SOFTWARE: A General-Purpose External Function for PDP-11 BASIC

E. C. Oakley
R. F. Systems Development Section

This article describes a new tool to dramatically simplify the test and software development phases in computer-controllable subsystems for the DSN. This tool does not add to the endless computer language proliferation, but instead adds dimension to a well-established, high-level, moderately sophisticated language to enable simplified control of minicomputer peripherals. Some of its versatility is demonstrated by programs used to aid its own implementation in hardware exercisers.

I. Introduction

The DSN is being called upon to provide multi-mission support capability with a minimum of down-time and spacecraft data loss due to system reconfiguration. The speed and accuracy required dictate that the subsystems involved be computer-controllable.

The general-purpose minicomputer is ideal for the test and control of such subsystems by providing the speed, versatility, and availability to be solely dedicated to the controlling task. The minicomputer generally has less memory available than larger computers, usually requiring that assembly-language programming be employed to realize best core utilization.

Assembly-language programming (Ref. 1) is a formidable obstacle to the speedy development of computer-controllable hardware (Ref. 2), because of the portion of time required for software development. Frequent program changes necessitate long edit/assembly cycles.

A high-level interpretive programming language, such as BASIC, offers the hardware designer a software environment providing ease of editing with immediate "RUN" capability. Helpful error diagnostics are supplied, and programming errors can rarely reach "self-destruct" proportions. Higher-level languages usually do not provide for control of nonstandard external peripherals. BASIC (Refs. 3 and 4), as implemented for the Digital Equipment Corp. PDP-11 (Ref. 5), does provide for a user-designed external function (EXF) to perform a specified operation.

II. Design

A general-purpose EXF has been written for BASIC in PDP-11 assembly language (Fig. 1 and Ref. 6), and uses 48_{10} words of core. It can move data to and from any peripheral device located along the PDP-11 Unibus. This includes analog-to-digital converters (ADCs), digital-to-analog converters (DACs), the general-purpose interface DR-11A, plus all the standard peripherals including the core memory itself.

The EXF call in BASIC user-programs consists of a parenthetical expression, such as:

(LINE #) LET X = EXF (A,D,M)

OR

(LINE #) PRINT EXF (A,D,M)

OR

(LINE #) IF EXF (A,D,M) >X THEN STOP

The expression contains three arguments separated by commas, any one of which may be numbers, assigned-variable letters (with or without subscripts), or zero (Ø), but all arguments must always be present. The first argument, A, is the decimal ADDRESS of the location to be accessed. The second argument, D, is the decimal value of the DATA to be moved. The third argument, M, is the command MODE of the function to be performed on the data. M's value may equal +1 for the LOAD mode, −1 for the READ mode, or Ø for an optional special mode function to be described in later paragraphs.

For the first EXF call cited (LINE #) LET X = EXF (A,D,M), the variable X will assume the decimal value of data at the address being accessed. This means that the second argument (DATA field) of an EXF call to READ an address will normally be zero (Ø) since the decimal value of the data will be returned to BASIC and be assigned to the variable X, as follows:

(LINE #) LET X = EXF (A,Ø,−1)

The ADDRESS field argument A must be an even number so that even addresses (whole words) will be accessed. An attempt to access data at odd addresses (upper bytes) will cause a fatal error, and EXF will not return control to the BASIC interpreter.

Data arbitrarily loaded into core locations occupied by EXF, BASIC, its user area, or stack may cause them to be destroyed, although careful, deliberate modification of BASIC or EXF core locations by using EXF calls under program control can yield considerable expansion of BASIC's capabilities.

Decimal address and data values are restricted by the 16-bit word size to integers in two's-complement form in the range ±32767.

The software interface for the EXF in BASIC is defined mainly in Chapter 8 of Ref. 3.

EXF is fully re-entrant. That is, a statement which requires the call to be re-entered before the first call is complete:

(LINE #) LET X = EXF (A1, EXF (A2,Ø,−1), +1)

will fetch (READ) data from one peripheral, and immediately output (LOAD) it into a second peripheral with minimum interpreter execution time.

III. Applications

The EXF conditional expression

(LINE #) IF EXF (A,D,M) <>X THEN STOP

could be used to test the value of a status word (input to a DR-11A general-purpose interface) from a typical DSN subsystem. As an example, assume that the value of the correct status for the subsystem is X = 25732, and a test of the input DR-11A reveals a mismatch; an alarm can then be sounded by continuously ringing the teletype bell as follows:

```
1ØØ LET S = −8188: REM − DR-11A AT LOC. 16ØØØ4[8].
11Ø IF EXF (S,Ø,−1) <>25732 THEN PRINT "": GOTO 11Ø
12Ø REM − CTRL/G (BELL) BETWEEN QUOTES IN
13Ø REM − LINE 11Ø; ALARM
```

By subtraction, the improper status bit can be detected, a diagnostic message printed to alert the user, and corrective action initiated to re-zero the errant phase shifter:

```
11Ø IF 25732-EXF (S,Ø,−1) = 8 THEN GOTO 2ØØ
.
.
.
2ØØ PRINT "PHASE SHIFTER TO BE RE-ZEROED..."
```

```
21Ø REM–RE-ZEROING ROUTINE FOLLOWS:
22Ø
.
.
.
```

Figure 2 shows a program for exercising the digital step attenuator (Ref. 2) of the Ranging Demodulator Assembly. It is used for generating time-linear control ramps and steps to aid in the test of attenuator parameters such as monotonicity and phase shift. Instructions are printed to direct the user, and convenient pauses are introduced to allow instrumentation range changes to properly display the results on a strip-chart recorder, as measured by a vector voltmeter.

Figure 3 shows a BASIC-EXF program for taking voltage measurements with the Digital Equipment Corp. ADØ1-D analog-to-digital converter. The program assembles a control word from Channel and Gain data input by the user. This peripheral control technique is straightforward, requiring separate EXF statements for initiating ADØ1-D conversion, testing for 'DONE' and then reading the data buffer.

If the user requires more program execution speed than this straightforward method can yield, an interesting alternative is available. EXF can be used to append a short Assembly-Language driver to the BASIC user program to improve peripheral handling speed by several orders of magnitude. This is accomplished by first writing the Assembly-Language program and then converting the octal machine instructions to decimal values. These values are entered into the BASIC user program as DATA statement elements. These data are loaded into the Loader core area (which is otherwise totally unavailable to BASIC), by using the EXF(A,D,+1) LOAD mode. During program execution, these machine instructions are then branched-to by using the EXF(A,D,Ø) ZERO mode, yielding about a thousand-fold increase in execution speed for servicing the desired peripheral. Figure 4 shows such a program which will cycle the ADØ1-D analog-to-digital converter at a much higher rate than the straightforward program shown in Fig. 3.

Using the technique just described, Fig. 5 shows a program for providing power-fail protection for any BASIC user program. In this case, the power-fail-save machine-language routine is branched-to only when the computer hardware detects that a power-down or subsequent power-up condition exists. This software routine will prevent catastrophic execution of a DSN subsystem test or control program, saving the user much time required for system re-initialization. Of course, the user must not HALT the processor at the console front panel.

IV. Concluding Remarks

EXF greatly simplifies input–output programming tasks, within BASIC's language environment. Examples of techniques for implementing programs for the test and control of DSN subsystems have been shown. EXF can add much flexibility to ordinary computational BASIC programs not requiring peripheral control capability.

References

1. *PDP-11 Paper Tape Software Programming Handbook,* DEC-11-GGPA-D. Digital Equipment Corporation, Maynard, Mass., June 1970.
2. Oakley, E. C., "Digital Step Attenuator," in *The Deep Space Network Progress Report,* Technical Report 32-1526, Vol. III, pp. 211–214. Jet Propulsion Laboratory, Pasadena, Calif., June 15, 1971.
3. *PDP-11 BASIC Programming Manual,* DEC-11-AJPB-D. Digital Equipment Corporation, Maynard, Mass., Dec. 1970.
4. Knight, D., and Scott, J., *Listing of PDP-11 BASIC VØØ7A,* DEC-11-AJPB-LA. Digital Equipment Corporation, Maynard, Mass., Nov. 5, 1970.
5. *PDP-11 Handbook,* Second Edition, No. AJ0. Digital Equipment Corporation, Maynard, Mass., 1969.
6. Program Assembly Listing, External Function for PDP-11 BASIC, JPL sketch No. B137631.

```
; PDP-11 BASIC EXTERNAL FUNCTION.  2-7-72.
; USES 96 BYTES (48 WORDS) OF CORE.
; FORM:   EXF(ARG1,ARG2,ARG3)
; ARG1 IS THE ADDRESS TO BE ACCESSED
; ARG2 IS THE DATA
; ARG3 IS THE FUNCTION TO BE PERFORMED.  CODES ARE:
;      -1 = READ
;       1 = LOAD
;       0 = SPECIAL.  LOADED BY USER WITH BASIC USING
;           EXF 1 (LOAD).  MUST BEGIN AT 37500 AND
;           END WITH 'BR FIN' TO LOCATION 37440.
        R0=%0
        R1=%1
        R2=%2
        R3=%3
        R4=%4
        SP=%6; STACK POINTER
        FLT=104436  ; BASIC INTERNAL ROUTINE
        FIX=104440  ; BASIC INTERNAL ROUTINE
        EVAL=104536 ; BASIC INTERNAL ROUTINE
        .=50; TRAP LOC. FOR EXF STARTING ADDRESS
        .WORD FX
        .=37342; STARTING ADDRESS FOR EXF ROUTINE
FX:     MOV R2,-(SP); SAVE DATA IN REGISTERS
        MOV R3,-(SP)
        MOV R4,-(SP)
        MOV (R0)+,R2; PREPARE TO CONVERT TO INTEGER
        MOV (R0)+,R3
        MOV (R0),R4
        FIX; CONVERT 3-WORD FPN TO SWI
        MOV R0,-(SP); STORE ADDRESS ON STACK
        CLR -(SP); INDICATE ARG BEING INTERPRETED
LOOP:   INCB R1; PAST COMMA (POINTER TO STRING IN R1)
        EVAL; BEGINS AT (R1), RESULT IN R2, R3, R4
        FIX; CONVERT 3-WORD FPN TO SWI
        MOV R1,R2; MOVE 'EVAL' SCAN POINTER TO R2
        TST (SP)+; TEST ARG COUNTER ON STACK
        BGT ARG3; BRANCH IF FUNCTION ARGUMENT
        MOV R0,ARG2; STORE DECIMAL INTEGER VALUE OF DATA
        INC -(SP); INCREMENT ARG COUNTER ON STACK
        BR LOOP; LOOP BACK FOR NEXT ARGUMENT
ARG3:   TST R0; TEST FUNCTION CODE.
        BMI READ; IF FUNCTION CODE IS NEGATIVE
        BEQ 37500; IF FUNCTION CODE IS ZERO
LOAD:   MOV ARG2,@(SP); DATA IN ARG2 TO ADDR IN ARG1
        MOV @(SP)+,R1; MOVE TO R1 TO CONVERT TO FPN
        BR FIN
READ:   MOV @(SP)+,ARG2; CONTENT OF ADDR IN ARG1 TO ARG2
        MOV ARG2,R1; FOR CONVERSION TO FLOATING POINT
FIN:    MOV #ARG2,R0; NEED A CORE ADDRESS FOR FLT
        MOV R2,-(SP); SAVE 'EVAL' SCAN POINTER ON STACK
        FLT; REFLOAT THE RESULT -- SWI TO FPN
        MOV (SP)+,R1; RESTORE 'EVAL' SCAN POINTER
        MOV (SP)+,R4; POP STACK TO RESTORE REGISTERS
        MOV (SP)+,R3
        MOV (SP)+,R2
        MOV -(R0),-(SP); STACK RESULT TO OUTPUT TO BASIC
        MOV -(R0),-(SP)
        MOV -(R0),-(SP)
        JMP 52; RETURN TO BASIC INTERPRETER
ARG2:   0,0,0; DATA / FLT STORAGE
        .END 52
```

Fig. 1. BASIC-EXF program assembly listing

```
LIST

1 REM--DIGITAL ATTENUATOR EXERCISER.  2-14-72.
5 REM--THIS PROGRAM CONTROLS AN 8-BIT DIGITAL ATTENUATOR THROUGH
10 REM--A DR-11A INTERFACE AT LOCATION 160002[8] (-8190[10]).
20 PRINT: PRINT "<*> DIGITAL ATTENUATOR CONTROL PROGRAM <*>": PRINT
045 PRINT "DO YOU DESIRE TO EXERCISE THE ATTENUATOR BY:"
050 PRINT ,"(KEY 1) AUTO-RAMP FOR PHASE/CONFIRMATION CHECK;"
055 PRINT ,"(KEY 2) AUTO-RAMP WITH HALTS AT FIVE EQUAL POINTS"
060 PRINT ,"         (TO CHANGE VOLTMETER RANGE) FOR CHECKING"
065 PRINT ,"         MONOTONICITY;"
070 PRINT ,"(KEY 3) AUTO MAJOR STEP;"
075 PRINT ,"(KEY 4) MANUAL STEPPING;";
080 INPUT D: IF D=1 THEN 100
082 IF D=2 THEN 200
085 IF D=3 THEN 300
090 IF D=4 THEN 400
095 PRINT "TRY AGAIN...1,2,3 OR 4!";: GOTO 080
100 LET C=EXF(-8190,0,1): REM--SET TO MIN. INS. LOSS.
105 PRINT "AUTO-RAMP (PHASE/CONFIRM).  RETURN WHEN READY...";: INPUT H
110 FOR B=1 TO 255: LET C=EXF(-8190,B,1)
115 LET T=5: REM--RAMP RUNS 64 SECS.
120 LET T=T-1: IF T>0 THEN 120
125 IF B=C THEN 155
130 PRINT: PRINT "" "CONFIRM LOST AT STEP" B "(" B/5 "DB. )"
135 REM--FIVE CTRL/G'S (BELL) BETWEEN FIRST QUOTES IN LINE 130.
155 NEXT B
160 LET C=EXF(-8190,0,1): REM--SET TO MIN. INS. LOSS.
165 PRINT "RAMP IS COMPLETE.  CONFIRM IS CORRECT, EXCEPT AS NOTED."
170 PRINT: GOTO 080
200 LET C=EXF(-8190,0,1): REM--SET TO MIN. INS. LOSS.
205 PRINT "HALTING AUTO-RAMP.  RETURN WHEN READY...";: INPUT H
210 LET A=0.2
215 FOR K= 1 TO 5
220 FOR A= A TO A+10 STEP 0.2
225 LET C=EXF(-8190,A*5,1)
228 LET T=10: REM--ONE-FIFTH RAMP RUNS 1 MIN., 29 SECS.
232 LET T=T-1: IF T>0 THEN 232
235 NEXT A
240 PRINT: PRINT "CHANGE VOLTMETER -10 DB.  RETURN WHEN READY...";
245 INPUT H: LET A= A+0.2: NEXT K
250 PRINT "RAMP IS COMPLETE.": PRINT: GOTO 080
300 LET C=EXF(-8190,0,1): REM--SET TO MIN. INS. LOSS.
305 PRINT "AUTO MAJOR STEP.  RETURN WHEN READY...";: INPUT H
310 FOR Q= 0 TO 7
315 PRINT EXF(-8190,2↑Q,1)
320 LET T=150: REM--EACH STEP RUNS 9.2 SECS.
325 LET T=T-1: IF T>0 THEN 325
335 NEXT Q
340 LET C=EXF(-8190,0,1): REM--SET TO MIN. INS. LOSS.
350 PRINT "RAMP IS COMPLETE.": PRINT: GOTO 080
400 PRINT "WHAT IS THE DESIRED ATTENUATION (A) IN DECIBELS?"
405 PRINT "(RESTRICT RANGE FROM 0.0 TO 51.0 DB. IN 0.2 DB. STEPS.)"
410 PRINT "A=";: INPUT A: PRINT "  ";
415 IF A<51.2 THEN 450
420 LET A=A-51
425 IF A>51 THEN 420
450 LET C=EXF(-8190,A*5,1)
455 PRINT C/5 "DB. CONFIRMED."
460 GOTO 410
8191 END
READY
```

Fig. 2. Digital attenuator exerciser program

```
LIST

001 REM--DATE OF THIS REVISION IS 12-6-71.
050 PRINT
060 PRINT "ADO1-D ANALOG-TO-DIGITAL CONVERTER EXERCISER."
070 PRINT "AVERAGES TEN READINGS.": PRINT
095 PRINT "SELECT CHANNEL (0 TO 15), AND GAIN (1,2,4 OR 8);"
100 PRINT "INPUT IN THIS FORM--   C,G"
110 INPUT C,G
120 IF INT(C)>15 THEN GOTO 95
130 IF C<0 THEN GOTO 110
140 LET B=INT(C)*256
210 IF G=1 THEN GOTO 260
220 IF G=2 THEN GOTO 260
230 IF G=4 THEN GOTO 260
240 IF G=8 THEN GOTO 260
250 GOTO 95
260 IF G=1 THEN LET A=0
270 IF G=2 THEN LET A=8
280 IF G=4 THEN LET A=16
290 IF G=8 THEN LET A=24
300 LET Y=0
350 FOR N=1 TO 10
400 GOSUB 1050
450 LET Y=Y+V
500 NEXT N
550 LET V=Y/10: REM--AVERAGE OF 10 READINGS.
700 IF G<3 THEN PRINT INT(V*100+.5)/100 "VOLTS": PRINT
800 IF G>3 THEN PRINT INT(V*1000+.5)/1000 "VOLTS": PRINT
900 REM--ROUND-OFF TO PROPER SIGNIFICANCE.
1000 GOTO 110
1040 REM
1050 LET X=EXF(-520,A+B+1,1)
1060 REM--LOAD ADC C&SR WITH GAIN, CHANNEL AND START.
1100 IF (EXF(-520,0,-1))<128 THEN 1100
1110 REM--TEST ADC ERROR AND DONE BITS.
1120 LET D=EXF(-518,0,-1)
1130 REM--READ ADC DATA REGISTER.
1200 LET V=D*9.765625E-3/G
1210 REM--RESCALE AND PROPORTION DATUM.
1220 RETURN
1230 END
READY
```

Fig. 3. Analog-to-digital converter exerciser program (slow)

```
LIST

 1 REM--AD01-D EXERCISER WITH EXF0.  3-6-72.
 10 PRINT: GOTO 8000
 60 PRINT "AD01-D ANALOG-TO-DIGITAL CONVERTER EXERCISER.": PRINT
 70 PRINT "AVERAGE HOW MANY READINGS";: INPUT Q
 95 PRINT "SELECT CHANNEL (0 TO 15), AND GAIN (1,2,4 OR 8);"
 100 PRINT "INPUT IN THIS FORM ";
 110 PRINT "(C,G)";: INPUT C,G
 120 IF INT(C)>15 THEN 95
 130 IF C<0 THEN 95
 140 LET B=INT(C)*256
 210 IF G=1 THEN GOTO 260
 220 IF G=2 THEN GOTO 260
 230 IF G=4 THEN GOTO 260
 240 IF G=8 THEN GOTO 260
 250 GOTO 95
 260 IF G=1 THEN LET A=0
 270 IF G=2 THEN LET A=8
 280 IF G=4 THEN LET A=16
 290 IF G=8 THEN LET A=24
 300 LET Y=0: LET D=A+B+1
 350 FOR N=1 TO Q: LET Y=Y+EXF(D,0,0): NEXT N
 600 LET V=Y*9.765625E-3/(G*N)
 650 REM--RESCALE, PROPORTION, AVERAGE DATA.
 700 IF G<3 THEN PRINT INT(V*100+.5)/100 "VOLTS": PRINT
 800 IF G>3 THEN PRINT INT(V*1000+.5)/1000 "VOLTS": PRINT
 900 REM--ROUND-OFF TO PROPER SIGNIFICANCE.
 1000 GOTO 110
 6192 DATA 5559,-16716,-29705,-16720,1277,7617,-16724,488
 8000 FOR A=16192 TO 16206 STEP 2: READ D
 8050 LET X=EXF(A,D,1): NEXT A: REM--LOAD EXF0 PROGRAM.
 8100 GOTO 60
 8191 END
 READY
```

Fig. 4. Analog-to-digital converter exerciser program (fast)

```
LIST

 1 REM--BASIC-EXF POWER-FAIL HANDLER.  2-16-72.
 10 PRINT "POWER-FAIL HANDLER WILL NOW BE LOADED...": PRINT
 100 LET X=EXF(20,16328,1): REM--LOAD POWER-FAIL TRAP VECTOR.
 200 FOR A=16328 TO 16382 STEP 2
 300 READ D
 400 LET X=EXF(A,D,1): REM--LOAD MACHINE INSTRUCTIONS.
 500 NEXT A
 1000 PRINT "THE POWER-FAIL HANDLER IS NOW LOADED."
 1010 PRINT "YOUR BASIC USER PROGRAM MAY NOW BE READ-IN"
 1020 PRINT "USING THE 'OLD' COMMAND, AND IT WILL BE"
 1030 PRINT "PROTECTED FROM POWER FAILURE.": PRINT
 1040 PRINT "THE BASIC USER PROGRAM WILL BE RETAINED IN"
 1050 PRINT "CORE, AND NOT BE DESTROYED WHEN THE MACHINE IS"
 1060 PRINT "TURNED-OFF, PROVIDED THAT THE MACHINE HAS NOT"
 1070 PRINT "BEEN HALT-ED AT THE CONSOLE.": PRINT
 6328 DATA 4134, 4198
 6332 DATA 4262, 4326
 6336 DATA 4390, 4454
 6340 DATA 4535, 38
 6344 DATA 5623, 16352
 6348 DATA -16330, 0
 6352 DATA 2560, 2688
 6356 DATA -31490, 7622
 6360 DATA 20, 5509
 6364 DATA 5508, 5507
 6368 DATA 5506, 5505
 6372 DATA 5504, 5623
 6376 DATA 16328, -16360
 6380 DATA 2, 0
 8191 END
 READY
```

Fig. 5. Power-fail-save program