digital

lab8/e functions for os/8 basic

lab8/e
lab8/e
lab8/e
lab8/e
lab8/e
lab8/e
lab8/e
lab8/e
lab8/e
lab8/e
lab8/e
lab8/e
lab8/e
lab8/e
lab8/e

digital equipment corporation

DEC-8E-ALOSA-A-D

LAB8/E

FUNCTIONS FOR

OS/8 BASIC

For additional copies, order No. DEC-8E-ALOSA-A-D from Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

First Printing
September, 1972

CONTENTS

INTRODUCTION

The addition of LAB8/E functions to OS/8 BASIC enables the user to solve a range of real-time and pseudo real-time problems using a higher-level language. The benefits of approaching real-time problems using BASIC are numerous. A novice programmer can solve problems with little or no assembly language expertise, and in general, the programming effort required for specific problems is dramatically reduced.

The approach taken for specifying each function was to maximize functional flexibility rather than to stress simplicity. Slaving the computer to external events is accomplished by recognizing Schmitt Trigger 'firings'. One of the design goals for the LAB8/E functions was to utilize memory efficiently for single precision and displayable data arrays. Another design goal was to incorporate a masking ability for the recognition of bit patterns when reading digital data. This feature, for example, allows easy conversion of decimal data into floating point format when data is received from decimal devices interfaced to the LAB8/E's digital input registers (DR8-E's).

I. GENERAL DESCRIPTION

This program contains a set of twelve functions which enable a user of OS/8 BASIC to utilize the following peripherals on a LAB8/E: A/D converter, VC8-E display control, DK8-ES real-time clock, and DR8-EA 12-channel buffered digital I/O. All functions, contained in an overlay called BASIC.UF, reside in the overlay area of BASIC (34ØØ-4577) with the understanding that the entire set of functions is in core whenever a given function is in use. Each function is called by a suitable three-character name, followed by any necessary arguments.

General regulations on arguments passed by the user functions in this package:

1. All arguments must lie within the following range:

   Ø≤ARGUMENT≤4Ø95

   Hence, negative arguments (<Ø) will cause a fatal error, FM; and positive arguments greater than 4Ø95 (>4Ø95) will cause the fatal error, FO. Fatal errors terminate program execution and return the user to command mode.

2. Furthermore, certain functions in this package require that the arguments be further restricted. These restrictions will be stated along with the discussion of each function later on. Argument errors due to these added restrictions will cause the fatal error, IA (illegal argument).

II. PREPARING BASIC FOR LAB8/E FUNCTIONS

The Basic Run Time System (BRTS) has made provision for one overlay area (see section 10.3.2) of the BASIC User's Manual) and has divided a set of infrequently used functions into three separate overlays; namely, BASIC.AF, BASIC.SF and BASIC.FF. Since a logical need for user-written assembly language subroutines exists, a last overlay, BASIC.UF was reserved. It is this last overlay that contains the twelve functions for LAB8/E support. Since the subroutines of this last overlay are determined apart from BRTS, it is necessary that BRTS be given a list of core addresses for each of the user subroutines. (See section 10.10 of BASIC Manual.) It is critical that the order of specifying these links or addresses be in the same order that the UDEF statements will appear in the program that calls the functions.

Consequently, before writing any program using these functions, it is absolutely necessary to modify BRTS. The following example illustrates how this is done. Take notice in the test programs at the end that the order in the UDEF statements is the same as the ordering of the addresses here. In the example any response by OS/8 or ODT will be underlined. A set of four *'s indicates the current contents of a location which is to be changed with the link addresses. A list of the names of the functions associated with each address is specified to the right for the sake of clarity only[1].

```
.GET SYS BRTS.SV
.OD
1/****54Ø2                      used for interrupts
ØØØ2/**** 4456
156Ø/**** 34ØØ                  INI
1561/**** 3454                  PLY
1562/**** 3473                  DLY
1563/**** 36ØØ                  DIS
1564/**** 4ØØØ                  SAM
1565/**** 41ØØ                  CLK
1566/**** 3541                  CLW
1567/**** 3521                  ADC
157Ø/**** 44ØØ                  GET
1571/**** 4432                  PUT
1572/**** 4271                  DRI
1573/**** 4313                  DRO
↑C
.SAVE SYS BRTS.SV
```

[1]The symbol indicates that the RETURN key is pressed, and the symbol indicates that the LINE FEED key is pressed.

A warning: Since many of BASIC's functions also reside in overlays, one is cautioned about using a function that will cause the current set of functions to be overlayed and thereby destroy any useful information.

EX1 One cannot calculate a set of cosine values and pass them to the PLY function to be stored away, because COS resides in BASIC.AF overlay and PLY resides in BASIC.UF.

EX2 Refer to TST18A.PG at the end. Notice that the INI call of line #29 occurs after the file functions were called. If INI were specified instead at line #1Ø, its information would have been destroyed by the file functions.

III. DEFINITION OF LAB8/E SUPPORT FUNCTIONS

A. Once BRTS has been modified to recognize the user function from the BASIC.UF overlay, BASIC programs making use of these functions may be written. Part B contains a complete description of each of the twelve functions. Part C contains a set of BASIC programs illustrating the use of the various functions.

Once again it is very important to reiterate a rule concerning the UDEF statements (see NOTE of section 10.10 of the BASIC User's Manual DEC-S8-LBASA-A-D). If a program requires the use of the Nth function in the ordered list of links, the first (N-1) functions of the list must be defined by UDEF statements or a set of (N-1) dummy-named functions must precede the defining of the Nth function.

EX. Referring to the ordered list of functions in the previous section, if the ADC function is the only one to be used in a particular BASIC program, the UDEF statements must be:

```
            .
            .
1Ø  UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
11  UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
            .
            .
                    OR
            .
            .
1Ø  UDEF DUA(N),DUB(N),DUC(N),DUD(N)
11  UDEF DUE(N),DUF(N),DUG(N),ADC(N)
            .
            .
```

However, it is recommended that one always use the complete set of UDEF's each time one requires one or more functions in a program. This is recommended solely to keep careless omissions to a minimum. This is done in each of the BASIC programs illustrated in Part C.

B. FUNCTIONS

1. INI(N): The *initialize* function has a twofold purpose.

Its main purpose is to locate the address of the array specified by BASIC's USE statement and retain that address until BASIC.UF is overlayed by one of the other three overlays.

A secondary purpose is to set a pointer to the first location of the array. Consequently, an array may be used to store one set of data followed immediately by a second set of data, provided the INI function was called only once. This means that displayable data (10 bits), and fixed point data (12 bits) may share the user array at the user's discretion. If, however, the INI function was again specified at the end of the first data run, the first set of data is overwritten by the second set of data. Hence, INI effectively zeros the array in this case. Whenever an array is to be used in conjunction with one or more of the functions in the BASIC.UF overlay, one first dimensions the array and eventually employs the USE statement (see section 10.8.1 of the OS/8 BASIC User's Manual) before the INI function can have meaning.

Ex.
```
DIM A(3)
  .
  .
  .
USE A
  .
  .
  .
X=INI(Ø)
  .
  .
  .
```

The argument N, for INI, is a dummy argument, and may be any integer; Ø, 1, 2, ...

N.B. 1. Whenever the functions PLY, DIS, SAM, GET, and PUT are used, make sure that the INI function has been previously called at least once.

2. A reminder: when an array is given the dimension N, BASIC allocates (N+1) floating point words of memory which is actually 3(N+1) single memory locations. Thus, in the example above, BASIC allocates 4 floating point words or 12 single memory locations for the array.

3. Each data value deposited into the user's array by the user functions is a single precision value (uses one memory word).

2. PLY(Y): The purpose of the *plot* function is to enable a BASIC program to create y-data values and enter them into the user array sequentially, beginning with the first unused location of the array. Each floating point value is fixed to a ten (1Ø) bit single precision value before it is put into the array.

The range of the y-data values must be:

Ø≤y<1.Ø

This is easily accomplished by inserting a scaling factor. (Refer to line numbers 26 and 64 of TESTØA.PG in Part C.)

The data in the user array can be displayed as it is being passed to the array (see DLY function) and/or be refreshed continuously once all values have been entered into the array (see DIS function).

3. DLY(N): The *delay* function is used only in conjunction with the PLY function. It causes the scope to be refreshed with the contents of the user array after each point is processed, so that the graphical progress of data can be observed.

N is an integer such that 1≤N≤1Ø24. It specifies the maximum number of points to be eventually displayed. Implied here is the fact that the display will contain only the first N points even if the arrays contain more than N points.

BASIC programs in Part C illustrating the use of DLY are TESTØA.PG, TEST2A.PG, TEST7A.PG, TEST9A.PG and TST19A.PG.

4. DIS(S,E,N,X): The *display* function is used to set up parameters for the displaying of y-data stored in the user array. The display will begin with the desired starting point, S of the array and display every Nth point while not exceeding the desired endpoint, E (where N = 1, 2, 3, ...).

Depending on the value of X, the DIS function has two separate operations.

a. Operation when X equals zero (X=Ø): Indication is given to the user-overlay-functions that a SAM function will be the next BASIC instruction. Consequently the parameters mentioned above are set up so that exactly one of the sampled channels can be displayed 'on the fly'. To understand the use of the arguments S,E,N,X; it is necessary to know how the A/D data is stored in the user array.

EX. Assume 1ØØ samples/channel in each case.

ARRAY	CASE 1 SAM CH#Ø	CASE 2 SAM CH #3,4,5
WD1	CH#Ø	CH#3
WD2	CH#Ø	CH#4
WD3	CH#Ø	CH#5
WD4	CH#Ø	CH#3
WD5	CH#Ø	CH#4
WD6	CH#Ø	CH#5
⋮	⋮	⋮
WD1ØØ	CH#Ø	CH#3

To display CASE1, once sampling begins:

DIS(1,1ØØ,1,Ø)

To display CH#4 of CASE2, once sampling begins:

DIS(2,1ØØ,3,Ø)

b. Operation when X is greater than zero (X>Ø): A user array of y-data is to be displayed immediately. The display is continually refreshed (no return to BASIC) until the operator types CTRL/N on the keyboard.

Note 1. Displayable y-data values are assumed to be 1Ø-bit single precision data words.

2. The x-coordinate for each y-data value is determined by a DELTAX value found as follows:

DELTAX = 1Ø23/[(E-S)/N]

Due to the outcome of DELTAX, the display may not always use the full width of the scope. However, the display is always centered.

3. $S \geq 1$; $E \geq S$; $(E-S)/N \leq 1Ø23$. At least one point must be displayed and no more than 1Ø24 points may be displayed.

5. SAM(C,N,P,T): The *sample* function is used solely to set up parameters for subsequent sampling of the ADC's or for subsequent sampling of digital input registers (Ø,1,2) depending on the value of T.

a. TASK 1 (T=Ø): Sample the ADC's

C = First channel # to be sampled; $Ø \leq C \leq 17_8$.

N = Number of consecutive channels to sample; $1 \leq N \leq (2Ø_8 - C)$.

P = Number of sample points/channel; $P \neq Ø$.

b. TASK 2 ($T \neq 0$): Sample digital input registers.

C = First register # to be sampled; $0 \leq C \leq 2$.

N = Number of consecutive input registers to sample; $1 \leq N \leq (3-C)$.

P = Number of samples/register; $P \neq 0$.

Note 1. Anytime a SAM instruction is used to sample the ADC's, exactly one channel must be displayed on the fly. However, the sampling rate is not slowed down by this requirement. Hence a DIS function call must precede a SAM function call whenever TASK 1 is chosen.

2. It is possible to display digital input data as long as it is understood that only the least significant 10 bits will be displayed. However, this data can not be displayed 'on the fly' and can only be displayed via the DIS function once all data is in the array.

6. CLK(R,O,S): The clock function sets up the clock to be used for A/D sampling, for digital input sampling, or as a simple timing device.

R(rate) = desired frequency at which to run the clock

Value of R	Frequency
1	External input
2	100 HZ
3	1K HZ
4	10K HZ
5	100K HZ
6	1M HZ

O(overflow CNT) = number of clock ticks per interrupt with the clock running at the desired frequency, R. $0 \leq O \leq 4095$

S (Schmitt trigger) ($S \neq 0$) = Activate all Schmitt triggers and start the clock when any one of the three Schmitt triggers fires.

($S=0$) Do not activate any Schmitt triggers and start up the clock immediately.

As mentioned above, this single clock function is used to set the clock for one of three separate tasks.

TASK1: Sample the ADC's.

The interrupts are turned on[1] and the program waits in the display loop for a clock overflow; at which time the A/D channel(s) is (are) sampled. The display loop will display the data for the channel specified by the user in the DIS function. When all channels have been sampled the requested number of times, the CLK function returns to BASIC.

TASK2: Sample digital input registers.

At each clock overflow, the digital input register(s) is (are) sampled. When all registers have been sampled the requested number of times, the CLK function returns to BASIC.

N.B. The sampled data from the ADC's or the digital input registers is stored sequentially in the user's array.

TASK3: A simple timing device.

The clock is set up and started (unless it is to be started when a Schmitt trigger fires) and then returns to BASIC.

The following illustrates what sequence of instructions are needed for each task.

```
TASK1               TASK2               TASK3
  :                   :                   :
DIM A(n)            DIM A(n)            Z=CLK(R,O,S)
USE A               USE A                 :
  :                   :
W=INI(Ø)            W=INI(Ø)
X=DIS(C,N,P,T)      Y=SAM(C,N,P,1)
Y=SAM(C,N,P,Ø)      Z=CLK(R,O,S)
Z=CLK(R,O,S)          :
  :
```

[1]When interrupts are turned on, the only possible valid interrupts can be caused by the keyboard or the clock. Hence, any other interrupt is an uncontrollable, spurious interrupt (faulty hardware) which will cause a HLT at location 4466. If this happens, do the following:

a) set SWITCH REGISTER to 4476 and hit the ADDR LOAD switch on the console.

b) Next, hit the CLEAR and CONT switches on the console. This will return you to BASIC.

c. Typing CTRL/C will return you to the OS/8 Monitor.

7. <u>CLW(N):</u> With the clock having been set up by CLK as a simple timer, this *clock wait* function, when called, simply returns to BASIC whenever a clock overflow occurs; and/or whenever a Schmitt trigger fires provided S was a non-zero argument in CLK.

Upon return to BASIC, a number is returned to the caller indicating whether the return was due to a clock overflow, a Schmitt trigger, or a clock overflow and the firing of a Schmitt trigger simultaneously. The number also indicates whether one of the above conditions occurred before or after the CLW function was called. N is a dummy argument (N=Ø,1,2 ...).

Below is a table illustrating the various numbers returned.

a. Case 1: Clock overflowed or a Schmitt trigger fired <u>after</u> CLW is called.

Overflow only	Schmitt Trigger Only	Simultaneously
Ø	1 (Trigger 1 fired)	-1
	2 (Trigger 2 fired)	-2
	3 (Trigger 1&2 fired)	-3
	4 (Trigger 4 fired)	-4
	5 (Trigger 1&4 fired)	-5
	6 (Trigger 2&4 fired)	-6
	7 (Trigger 1,2&4 fired)	-7

b. Case 2: Clock overflowed or a Schmitt trigger fired <u>before</u> CLW is called.

Overflow only	Schmitt Trigger only	Simultaneously
-8	9	-9
	10	-10
	11	-11
	12	-12
	13	-13
	14	-14
	15	-15

In Part C, TEST4A.PG and TEST5A.PG make use of the CLW function.

The CLW function has many useful applications. Subroutine timing may be accomplished by starting the clock with a specific rate and overflow count. The subroutine is called, and at the end of the subroutine the CLW function is called to see if an immediate return is obtained. This timing is empirical in so far as one would keep changing the rate and/or overflow count until Case 2 occurred. Secondly, Schmitt trigger firing may be used to branch to a particular subroutine or to notify the program to proceed with specific tasks such as reading digital data or sampling an analog input. Thirdly, time interval histograms and and post stimulus histograms are also possible (see TST2ØA.PG of Part C).

8. ADC(N): This function is issued any time one wishes to sample A/D channel N. The 1Ø bit data value is floated and returned to the caller for immediate examination. $Ø \leq N \leq 17_8$.

The BASIC statement W=ADC(3) asks that A/D channel #3 be sampled and the floating point value be assigned to W.

TEST5A.PG of Part C illustrates one use of the ADC function.

9. GET(M,L): This function is used to get one 12 bit word from the user array, mask out certain bits and return the result as a floating point number to the caller.

L = Lth location of the user array. Hence, if an array has N single precision words, L can take on meaningful values of 1,2,3,...,N.

Note: Although BASIC allows Ø to be a meaningful value in a dimension statement such as DIM A(Ø), it must be understood that L always begins with 1, where 1 stands for the first single-word location of the array. Thus DIM A(Ø) specifies an array of one floating point word (three one-word locations).

M = A masking number such that $Ø \leq M \leq 4Ø95$. This floating point number is converted to a 12 bit binary number between Ø and 7777. Those bits that are zero will mask out or eliminate those bits in the array value. If M=Ø, then no masking is done and the 12 bit array value is returned in tact. M=Ø and M=4Ø95 have the same meaning.

The BASIC statement Y=GET(15,2) gets the second word of the user array, masks out all bits except bits 8,9,10,11 and assigns the floating point result to Y. Consequently, if an array is as follows:

$$\left.\begin{array}{ll} \text{single prec WD1} & \{\ 5678 \\ \text{single prec WD2} & \{\ 1234 \\ \text{single prec WD3} & \{\ 4455 \end{array}\right\} \text{Fl. pt. word } Ø$$

⋮

$WD2 = 1234_8 = 001010011100_2$

$MASK = 15_{10} = 17_8 = 000000001111_2$

The 12 bit value after masking is:

$000000001100_2 = 12_{10}$

Hence, Y=12

Note: For user assistance in understanding decimal to octal to binary conversions one is referred to the *Introduction to Programming Handbook*.

10. PUT(M,L): This function enables a floating point number to be fixed to a single 12 bit word and put into the user's array.

L = Lth location of the user's array. For an array of N single precision words, L can take on meaningful values of 1,2,3,...,N.

M = The floating point number to be fixed and stored in the array. $\emptyset \le M \le 4\emptyset 95$.

N.B. Both GET and PUT functions imply that a user's array must not exceed 4Ø96 memory locations, because of the general restriction on any argument for these user functions.

The BASIC statement Y=PUT(128,4) means fix 128 to 12 bits $(000\ 010\ 000\ 000_2)$ and put the value into the 4th word of the user array.

TST15A.PG, TST16A.PG, TST17A.PG and TST18A.PG illustrate the use of functions GET and PUT.

11. DRI(N): This function is issued any time one wishes to sample a digital input register, N $(\emptyset \le N \le 2)$. The 12 bit digital value is returned to the user as a floating point number.

Basic statement: X=DRI(Ø) means that input register #Ø is sampled and the floating point result is assigned to X.

12. DRO(M,N): This function is issued any time one wishes to set the bits of a digital output register, N$(\emptyset \le N \le 2)$. The output register bits are set via the value of M $(1 \le M \le 4\emptyset 95)$. If M=Ø, the output register is cleared, otherwise the bits of the register remain set. Hence, additional bits of the register can be set while maintaining those set earlier.

Basic statement: Z=DRO(9,1) means set bits 8 and 11 of output register #1 if not already set.

$9_{10} = 000000001001_2$

TST13A.PG and TST15A.PG illustrate the use of the DRI and DRO functions.

C. The following set of BASIC programs illustrates a number of ways the user functions may be implemented. Each program has been kept as simple as possible.

It should be pointed out that for TST12A.PG, TST13A.PG and TST15A.PG a battery powered 'black box' was used to interact with the digital I/O registers. The box contained a set of 12 switches which could set any combination of bits for the digital input register, and it also contains a row of 12 lights that were lighted by the contents of the 12 bit digital output register. When running TST18A.PG, use the data from TST17A.PG.

```
1 REM-      PROGRAM NAME:TEST0A.PG
2 REM-
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
6 DIM A(342)
9 REM-
10 REM-CALC 1024 PTS & DISPLY ON FLY.
11 REM- WHEN DONE DISPLY EVERY 10TH PT.
12 REM-
20 USE A
22 Z=INI(0)
24 FOR N=1 TO 1024
26 Y=(3*N-2)/3071
28 X=PLY(Y)
30 W=DLY(1024)
32 NEXT N
34 V=DIS(1,1024,10,1)
49 REM-
50 REM-CALC 30 PTS & DISPLY ONLY WHEN DONE.
51 REM-
60 Z=INI(0)
62 FOR N=1 TO 30
64 Y=(2*N+1)/61.1
66 Z=PLY(Y)
68 NEXT N
70 V=DIS(1,30,1,1)
80 END
```

```
1 REM=  PROGRAM NAME:TEST1A.PG
2 REM=
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,D,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
6 DIM A(342)
10 REM=
11 REM=SAMPLE CHAN 0 (1024 TIMES); DISPLY ALL PTS
12 REM=ON THE FLY.
13 REM=10 INTERRUPTS/SEC
14 REM=
20 USE A
21 W=INI(0)
22 W=DIS(1,1024,1,0)
24 X=SAM(0,1,1024,0)
26 Y=CLK(3,100,0)
28 Z=DIS(1,1024,1,1)
40 REM=
41 REM=SAMPLE CHANNELS 0,1 (100 TIMES EACH).
42 REM=10 INTERRUPTS/SEC; DISPLY CHAN 0 WHILE SAMPLING.
43 REM=WHEN DONE SHOW THREE DIFF DISPLYS:
44 REM=DISPLAY CHAN 0--HIT AN DISPLAY CHAN 1--HIT AN DISPLAY CHANS 0&1.
50 USE A
51 W=INI(0)
52 W=DIS(1,200,2,0)
54 X=SAM(0,2,100,0)
56 Y=CLK(3,100,0)
58 Z=DIS(1,200,2,1)
60 U=DIS(2,200,2,1)
62 V=DIS(1,200,1,1)
70 END
```

```
1 REM=     PROGRAM NAME:TEST2A.PG
2 REM=
3 UDEF INI(N),PLY(Y),OLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
6 DIM A(342)
10 REM=
11 REM=CALC A PARABOLA OF 601 PTS AND DISPLY ON FLY.
12 REM= WHEN DONE DISPLY EVERY 10TH PT OF PARABOLA.
13 REM=
20 USE A
22 Z=INI(0)
24 FOR N=-300 TO 300
26 Y=(N*N)/100000
28 X=PLY(Y)
30 W=OLY(601)
32 NEXT N
34 V=DIS(1,601,10,1)
50 REM=
51 REM=CALC A CUBIC OF 601 PTS & DISPLY ON FLY
52 REM=WHEN DONE DISPLY EVERY 10TH PT.
53 REM=
60 Z=INI(0)
62 FOR N=-300 TO 300
64 Y=(N*N*N+27000000)/54000010
66 X=PLY(Y)
68 W=OLY(601)
70 NEXT N
72 V=DIS(1,601,10,1)
80 END
```

```
1 REM-     PROGRAM NAME:TEST3A.PG
2 REM-
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
6 DIM A(342)
10 REM-
11 REM-ILLUSTRATE ABILITY TO ACCESS USER BUFFER.
12 REM-PUT NUMBERS 1-10 IN TO BUF IN THAT ORDER
13 REM-& READ THEM OUT IN THE REVERSE ORDER.
14 REM-
20 Z=INI(0)
22 FOR N=1 TO 10
24 PRINT N
26 T=N
28 R=PUT(T,N)
30 NEXT N
32 FOR N=1 TO 10
34 M=11-N
36 P=GET(0,M)
38 PRINT P
40 NEXT N
50 END
```

```
1 REM-TEST4A.PG
2 REM-
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,D,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
6 REM-SAMPLE CHAN 0 IF CLOCK O.F.
7 REM-SAMPLE CHAN 1 IF SCHMITT ONLY
8 REM-SAMPLE CHAN 2 IF BOTH FIRE
9 REM-IF EARLY, TELL USER
10 REM-ROUTINE ALSO OUTPUTS Z
11 X=CLK(3,4000,1)
12 FOR N=1 TO 10
14 Z=CLW(0)
15 PRINT "Z=";Z
16 IF Z=0 GOTO 30
18 IF Z<0 GOTO 24
19 IF Z<8 GOTO 34
20 IF Z=8 GOTO 40
21 GO TO 40
24 IF Z<-8 GOTO 40
26 W=ADC(2)
28 GO TO 36
30 W=ADC(0)
31 GO TO 36
34 W=ADC(1)
36 PRINT W
37 GO TO 42
40 PRINT "EARLY"
42 NEXT N
50 END
```

```
1 REM-     PROGRAM NAME:TEST5A.PG
2 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
3 UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
4 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
5 DIM A(342)
10 REM-
11 REM-USE CLK AS A SIMPLE TIMER.
12 REM-SAMPLE CHAN 0 EVERY 4TH SEC & PUT VAL TO TTY.
13 REM-DO THIS 10 TIMES.
14 REM-
20 X=CLK(3,4000,0)
22 FOR I=1 TO 10
24 Y=CLW(0)
26 Z=ADC(0)
28 PRINT Z
30 NEXT I
40 REM-
41 REM-USE CLK AS A SIMPLE TIMER.
42 REM-SAMPLE CHAN 1 TEN TIMES & SYNC OFF ANY SCHMITT TRIG.
43 REM-
50 X=CLK(4,4000,1)
52 FOR I=1 TO 10
54 Y=CLW(0)
56 Z=ADC(1)
58 PRINT Z
60 NEXT I
70 END
```

```
1 REM-    PROGRAM NAME:TEST7A.PG
2 REM-
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,D,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
6 DIM A(342)
7 USE A
8 REM-DISPLAY A TRIANGLE
10 Z=INI(0)
12 FOR N=1 TO 30
14 Y=N/30.1
16 W=PLY(Y)
18 Z=1/30.1
20 U=PLY(Z)
22 P=DLY(118)
24 NEXT N
26 FOR N=1 TO 29
27 M=30-N
28 Y=M/30.1
30 W=PLY(Y)
32 Z=1/30.1
34 U=PLY(Z)
36 P=DLY(118)
38 NEXT N
40 V=DIS(1,118,1,1)
42 END
```

```
1 REM-     PROGRAM NAME:TEST8A,PG
2 REM-
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,D,S),CLW(N),ADC(N)
6 DIM A(342)
10 REM-
11 REM-SAMPLE CHAN 0 100 TIMES; DISPLY;
12 REM-HOWEVER SYNC OFF SCHMITT TRIGS.
14 REM
32 USE A
34 W=INI(0)
36 W=DIS(1,100,1,0)
38 X=SAM(0,1,100,0)
40 Y=CLK(3,100,1)
42 Z=DIS(1,100,1,1)
50 END
```

```
1 REM-   PROGRAM NAME:TEST9A.PG
2 REM-
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
6 DIM A(342)
10 REM-
11 REM-CALC A PARABOLA OF 401 PTS AND DISPLY ON FLY
13 REM-
20 USE A
22 Z=INI(0)
24 FOR N=-200 TO 200
26 Y=(N*N)/40001
28 X=PLY(Y)
30 W=DLY(401)
32 NEXT N
50 REM-
51 REM-CALC A CUBIC OF 401 PTS & DISPLAY ON FLY. SHOW PARABOLA TOO.
52 REM-WHEN DONE DISPLY EVERY PT & THEN EVERY 10TH PT.
53 REM-
62 FOR N=-200 TO 200
64 Y=(N*N*N+8000000)/16000010
66 X=PLY(Y)
68 W=DLY(802)
70 NEXT N
72 V=DIS(1,802,1,1)
74 V=DIS(1,802,10,1)
80 END
```

```
1 REM-TST10A.PG
2 REM-
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,D,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
6 DIM A(3)
7 REM-THIS ROUTN RETURNS 4 DIGITS-3BITS/DIGIT
10 USE A
11 Z=INI(0)
12 PRINT "VALUE"
14 INPUT Y
16 Z=PUT(Y,1)
18 P=GET(7,1)
19 PRINT P
20 P=GET(56,1)
21 PRINT P
22 P=GET(448,1)
23 PRINT P
24 P=GET(3584,1)
25 PRINT P
26 GO TO 12
30 END
```

```
1 REM-     PROGRAM NAME TST12A.PG
2 REM-
3 REM-THIS ROUTN SAMPLES DIGITAL BOARD #1 TEN TIMES
4 REM-ONCE EVERY 4 SEC & PUTS THE VALUES INTO USER BUF
5 REM-THEN IT PRINTS OUT THE 10 VALUES
10 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
11 UDEF SAM(C,N,P,T),CLK(R,D,S),CLW(N),ADC(N)
12 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
20 DIM A(342)
22 USE A
23 W=INI(0)
24 X=SAM(1,1,10,1)
26 Y=CLK(3,4000,0)
28 FOR N=1 TO 10
30 W=GET(0,N)
32 PRINT W
34 NEXT N
40 END
```

```
1 REM=     PROGRAM NAME:TST13A.PG
2 REM=
3 REM-TEST THE OUTPUT REG-SEE THE LIGHTS LITE UP
4 REM=OCTAL INPUT LIGHTS THE LIGHTS AND THE LAMP?
5 REM= AN INPUT OF 0 CLEARS THE OUTPUT REG
10 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
11 UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
12 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
14 W=DRO(0,1)
16 PRINT "NUMBER"
18 INPUT Y
19 IF Y=0 GOTO 14
20 W=DRO(Y,1)
22 GO TO 16
30 END
```

```
1 REM=    PROGRAM NAME:TST15A.PG
2 REM=
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,D,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
6 DIM A(3)
7 REM=THIS ROUTN RETURNS 3 DIGITS=4BITS/DIGIT(MASKING)
8 REM=IT FIRST OUTPUTS THE DECIMAL EQUIV OF THE NUMBER
10 USE A
11 Z=INI(0)
12 W=DRI(1)
13 PRINT W
16 X=PUT(W,1)
18 P=GET(15,1)
19 PRINT P
20 P=GET(240,1)
21 PRINT P
22 P=GET(3840,1)
23 PRINT P
24 PRINT "WASTE TIME"
25 INPUT R
26 GO TO 12
30 END
```

```
1 REM-    PROGRAM NAME    TST16A.PG
2 REM-
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
6 DIM A(3)
7 REM-THIS ROUTN SHOWS THAT ANY N;0<=N<=4095 PUT INTO
8 REM-A USER BUF IS RETURNED AS THE SAME VALUE.
10 USE A
11 Z=INI(0)
12 PRINT "NUMBER"
14 INPUT Y
16 X=PUT(Y,1)
18 Z=GET(0,1)
20 PRINT Z
26 GO TO 12
30 END
```

```
1 REM=    PROGRAM NAME: TST17A.PG
2 REM=FILL AN ARRAY OF 30 WORDS WITH THE FIRST 30 INTEGERS.
3 REM=WRITE THE ARRAY OUT TO DECTAPE.
5 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
6 UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
7 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
8 DIM A(9)
9 USE A
10 X=INI(0)
11 FOR N=1 TO 30
12 PRINT N
13 X=PUT(N,N)
14 NEXT N
16 FILEVN#1:"DTA1:DATA.PG"
22 FOR I=0 TO 9
24 PRINT #1:A(I)
26 NEXT I
28 CLOSE #1
30 END
```

```
1 REM PROGRAM NAME: TST18A,PG
2 REM=READ INTO AN ARRAY 10 FL PT WDS(30 INTEGERS FROM MS)
3 REM=WRITE OUT THE 30 INTEGERS ON TTY
5 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
6 UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
7 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
8 DIM A(9)
9 USE A
20 FILEN #1:"DTA1:DATA,PG"
22 FOR I=0 TO 9
24 INPUT #1:A(I)
26 NEXT I
28 CLOSE #1
29 X=INI(0)
30 FOR N=1 TO 30
32 X=GET(0,N)
34 PRINT X
36 NEXT N
40 END
```

```
1 REM-   PROGRAM NAME : TST19A.PG
2 REM-
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
6 DIM A(16)
10 REM-SAMPLE CHAN 0 50 TIMES; SYNC OFF SCHMITT;
11 REM-10 INTERRUPTS/SEC; WHEN DONE DISPLY TILL AN;
12 REM-THEN WRITE OUT DATA TO DTA1;
20 USE A
21 W=INI(0)
22 W=DIS(1,50,1,0)
24 X=SAM(0,1,50,0)
26 Y=CLK(3,100,1)
28 Z=DIS(1,50,1,1)
29 FILEVN #1:"DTA1:SAM.DA"
30 FOR I=0 TO 16
32 PRINT #1:A(I)
34 NEXT I
36 CLOSE #1
38 REM-DISPLAY A PRABOLA
40 P=INI(0)
42 FOR N=-25 TO 25
44 Y=(N*N)/625.1
46 X=PLY(Y)
48 W=DLY(51)
50 NEXT N
52 V=DIS(1,51,1,1)
54 REM-READ DATA BACK IN & DISPLAY IT AS BEFORE.
56 FILEN #1:"DTA1:SAM.DA"
58 FOR I=0 TO 16
60 INPUT #1:A(I)
62 NEXT I
64 CLOSE #1
65 W=INI(0)
66 Z=DIS(1,50,1,1)
68 END
```

```
1 REM-PROGRAM NAME:TST20A.PG
2 REM-
3 UDEF INI(N),PLY(Y),DLY(N),DIS(S,E,N,X)
4 UDEF SAM(C,N,P,T),CLK(R,O,S),CLW(N),ADC(N)
5 UDEF GET(M,L),PUT(M,L),DRI(N),DRO(M,N)
10 DIM X(100),Y(100),A(67)
11 REM-J1=BINS IN LATENCY(#EPOCHS TILL DONE)
12 REM-T1=BIN WIDTH(TIH) IN MS(#MS/CLK O.F.)
13 REM-T2=BIN WIDTH OF LATENCY(#CLK O.F./EPOCH)
16 PRINT "J1,T1,T2?"
18 INPUT J1,T1,T2
20 I=0
21 J=0
22 K=0
23 Y=CLK(3,T1,1)
25 Z=CLW(0)
30 IF Z=0 GOTO 100
32 IF Z<0 GOTO 36
34 IF Z<8 GOTO 300
35 GO TO 36
36 IF Z>=8 GOTO 200
37 REM-INCR UNDERFLO BIN 0
38 I=0
39 GO TO 300
99 REM-CLK O.F. ONLY;BMP HIST BIN
100 I=I+1
102 IF I<>100 GOTO 110
103 REM-END OF TIME,BMP OVERFLO BIN 100
104 X(100)=X(100)+1
106 I=0
109 REM BMP LATENCY CTR
110 K=K+1
112 IF K<>T2 GOTO 25
113 REM-AN EPOCH IS DONE
114 K=0
116 J=J+1
117 REM-ALL DONE?
118 IF J=J1 GOTO 500
119 REM-MORE EPOCHS TO GO?
120 GO TO 25
199 REM-CLK O.F. AND SCHMITT TRIG
200 X(I)=X(I)+1
202 Y(J)=Y(J)+1
204 GOTO 100
299 REM-SCHMITT TRIG ONLY
300 X(I)=X(I)+1
302 Y(J)=Y(J)+1
304 GO TO 25
498 REM-GET LARGEST BIN VALUE TO BE USED AS A
499 REM-SCALE FACTOR FOR DISPLAY
500 USE A
503 Q=0
504 FOR I=0 TO 100
506 Z=X(I)
508 IF Q>=Z GOTO 516
510 Q=Z
516 NEXT I
549 REM-SCALE ALL BIN VALUES FOR MAX DISPLAY
550 W=INI(0)
551 FOR I=0 TO 100
552 Z=X(I)
```

```
554 Y=Z/(Q+1)
555 W=PLY(Y)
556 NEXT I
598 REM-GET LARGEST LATENCY VALUE TO BE
599 REM USED AS A SCALE FACTOR FOR DISPLAY
600 Q=0
602 FOR I=0 TO 100
604 Z=Y(I)
606 IF Q>=Z GOTO 610
608 Q=Z
610 NEXT I
699 REM-SCALE ALL LATENCY VALUES FOR MAX DISPLAY
700 FOR I=0 TO 100
702 Z=Y(I)
704 Y=Z/(Q+1)
706 W=PLY(Y)
708 NEXT I
710 REM-DISPLAY 'TIH'
711 V=DIS(1,101,1,1)
712 REM-DISPLAY LATENCY
720 V=DIS(102,202,1,1)
725 REM-DISPLAY BOTH 'TIH' & LATENCY SIDE BY SIDE
726 V=DIS(1,202,1,1)
800 END
```

IV. GETTING ON THE AIR WITH OS/8 BASIC

A. DECtape users:

Transfer the user overlays, BASIC.UF from the DECtape, provided with the software kit to the OS/8 system device.

```
.R PIP
*SYS:BASIC.UF<DTAn:BASIC.UF/I (where n=Ø,1,2,...,7)
*↑C
```

B. Papertape users:

Use the ABSLDR to read into core the user overlays which are in binary format on the paper tape, provided with the software kit. Then create a 'save file' on the system device.

```
.R ABSLDR
*PTR:$↑   (where $ symbolizes striking the ALT MODE key)
.SAVE SYS BASIC.UF 34ØØ-4577
```

V. LAB8/E FUNCTION SUMMARY

FUNCTION	EXPLANATION
INI(N)	Locate the address of the user array and initialize a pointer to start of the array. N is a dummy argument.
PLY(Y)	Y-data created via the BASIC program is deposited into the user array sequentially. Ø≤Y<1.Ø
DLY(N)	Used in conjunction with PLY, the scope is refreshed with the contents of the user array after each point is processed. 1≤N≤1Ø24 and N specifies the maximum number of points to be eventually displayed.
DIS(S,E,N,X)	Meaning #1 (X=Ø). Set up paramaters to display ADC data once sampling begins. Meaning #2 (X≠Ø). An array of y-data is to be displayed immediately. In both cases, the display begins with point S of the array, and every Nth point is displayed while not exceeding the desired end point E.
SAM(C,N,P,T)	Used to set up parameters for subsequent sampling of the ADC's (T=Ø) or sampling of digital input registers (T≠Ø). C is the first channel # or digital input register #. N is the number of consecutive channels or registers to sample. P is the number of samples per channel or register.

FUNCTION	EXPLANATION
CLK(R,O,S)	Set up the clock for A/D sampling, digital input sampling or for use as a simple timer. R is the desired rate, O is the overflow count and S activates the Schmitt triggers.
CLW(N)	This function returns to the caller a number, indicating whether the clock overflowed or a Schmitt trigger fired and whether these occurred before or after CLW was called.
ADC(N)	This function is issued any time one wishes to sample A/D channel N.
GET(M,L)	A twelve (12) bit number from the user array at location L is masked with the number M and returned to the caller.
PUT(M,L)	A floating point number, M, is fixed to 12 bits and stored in the user array at location L.
DRI(N)	This function is used any time one wishes to sample a digital input register N.
DRO(M,N)	The bits of digital output register N are set via the value of M.

HOW TO OBTAIN SOFTWARE INFORMATION

Announcements for new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters.

Digital Software News for the PDP-8 & PDP-12
Digital Software News for the PDP-11
Digital Software News for the PDP-9/15 Family

These newsletters contain information applicable to software available from Digital's Program Library, Articles in Digital Software News update the cumulative Software Performance Summary which is contained in each basic kit of system software for new computers. To assure that the monthly Digital Software News is sent to the appropriate software contact at your installation, please check with the Software Specialist or Sales Engineer at your nearest Digital office.

Questions or problems concerning Digital's Software should be reported to the Software Specialist. In cases where no Software Specialist is available, please send a Software Performance Report form with details of the problem to:

Software Information Service
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

These forms which are provided in the software kit should be fully filled out and accompanied by teletype output as well as listings or tapes of the user program to facilitate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

Orders for new and revised software and manuals, additional Software Performance Report forms, and software price lists should be directed to the nearest Digital Field office or representative. U.S.A. customers may order directly from the Program Library in Maynard. When ordering, include the code number and a brief description of the software requested.

Digital Equipment Computer Users Society (DECUS) maintains a user library and publishes a catalog of programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information please write to:

DECUS
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback -- your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability.

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

How can this manual be improved?

Other comments?

Please state your position. ______________________ Date: __________

Name: ______________________ Organization: ______________________

Street: ______________________ Department: ______________________

City: ______________ State: ______________ Zip or Country ______________

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

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754