SAND76-8265
Unlimited Release

FORODT - Fortran Debug Routine for the PDP-11 (RT-11)

D. N. Tanner

Prepared by Sandia Laboratories, Albuquerque New Mexico 87115
and Livermore, California 94550 for the United States Energy Research
and Development Administration under Contract AT (29-1)-789

Printed January 1977

Sandia Laboratories

SF 2900 Q(7-73)

Issued by Sandia Laboratories, operated for the United States Energy Research and Development Administration by Sandia Corporation.

SAND76-8265
Unlimited Release
Printed January 1977

FORODT - FORTRAN DEBUG ROUTINE FOR THE PDP-11 (RT-11)

Duncan N. Tanner
Electronics Development Division 8159
Sandia Laboratories, Livermore

ABSTRACT

FORODT provides run time debug features to PDP-11 Fortran programs running with the RT-11 operating system. Digital Equipment Corporation's ODT program has been extended to include Fortran breakpoints, decimal integer and floating point data input, and output options.

CONTENTS

FORODT - FORTRAN DEBUG ROUTINE FOR THE PDP-11 (RT-11)

Introduction

FORODT allows breakpoints to be set in Fortran as well as assembly language programs running on the PDP-11 with the RT-11 operating system. FORODT features are in addition to those provided in DEC ODT V01-02. The new features include the setting and clearing of Fortran breakpoints, a Fortran "GOTO" command, and the change of type-in/type-out radix. Octal, integer, real (2-word floating point), and double (4-word floating point) I/O conversions are available.

These features are summarized in Appendix I. Appendices II and III contain detailed examples. It is assumed that the reader is familiar with ODT. The following descriptions apply to the FORODT features.

FORODT occupies about 2100 decimal words, about 600 more than ODT. Two Fortran OTS routines require an additional 633 words, but are usually needed by the Fortran program so the effect is user-dependent.

Fortran Breakpoints

ODT was designed to be used to debug assembly language programs. The RT-11 Fortran compiler does not generate assembly language instructions; it generates a list of subroutine entry points and arguments which perform the desired function when executed in order. A pointer is used to thread through the list. Actual execution occurs within subroutines which come from the Fortran Library (FORLIB). Normal ODT breakpoints cannot be used because the breakpoints must be at an executable location.

FORODT allows up to eight breakpoints to be set in Fortran threaded code. This is in addition to the eight breakpoints available for normal assembly language routines.

The Fortran compiler has an option which provides the information needed to set Fortran breakpoints. The following program will be used as an example. It is compiled

```
ODTDEM,ODTDEM/L:7=ODTDEM
```

The L switch (/L:7) tells the Fortran compiler to generate the generated code listing and the storage map in addition to the listing. The listing of ODTDEM is shown in Figure 1, and a list of two subroutines is shown in Figure 2.

Setting Threaded Code Breakpoints

The breakpoint must be chosen from the generated code listing of the Fortran program. Once the absolute or relative location is found the breakpoint is set:

```
r;nV
```

r is the address of the breakpoint
n is the breakpoint number 0 to 7.

The address of the breakpoint is either absolute or relative form. The absolute address is the octal location in memory. The relative form is K,ℓ;nV where K is the relocation register and ℓ is the location relative to the contents of relocation register K. It will be assumed that relocation registers have been set for the example program:

```
Register 0   Main
Register 1   Subroutine SUB
Register 2   Function FUN
```

In this way, the relative locations for data and breakpoints can be used.

The program is linked by:

```
ODTDEM,ODTDEM=FORODT,ODTDEM/F
```

Figure 3 shows the memory map. The main program starts at 11212 and a relocation register is set 11212;0R.

Refer to the generated code listing in Figure 1. ISN#0007 in the listing refers to line 7 in the source listing. The generated code follows the relative octal location followed by the contents. Line 7 would look like this:

```
      C
      C     PROGRAM TO DEMONSTRATE THE THREADED
      C     CODE BREAK POINT
      C
0001        COMMON /LCOM/IA(10)
0002        COMMON A(10)
0003        DO 10 I=1,10
0004        K=I-1
0005        CALL SUB(K,B)
0006        A(I)=B
0007        IA(I)=K
0008  10    CONTINUE
0009        END


FORTRAN IV      STORAGE MAP

NAME    OFFSET  ATTRIBUTES

I       000006  INTEGER*2 VARIABLE
K       000010  INTEGER*2 VARIABLE
SUB     000000  REAL*4    PROCEDURE
B       000012  REAL*4    VARIABLE

COMMON BLOCK //         LENGTH 000050

A       000000  REAL*4    ARRAY (10)

COMMON BLOCK /LCOM/     LENGTH 000024

IA      000000  INTEGER*2 ARRAY (10)
```

```
FORTRAN IV      GENERATED CODE

ISN #0003
000020  LSN$    #000003
000024  MOI$1M  000006

ISN #0004
000030  LSN$    #000004
000034  MOI$MM  000006  000010
000042  DCI$M   000010

ISN #0005
000046  ISN$
000050  REL$    000012
000054  REL$    000010
000060  CAL$    #000002 SUB+#000000

ISN #0006
000066  ISN$
000070  SAF$MM  000006  .$$$$.+#177774
000076  MOF$MA  000012

ISN #0007
000102  ISN$
000104  SAI$MM  000005  LCOM+#177776
000112  MOI$MA  000010

ISN #0008
000116  LSN$    #000010
000122  NMI$1I  000006  #000012 000030

ISN #0009
000132  ISN$
000134  RET$
```

FIGURE 1. MAIN PROGRAM EXAMPLE

```
      C
      C     SAMPLE SUBROUTINE
      C
0001        SUBROUTINE SUB(K1,B1)
0002        J= K1 + 1
0003        B1=FUN(J)
0004        RETURN
0005        END
```

```
FORTRAN IV      STORAGE MAP

NAME    OFFSET  ATTRIBUTES

K1      000014  INTEGER*2 PARAMETER VARIABLE
B1      000016  REAL*4    PARAMETER VARIABLE
J       000020  INTEGER*2 VARIABLE
FUN     000000  REAL*4    PROCEDURE
```

```
FORTRAN IV      GENERATED CODE

ISN #0002
000022  LSN$    #000002
000026  MOI$PM  000014  000020
000034  ICI$M   000020

ISN #0003
000040  ISN$
000042  REL$    000020
000046  CAL$    #000001 FUN+#000000
000054  MOF$RP  000016

ISN #0004
000060  ISN$
000062  RET$
```

```
      C
      C     SAMPLE FUNCTION
      C
0001        FUNCTION FUN(J2)
0002        A1=2.0**J2
0003        FUN=A1
0004        RETURN
0005        END
```

```
FORTRAN IV      STORAGE MAP

NAME    OFFSET  ATTRIBUTES

FUN     000016  REAL*4    VARIABLE
J2      000014  INTEGER*2 PARAMETER VARIABLE
A1      000022  REAL*4    VARIABLE
```

```
FORTRAN IV      GENERATED CODE

ISN #0002
000026  LSN$    #000002
000032  MOF$IS  #040400
000036  MOI$PS  000014
000042  XFI$
000044  MOF$SM  000022

ISN #0003
000050  ISN$
000052  MOF$MM  000022  000016

ISN #0004
000060  ISN$
000062  RET$F   000016
```

FIGURE 2. SUBROUTINES EXAMPLE

```
RT-11 LINK    V04-04A    LOAD MAP
ODTDEM.SAV               12-NOV-76

SECTION ADDR    SIZE    ENTRY   ADDR    ENTRY   ADDR    ENTRY   ADDR

. ABS.  000000  001000  $USRSW  000000  $RF1C1  000000  $HRDWR  000000
                        .V011A  000001  $NLCHN  000006  $WASIZ  000075
                        $LRECL  000210  $TRACE  004737
        001000  010116  O.ODT   001312
.$$$$.  011116  000050
LCOM    011166  000024
        011212  000136
        011350  000054  SUB     011350
        011434  000066  FUN     011434
        011522  000036  CAI$    011522  CAL$    011530
        011560  000064  MOI$IP  011560  MOI$SP  011562  MOI$PP  011570
                        MOI$MP  011574  MOI$PS  011604  MOI$PM  011612
                        MOI$PA  011620  MOI$0P  011626  MOI$IP  011634
        011644  000044  NMI$IM  011644  NMI$II  011654  BLE$    011662
                        BEQ$    011664  BGT$    011672  BGE$    011674
                        BRA$    011676  BNE$    011702  BLT$    011704
        011710  000026  MOF$RS  011710  MOF$RM  011716  MOF$RA  011726
                        MOF$RP  011732
        011736  000014  MOF$IS  011736  MOF$0S  011744
        011752  000014  MOF$SM  011752  MOF$SP  011762
        011766  000056  $OTIS   011766
        012044  000102  MOI$SS  012044  MOL$SS  012044  MOI$SM  012050
                        MOI$SA  012054  MOI$IS  012060  MOL$IS  012060
                        REL$    012060  MOI$IM  012064  MOI$IA  012070
                        MOI$MS  012074  MOI$MM  012100  MOI$MA  012104
                        MOI$0S  012110  MOI$0M  012114  MOI$0A  012120
                        MOI$1S  012124  MOI$1M  012132  MOI$1A  012140
        012146  000160  ISN$    012146  $ISNTR  012152  LSN$    012166
                        $LSNTR  012172
        012326  000034  ICI$S   012326  ICI$M   012332  ICI$P   012336
                        ICI$A   012340  DCI$S   012344  DCI$M   012350
                        DCI$P   012354  DCI$A   012356
        012362  000042  MOF$MM  012362  MOF$MA  012374  MOF$MP  012402
                        MOF$PM  012410  MOF$PA  012414  MOF$PP  012420
        012424  000044  RET$L   012424  RET$F   012430  RET$I   012436
                        RET$    012440
        012470  000414  OCI$    012470  ICI$    012476  $ECI    012512
                        OCO$    012672  ICO$    012700
$FIO2   013104  000036
        013142  000120  STP$    013142  FOO$    013146  $EXIT   013166
$CLO2   013262  000002
        013264  000032  SAI$IM  013264  SAI$SM  013266  SVI$IM  013274
                        SVI$SM  013276  SAI$MM  013306  SVI$MM  013312
        013316  000044  SAF$IM  013316  SAF$SM  013320  SVF$IM  013330
                        SVF$SM  013332  SAF$MM  013352  SVF$MM  013356
        013362  000002  $AOTS   013362
        013364  001512  $OTI    013412
        015076  002750  $ERRTB  015076  $ERRS   015176
        020046  001710  RCI$    020046  GCO$    021016  FCO$    021024
                        ECO$    021030  DCO$    021036
        021756  000256  XFI$    021756  $PWRI   021756
        022234  000306  DIF$PS  022234  DIF$MS  022240  DIF$IS  022250
                        DIF$SS  022254  $DVR    022254
        022542  000350  MUF$PS  022542  MUF$MS  022546  MUF$IS  022556
                        MUF$SS  022562  $MLR    022562

TRANSFER ADDRESS = 001312
HIGH LIMIT = 023112
```

FIGURE 3. LOAD MAP EXAMPLE

Relative Location	Contents	Absolute Location	Actual Contents
102	ISN$	011314	012146
104	SAI$MM	011316	013306
106	000006	011320	000006
110	LCOM-2	011322	011164
112	MOI$MA	011324	012104
114	000010	011326	000010

The absolute location and actual contents are determined by the linker and are found from the load map (Figure 2). The absolute location is found by adding the relative location to the start of the routine (11314 = 102 + 11212).

To set a breakpoint, the relative location of line 7 is found in the generated code list. It is set with 0,102;V. It is important to note that breakpoints could also be set at 0,104 and 0.112, but not at 0,106, 0,110, and 0,114 since these words contain arguments for the threaded code, not threaded code routine names or entry points.

The Fortran breakpoint command r;V operates identically to the normal ODT breakpoint command r;B.

Program Start and Continue

When the program is run, it will start up in FORODT. Breakpoints can be set and the program is started at the beginning of the main program for Fortran programs. In our example, the program is started at 011212 or 0,0. 0,0,;G will start the program. When a breakpoint occurs, FORODT types

Vn;r

where r is the location of the breakpoint and n is the breakpoint number. This is identical in form to the ODT breakpoint Bn;r.

The program execution is continued by typing ;P or n;P. This is also identical to ODT. FORODT knows which type of breakpoint occurred.

Single Instruction Mode

The ODT single instruction mode cannot be set when the breakpoint is a Fortran breakpoint (;V). The single instruction mode is available following an ODT breakpoint (;B).

Fortran GOTO

The "GOTO" command allows program execution to proceed at any Fortran instruction. The command causes execution to assume at a threaded code location with the same rules which apply to the Fortran breakpoint (;V). The GOTO command is

r;T

where r is the location of the start of the Fortran instruction.

While the command allows the user to change the order of program execution, there are some pitfalls which must be considered. Some of the restrictions and precautions are:

1. The Fortran program should always start initially with the r;G command because the first few words of the program are instruction which initializes the Fortran work space and tables and start the threaded code execution.

2. Many Fortran instructions are meant to be executed only once. For example, instructions which set up and open I/O channels. A second execution of these instructions could produce fatal or unpredictable results.

3. Disrupting the normal subroutine nesting is risky. As an example:

   Subroutine A calls B
   Subroutine B calls C
   If we GOTO an instruction in Subroutine B, the return link to Subroutine A will not be set up and the program will not run correctly.

4. The GOTO should not start in the middle of a Fortran statement. In line 7 location 0,102 is the start of the statement and 0,102;T is valid. Resuming at 0,104, 0,110 and 0,112 are not recommended unless the user is fully aware of the requirements of the routines skipped in that line. Many of the OTS routines require that information be placed on the stack in correct order, and this may not be accomplished correctly if execution startes in the middle of the line.

Restarting a Program

Once the program has been started using the r;G command, it should not be attempted again without exiting to Monitor (•).

This is necessary because the first action Fortran takes is to set up the Fortran OTS workspace and tables. This cannot be done more than once.

The r;T command can be used to restart as long as the restrictions which apply to this command are observed. If the restart follows a breakpoint in a subroutine, stack overflow could result. By returning to the main part of the program subroutine, linkage problems are avoided, but the stack pointer is not reset to clear out the linkage. If this is done several times the stack will continue to grow to the point where the hardware flags the overflow.

Data Access

ODT and FORODT require that the absolute or relative location be known for each word to be opened. The Fortran listing provides this information in the storage map. In the example, the variable I is located at relative location 6 in the main portion of the program. This variable can be examined by typing

```
0,6/
```

The value of I is displayed following the /.

There are four variable storage classifications:

1. Local variables.
2. Variables stored in blank common.
3. Variables stored in labeled common.
4. Variables which are parameters or arguments in a subroutine.

The Fortran storage map listing and the load map are used to determine the actual location of each variable.

Local Variables

The Fortran storage map gives the relative location. In our example, the variable I is located at 6 and can be examined by typing

```
0,6/
```

Blank Common

Blank common appears in the load map as the .$$$$. and any variable in blank common is referenced relative to that location. In the example, blank common starts at 11116 and variable array A starts there.

Labeled Common

Labeled common appears in the load map with the label. The variable is referenced relative to that location. In the example, labeled common block LCOM starts at 11166 and array IA starts there.

Dummy Variables

Dummy variables (arguments in a subroutine) are referenced differently. Within the subroutine the storage map indicates the argument is a parameter variable. The relative location given does not contain the value, it contains the location of the value.

In our example, relocation register 1 is set to the start of subroutine SUB. To look at the parameter variable K1 in subroutine SUB, we type

```
1,14/011364@

0,10/XXXX          the value of K1 is displayed
```

The character @ directs FORODT to open the location contained in word 1,14 which is in fact the location of the variable K1 in the main program. This address scheme is meaningful only if the current breakpoint is within the subroutine or routines called by the subroutine.

Input/Output Radix

FORODT has the feature to change the I/O radix from octal to decimal integer or floating point. When FORODT is first started, the default I/O radix is octal. When a word is opened the contents are displayed in the current radix.

To better explain the I/O radix operation, it is necessary to define two terms:

1. Current Radix

   The I/O radix which is in effect for when a word is open and is the radix used last to display the contents to that word.

2. Default Radix

The current radix is restored to the default radix when FORODT types an *. If successive words are opened by typing a line feed, the current radix is retained until a carriage return is typed or an error occurs.

Both the current radix and default radix can be changed. The current radix is changed by typing a single # followed by the letter I, F, D, or O. This will cause the contents of the current word to be displayed in the new current radix. Input to change a word must always be in the current radix. Word addresses are always octal. For example, the default radix is octal and the location 1202 is opened

1202/000020 #I 16 18<CR>

Location 1202 contained 20 octal. The #I changed the current radix to decimal integer, and the contents of 1202 were displayed as 16 decimal. 18<CR> changed the contents of 1202 to 18 decimal or 22 octal.

Real and double floating point numbers occupy two or four words, so if a location is opened with a real radix, that word and the next word are opened and displayed in floating point. The reply is converted back and two words are modified. If a line feed is typed, the next real number is opened and displayed. Example:

*1012/ 034100 #F 0.1148000E-04 1.158E-5<LF>

1016/0.1296000E-03

The second location opened is four more than the first.

The current radix determines the number of words displayed and modified. Example:

*1044/ 000000 #F 0.0000000 #I 0 1<LF>

In this case, the current radix was changed from real to integer and only one word was modified. The line feed caused the next word to open and it was displayed as a decimal integer.

The default radix is set by typing two #'s and the letter I, F, D, or O. When FORODT is first run, the default radix is octal.

```
*##F                          Set the default radix to
                              2 word floating point.
*1012/0.1158000E-04<LF>

 1014/0.129000E-03

*##F

*1012/0.158000E-04 #I 14400 1<CR>
```

Here only location 1044 is changed because the current radix was changed to decimal integer (1 word).

```
*##I

*1012/ 14400 #F 0.158000E-4 1.23<CR>
```

Here the floating point number 1.23 is stored in locations 1012 and 1014.

Restrictions

Programs which use overlays require special caution. Breakpoints cannot be set in overlay regions if a new segment will be swapped in, as the breakpoint will be destroyed. Breakpoints in the root segment will work fine.

FORODT will operate with assembly language programs if the decimal I/O features are desired, but the program must be linked with the F switch because FORODT uses Fortran I/O conversion routines.

APPENDIX I - COMMAND SUMMARY

Command	Format	Explanation
V	;V	Removes all Fortran breakpoints.
	r;V	Sets Fortran breakpoint at absolute location r. The next available breakpoint number is used.
	r;nV	Sets Fortran breakpoint n at location r. The location r is either the absolute location (octal) or the location relative to a relocation register. r ⇒ K,ℓ where K is the relocation register number and ℓ is the relative location. r = ℓ + contents of relocation register K.
	;nV	Removes breakpoint n.
##	##I	Sets default I/O radix to 1-word decimal integer.
	##F	Sets default I/O radix to 2-word floating point (real).
	##D	Sets default I/O radix to 4-word floating point (double).
	##O	Sets default I/O radix to 1-word octal.
#		Used only when a location is open to change the current I/O radix. The currently opened location is displayed in the radix indicated following the #. This new current I/O radix is maintained until a carriage return is typed or FORODT types an *.
	#I	Sets current I/O radix to 1-word decimal integer.
	#F	Sets current I/O radix to 2-word floating point (real).
	#D	Sets current I/O radix to 4-word floating point (double).

Command	Format	Explanation
	#O	Sets current I/O radix to 1-word octal.
$	$V/	Opens the first word of the Fortran breakpoint table.
T	r;T	GOTO -- resumes program at threaded code location r.

ODT Commands

All of the ODT commands are available in FORODT. Those most useful to the Fortran programmer are summarized here.

Command	Format	Explanation
/	r/	Opens the word at location r.
		The contents displayed in the current radix. The location may be modified by typing a new value in the current radix followed by a <CR> or <LF>.
<CR>		Carriage return. Closes an open location modifying it if a number was typed and it types an * for the next command.
<LF>		Line feed. Closes the current location, modifying it if a number was typed and opens the next sequential location. The F and D radix options modify two and four words and open the next two or four word group.
R	;R	Resets all relocation register to -1 (unassigned).
	;nR	Resets relocation register n to -1.
	r;nR	Sets relocation register n to value of r.
G	r;G	Goes to location r and starts the program. r;G should be to the beginning of the main Fortran program. r;G to threaded code locations will not operate properly.

Command	Format	Explanation
P	;P	Proceed with program execution from the breakpoint B or V.
	K;P	Proceed with program execution from the breakpoint B or V; and skips K-1 encounters with this breakpoint

APPENDIX II - FORODT EXAMPLE

An example of FORODT with the example program. The user inputs are underlined.

.R ODTDEM<CR>	Run example program.
FORODT V01-02	
*11212,0R	Set relocation register 0 to start of main program.
*11350,1R	Set relocation register 1 to start subroutine SUB.
*11434,2R	Set relocation register 2 to start of function FUN.
*11116,3R	Set relocation register 3 to start of blank common.
*11166,4R	Set relocation register 4 to start of common block LCOM.
*0,116,V	Set Fortran breakpoint 0 to line 8 in main program.
*1,60V	Set Fortran breakpoint 1 to line 4 in subroutine SUB.
*2,60V	Set Fortran breakpoint 2 to line 4 in function FUN.
*3,0/ ** #F ** 0<LF>	Initialize arrays A and IA to zero. ** indicates the number could be any value.
3,000004/ ** 0<LF>	
3,000010/ ** 0<LF>	
3,000014/ ** 0<LF>	
3,000020/ ** 0<LF>	
3,000030/ ** 0<LF>	
3,000034/ ** 0<LF>	
3,000040/ ** 0<LF>	
3,000044/ ** 0<LF>	

```
4,000000/ ** #I ** 0<LF>
4,000002/ ** 0<LF>
4,000004/ ** 0<LF>
4,000006/ ** 0<LF>
4,000010/ ** 0<LF>
4,000012/ ** 0<LF>
4,000014/ ** 0<LF>
4,000016/ ** 0<LF>
4,000020/ ** 0<LF>
4,000022/ ** 0<CR>
```

*0,0;G	Start program.
V2;2,000060	Breakpoint occurred at line X in function FUN.
*2,14/011370 @	Examine dummy variable J2 in function FUN.
1,000020 /000001<CR>	Actual variable is J in sub-routine SUB.
*2,22/040400 #F 2.000000<CR>	Examine A1.
*2,16/ 2.000000<CR>	Examine FUN value.
*;P	Continue execution.
V1;1,000060	Breakpoint occurred at line 4 in SUB.
*1,14/011164 @	Examine dummy variable K1.
0,000010 /000000<CR>	Actual vairable is K in main.
*1,16/011224 @	Examine dummy variable B1.
0,000012 /040400 #F 2.000000	Actual variable is B in main.
*1,20/000001<CR>	J = 1.
*;P	Continue program.
V0;0,000116	Breakpoint at line 8 in main program.
*0,6/000001<LF>	Examine I.
0,000010 /000000<LF>	Examine K.
0,000012 /040400 #F 2.000000<CR>	Examine B.

```
*3,0/040400 #F  2.000000<LF>
```
Examine array A in blank common.

```
3,000004 / 0.0000000<LF>
3,000010 / 0.0000000<LF>
3,000014 / 0.0000000<LF>
3,000020 / 0.0000000<LF>
3,000024 / 0.0000000<LF>
3,000030 / 0.0000000<LF>
3,000034 / 0.0000000<LF>
3,000040 / 0.0000000<LF>
3,000044 / 0.0000000<LF>
```

```
4,000000 / 0.0000000 #I 0<LF>
```
Examine array IA in common block LCOM.

```
4,000002 /      0<LF>
4,000004 /      0<LF>
4,000006 /      0<LF>
4,000010 /      0<LF>
4,000012 /      0<LF>
4,000014 /      0<LF>
4,000016 /      0<LF>
4,000020 /      0<LF>
4,000022 /      0<CR>
```

```
*;P
```
Continue program.

```
V2;2,000060
```
Breakpoint at line 4 in FUN.

```
*2,14/011370 @
```
Dummy variable J2 = 2.

```
1,000020 /000002
```

```
*2,16/040600 #F  4.000000<LF>
```
A1.

```
2,000022 / 4.000000<CR>
```
FUN.

```
*;P
```
Continue.

```
V1;1,000060
```
Breakpoint at line 4 in SUB.

```
*;P
```
Continue.

```
V0;0,000116
```
Breakpoint at line 8 in main.

```
*;P
V2;2,000060
*;P
V1;1,000060
*;P
```

```
V0;0,000116
```
Breakpoint at line 8 in main.

```
*##F                                    Set radix to 2-word floating
                                        point.

*3,0/  2.000000 LF                      Examine array A in blank common.

3,000004 /  4.000000<LF>
3,000010 /  8.000000<LF>
3,000014 / 0.0000000<LF>
3,000020 / 0.0000000<LF>
3,000024 / 0.0000000<LF>
3,000030 / 0.0000000<LF>
3,000034 / 0.0000000<LF>
3,000040 / 0.0000000<LF>
3,000044 / 0.0000000<LF>

4,000000 / 0.0000000 #I 0<LF>           Array IA (integer).

4,000002 /      1<LF>
4,000004 /      2<LF>
4,000006 /      0<LF>
4,000010 /      0<LF>
4,000012 /      0<CR>

*##0                                    Set radix to octal.

*0,6/000003<LF>                         Examine I.

0,000010 /000002<LF>                    Examine K.

0,000012 /041000 #F 8.000000<CR>        Examine B.

*;P                                     Continue.

V2;2,000060                             Line 4 in FUN.

*3;P                                    Continue skipping breakpoint 2
                                        two times.

V1;1,000060                             Line 4 in SUB.

*3;P                                    Continue skipping breakpoint 1
                                        two times.

V0;0,000116                             Line 8 in main.

*;P

V0;0,000116                             Line 8 in main.

*;P

V0;0,000116                             Line 8 in main.
```

```
*;P
V2;2,000060                          Line 4 in FUN.

*;P
V1;1,000060                          Line 4 in SUB.

*;P
V0;0,000116                          Line 8 in main.

*0,6/000007<LF>                      Examine I.

0,000010 /000006<LF>

0,000012 /042000 #F 128.0000<CR>

*##F

*3,0/  2.000000<LF>                  Examine array A.

3,000004 /  4.000000<LF>
3,000010 /  8.000000<LF>
3,000014 /  16.00000<LF>
3,000020 /  32.00000<LF>
3,000024 /  64.00000<LF>
3,000030 /  128.0000<LF>
3,000034 / 0.0000000<LF>
3,000040 / 0.0000000<LF>
3,000044 / 0.0000000<LF>

4,000000 / 0.0000000 #I 0 LF         Examine array IA.

4,000002 /     1<LF>
4,000004 /     2<LF>
4,000006 /     3<LF>
4,000010 /     4<LF>
4,000012 /     5<LF>
4,000014 /     6<LF>
4,000016 /     0<LF>
4,000020 /     0<LF>
4,000022 /     0<CR>

*##I

*0,6/    7 9<CR>                     Change I to 9.

*;V                                  Clear all breakpoints.

*0,132;V                             Set breakpoint at line 9 in main
                                     program.
```

```
*;P                                  Continue.

V0;0,000132                          Breakpoint at line 9.

*0,6/     11<LF>                     I = 11.

0,000010 /      9<LF>

0,000012 / 17792 #F  1024.000

*##F

*3,0/  2.000000<LF>                  Examine A.

3,000004 /  4.000000<LF>
3,000010 /  8.000000<LF>
3,000014 /  16.00000<LF>
3,000020 /  32.00000<LF>
3,000024 /  64.00000<LF>
3,000030 /  128.0000<LF>
3,000034 / 0.0000000<LF>      } Note that these two did not get
3,000040 / 0.0000000<LF>      } set since we changed I from 7 to 9.
3,000044 /  1024.000<LF>

4,000000 / 0.0000000 #I 0<LF>        Examine IA.

4,000002 /      1<LF>
4,000004 /      2<LF>
4,000006 /      3<LF>
4,000010 /      4<LF>
4,000012 /      5<LF>
4,000014 /      6<LF>
4,000016 /      0<LF>         } These two also did not get set.
4,000020 /      0<LF>         }
4,000022 /      9<CR>

*;P                                  Continue.

STOP --                              Exit.
```

APPENDIX III - GOTO EXAMPLE

Example program showing the use of the GOTO command (r;T):

```
.R ODTDEM CR
FORODT V01-02
*11212;0R                    Set relocation register to main
                             program.
*0,66;V                      Set breakpoint at main line 6.
*0,0;G                       Start program.
V0;0,000066                  Breakpoint at line 6.
*0,116;T                     Skip to line 8, arrays A and IA
                             do not get set for I = 1.
V0;0,000066                  Breakpoint at line 6.
*2;P                         Continue skipping one breakpoint.
V0;0,000066                  Breakpoint at line 6.
*0,6/000004                  I = 4.
*0,116;T                     Skip to line 8.
V0;0,000066                  Breakpoint at line 6.
*0,6/000005<CR>              I = 5.
*;0V                         Clear breakpoint at line 6.
*0,132;V                     Set breakpoint at line 9.
*;P                          Continue
V0;0,000132                  Breakpoint at line 9.
*0,6/ 000013  #I  11<CR>     I = 11.
*##F                         Set default I/O to real.
```

```
*3,0/ 0.0000000<LF>          Would have been 2.0

3,000004/ 4.0000000<LF>
3,000010/ 8.0000000<LF>
3,000014/0.00000000<LF>      Would have been 16.0
3,000020/ 32.000000<LF>
3,000024/ 64.000000<LF>
3,000030/ 128.00000<LF>
3,000034/ 256.00000<LF>
3,000040/ 512.00000<LF>
3,000044/ 1024.0000<CR>

*
```

The two positions in Array A were not set since ;T was used to skip those steps.

UNLIMITED RELEASE

INITIAL DISTRIBUTION

Bendix Corporation
Kansas City Division
P. O. Box 1159
Kansas City, MO 64141
Attn: R. Douglass, D/845
R. Hayes, D/131
D. R. Lemon, D/131
W. E. Simes, D/131
J. J. Zimmerman, D/131

Decus Program Librarian
Digital Equipment Computer User's Society
146 Main Street
Maynard, MA 01754

Mead Technology Laboratories
3481 Dayton Xenia Road
Dayton, OH 45432
Attn: Martin W. Roth

R. G. Bradley, 1245
R. A. Hayenga, 2532
R. W. Roberts, 2534
G. D. Horne, 2634
F. I. Magee, 2643
T. B. Cook, Jr., 8000
L. Gutierrez, 8100
D. L. Hartley, 8115
D. E. Gregson, 8150
C. D. Skoog, 8159
M. A. Soderstrand, 8159
D. N. Tanner, 8159 (20)
L. M. Watkins, 8159
K. L. Burris, 8166
H. E. Schoeppe, 8166
T. W. Sneddon, 8166
R. S. Tilley, 8166
L. D. Zirkle, 8166
R. J. Tockey, 8181
C. H. DeSelm, 8200
B. F. Murphey, 8300
R. Y. Lee, 8322
H. D. Jones, 8322
H. G. Short, 8323
R. E. Huddleston, 8325
K. Berkbigler, 8325
J. E. Marion, 8332
P. D. Gildea, 8335
V. Barr, 8342

INITIAL DISTRIBUTION (Continued)

D. Benthusen, 8342
W. C. Scrivner, 8400
J. D. Benton, 8411
R. W. Kelley, 8411
F. J. Cupps, 8265/Classification and Technical Library Processing Division, 3141
Technical Publications and Art Division, 8265, for TIC (2)
Classification and Technical Library Processing Division, 3141 (2)
Library and Security Classification Division, 8266-2 (3)