Behavior Research Methods & Instrumentation
1975, Vol. 7 (2), 137-142

4K Laboratory FOCAL

S. W. LINK
McMaster University, Hamilton, Ontario, Canada

While FOCAL has become a commonplace calculational language in many small computer laboratories, the use of FOCAL as an experimental control system has been largely ignored. In the present paper, a 4K FOCAL laboratory system is described. Extensions to the FOCAL function and command lists provide for integer manipulations, high- and low-speed output, and a variable DO statement. By providing access to experimental peripheral devices, FOCAL can be used as a powerful operating system. As an illustration, a two-choice reaction time experiment is discussed, a FOCAL control program for this experiment is described, and an assembly language listing illustrates how changes to FOCAL can be made.

Like many of you, I began my computer training on a small computer. The selection of a small computer was not so much a matter of choice as of necessity, for at the time most computers were small—if not physically, then at least computationally. The Bendix G15-D was such a computer. The D signified the presence of a drum that revolved to expose, slowly, each of 1,024 words of bulk storage. The machine was user operated, and, because of modest operating tolerances, was abused by the vagaries of both humidity and temperature. We programmed it in hexadecimal, without an assembly language, but usually with what we conceived of as consummate skill and lofty logic. We soon learned the canard that while people are smarter than computers, computers are smarter than programmers.

As computers and their operating systems grew, so grew our nostalgia for the "hands on" feature of a small computer. We progressed to the 709, the 7090, the B5500, and a variety of CDC machines only to abandon them at the arrival of the first PDP-1. Our nostalgia abated, we advanced toward the use of small laboratory computers that were fast, simple to operate, reliable, but often contained only 4K of core memory.

It was with this background that I approached reviewing Bernard Weiss' very fine new book, *Digital Computers in the Behavioral Laboratory* (1973) (Link, 1974). About the only criticism I had concerned the omission from the book of a chapter describing how FOCAL could be modified to provide a laboratory operating system. I claimed that FOCAL could be a powerful laboratory language; Bernard Weiss asked me to demonstrate how—so here we are.

My purpose, then, is to illustrate in rather simple terms how 4K FOCAL has been used in a laboratory where choice responses and their response times are always gathered and where the number of within-subject experimental outcomes is at most 30—about the largest number that can be conveniently investigated during a 1-h experimental session. The point of this example is to illustrate that a considerable savings in programming effort can be obtained by the modification of one manufacturer's user-oriented software.

The application of FOCAL to experimental control is not new. A number of previous papers have pointed to FOCAL applications, and some have even included illustrations of FOCAL programming together with patches designed to implement new instructions or permit input from external experimental devices (e.g., Reece, 1973; Siegel, 1972). At least one author has suggested, however, that FOCAL will require more than 4K of storage for all but the simplest of experiments (Doll, 1972). Yet, for many laboratories, simple experiments are quite common, and the use of FOCAL will provide faster set-up time than could be obtained by assembly language programming or by the modification of some quite sophisticated laboratory packages (Matthews & Wescourt, 1974; Millman, 1971).

The laboratory environment will determine how often, and how many additional, peripheral devices must be sampled or supplied with data by FOCAL. In this regard, our laboratory is similar to laboratories for which large operating systems have been written. Minor inspection would reveal the usual jungle of computer (PDP-8I), peripheral equipment, and assorted, if not tangled, cables connecting the computer to an experimental room. We run one subject at a time, in part because the stimulus equipment must sometimes be calibrated for each subject, but mostly because our experiments are lengthy within-subject psychophysical studies which do not require more than a single subject station. In addition to a high-speed reader/punch and ASR-33, we have a calligraphic display system (Griffin, 1968; Link, 1969), auditory and tactual stimulus control devices, indicator lights, various types of response panels, a programmable clock, and a Tri-Data Cartrifile that has been taught to perform like a DEC disk monitor system.

The research described in this paper was supported by the National Research Council of Canada. The paper was aided by the support of Rockefeller University during my tenure as visiting Associate Professor of Psychology. Reprint requests should be sent to the Department of Psychology, McMaster University, Hamilton, Ontario, Canada.

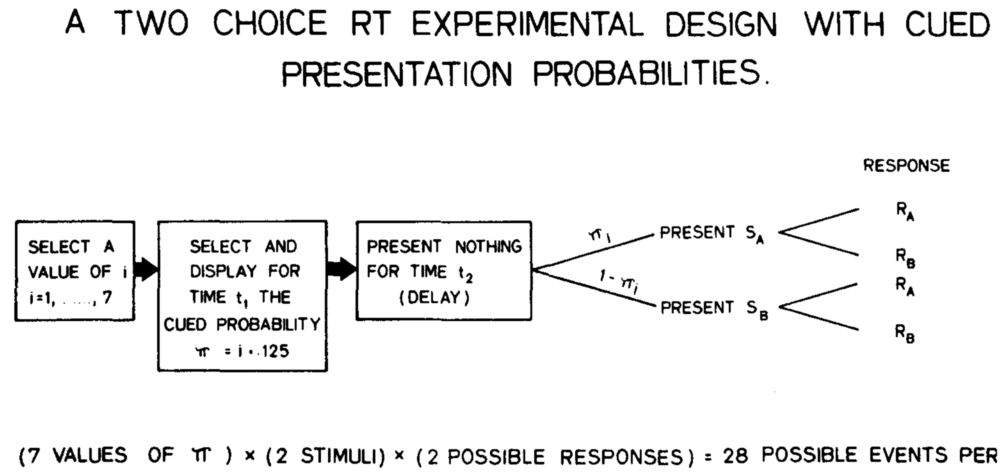

Figure 1: A two-choice reaction time experimental design.

A typical laboratory experiment is illustrated in Figure 1. The purpose of this two-choice response time experiment was to bias a subject on a trial by trial basis toward one of two response alternatives, R_A or R_B. A point of major theoretical interest was how correct and error response times would change as a function of response bias (Link, 1975).

The program controlling the experiment must perform the following tasks. At the start of each self-paced trial, the probability of presenting stimulus S_A, on the current trial, must be presented to the subject for a convenient length of time (T1). Thereafter, a blank display screen is presented (for time T2) to clear the visual field and provide a warning signal. Then stimulus S_A is presented with a marginal probability equal to the displayed probability. The stimulus remains visible until the subject responds R_A or R_B, and both the response made and the response time must be recorded and saved for later analysis. However, to provide the experimenter and the subject with a summary of results, we accumulated for each displayed probability value, $\pi_i(i = 1, \ldots, 7)$, the numbers of responses of each type to each stimulus and the associated response times. Since there were seven values of π, two stimuli, two responses, and two response measures, the results from a single trial would occupy 2 of 56 data collection bins. At the end of the subject session, response frequencies and mean RTs for each stimulus-response combination and each value of π were to be reported on the Teletype.

The first question we face in adapting FOCAL to our experimental needs is what changes to make. Any changes that are made must either replace or modify existing FOCAL routines or be patched to FOCAL in any unused space. It is, therefore, of primary importance that we have a clear idea of how free space can be obtained.

There are three methods of increasing the amount of available memory space. First, by deleting the extended functions (LOG, EXP, ATN, SINE, and COSINE), an area from 3206_8 to 5400_8 is made available. The size of this area is greater than the space occupied by Symbolic Editor. Second, by limiting the memory area used for storage of the FOCAL program, variables, and the push down list, we can reserve a memory area for patches or data storage. Last, there are many sections of 4K-FOCAL 69 that are unused, and some command routines, such as the L command, provide additional free space when deleted. By these simple methods, well over one-quarter of the 4K memory is free to be used for storage of the FOCAL program, variables, push down list, and user defined routines.

Having determined the available free space, we now wish to use this space efficiently.

When a large number of variables are to be defined, and the values assumed by these variables are less than 4095, an efficient use of free memory space can be made by treating these variables as integers. In contrast to variables defined by FOCAL, which require five contiguous memory locations, our integers each occupy only a single location. Naturally, deviating from the FOCAL definition of a variable will require additional programming space (approximately 18 locations), but this increase is offset by more efficient storage.

In the sample program below, integer manipulation is accomplished by defining two new FOCAL functions. The function FPUT (X,Y) will convert to an integer $0 < Y < 4095$ either the value Y or a fixed numerical value and place the integer value in Memory Location X (specified in decimal). An example of FPUT is shown in statement, 01.03 of the sample FOCAL program. We wished to fill Memory Locations 2342 to 2453 with zero. It can be seen that rather than using a FOCAL variable set equal to zero, a fixed integer value of 0 was used. To retrieve an integer already in memory, another FOCAL function FLST (x) will convert the integer found in (decimal) Location X to a FOCAL variable. For example, the command S Z=FLST(X) will set the

FOCAL variable Z equal to the integer value found in Location X. These two FOCAL functions control input to and output from user selected areas of memory and vastly increase the capability of FOCAL to store experimentally obtained data values such as frequencies of stimulus-response pairings.

In addition to functions providing for integer storage and retrieval, other functions can increase the convenience of Laboratory FOCAL. To the version of FOCAL controlling the two-choice reaction time experiment, we have added a new random number generator and a routine to switch between high- and low-speed output. By setting Z=FRAN( ), a pseudorandom number bounded by 0 and 1 will be generated. The random numbers so obtained have satisfied marginal probability, runs, and sequential tests for randomness. Switching between low- and high-speed output is accomplished by the instruction S Z=FSWP( ). Each execution of this instruction promotes a change from the current to the alternative output mode. High-speed output will, in many cases, obviate the need to devote large memory areas to storage of trial by trial results.

Other efficiencies are to be had by modifying or augmenting the FOCAL command list. In the present case, only two changes have been made. The first change is a modification of the Comment (C) command. Normally, whenever a C is encountered, the FOCAL processor will simply ignore any subsequent characters up to the next text terminator. With a rather minor modification, the Comment command can also be made to clear all device flags. If peripheral devices are not to be serviced by a FOCAL interrupt handler, then it is particularly important that these devices and their flags be cleared at the beginning of a FOCAL program. Were these devices not cleared, FOCAL would sense an "illegal interrupt" and become quite confused.

The second command change provides for multiple branching beyond that offered by use of an IF statement. Suppose, for example, that on the basis of calculation from random numbers, experimental trial outcomes, or other methods, any 1 of 20 different resultant computations must be performed. A chain of IF statements would, of course, eventually lead to the desired computation. On the other hand, if the calculation yields a number that can be put into correspondence with numbers ranging from 1 to 20, then a single branching statement could provide direct transfer to the desired computational sequence. To effect this operation, we have replaced the usual Library function, L, with a routine which will transfer program control to an arbitrary FOCAL group number. Execution of the statement L X will force a transfer of control to Group Number X. After execution of Group X, control is transferred back to the statement following L X. Thus, the multiple branching statement can be considered similar to a variable DO statement.

```
C-FOCAL,1969

Ø1.Ø1 C PROGRAM FOR TWO CHOICE RT WITH PROBABILITY DISPLAY
Ø1.Ø2 C ERASE INTEGER STORAGE AREA
Ø1.Ø3           F I=2342,2453;S Z=FPUT(I,Ø)
Ø1.Ø5 C SET UP STIMULUS LIST
Ø1.Ø6           S J=2341;F K=Ø,6;D 8
Ø1.Ø9 C RUN 5 BLOCKS OF 56 TRIALS EACH
Ø1.1Ø           F K=1,5;F I=Ø,55;D 5
Ø1.19 C PRINT OUT SUMMARY OF RESULTS
Ø1.2Ø           T !!!;F K=Ø,6;D 2
Ø1.3Ø C ALL FINISHED SO; QUIT

Ø2.Ø5 C GROUP 2 CONTROLS SUMMARY PRINTOUT
Ø2.1Ø           T !;F I=K*4+1,(K+1)*4;D 3
Ø2.2Ø           R

Ø3.Ø1 C GROUP 3 PRINTS ONE LINE OF RESULTS
Ø3.Ø5           S J=I+2397
Ø3.1Ø           I (FLST(J))3.3,3.3;S T(I)=T(I)/FLST(J)
Ø3.3Ø           T %5,FLST(J),T(I);R

Ø4.Ø1 C GROUP 4 RANDOMLY CHOOSES THE STIMULUS AND
Ø4.Ø2 C PERMUTES THE STIMULUS LIST TRIAL BY TRIAL
Ø4.Ø3 C FRAN() YIELDS A PSUEDO-RANDOM NUMBER >Ø,<1.
Ø4.1Ø           S J=56-I;S M=FITR(FRAN()*J+2342);S X=2341+J
Ø4.2Ø           S R=FLST(M);S Z=FPUT(M,FLST(X));S Z=FPUT(X,R)
Ø4.3Ø           R

Ø5.Ø1 C GROUP 5 CONTROLS A SINGLE TRIAL BY FIRST OBTAINING
Ø5.Ø2 C A STIMULUS LIST NUMBER AND CONVERTING IT TO A
Ø5.Ø3 C STIMULUS NUMBER,S, AND TO A PROBABILITY DISPLAY
Ø5.Ø4 C POINTER,X.
Ø5.Ø5           D 4;S X=FITR((R-1)/2);S S=R-2*X
Ø5.15 C ENTER REAL TIME STIMULUS PRESENTATION ROUTINE
Ø5.16 C X*3 POINTS TO A DISPLAY VECTOR CONTAINING THREE
Ø5.17 C     CHARACTERS;125,25Ø,375,5ØØ,625,75Ø,875.
Ø5.18 C T1 IS THE DISPLAY DURATION
Ø5.19 C T2 IS THE DELAY FROM DISPLAY TO STIMULUS
Ø5.2Ø C  S IS THE STIMULUS(1 OR 2)
Ø5.25           S Z=FTRL(X*3,T1,T2,S)
Ø5.3Ø C GET RESPONSE AND RT FROM KNOWN MEMORY LOCATIONS
Ø5.35           S R=FLST(2745);S RT=FLST(2746)
Ø5.39 C OUTPUT TRIAL RESULTS ON HIGH SPEED PUNCH
Ø5.4Ø           S Z=FSWP();T %1," ",X,S,R;T %4,RT;S Z=FSWP()
Ø5.59 C ACCUMULATE RESULTS
Ø5.6Ø           S Y=X*R+(S-1)*2+R;S T(Y)=T(Y)+RT;S Y=Y+2397
Ø5.7Ø           S Z=FPUT(Y,FLST(Y)+1)
Ø5.8Ø           R

Ø8.Ø1 C THIS ROUTINE COMPUTES AN ARRAY (LINEARIZED) OF 7 ROWS
Ø8.Ø2 C AND 8 COLUMNS. ODD NUMBERS REFER TO STIMULUS 1 AND EVEN
Ø8.Ø3 C NUMBERS TO STIMULUS 2. THE MAGNITUDE OF A NUMBER INDICATES
Ø8.Ø4 C A ROW OF THE ARRAY AND ALSO DETERMINES THE PROBABILITY
Ø8.Ø5 C DISPLAY TO BE SHOWN TO THE SUBJECT(CALCULATED IN GROUP 4).
Ø8.1Ø           F I=1,7-K;S J=J+1;S Z=FPUT(J,2*K+1)
Ø8.2Ø           F I=Ø,K;S J=J+1;S Z=FPUT(J,2*K+2)
Ø8.3Ø           R
*
```

Figure 2: A 4K Laboratory FOCAL program for a two-choice reaction time experiment with cued presentation probabilities.

As an illustration of this command, let Y be a random number uniformly distributed between zero and one. If we want to execute 20 different, but equally likely, calculations based on the value of Y, we set X=20*Y + 10. The value of X will range from 10 through 29, and execution of the L X command will transfer control to Group Number X where the appropriate calculations can be performed.

A more powerful use of the L command is in sequencing program operation. The execution of the instruction F I=1, N; L X(I) provides an example. In this statement, the first group of statements to be executed is defined by X(1). A statement in Group X(1) could modify any values in the vector X(I). Furthermore, since the values of I and N can be changed by statements within Group X(1), the sequence of program operation, after the completion of Group X(1), can be altered. Since any group may operate on I,N, or any value of X(I), a program can be thought of as series of transitions from one group to another, where each group is sensitive to the current state of the program vector X. The use of the L command can provide great programming flexibility and can greatly simplify complicated multiple branching structures.

The final problem to be faced in adapting FOCAL to the laboratory centers on access to experimental control devices. Access from the FOCAL control program can be provided by extension of the FOCAL function and command repertoire. At a minimum, a single function could pass to an assembly language subroutine all parameters required for execution of a single experimental trial while the subroutine would return to FOCAL the observed response measures. In this instance, the subroutine would assume complete control during an experimental epoch by disabling the FOCAL processor and operating in real time.

A second, and often preferable, method is to define a FOCAL function or command which will control a single experimental device such as a display device, a clock, and so forth. This method has the advantage of providing a single FOCAL operating system that can be used by experimenters not acquainted with assembly language programming. Although the implementation of this method may require substantial modification to the FOCAL interrupt handler, it provides a quite flexible experimental language (cf. Reece, 1973).

A FOCAL EXAMPLE

Figure 2 represents a FOCAL program which uses many of the function and command features described above. The program has been used to obtain two-choice RT data in an experiment designed to bias a subject toward one or the other of the two response alternatives. In this program, a single FOCAL function passes parameter values to an assembly language subroutine, FTRL, which assumes complete computer control during an experimental trial. Each group of statements is self-explanatory, given the numerous comments, but a short description of the main features of the program may be of some value.

Briefly, the FOCAL function FTRL (Y, T1, T2, S) was defined so that the arguments of the function could be computed in the body of the FOCAL program. These arguments refer to the presentation probability to be displayed to the subject, Y; the duration of that display, T1; the time interval between the termination of the probability display and the presentation of the stimulus, T2; and the value of the stimulus, S. Upon execution, FTRL controls all within-trial experimental events, waits for the subject's response, and then stores the response and response time in absolute memory locations. When FTRL has completed execution, the main body of the FOCAL program retrieves the response and response time. FTRL uses one page of memory.

The FOCAL program consists of two main sections. Group 1 is an executive routine that first clears all device flags and then initializes Locations 2342 to 2453 to zero. Then a 7 by 8 linearized array is filled with numbers which simultaneously indicate presentation probability and stimulus values. After these initial computations have been completed, five blocks of 56 experimental trials are run, and then summary results are printed on the Teletype.

Group 5 executes a single trial. First, a value is selected at random from the linearized array filled by Group 1. The value obtained is decomposed into a presentation probability indicator ranging from 1 to 7 and a stimulus value S. These values, together with T1 and T2, are passed to the assembly language trial controller by FTRL. After a response, control returns to FOCAL, and the values of the response and response time are obtained from Absolute Locations 2745 and 2746. All data summarizing the trial are then punched on paper tape, and summary statistics are gathered in two linearized arrays. At the end of Group 5, control is returned to Statement 1.10.

What has been gained through this approach to the laboratory use of FOCAL is a bookkeeping system that relegates to FOCAL data handling operations. The control of a single trial is in the hands of FTRL where real time operation can be effected. Taking advantage of the great flexibility provided by FOCAL saves valuable programming time and yet allows for hands-on operation by laboratory personnel who are unfamiliar with assembly language programming. Furthermore, debugging time is greatly reduced since the experimenter can simultaneously act as programmer and subject and can easily modify his FOCAL program to meet the demands of a new experimental design.

The major question most computer users ask concerning this application of FOCAL is how to get started. My experience is probably similar to that of others who have adapted FOCAL to their own purposes. First, one needs a model illustrating the assembly language programming required in defining new commands and functions. One of the best models is that of Reece (1973). However, to provide other illustrations, the appendix to this paper contains a PAL8 listing of the FOCAL modifications required to define FRAN, FLST, FPUT, FSWP, and changes to the C and L commands. In order to write a patch to FOCAL, one should have available a listing of FOCAL, the *Advanced Focal* manual (DEC-08-AJBB-DL, 1969). and the very useful monograph by Wrege entitled "FOCAL: How to write new subroutines and use internal routines." With these programming aids, an experimenter should have little difficulty in adapting FOCAL to his laboratory needs.

APPENDIX

The program listed below provides examples of patches to 4K-FOCAL 69. Some routines are identical to those employed by Reece (1973), and some other routines will be recognized by FOCAL users.

```
/PATCH TO FOCAL FOR TWO CHOICE  PAL8-V7

            /PATCH TO FOCAL FOR TWO CHOICE RT EXPERIMENT

            /DEFINITIONS
      ØØ53  INTEGER=ØØ53
      Ø136  EFUN31=Ø136
      456Ø  SPNOR=456Ø
      4542  PUSHA=4542
      4543  PUSHF=4543
      454Ø  PUSHJ=454Ø
      1413  POPA=1413
      4544  POPF=4544
      5541  POPJ=5541
      4557  RTL6=4557
      ØØ22  PC=22
      ØØ65  NAGSW=65
      ØØ67  LINENO=67
      ØØ4Ø  EX1=4Ø
      ØØ44  EXP=44
      Ø42Ø  DO=42Ø
      1613  EVAL=1613
      157Ø  ATLIST=157Ø
      Ø374  FNTABF=Ø374
      2165  FNTABL=2165
      1163  COMGO=1163
      26ØØ  SAVAC=26ØØ
      3Ø52  FRAN=3Ø52
            /
            /SPECIAL DEFINITIONS FOR PATCH
      457Ø  ARG=JMS I XARG1
      52ØØ  XTRL=52ØØ
      616Ø  XPUT=616Ø
      75Ø3  SWAP=75Ø3
      4626  LCOM=4626
      2564  XLST=2564
      1343  XARG=1343
      ØØ11  CT=11
      Ø127  M2P=127
            /
            /CHANGES TO LOW CORE FOCAL
      ØØØ1  *1
ØØØØ1 6451          6451     /CLEAR RESPONSE FLAG
ØØØØ2 7ØØØ          NOP

ØØØØ3 54Ø4          JMP I 4
ØØØØ4 26Ø3          26Ø3     /FOCAL INTERRUPT HANDLER

      ØØ35  *35
ØØØ35 4423  BOTTOM, LCOM-2Ø3         /LAST FOCAL LOCATION

      Ø17Ø  *17Ø
ØØ17Ø 1343  XARG1,  XARG
ØØ171 ØØØØ  SHOW,   Ø        /LOCATION FILLED WITH FWA
                             /OF DISPLAY ROUTINE
ØØ172 5733  ADD,    5733     /ADDITION ROUTINE
ØØ173 4421  RANDOM, 4421     /CURRENT FLOATING
ØØ174 3Ø4Ø          3Ø4Ø     /REPRESENTATION OF
ØØ175 ØØØ1          ØØØ1     /RANDOM NUMBER

            /CHANGES TO FUNCTION TABLES
      Ø377  *FNTABF+3
ØØ377 52ØØ          XTRL     /MAIN TRIAL SEQUENCER
      217Ø  *FNTABL+3
Ø217Ø 27ØØ          27ØØ     /CODE FOR FTRL=4*T+2*R+L
      Ø4ØØ  *FNTABF+4
ØØ4ØØ 3Ø52          FRAN     /NEW RANDOM NUMBER GENERATOR
      Ø4Ø1  *FNTABF+5
ØØ4Ø1 75Ø3          SWAP     /SWITCH OUTPUT MODE
      2172  *FNTABL+5
Ø2172 2712          2712     /CODE FOR FSWP=4*S+2*W+P
      Ø411  *FNTABF
                +15
ØØ411 2564          XLST     /ROUTINE TO RETURN LOCATION CONTENTS
       22Ø2  *FNTABL+15
Ø22Ø2 2652          2652     /CODE FOR FLST=4*L+2*S+T
      Ø412  *FNTABF+16
ØØ412 616Ø          XPUT     /ROUTINE TO PUT INTO CORE
      22Ø3  *FNTABL+16
Ø22Ø3 2676          2676     /CODE = 4*P+2*U+T
      1574  *ATLIST+4
Ø1574 Ø614           614     /REPLACE ADDR OF SYMBOL TABLE TYPEOUT
                             /(3Ø52)WITH ADDR OF EXIT FROM A CALL.
       117Ø  *COMGO+5
Ø117Ø 4643          XCLR     /C COMMAND NOW CLEARS FLAGS AND BUFFERS
       1173  *COMGO+1Ø
Ø1173 4626          LCOM     /THE L COMMAND NOW EXECUTES A DO X.
       1217  *1217
Ø1217 76ØØ          76ØØ     /ERASE COLONS
      6ØØ2  *6ØØ2

Ø6ØØ2 76ØØ          76ØØ     /ERASE EQUAL SIGN

            /ROUTINE TO GET AN ARGUMENT FROM A FUNCTION CALL.

      1343  *XARG
Ø1343 ØØØØ  XARG,   Ø        /ENTRY POINT
Ø1344 73ØØ          CLL CLA
Ø1345 456Ø          SPNOR
Ø1346 454Ø          PUSHJ
Ø1347 1612          EVAL-1
Ø135Ø 4453          JMS I INTEGER
Ø1351 5743          JMP I XARG       /RETURN

            /ROUTINE TO GET CONTENTS OF A CORE ADDRESS.
            /S Z=FLST(I), WHERE I IS ADDRESS IN DECIMAL

      2564  *XLST
Ø2564 4453  XLST,   JMS I INTEGER   /GET I
Ø2565 3377          DCA XLIST1
Ø2566 1777          TAD I XLIST1    /GET CONTENTS OF I
Ø2567 711Ø          CLL RAR /NORMALIZE IN FLAC
Ø257Ø 3Ø45          DCA 45
Ø2571 7Ø1Ø          RAR
Ø2572 3Ø46          DCA 46
Ø2573 1376          TAD XFLCN1
Ø2574 3Ø44          DCA 44
Ø2575 5536          JMP I EFUN31    /RETURN
Ø2576 ØØ14  XFLCN1, 14
Ø2577 ØØØØ  XLIST1,  Ø

            ///RANDOM NUMBER GENERATOR: FRAN///
      3Ø52  *FRAN
Ø3Ø52 3Ø44  FRAN,   DCA 44
Ø3Ø53 1174          TAD RANDOM+1
Ø3Ø54 3Ø45          DCA 45
Ø3Ø55 1175          TAD RANDOM+2
Ø3Ø56 3Ø46          DCA 46
Ø3Ø57 4543          PUSHF   /FLAC=R*2↑12
Ø3Ø6Ø Ø173          RANDOM
Ø3Ø61 4544          POPF
Ø3Ø62 ØØ41          EX1+1
Ø3Ø63 1313          TAD M4  /MULT BY 2↑4
Ø3Ø64 3Ø11          DCA CT
Ø3Ø65 4527          JMS I M2P
Ø3Ø66 2Ø11          ISZ CT  /TO GET R*2↑16
Ø3Ø67 5265          JMP .-2
Ø3Ø7Ø 4572          JMS I ADD       /+R=R*(2↑16+1)
Ø3Ø71 4527          JMS I M2P       /*2=R*(2↑17+2)
Ø3Ø72 4572          JMS I ADD       /+R=R*(2↑17+3)
Ø3Ø73 4543          PUSHF
Ø3Ø74 ØØ45          EXP+1
Ø3Ø75 4544          POPF
Ø3Ø76 Ø173          RANDOM  /NEW RANDOM NUMBER
Ø3Ø77 3Ø47          DCA EXP+3       /CHOP TO 2 WORDS
Ø31ØØ 3Ø44          DCA EXP /MAKE A FRACTION
Ø31Ø1 1Ø45          TAD EXP+1       /CHECK SIGN
Ø31Ø2 77ØØ          SMA CLA
Ø31Ø3 5536          JMP I 136       /RETURN
Ø31Ø4 1Ø45          TAD EXP+1       /TAKE 1'S COMPLEMENT
Ø31Ø5 714Ø          CLL CMA
Ø31Ø6 3Ø45          DCA EXP+1
Ø31Ø7 1Ø46          TAD EXP+2
Ø311Ø 7Ø4Ø          CMA
Ø3111 3Ø46          DCA EXP+2
Ø3112 5536          JMP I 136       /ALWAYS RETURN A POSITIVE NUMBER
 Ø3113 7774  M4,     -4

            /ROUTINE TO EXECUTE A COMPUTED DO X WITH X AS
            /A VARIABLE. CALL AS L X WHERE X IS A PREVIOUSLY
            /DEFINED GROUP NUMBER. THIS ROUTINE REPLACES THE
            /L COMMAND.

      4626  *LCOM
Ø4626 73ØØ  LCOM,   CLL CLA
Ø4627 1Ø22          TAD PC  /SAVE FOCAL PROGRAM COUNTER
Ø463Ø 4542          PUSHA   /ON THE PUSHDOWN LIST FOR RETURN
Ø4631 457Ø          ARG     /EVALUATE THE SYMBOL FOLLOWING L
Ø4632 4557          RTL6    /CONVERT TO GROUP NUMBER
Ø4633 7ØØ4          RAL
Ø4634 3Ø67          DCA LINENO
Ø4635 3Ø65          DCA NAGSW       /SET ALL GROUP SWITCH
Ø4636 454Ø          PUSHJ   /EXECUTE THE GROUP
Ø4637 Ø421           DO+1
Ø464Ø 1413          POPA    /GET RETURN
Ø4641 3Ø22          DCA PC  /RESTORE PROGRAM COUNTER
Ø4642 5541          POPJ    /EXIT

            /ROUTINE TO CLEAR ALL FLAGS AND BUFFERS. THIS
            /ROUTINE REPLACES THE C COMMAND IN FOCAL.

Ø4643 73ØØ  XCLR,   CLL CLA
Ø4644 6Ø12          6Ø12
Ø4645 6Ø22          6Ø22
Ø4646 6Ø32          6Ø32
Ø4647 6Ø42          6Ø42
Ø465Ø 6132          6132    /DISABLE CLOCK
Ø4651 6451          6451    /CLEAR AND SKIP ON RESP FLAG
Ø4652 6356          6356    /FREEZE SCOPD
Ø4653 6456          6456    /CLEAR OUTPUT BUFFER
Ø4654 5655          JMP I XCLR1     /RETURN
Ø4655 Ø614  XCLR1,   614    /RETURN FOR A CALL

            /EXPERIMENTAL ROUTINE FOR TWO CHOICE RT
            /CALL WITH S Z=FTRL(X*3,T1,T2,S)
            /X=Ø,...,6 SPECIFIES THE TRIAL TYPE, *3 GIVES TABLE LOCATION
             /T1=DURATION OF PROBABILITY DISPLAY IN .1 SEC. UNITS
             /T2=INTERVAL BETWEEN PROB. DISPLAY AND STIM.(.1 SEC. UNITS)
             / S=STIMULUS(1 OR 2)
            /
            /THE CODING OF THIS ROUTINE WILL BE UNIQUE TO INDIVIDUAL
             /LABORATORIES AND HAS BEEN OMITTED FROM THIS LISTING.
             /FOR THE PURPOSE OF THIS LISTING THE AVAILABLE SPACE
             /IS FROM 4656 TO 54ØØ.

            /ROUTINE TO PUT DATA INTO CORD LOCATION.
            /CALL AS S Z=FPUT(X,Y), WHERE X IS ADDRESS
            /(IN DECIMAL) AND Y IS DATA VALUE.

      616Ø  *XPUT
Ø616Ø 4453  XPUT,   JMS I INTEGER   /GET X
Ø6161 3365          DCA XPUT1
Ø6162 457Ø          ARG     /GET DATA VALUE
Ø6163 3765          DCA I XPUT1
Ø6164 5536          JMP I EFUN31    /RETURN FROM CALL
Ø6165 ØØØØ  XPUT1,  Ø

            /ROUTINE TO CHANGE OUTPUT MODE FROM LOW TO HIGH OR FROM
             /HIGH TO LOW SPEED. EACH CALL REVERSES THE MODE.

      75Ø3  *SWAP
Ø75Ø3 1Ø16  SWAP,   TAD 16  /WAIT FOR OUTPUT TO FINISH
Ø75Ø4 764Ø          SZA CLA
Ø75Ø5 53Ø3          JMP .-2
Ø75Ø6 1324  SWITCH, TAD CURDEV      /(CURDEV)=2Ø OR -2Ø
Ø75Ø7 7Ø41          CIA
Ø751Ø 3324          DCA CURDEV
Ø7511 1325          TAD ADDRS
Ø7512 3Ø11          DCA 11
Ø7513 1411  LOOP,   TAD I 11
Ø7514 745Ø          SNA
Ø7515 5536          JMP I EFUN31    /RETURN
Ø7516 3323          DCA PLACE
Ø7517 1723          TAD I PLACE     /GET IOT
Ø752Ø 1324          TAD CURDEV
Ø7521 3723          DCA I PLACE
Ø7522 5313          JMP LOOP
Ø7523 ØØØØ  PLACE,  Ø
Ø7524 ØØ2Ø  CURDEV, 2Ø      /SET INITIALLY FOR LOW TO HIGH SPEED
 Ø7525 7525  ADDRS,  ADDRS
```

```
Ø7526  26Ø6        26Ø6   /6Ø41
Ø7527  261Ø        261Ø   /6Ø42
Ø7530  2615        2615   /6Ø44
Ø7531  2711        2711   /6Ø46
Ø7532  2762        2762   /6Ø46
Ø7533  ØØØØ        ØØØØ
```

```
ADD     Ø172
ADDRS   7525
ARG     457Ø
ATLIST  157Ø
BOTTOM .ØØ35
COMGO   1163
CT      ØØ11
CURDEV  7524
DO      Ø42Ø
EFUN3I  Ø136
EVAL    1613
EXP     ØØ44
EX1     ØØ4Ø
FNTABF  Ø374
FNTABL  2165
FRAN    3Ø52
INTEGE  ØØ53
LCOM    4626
LINENO  ØØ67
LOOP    7513
M2P     Ø127
M4      3113
NAGSW   ØØ65
PC      ØØ22
PLACE   7523
POPA    1413
POPF    4544
POPJ    5541
PUSHA   4542
PUSHF   4543
PUSHJ   454Ø
RANDOM  Ø173
RTL6    4557
SAVAC   26ØØ
SHOW    Ø171
SPNOR   456Ø
SWAP    75Ø3
SWITCH  75Ø6
XARG    1343
XARG1   Ø17Ø
XCLR    4643
XCLR1   4655
XFLCN1  2576
XLIST1  2577
XLST    2564
XPUT    616Ø
XPUT1   6165
XTRL    52ØØ
```

REFERENCES

Doll, T. J. A 4-K computer language for experimentation with human subjects. Behavioral Research Methods & Instrumentation. 1972, 4, 27-31.

Griffin, J. D. A. Investigation of CRT control room displays using a computer. Second Canadian Symposium Proceedings, DECUS, Toronto, 1968.

Link, S. W. A computer controlled laboratory for visual perception and human learning. Third Canadian Symposium Proceedings, DECUS, Toronto, 1969.

Link, S. W. Deus Ex Machina. Contemporary Psychology, 1974, 19, 8, 596-597.

Link, S. W. The relative judgment theory of two choice response time. Journal of Mathematical Psychology. 1975, 12.

Matthews, P., & Wescourt, K. Imlac control program for psychological experiments. Department of Psychology. Stanford uiversity, Stanford, California, 1974.

Millman, B. PSYPAL: A computer language for the control of psychological experiments. Department of Psychology Technical Report, University of Calgary, Alberta, 1971.

Reece, P. Some simple I/0 patches for 4K FOCAL. Decuscope, 1973, 12, 23-29.

Siegel, W. Combining FOCAL and assembly language. Behavioral Research Methods & Instrumentation. 1972, 4, 105-106.

Weiss, B. *Digital computers in the behavioral laboratory*. New York: Appleton-Century-Crofts, 1973.

Wrege, D. FOCAL: How to write new subroutines and use internal routines. DECUS: FOCAL-17.

Advanced FOCAL technical specifications. Maynard, Mass: Digital Equipment Corporation, DEC-08-AJBB-DL, 1969.