may be used except upon initial input.

3. The Push-Down List

The Push-Down List is limited to Q words where Q may be approximated by Q = N*7 + M*5 + L*10 and must be less than 160.

N is the depth of subroutine calls (*DO*)
M is the depth of paren.pairs
L is the depth of nested *FOR* loops.

4. Formats

a. Variable formats may be created by using a variable in place of the format description:

```
* TYPE %DIGITS, ALPHA,!
```

b. When the right hand margin (70 spaces) is encountered on the teletype, an extra carriage-return/line-feed pair is printed. This does not apply to the line printer nor to the high-speed punch.

5. Patch Information

Location BOTTOM contains last core address used. Change the content of TYPE+6 to 104440 to suppress "=" and print a space instead. Change the content of ASK+6 to 000400 to suppress ":".

6. Speed

A 10 x 10 matrix can be inverted in a 4K configuration in 65 seconds. Such programs may be 40% faster than FOCAL-8.

7. Storage Space and Variables

a. In a 4K machine there are about 1400 word locations for program text and/or variables. The space that is not used by the text is all used for the variables.

b. Each variable uses 4 words; each character uses 1 byte.

c. Variables may remain intact after a *MODIFY* command. This can make certain programs easier to debug when data is not lost while correcting the program.

8. Arithmetic

a. Arithmetic significance (6 digits) is the same as that for FOCAL-8, but the rounding algorithm is improved.

b. The range is ten to the plus or minus 38.

9. Starting Addresses

a. The program is self-starting at load time and upon power-fail plus auto-restart.

b. Manual restart is from location zero.

10. Input Data

a. Expressions beginning with + or - as well as numbers may be used as input to an *ASK* command: +X↑2-X+3.1. Double subscripted variables may not be used as input.

b. Rubout may be used to delete single characters of input.

11. New Commands and New Functions

a. The OPERATE command is used to select I/O devices.

b. The XECUTE command is used like a Fortran CALL to execute functions as subroutine calls.

c. The time-of-day clock can be started by "XECUTE FX(1,177546,64)". It can be stopped by "XECUTE FX(1,177546,0)".

d. The ERASE TEXT command is new.

e. FX(func,addr,data) to access the UNIBUS.

f. FNEW(arg,group) to call a user function

g. FADC(gain,channel) to read the A/D.

h. FCLK( ) to read the internal clock.

i. FRAN( ) to have a random number 0 to 1.

12. Future Commands

a. The *BEGIN* command is used to start a FOCAL process (i.e. to "DO" a group as a subroutine) when a certain time has been reached or a delay expired (command is available only with the RSX and FOCAL-11 overlay). e.g. "BEGIN 45 AT X", where X is the time-of-day in hundredths of a second.

b. The *KILL* command is used to halt all processes begun by the BEGIN command.

c. The *LIBRARY* command interacts with the disk operating system. It is available only with the DOS and FOCAL-11 overlay:

```
LIBRARY CALL  file
LIBRARY DELETE  file
LIBRARY GOSUB  file (line)
LIBRARY LIST  dev:
LIBRARY RUN  dev:file (line)
LIBRARY SAVE  dev:file
LIBRARY EXIT
LIBRARY OUTPUT dev:file
LIBRARY INPUT  dev:file
```

Appendix C

SYMBOL TABLE SUBSCRIPT PRINTING

Single subscripts and negative subscripts are printed as double subscripts. Numbers greater than 255 are printed as excess 255.

Single Subscript Storage : A(I) : 16 bits
Double Subscript Storage : A(I,J) : 8 bits , 8 bits

All output is treated as having double subscripts:

Usage	$ Printout		Type of Subscript
B(128)	B (128, 00)	=	single < 255
B(0,1)	B (00, 01)	=	double < 255
B(256)	B (00, 01)	=	single > 255
B(+1, -1)	B (01, 255)	=	negative double
B(-1)	B (255, 255)	=	negative single
B(-256)	B (00, 255)	=	negative > 255
B(257,3)	B (01, 03)	=	double > 255

Appendix D

HARDWARE NOTES

1. Using a non-existent device (on a particular configuration) will cause error ?09.

2. An error diagnostic or CONTROL/C resets I/O device selection to teletype output and keyboard input.

3. An AD01 error flag will cause FADC(,) to hang. Recovery is only by restarting at zero or by a power-off-on cycle.

4. At least one character must be typed on the keyboard before the low speed reader will read.

5. Power-fail or manual START will stop the clock.

Appendix E

EXPANSION FUNCTIONS

E.1 Example of a Recursive Function

```
1.1 SET N = 5
1.2 TYPE FNEW(N,5);C-FACTORIAL FUNCTION
1.3 QUIT

5.1 IF (1-&)5.2;R
5.2 SET &=&*FNEW(&-1,5)
```

E.2 Some Transcendental Functions

```
11.01 C TAN: DO 11 OR FNEW(ARG,11)
11.10 I (&↑2-.01)11.2;S &=&/2;D 11;S &=2*&/(1-&↑2+1E-20);R
11.20 S &=&+&↑3/3+&↑5/7.5+&↑7/315

12.01 C ASIN:DO 12   ACOS:D 12.3 OR FNEW (ARG,12.3)
12.10 I (&↑2-.01)12.2;S &=&/(FSQT(1+&)+FSQT(1-&));D 12;S &=2*&;R
12.20 S &=&+&↑3/6+.075+&↑7/22.4;R
12.30 D 12;D 10.1

13.01 C ATAN: DO 13 OR FNEW(ARG,13)
13.10 I (&↑2-.01)13.2;S &=&/(1+FSQT(&↑2+1));D 13;S &=2*&;R
13.20 S &=&-&↑3/3+&↑5/5-&↑7/7

14.01 C EXP: DO 14
14.10 I (&↑2-.01)14.2;S &=&/2;D 14;S &=&↑2;R
14.20 S &=1+&+&↑2/2+&↑3/6+&↑4/24+&↑5/120+&↑6/720

15.01 C LOG: DO 15
15.10 I (&↑2-2.04*&+1)15.2;S &=FSQT(&);D 15;S &=2*&;R
15.20 S &=(&-1)/(&+1);S &=2*(&+&↑3/3+&↑5/5+&↑7/7)

16.01 C SINH:DO 16   COSH:DO 16.3
16.10 I (&↑2-.01)16.2;S &=&/3;D 16;S &=3*&+4*&↑3;R
16.20 S &=&+&↑3/6+&↑5/120;R
16.32 D 16;S &=FSQT(1+&↑2)
*
*
```

Example of a Device Control Function

```
*
*
21.01 C THIS PROGRAM PRINTS THE BIT PATTERN IN THE S R
21.10 SET Z=FX(1,177570,-1)
21.20 IF (Z-Z1)21.5,21.1,21.5
21.50 S Z1=Z;T !;S Z=FNEW(Z,40);G 21.1

40.10 S &1=14
40.20 IF (&-2↑&1)40.3;S Z=FCHR(49)
40.21 S &=&-2↑&1
40.22 S &1=&1-1
40.25 IF (&)X;G 40.2
40.30 S Z=FCHR(48);G 40.22
*
*
*GO

0000000000000101
0000001111000111
1111001111000111
0110001111000111
0111111111000111
0111111101000000
0000000000000000
```

Example of a Timing Routine

```
1.10 X FX(-1,177546,64)
1.20 S X=FCLK( )
1.30 F I=1,1000; DO 2
1.40 S X=FCLK( )-X
1.50 TYPE !%7.06, X/6083-30.3E-9*X," SECONDS PER CYCLE."!!
1.70 QUIT

2.01 C SUBROUTINE TO BE TIMED
2.10 X A+B
```

Appendix F

FOCAL-11 Commands

```
                ;ENGLISH  ;FRENCH    ;SPANISH     ;GERMAN
ASK             ;ASK      ;DEMANDE   ;INTERROGUE  ;FRAGE
ERRORC          ;BEGIN    ;LEVE      ;XECUTE      ;COMMENCE
PCI             ;COMMENT  ;COMMENTE  ;COMENTARIO  ;KOMMENTAR
DO              ;DO       ;FAIZ      ;HAGA        ;MACHE
ERASE           ;ERASE    ;BIFFE     ;BORRE       ;LOSCHE
FOR             ;FOR      ;QUAND     ;PARA        ;DAFOR
GOTO            ;GOTO     ;VA        ;ADELANTE    ;GEHZU
ERRORC          ;H-       ;K-        ;G-          ;U-
IF              ;IF       ;SI        ;SI          ;WENN
ERRORC          ;J-       ;J-        ;J-          ;I-
ERRORC          ;KILL     ;HALTE     ;HALTE       ;HALT
ERRORC          ;LIBRARY  ;ENTERPOSE ;LAZO        ;BILBLIOTHEK
MODIFY          ;MODIFY   ;MODIFIE   ;MODIFIQUE   ;ÄNDERE
ERRORC          ;N-       ;G-        ;V-          ;J-
PROGIO          ;OPERATE  ;PRATIQUE  ;OBRARE      ;OBERATE
ERRORC          ;P-       ;N-        ;N-          ;N-
STARTX          ;QUIT     ;ARRETE    ;DETENGASE   ;ENDE
RETURN          ;RETURN   ;RETOURNE  ;RETOURNE    ;QUITTE
SET             ;SET      ;ORGANIZE  ;UBIQUE      ;SETZE
TYPE            ;TYPE     ;TAPE      ;TIPPEE      ;RECHNE
ERRORC          ;U-       ;U-        ;W-          ;P-
ERRORC          ;V-       ;W-        ;Q-          ;V-
WRITE           ;WRITE    ;INSCRIS   ;ESCRIBA     ;TIPPE
XECUTE          ;XECUTE   ;XECUTE    ;FLUIR       ;XECUTE
;OTHERS UNUSED: ;YZ       ;YZ        ;YZ          ;YZ

;TO CHANGE LANGUAGE, ALPHABETIZE ON THE APPROPRIATE COLUMN.
```