pdp11

FORTRAN
Debugging Technique

Order Number: AA-H069A-TC

digital

September 1978

This document describes the purpose and use of the FORTRAN Debugging Technique.

FORTRAN
Debugging Technique

Order Number: AA-H069A-TC

SUPERSESSION/UPDATE INFORMATION: This is a new document.

OPERATING SYSTEM AND VERSION: RT-11 V03

SOFTWARE VERSION: FDT V02

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, September 1978

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10

CONTENTS

			Page
CHAPTER	1	INTRODUCTION	1-1
	1.1	USING FDT	1-1
	1.2	FDT COMMAND TYPES	1-2
	1.2.1	Program Control Commands	1-2
	1.2.2	Information Transfer Commands	1-3
	1.2.3	FDT Control Commands	1-3
	1.3	FDT CONVENTIONS AND TERMINOLOGY	1-3
	1.3.1	Syntax Conventions	1-3
	1.3.2	Current Procedure	1-4
	1.3.3	The Location Specification	1-4
		Offset Location	1-5
		Named Location	1-5
		Relative Location	1-6
		Subscripted Name Location	1-6
	1.3.4	Mode Codes	1-7
	1.3.5	FDT Pause Definition	1-9
	1.4	CAUTIONS AND PITFALLS	1-10
CHAPTER	2	DESCRIPTION OF THE FDT COMMANDS	2-1
	2.1	ACCEPT	2-2
	2.2	CONTINUE	2-5
	2.3	DIMENSION	2-6
	2.4	ERASE	2-7
	2.5	GOTO	2-8
	2.6	IF	2-9
	2.7	MACRO	2-10
	2.8	NAME	2-12
	2.9	PAUSE	2-13
	2.10	RESET	2-16
	2.11	START	2-17
	2.12	STEP	2-18
	2.13	STOP	2-19
	2.14	TYPE	2-20
	2.15	WATCH	2-21
	2.16	WHAT	2-22
CHAPTER	3	ADVANCED TECHNIQUES	3-1
	3.1	NAMED COMMON	3-1
	3.2	HOW FDT GENERATES ADDRESSES	3-2
	3.2.1	Octal Offsets	3-2
	3.2.2	Names	3-3
	3.2.3	Relative Addressing	3-3
	3.2.4	Subscript Addressing	3-4
	3.2.5	Indirect Addressing	3-4

CONTENTS (CONT.)

			Page
	3.3	FORMAT CONVERSION ROUTINES	3-4
	3.4	ON-LINE DEBUGGING TECHNIQUE (ODT)	3-4
	3.5	EXECUTION SPEED	3-5
APPENDIX A		FDT COMMAND SUMMARY	A-1
APPENDIX B		FDT LOCATION SPECIFICATION FORMATS	B-1
APPENDIX C		FDT MODES	C-1
APPENDIX D		FDT ERROR MESSAGES	D-1
INDEX			Index-1

TABLES

TABLE	1-1	FDT Mode Codes	1-7

CHAPTER 1

INTRODUCTION

The FORTRAN Debugging Technique (FDT) is a sophisticated interactive debugging tool for FORTRAN IV programs. FDT gives you step-by-step control of the execution of your program and the ability to examine and change the contents of any variable in your program during program execution.

FDT runs on any PDP-11 with FORTRAN IV under the RT-11 or RSTS/E operating system. FDT requires approximately 2K words of memory space during a debugging session.

To use FDT successfully, you need to know the FORTRAN IV programming language. You do not need to know details of internal data formats, machine operation, or the FORTRAN compilation process.

1.1 USING FDT

If you have a program that does not work, follow these steps to begin an FDT debugging session:

1. Compile your FORTRAN program. You must use the FORTRAN compiler option that produces threaded code; FDT does not work with inline code. Do not use the compiler option that suppresses internal statement numbers because FDT needs them. Obtain a source program listing and a storage map listing. You do not need the generated code listing.

2. Link your program units. Include FDT among the input files to be linked. If you are using overlays, place FDT in the root segment of your program. Generate a linker map if your program has named common blocks or assembly language subroutines that you might need to examine.

3. Run your FORTRAN program. FDT takes control and executes an automatic FDT pause before the first executable statement of the program. The following message appears on the terminal:

```
FDT V02-01
FDT PAUSE AT ISN xx IN mainprog
!
```

The variable components of the message are:

xx The internal statement number of the first executable statement in the FORTRAN program.

mainprog The main program being debugged. FDT refers to the main program with the name assigned to it in the FORTRAN PROGRAM statement. If you do not use the PROGRAM statement to name the program, FDT uses the default main program name, .MAIN.

! The prompt FDT issues to indicate it is ready to accept a command.

NOTE

If you do not follow the instructions, FDT may not be able to get control, and the FORTRAN program will then run without FDT. Sometimes FDT gets only partial control (as happens when the main program is compiled without internal statement numbers) and cannot continue. In this case, FDT issues the message FDT START FAIL, and exits to the monitor.

4. Begin debugging your program. Type in the FDT commands that you have decided to use to start solving your problem. At this point, you must type in at least one command that causes an FDT pause, or else the program runs to completion without allowing you to enter any FDT commands. If you want to execute your program without FDT intervention, type CONTINUE or START.

1.2 FDT COMMAND TYPES

There are three classes of FDT commands:

- Program control commands
- Information transfer commands
- FDT control commands

1.2.1 Program Control Commands

Program control commands allow you to control the execution of your FORTRAN program. You can use the program control commands to halt program execution, to continue with the next executable statement or restart from the beginning, to step through the program one or more statements at a time, or to end a debugging session and exit to the operating system monitor. The program control commands are START, STOP, CONTINUE, STEP, PAUSE, RESET, and WATCH.

1.2.2 Information Transfer Commands

Information transfer commands allow you to examine the contents of any variable or array element in your program and to modify its value. As part of the command, you can define the data type of the variable, so that the contents of the variable appear in familiar notation (see Section 1.3.4 on mode codes and Section 1.3.3 on location specification). The information transfer commands are NAME, DIMENSION, TYPE, ACCEPT, and ERASE.

1.2.3 FDT Control Commands

FDT control commands allow you to control the operation of sequences of FDT commands. Using these commands, you can define and execute macros composed of FDT commands, and can branch, conditionally or unconditionally, to another FDT command. The FDT control commands are GOTO, IF, MACRO, and WHAT.

1.3 FDT CONVENTIONS AND TERMINOLOGY

The following sections describe syntax conventions and terminology that you should be familiar with in using FDT.

1.3.1 Syntax Conventions

The general form of an FDT command is the command name followed by its parameters. A simple example is:

```
STEP 3
```

where STEP is the command name and 3 is the parameter. You can enter a command whenever FDT has issued its exclamation-point prompt.

The syntax conventions for entering FDT commands appear in the following paragraphs:

- Spaces between the prompt and the command are optional. You can place the command directly after the prompt, or you can separate the command from the prompt by any number of spaces.

  The following forms are correct:

  ```
  !command
  !     command
  ```

- FDT recognizes only the first three letters of any command. You can abbreviate any FDT command name to its first three letters. There cannot be any blanks between the letters in the command name.

  The following examples are correct:

  ```
  !START
  !STA
  !STAT     FDT interprets this command as STA.
  ```

The following examples are incorrect:

```
!ST A      There is an embedded space in the command.
!CO        The abbreviation is too short.
```

- Spaces are required between a command name and its parameter(s). Some FDT commands can accept parameters. You must leave at least one space between the command name and the parameter.

  The following examples are correct:

  ```
  !MACRO 1
  !PAUSE .MAIN.,20 MACRO 2 AFTER 10
  ```

  The following example is incorrect:

  ```
  !MACRO1
  ```

- There are two formats for a series of FDT commands. You can enter the commands as a list with one command on each line, or you can enter several commands on a line, separating the commands with semicolons.

  The following example shows two commands on a line:

  ```
  !MACRO 1;  TYPE I,J
  ```

- The maximum command length is one line. There is no continuation character to permit a command longer than one line. MACRO definitions (but not the MACRO command) can continue onto more than one line (see Section 2.7).

If you type a command that is incorrectly spelled or incorrectly spaced, FDT prints the error message ?UNDEFINED and prompts for a new command.

1.3.2 Current Procedure

An important concept in using FDT is the "current procedure." As its name implies, the current procedure is the procedure (main program, subroutine, or function) being executed at a given time. FDT defines the current procedure as the procedure being executed when an FDT pause occurs (see Section 1.3.5). You need to know what the current procedure is when you are defining locations for FDT (see Sections 1.3.3, 1.3.4, and 3.2).

1.3.3 The Location Specification

The FDT information transfer commands require you to specify the location of a variable or array element in the FORTRAN program. The location specification consists of two parts, the location itself, and the data type for the location.

There are several ways to specify a location. For a variable, you can use an offset location, a named location, or a relative location. For an array element, you use a subscripted name location.

Offset Location

The most direct way of specifying a variable's location is to give its offset. The offset is the difference between the address of the variable and the base location of the current procedure's data block. The offset for any variable appears in the storage map produced by the FORTRAN compiler.

You specify the offset of a location by typing an octal number:

```
nnn
```

where nnn is the offset of the variable as shown on the FORTRAN storage map. (The offsets in the storage map are expressed in octal bytes.) You need not enter leading zeros.

The following is a fragment from a FORTRAN IV storage map:

```
FORTRAN IV      Storage Map for Program Unit .MAIN.

Local Variables, .PSECT $DATA, Size = 000322 (  105. words)

Name   Type  Offset     Name   Type  Offset     Name   Type  Offset
DAY    I*2   000306     I      I*2   000314     I4     I*2   000312
J      I*2   000316     MONTH  I*2   000304     YEAR   I*2   000310
```

The offset of the variable DAY is 306; the offset of the variable YEAR is 310.

You can specify the offset of a location only for unsubscripted variables in the current procedure. You cannot use a location offset specification for subscripted array elements, variables in common, or variables in a FORTRAN routine that is not the current procedure. The two exceptions to this rule are for variables in blank common and variables in the main program. (Section 3.1 describes special techniques for specifying the offsets of variables in named common.)

1. Variable Offsets in Blank Common

   To specify an offset location for a variable in blank common, type:

   ```
   .BCOM.+nnn
   ```

   where nnn is the offset of the variable (as shown in the FORTRAN storage map).

2. Variable Offsets in the Main Program

   Any variable in the main program can be referenced at any time by typing:

   ```
   mainprog+nnn
   ```

   where mainprog is the name of the main program and nnn is the offset of the variable.

Named Location

You can specify the location of a variable by using a name that you have associated with the location. You use the NAME command in FDT to define a name and associate it with a location (see Section 2.8). Once you have defined a name, you can use it in any location specification. The name is valid in any procedure.

Relative Location

You can specify the location of a variable relative to a previously named variable. To specify a location relative to a previously defined name, you enter the name and the displacement of the variable from the named location. This practice is useful primarily for referencing sequences of variables in consecutive locations.

The format of the specification is:

```
name+nnn
```

The components of the specification are:

name — The named location previously defined in a NAME command.

nnn — The displacement in octal between the location to be specified and the location associated with the name. (The displacement is the difference between the address of the variable and the address of the named location.)

Subscripted Name Location

You can specify the location of an array element by using a subscripted name. The FDT DIMENSION command allows you to define a subscripted name and associate it with the location of a FORTRAN array. Once you have defined a subscripted name and associated it with an array, you can specify the location of any element within the array with the subscripted name and the appropriate subscripts in parentheses. For example:

```
ARRAY(3,7)
```

There are two ways to specify the subscript of an array element:

- Use an integer constant.
- Use a name defined in a previous NAME command. If you use a name as a subscript, FDT assumes that the variable associated with the name has an integer value.

There are certain conventions you must follow for using a subscripted name location:

- Each subscript value must be within the range defined for that dimension. If the subscript value is outside this range, FDT prints the following warning message:

  ```
  %SUBSCR OUT OF BOUNDS
  ```

  Since the message is only a warning, FDT references the location using the subscripts specified.

- The number of subscripts you specify in a subscripted name must be less than or equal to the number of dimensions in the DIMENSION command that defined the name. If the number of subscripts is equal to the number of dimensions, FDT uses the FORTRAN subscript reference algorithm to locate the array element. If the number of subscripts is less than the number of dimensions, FDT assumes that the missing subscripts have a value of 1. For example, if ARRAY is defined as a three-dimensional array, ARRAY(3) is equivalent to ARRAY(3,1,1).

- A subscripted name retains its association with a location regardless of whether that location is in the current procedure. The name loses its association with the location only when you redefine the name with another NAME or DIMENSION command or when you cancel the name with an ERASE command.

1.3.4 Mode Codes

The second major component of a location specification is the data type or mode for the location.

To specify a mode for a location, type:

```
loc[/mode]
```

The components of the location specification are:

loc — The location (see Section 1.3.3).

mode — The FDT mode code for the location. The FDT modes are similar to FORTRAN data types. The default mode for a location is I, corresponding to the INTEGER*2 data type in FORTRAN. Table 1-1 lists the FDT modes and the nearest corresponding FORTRAN data type.

Table 1-1
FDT Mode Codes

Mode	FORTRAN Type	Description
I	INTEGER*2	16-bit value displayed in decimal
J	INTEGER*4	32 bits, first 16 bits displayed in decimal
L	LOGICAL*4	32 bits, displayed as T or F
M	LOGICAL*1	8 bits, displayed as T or F
E	REAL*4	32 bits, scientific notation
D	REAL*8	64 bits, scientific notation
C	COMPLEX	64 bits, real and imaginary parts
B	BYTE	8 bits, displayed in decimal
R	----	16 bits, displayed as three RAD50 characters
O	----	16 bits, displayed in octal
An	----	A string of n ASCII characters (where n is in the range 1 to 255)
Z	----	ASCIZ string (as used in the FORTRAN string handling package)
P	----	Dummy variable mode

When you need to indicate that the associated variable is a parameter (dummy variable), you can use the letter P preceding any of the FDT modes. In fact, you must specify this form for any variable listed as a parameter in the attributes section of the FORTRAN storage map.

Examples:

Specification	Meaning
204/E	A REAL*4 variable at offset 204.
16/PI	An INTEGER*2 parameter variable at offset 16.
POWER/C	A COMPLEX variable at a location associated with the name POWER.

Any location specification can include a mode specification. For example:

```
TYPE 202/E
```

The offset of the location is 202 and the data type of location offset 202 is REAL*4.

If a location specification occurs without a mode, there are two possible actions.

1. If the specification does not involve a name, then FDT assumes the default mode I. For example,

   ```
   TYPE 202
   ```

   is the same as

   ```
   TYPE 202/I
   ```

2. If the specification includes a name (a named location or a subscripted name) then FDT assumes the mode associated with that name. (The mode becomes associated with the name when you define the name in an FDT NAME or DIMENSION command.)

   For example,

   ```
   NAME PI,202/E
   TYPE PI
   ```

   has the same result as

   ```
   TYPE 202/E
   ```

   Similarly,

   ```
   NAME ANGLE,202
   TYPE ANGLE
   ```

   has the same result as

   ```
   TYPE 202
   ```

The original mode definition can be overriden in subsequent commands. For example, if the name definition is:

```
NAME PI,202/E
```

the following commands have the effects shown:

```
TYPE PI/D       Uses mode D for output.
TYPE PI         Uses originally defined mode E for output.
TYPE 202        Uses the default mode I for output.
```

The mode code is important for subscripted names in a DIMENSION command because the mode code determines how FDT locates the required array element. You can use the default mode in a DIMENSION command. However, it is better practice to specify the intended mode explicitly in the DIMENSION command.

You can specify a mode only in a location specification. Subscripts and other command parameters are not location specifications, so you cannot associate modes with them.

1.3.5 FDT Pause Definition

FDT contains a pause feature similar in operation to a FORTRAN PAUSE statement: the FDT pause halts execution of the program and allows the program to be continued by further commands. When the pause occurs, you can enter FDT commands. (Do not confuse the FDT pause with the FORTRAN pause. The FORTRAN PAUSE statement cannot cause an FDT pause.)

There are five ways to cause an FDT pause. The pause name indicates the situation causing the pause:

- automatic pause
- entry pause
- statement pause
- step pause
- watch pause

The automatic pause occurs before the first executable statement in the main program. (There is no FDT command for invoking the automatic pause.) The automatic pause occurs only during the first run of the program in a debugging session and not during subsequent runs. That is, subsequent FDT START commands or monitor START or REENTER commands do not invoke the automatic pause.

An entry pause occurs at the entry point of a subroutine or function. You use the PAUSE command to specify the procedure name, and FDT pauses before the procedure begins executing.

A statement pause occurs before execution of a particular FORTRAN statement. Using a PAUSE command, you specify the statement where you want the pause to occur, and FDT pauses when that statement is the next one to be executed.

A step pause occurs after a defined number of FORTRAN statements have executed. You specify the number of statements using the STEP command.

A watch pause occurs when the value of a variable being watched changes. You specify the variable to watch using the WATCH command.

When an FDT pause occurs, FDT prints the following message on the terminal:

```
FDT PAUSE AT ISN nnn IN proc
!
```

The variable components of the message are:

nnn — The internal statement number of the next executable FORTRAN statement.

proc — The name of the procedure in which the FDT pause occurred. A procedure is a main program, subprogram, or function. FDT defines procedure proc as the current procedure. This procedure remains the current procedure until the next FDT pause occurs.

1.4 CAUTIONS AND PITFALLS

The following are some general cautions you should observe while using FDT:

- Correct use of spaces is important in FDT.
- Invalid location specifications or constants out of range cause unpredictable results.
- Under the RT-11 SJ operating system, the ends of long lines of text printed at the terminal may be lost when the terminal reaches its right margin. FDT does not provide automatic carriage return or line feed.
- For purposes of subscripting and copying, FDT considers locations with ASCIZ mode (/Z) to have a length of one byte. For example, the command

  ```
  ACCEPT NAME/Z=INPUT
  ```

  copies only the first character from INPUT to NAME. To copy an ASCIZ string, use mode An, where n is equal to the maximum possible length of the ASCIZ string, including the terminating zero byte. For example, the command

  ```
  ACCEPT NAME/A25=INPUT
  ```

  copies 25 bytes from INPUT to NAME.
- An FDT error message does not indicate that an entire command was aborted. Some action may have occurred before discovery of the error. In this case, side effects can result. For example, the command

  ```
  PAUSE SUBR+12
  ```

  produces the error message ?UNDEFINED, but it acts like the command PAUSE SUBR.

- FDT ignores any commands following CONTINUE, START, or STEP. For example, if you type the command line:

```
STEP 3 ;  TYPE INDEX
```

FDT executes three FORTRAN statements and then pauses, without typing anything. However, commands following CONTINUE, START, or STEP are executed if FDT control branches around CONTINUE, START, or STEP. For example, if you type:

```
IF SPY<50;CONTINUE;TYPE INDEX
```

then FDT resumes execution if SPY is less than 50, but types the contents of INDEX and waits for another FDT command if SPY is greater than or equal to 50.

- You cannot place an FDT pause on the GO TO of a logical IF statement. For example, suppose the FORTRAN source listing contains the following statements:

```
0008        NEXT=NOW+1
0009        IF(ITEMS(NEXT).GT.ITEMS(NOW)) GO TO 50
0011        TEMP=ITEMS(NOW)
```

You cannot place an FDT pause on internal statement 10. That is, the FDT command:

```
PAUSE ,10
```

never causes a pause.

However, an FDT pause on statement 10 would be valid if the source listing contained the following statements:

```
0008        DO 50 NEXT=1,N
0009        IF (ITEMS(NEXT).GT.MAX) MAX=ITEMS(NEXT)
0011  50    CONTINUE
```

In this case, the FDT pause occurs before the value of MAX changes if the condition is true. If the condition is false, no FDT pause occurs.

CHAPTER 2

DESCRIPTION OF THE FDT COMMANDS

The FDT commands are described in the following sections. The commands are arranged alphabetically for convenient reference. Appendix D contains a reference summary of the commands.

Useful commands for a new FDT user are:

- PAUSE, STEP, and CONTINUE to control execution
- NAME to refer to variables
- TYPE and ACCEPT to display and change values

ACCEPT

2.1 ACCEPT

The ACCEPT command assigns new values to FORTRAN variables. There are three forms of arguments for the ACCEPT command. You can mix all three forms freely within a macro definition. (The third form is not valid outside a macro.) You can specify as many arguments as fit on a single command line. The arguments must be separated by commas.

The first form is:

```
ACCEPT loc=value
```

The arguments are:

loc — The location whose value is to be changed.

value — The new value to be assigned to loc. The new value may be a constant in the same data format as the mode of loc, or it may be a previously defined name or subscripted name.

The second form is:

```
ACCEPT 'text'
```

The argument is:

'text' The literal string to be printed.

This form of the ACCEPT command is identical in function to the analogous form of the TYPE command. The FORTRAN conventions for text literals apply.

The text literal form of ACCEPT is useful for FDT macros, where you can use the text as a prompt to enter new values.

The third form is:

```
ACCEPT loc
```

The argument is:

loc — The location specification for a value entered from the terminal.

This form of the ACCEPT command is valid only in FDT macros. It requires an input value from your terminal, and prompts for the value with a question mark. The value you enter must follow the same conventions required for other forms of the ACCEPT command.

The following mode conventions apply to the values in the ACCEPT command:

- The value may be a previously defined name. If it is, FDT copies its contents into loc. When you use a name as the value, its mode should match the mode of loc, but need not. If the modes of loc and name are different, the mode of loc determines the number of bytes copied from name into loc (see Table 1-1).

  For example:

Mode of loc	Data Type	Number of Bytes Copied
I	INTEGER*2	2
J	INTEGER*4	4
E	REAL*4	4
D	REAL*8	8
Z	ASCIZ	1

  When you use a name as a value, FDT ignores the mode of name and uses name only to find the address of the value to be copied. FDT never does conversions from one mode to another.

- String-mode constants (/Z or /An) must appear enclosed in single quotes. The FORTRAN conventions for text literals apply.

- RAD50 constants and logical constants (/R, /L, and /M) must be preceded by the character # to indicate that they do not represent names. For example, the logical constants T and F are represented by #T and #F.

- Complex constants appear as two real constants separated by a comma, in the order real, imaginary.

- Nonstring constants may include at most 40 characters. The number of characters represented by a string constant is limited by the length of the input line.

Examples:

Command	Meaning
ACCEPT 202=1	Sets the INTEGER*2 value at offset 202 to the value 1.
ACC DELTA=EPSILON	Replaces the contents of DELTA with the contents of EPSILON.
ACC 'DELTA=',DELTA	Prints DELTA=? on the terminal and waits for an input value. (Valid only within an FDT macro.)

Command	Meaning
ACC I=0,J=1,244/E=3.14159	Sets the values of locations named I and J to 0 and 1 respectively; sets the REAL*4 variable at offset 244 to 3.14159.
ACCEPT I=12,'J=',J	Sets the location named I to 12; prints J=? on the terminal and waits for input to set the location named J. (Valid only within an FDT macro.)

NOTE

Both ACCEPT and TYPE use subroutines that are loaded by FORTRAN to process FORMAT statements. If you are debugging a program without D, E, F, or G FORTRAN format specifications, then some of the format conversion routines are not loaded and FDT cannot accept or type variables with modes C, D, or E. If you attempt to use modes C, D, or E when the format conversion routines are not present, FDT prints the error message ?NO CONVERSION and continues execution. Section 3.3 describes a way to avoid this problem.

2.2 CONTINUE

The CONTINUE command resumes program execution after any FDT pause. The next statement executed is the statement whose internal statement number and procedure name appeared in the last pause message. When it executes a CONTINUE command, FDT ignores any commands remaining on the line.

If the current FDT pause occurred as the result of a PAUSE command, the CONTINUE command can have one optional parameter called the execution count. FDT ignores the execution-count parameter if the current pause is an automatic pause or if it occurred as a result of a STEP or WATCH command.

The form of the command and parameter is:

```
CONTINUE [ntimes]
```

The parameter is:

ntimes The execution count. The execution count is an integer specifying the number of times the FORTRAN program must reach this point before another FDT pause can occur here. If you omit the ntimes parameter or if ntimes = 1, an FDT pause occurs the next time control reaches this point (unless the PAUSE command is replaced or cancelled by subsequent commands).

DIMENSION

2.3 DIMENSION

The DIMENSION command associates a name and a list of dimensions with the FORTRAN array you want to define, allowing the name to be used as a subscripted name. FDT cannot access FORTRAN virtual arrays. The DIMENSION command is the subscripted-name equivalent of the NAME command.

The form of the DIMENSION command is:

```
DIMENSION name(i,j,...)[,loc]
```

The parameters are:

name	The FDT name to be associated with the array.
(i,j,...)	The list of dimensions associated with the array. There may be at most seven dimensions. Each dimension value in the list must be an integer in the range 1 through 32767.
loc	The location specification for the first element (base) of the array. The offset for the base of the array appears in the FORTRAN storage map. Offsets for FORTRAN virtual arrays are not valid loc parameters. If you omit the loc argument, FDT erases the name specified (see ERASE command, Section 2.4).

The mode code is an important part of the location specification for subscripted names. You can default the mode in a DIMENSION command. However, it is better practice to specify the intended mode explicitly in the DIMENSION command. FDT uses the specified mode in all subscript calculations referring to the array.

ASCII mode (/An) is not valid in a DIMENSION command because it has no meaning when used in subscript calculation. No error message appears if you specify ASCII mode in a DIMENSION command. However, when you attempt to use the subscripted name, FDT responds with the error message ?UNDEFINED.

Examples:

Command	Meaning
DIMENSION DATA(10,10),46/E	A REAL*4 array named DATA with 10 by 10 dimensions at offset 46.
DIM COLUMN(10),DATA(1,3)/E	A one-dimensional REAL*4 array COLUMN equivalenced to the third column of the array DATA.

The following example shows a useful trick for naming one element of an array.

NAME W,DATA(5,5)/E	A REAL*4 variable W equivalenced to array element (5,5) of DATA (see NAME command in Section 2.8).

2.4 ERASE

The ERASE command cancels the association of a name and a location, and frees the space in FDT's internal tables.

The ERASE command has the following form:

```
ERASE name1[,name2,...]
```

You can specify as many names or subscripted names in an ERASE command as can fit on a single line. The names must be separated by commas. Subscripted names must appear without subscripts; subscripts are invalid in the ERASE command. It is not possible to erase the name for part of an array.

Examples:

```
ERASE TIME
ERASE SPEED,DIST,DATA,ICOUNT
```

ERASE does not change the values of the variables whose associated names are erased.

GOTO

2.5 GOTO

The GOTO command changes the order of execution of commands within an FDT macro. It causes an unconditional transfer of control, analogous to that caused by the FORTRAN GO TO statement. (Do not confuse the FDT GOTO and the FORTRAN GO TO; the FDT GOTO cannot change the order of execution of FORTRAN statments.) Like the FORTRAN GO TO, the FDT GOTO requires numeric statement labels.

The form of the GOTO command is:

```
GOTO label
```

(Remember that FDT command names must not contain spaces; GO TO is an invalid FDT command.)

The parameter is:

label — The numeric label of the command to which control is transferred by the GOTO command. The label must be an integer in the range 1 to 32767. Embedded spaces are not valid.

If two or more labels have the same value, FDT uses the first occurrence of the label. If the label you reference does not exist within the current macro, FDT prints the error message ?LABEL.

You can use the IF command and the GOTO command to create loops of FDT commands analogous to simple FORTRAN loops or to FORTRAN DO loops (see Sections 2.6 and 2.7). For example, if the first command in the loop has the label 100, the last command in a loop might be IF NEXT<>0;GOTO 100. The only loops you can do are on conditions or values; there is no way to increment an index in FDT.

If FDT is executing an infinite loop, you can terminate execution only by typing CTRL/C twice to return to the monitor. The monitor's START or REENTER commands can then operate in the same way as the START command in FDT. However, in some cases, you will have to use the RUN command to reload the FORTRAN program and start the debugging session from the beginning.

Defining a Label

You label a command by preceding it with an integer. The label must follow either the left parenthesis that marks the beginning of the macro definition or the semicolon that delimits the preceding command. The label may be set off by spaces in its defining occurrence.

The following macro definition contains a label:

```
MACRO 1(TYPE MAX;  10 ACCEPT 'INIT=',INIT ;IF INIT>MAX;
GOTO 10;  S 3)
```

2.6 IF

The IF command requests conditional execution of another FDT command. You use IF to specify that another FDT command is to be executed only if a given condition is met.

The form of the command is:

```
IF loc<rel>value;FDT command
```

The parameters are:

loc	The location specification for the value to be compared. The mode of loc must be E, I, or J.
<rel>	The logical relation tested. The parameter <rel> is one of the six logical relations: = (equal), <> (not equal), > (greater than), >= (greater than or equal), < (less than), or <= (less than or equal).
value	The value to be compared to loc. The value parameter may be either a constant in the same mode as loc, or a previously defined FDT name. Subscripted names are valid. The mode of the name should (but need not) match the mode of loc. FDT assumes that name has the same mode as loc, and ignores the actual mode of name. It compares the contents of loc and name in the mode defined by loc. FDT never does conversions from one mode to another. (See also the discussion of mixed modes in the ACCEPT command, Section 2.1.)
FDT command	The single FDT command that is executed only if the logical relation specified between loc and value is true.

The entire IF command must fit on one line.

The IF command compares the value in loc to the value specified. If the relation is true, FDT executes the FDT command specified. If the relation is not true, FDT does not execute the command but skips to the next command.

Example:

```
IF PARM/I>5;WATCH COUNT;STEP
```

If PARM is greater than 5, FDT sets a watch on the location named COUNT, and executes the next FORTRAN statement; otherwise, FDT simply executes the next FORTRAN statement.

MACRO

2.7 MACRO

The MACRO command allows you to define, execute, or delete an FDT macro. An FDT macro is a sequence of FDT commands that is executed as a unit. Thus, an FDT macro is analogous to a FORTRAN subroutine or function subprogram.

1. Defining a macro

You can create a new macro, or change an existing macro, with a MACRO command by typing:

```
MACRO m(FDT commands)
```

The parameters are:

m	The number of the macro where m is a value in the range 0 to 7.
FDT commands	Any valid FDT command or series of commands.

There is no space between the macro number (m) and the left parenthesis. The FDT commands appear within the parentheses, separated by semicolons. There is no limit to the number of commands defining a macro. The definition may continue on as many lines as necessary. FDT prompts for each new line. The end of the macro is defined by the closing right parenthesis.

Examples:

The following example prints the values of the variables associated with the names TIME, SPEED, and DIST, and the first two elements of the array named DATA:

```
MACRO 3(TYPE TIME,SPEED,DIST,DATA(1),DATA(2))
```

The following example shows a multiple-line macro definition. The macro accepts a floating-point value VAR from the terminal, and tests whether VAR is within the limits required by the program. If VAR is outside the limits, FDT goes to the command labeled "100" and prompts for another value.

Previous definitions:

```
NAME VAR,234/E
NAME TOP,240/E
```

Macro definition:

```
MACRO 5(100 ACCEPT 'VAR=',VAR;  IF VAR<12.;  GOTO 100;
IF VAR>TOP;  GOTO 100;  CON)
```

2. Executing a macro

You can execute a macro either automatically or manually:

FDT executes a macro automatically whenever it executes an FDT pause that was set up by a PAUSE command containing a MACRO parameter (see Section 2.9).

You can execute a macro manually by typing the following command:

```
MACRO m
```

The parameter is:

- m The number of the macro to be executed. The value of m is a number in the range 0 to 7.

The specified macro executes until FDT reaches the end of the macro, at which time FDT prompts you for more commands by printing !, or until a START, STEP, or CONTINUE command within the MACRO is executed, at which time FDT resumes execution of the FORTRAN program.

3. Deleting a macro

You can delete a macro by redefining it with a null command string. The form of the command is:

```
MACRO m()
```

The parameter is:

- m The number of the macro in the range 0 to 7.
- () Null command string. The left and right parentheses must not enclose any characters or spaces.

Implicit Macro

MACRO 0 is a special macro number that you cannot associate explicitly with a PAUSE. Any current PAUSE command that you have not explicitly associated with another MACRO is implicitly associated with MACRO 0.

MACRO 0 is usually not defined. If you do define MACRO 0, every PAUSE command without a MACRO parameter executes MACRO 0. You can execute MACRO 0 manually.

NAME

2.8 NAME

The NAME command associates a name with a location and a mode. The name must be unique in the first six characters. Any characters after the sixth are ignored. The first character of the name must be a letter. The other characters can be either letters or digits.

The form of the NAME command is:

```
NAME name[,loc]
```

The parameters are:

name The name to be associated with the variable. If the name you specify has already been defined, this NAME command redefines the name. The old definition is lost.

loc The location specification (as described in Sections 1.3.3 and 1.3.4). If you omit the location specification, FDT cancels the name association (see ERASE command, Section 2.4).

Examples:

Command	Meaning
NAME I,202	INTEGER*2 variable I at offset 202.
NAM PARM,16/PI	INTEGER*2 parameter PARM at offset 16.
NAM DELTA,I+2/E	REAL*4 variable DELTA at offset 204. (Relative location I+2 is location 202+2.)
NAM EPSILON,DELTA+4/E	REAL*4 variable EPSILON at offset 210.
NAM PARM	Erases the definition of PARM.

2.9 PAUSE

The PAUSE command defines statement pauses and entry pauses (see Section 1.3.5).

The PAUSE command marks the statement in your program before which you want an FDT pause to occur. You mark the statement using its internal statement number. The pause occurs only when the marked statement is the next one to be executed.

You can define a statement pause for any executable FORTRAN statement in your program, even if that statement is located in a procedure that is nonresident or in a shared library. The location of the pause is stored internally by FDT. It is not stored in the FORTRAN program itself.

You can place an entry pause on the entry point of any FORTRAN subroutine or function. Whenever the procedure is called, FDT issues the message FDT PAUSE AT ISN 0 IN proc. You can use this feature to detect calls to a procedure without having to determine the first executable statement of the procedure.

The PAUSE command requires you to specify the location of the pause. There are two optional parameters. You can specify the optional parameters MACRO and AFTER in either order. The complete form of the PAUSE command is:

```
PAUSE {[proc],isn} [MACRO m][AFTER ntimes]
      { proc      }
```

The parameters are:

proc,isn — The location of the statement to be marked.

proc — The name of the procedure in which the pause occurs. The default value for the procedure name is the name of the current procedure. The current procedure is defined as the main program, subroutine, or function whose name was printed in the last FDT pause message.

If you specify a subroutine or function as the procedure and omit the isn parameter, FDT establishes an entry pause at the entry point of the FORTRAN procedure you named. You cannot place an entry pause on the entry point of an assembly language routine. (Note that "proc,0" is not equivalent to "proc" because internal statement number 0 is undefined.)

isn — The internal statement number of the FORTRAN source statement where you want the FDT pause to occur. The internal statement number appears to the left of each statement in the source listing.

NOTE

FDT does not give an error message if proc,isn specifies a nonexistent or nonexecutable statement in the FORTRAN program. The command containing an invalid internal statement number is effectively ignored; the nonexistent or nonexecutable point is not executed by the program and thus cannot cause an FDT pause to occur.

MACRO m

The FDT macro that is executed when the FDT pause occurs. An FDT macro is a sequence of FDT commands that is executed as a unit. It is analogous to a subroutine or function in FORTRAN. For more information on FDT macros, see Section 2.7.

The m argument specifies the number of the FDT macro you want to associate with the FDT pause you are creating. The macro number must be an integer in the range 1 to 7.

The MACRO m parameter is valid whether or not macro m exists. The macro can be defined or changed at any time without affecting the PAUSE command. If the macro does not exist when the pause occurs, FDT operates as if the macro parameter were not specified.

If the macro does exist when the pause occurs, FDT executes the macro without issuing an FDT pause message. If the macro contains a CONTINUE command, execution of the FORTRAN program resumes. If the macro does not contain a CONTINUE command, FDT prompts for more commands when it has finished executing the macro.

You can abbreviate the word MACRO to MAC. Only the first three characters are checked for spelling accuracy. (Remember that there must be a space between the word MACRO and the macro number, m.)

AFTER ntimes

The execution count. The ntimes parameter specifies the number of times that the program must reach (but not execute) the marked statement before an FDT pause occurs at that point. The ntimes value must be an integer in the range 1 to 32767. The default value for the AFTER ntimes parameter is 1, which causes an FDT pause the first time program control reaches the statement.

FDT decrements the execution count each time control reaches the internal statement number specified. An FDT pause occurs when the count is zero. When the pause occurs, FDT automatically resets the execution count to the default value of 1. If you want to override the default execution count, specify the required execution count in a CONTINUE ntimes command (see Section 2.2). (The pause definitions output by the WHAT command contain the current value of the execution count, not the initial value.)

You can abbreviate the word AFTER to AFT. Only the first three characters are checked for spelling accuracy. (Remember that there must be a space between the word AFTER and the execution count, ntimes.)

There can be at most eight active PAUSE commands in a program at any one time. If you attempt to define more than eight pauses in a program, FDT prints out the error message ?NO ROOM.

You cannot place two pause definitions at the same point in your program. If you try to place two pauses at the same point, the second pause definition cancels the first.

Statement pauses are disabled when FDT is in step mode (see STEP command, Section 2.12).

Example:

The following command line

```
PAUSE SUBR;PAUSE ,10
```

places an entry pause on procedure SUBR, and a statement pause on statement 10 of the current procedure.

RESET

2.10 RESET

The RESET command removes pauses created by the PAUSE command.

The form of the RESET command is;

```
RESET proc,isn
```

The parameters are:

proc,isn	The location in the FORTRAN program for which the pause was defined. The location specified must be exactly the same as that which appeared in the pause definition.
proc	The name of the procedure in which the pause was to occur.
isn	The internal statement number of the FORTRAN source statement where the pause was to occur.

For example, the command line

```
RESET ,10;RES SUBR
```

removes the statement pause on statement 10 of the current procedure and the entry pause on procedure SUBR.

2.11 START

You can issue the START command whenever FDT prints its command prompt. The FDT START command begins executing your FORTRAN program at the first executable statement in the main program. However, START might not work when the system is not in an initial or ready condition. For example, if there are open files, a START command causes an error message and FDT returns control to the monitor.

The START command has no parameters. When the START command is executed, FDT ignores any commands remaining in the line.

STEP

2.12 STEP

The STEP command continues execution of your program in step mode, and indicates the number of statements that you want to execute before the next FDT pause.

The forms of the command and its optional parameter are:

```
STEP [n]
S [n]
```

(Notice that the STEP command has the special abbreviation S.)

The parameter is:

n An integer in the range 1 to 32767. FDT executes n FORTRAN statements before pausing. The default value for n is 1, which results in single-step operation with a pause before each statement.

Step mode disables all statement pause definitions. You cancel step mode and reenable the pause definitions by entering any FDT command (other than another STEP command). The occurrence of an FDT pause also cancels step mode.

STEP counts only executable statements. A logical IF statement counts as one executable statement if it contains a GOTO; otherwise, a logical IF statement may be one or two executable statements (depending on the outcome of the IF test). The END statement is an executable statement. Nonexecutable statements are FORMAT statements, declaration statements, and SUBROUTINE and FUNCTION statements.

After it executes a STEP command, FDT ignores any commands remaining in the line.

2.13 STOP

You issue the STOP command to end a debugging session. The STOP command in FDT is equivalent to a STOP statement in the FORTRAN program. The STOP command closes any open logical units, performs necessary exit actions, and returns control to the operating system.

The STOP command has no parameters.

TYPE

2.14 TYPE

The TYPE command prints the values of variables in the FORTRAN program during a debugging session.

The forms of the TYPE command are:

```
TYPE loc[,more]

      or

TYPE 'text'[,more]
```

The parameters are:

loc	The location specification of the variable whose value you want to examine.
'text'	A literal text string to be printed on the terminal. The FORTRAN conventions for text literals apply to FDT literals.
more	Another loc or text parameter.

Multiple loc or text parameters may appear in a single TYPE command. The parameters must be separated by commas. The maximum length of the TYPE command is one line.

FDT finds each location specified and prints the contents of the location in the mode indicated by the location specification. FDT prints text literals exactly as you type them (except for the enclosing quotation marks). If multiple parameters appear in a TYPE command, FDT separates the values with commas.

Examples:

Command	Meaning
TYPE 202	Prints INTEGER*2 variable at offset 202.
TYPE 'HI'	Prints HI.
TYPE 'DON''T'	Prints DON'T.
TYPE DELTA,EPSILON	Prints two values separated by a comma.
TYPE DELTA/O	Prints first 16 bits of DELTA in octal.
TYPE 'DELTA=',DELTA	Prints DELTA= followed by the value of DELTA.
TYPE DATA(I,J)	Prints the value of the (i,j) array element in DATA.

2.15 WATCH

The WATCH command directs FDT to watch the contents of a location and to perform a watch pause whenever the value in that location changes.

The form of the WATCH command is:

```
WATCH [loc]
```

The parameter is:

loc — The specification of the location to be watched. (Sections 1.3.3 and 1.3.4 describe how to specify a location.) Only one location can be watched at any given time. A new WATCH loc command cancels the previous WATCH command. A WATCH command without a location specification cancels the previous WATCH loc command.

The location to be watched can have any FDT mode except Alpha (/An). Only the first character of the string is watched for locations in ASCIZ string mode (/Z).

When the value of a watched location changes, FDT prints the message WATCH PAUSE, and performs an FDT pause at the next executable FORTRAN statement. The value must actually change. For example, if IVALUE is equal to 5, then the statement IVALUE=5 does not cause a watch pause. If another FDT command changes the value of a watched location, a watch pause does occur. For example, you can deliberately trigger a watch pause by using FDT's information transfer commands to change the value in a watched location.

Each time a watch pause occurs, FDT cancels the WATCH command that initiated it. You must issue another WATCH command with the same location specification if you want FDT to continue watching the same location.

Watch pauses are independent of other FDT pauses. You can place a watch pause and any other command causing a pause (for example, PAUSE or STEP) on the same FORTRAN statement. If you do specify a watch pause and another pause for the same statement, FDT processes the other pause first. You must resume execution using CONTINUE, STEP, or START before the watch pause can occur.

Examples of the WATCH command:

```
WATCH MAX
WATCH ARRAY(3,2)
```

WHAT

2.16 WHAT

The WHAT command displays the current status of the FDT system. When you type WHAT, FDT returns the following information:

- A list of the definitions for all the active pauses. For each pause, FDT prints the location of the pause (the procedure name and optional internal statement number), the current value of the execution count (when it is greater than 1), and any associated macro number. The pause listing for an entry pause has no internal statement number.

- A list of all currently defined macro numbers and the contents of the macros.

CHAPTER 3

ADVANCED TECHNIQUES

The following sections describe some advanced techniques that you can use to extend FDT's power.

3.1 NAMED COMMON

FDT can access variables in a FORTRAN named common block after you define the block using the NAME command. Use the following procedure:

1. Locate the address of the named common block.

   The absolute address of the named common block appears in the link map. The name given to the block in the FORTRAN source code appears in the link map under the column labeled SECTION. The 6-digit number following the name is the address of the block.

   For RT-11 FB foreground programs, the named common block address appearing in the link map is a relative address. The desired absolute address is the sum of the link map relative address and the foreground job load address. Obtain the foreground load address by issuing the FRUN command with the /P option. (The /P option causes control to return to the monitor; type RESUME to execute the foreground job.)

2. Describe the named common block to FDT.

   A NAME command in the following form defines the block's location:

   ```
   NAME block,.ABS.+addr
   ```

   The arguments are:

   block — The FDT name of the common block. If the block name is the same as the name of a variable in a previous NAME command, FDT erases the previous name.

   addr — The 6-digit absolute address of the named common block.

3. Refer to locations in named common.

   The location of any variable in named common is expressed as follows:

   ```
   block+nnn
   ```

The components of the specification are:

block The FDT name of the common block.

nnn The offset of the variable as given in the FORTRAN storage map.

You can assign FDT names to variables in named common by using block+nnn as a location in a NAME command.

Examples:

Assume that the name of the FORTRAN named common block is COMBLK and the link map gives its address as 063524.

Describe the named common block to FDT:

```
NAME COMBLK,.ABS.+63524
```

Print the contents of an INTEGER*2 variable at offset 000010 in COMBLK:

```
TYPE COMBLK+10
```

Name the variable at location COMBLK+10:

```
NAME INDEX,COMBLK+10
```

Print the contents of the variable:

```
TYPE INDEX
```

3.2 HOW FDT GENERATES ADDRESSES

FDT uses the location specification to generate the address and mode information of a FORTRAN variable. The manner in which FDT generates the 16-bit PDP-11 address depends on the information in the location specification. The following section describes the address-generation process for the five kinds of location specifications.

3.2.1 Octal Offsets

The form of the location specification is:

oct An octal number in the range of 0 to 177777.

When you give an octal number as a location specification, FDT always interprets the number as an offset from the base address of the current procedure's data block. In other words, FDT adds the octal number to the base address to generate the 16-bit address. Thus, you can use the offsets produced by the FORTRAN compiler to generate addresses for any locations shown in the FORTRAN storage map.

3.2.2 Names

The form of the location specification is:

name An alphanumeric name unique in the first six characters.

The first character must be a letter. FDT ignores any characters after the sixth. The NAME command associates a name with a location. Once the association has been established, the name refers directly to the location of the variable.

3.2.3 Relative Addressing

The form of the location specification is:

name+oct An alphanumeric name unique in the first six characters, a plus sign, and an octal number in the range 0 to 177777.

This location specification is a form of relative addressing in which the base address is the location associated with the name. FDT adds the octal displacement to the base address to determine the location of the variable. Displacements in the range 100000 to 177777 are negative displacements. FDT assumes that all octal numbers represent two's complement binary integers. For example, the displacement 177776 represents a displacement of -2 bytes (the word preceding the name). You must determine the displacement yourself. It does not appear in the storage map.

You could use relative addressing for array subscripts. However, this is unnecessary because FDT allows you to define subscripted names. Relative addressing allows you to address variables in named common blocks once you have supplied the base address of the common block.

There are several predefined names in FDT that you can use as base addresses for relative addressing. These names are:

.MAIN. The default name assigned by FDT to the base address of the FORTRAN main program. You can reference any location in the main program using this base. If you named your main program with the FORTRAN PROGRAM statement, then FDT uses that name to find the base address for a relative address.

.BCOM. The name assigned by FDT to the base address of the FORTRAN blank common area. Use .BCOM. to reference variables in blank common.

.ABS. The zero address of memory. FDT uses this name as the base address for referencing any absolute memory location.

For example, the following command prints the contents of absolute location 56 (octal) in octal format:

```
TYPE .ABS.+56/O
```

3.2.4 Subscript Addressing

The form of the location specification is:

name(i,j,k,...)	A subscripted name for specifying arrays with at most seven dimensions. The base address for this form of addressing is the address of name(1,1,1,...).

FDT uses the FORTRAN subscripting algorithm to compute the displacement from the base address and to locate the specified array element.

3.2.5 Indirect Addressing

When you define a name as a parameter (by specifying its mode as P), FDT uses the location associated with the parameter name as an indirect address. That is, FDT regards the value associated with the name as the address of the desired value. For example, if the location of the parameter PARM contains 2000, and location 2000 contains 1234, then the command

```
TYPE PARM/I,PARM/PI
```

prints

```
2000,1234
```

3.3 FORMAT CONVERSION ROUTINES

TYPE, ACCEPT, and IF commands use the FORTRAN library format conversion routines. If the program you are debugging does not have D, E, F, or G format specifications, then the linker does not load the routines that FDT requires for C, D, and E modes. If you require C, D, or E modes you can force the linker to load the required routines with the following procedure:

1. When you link your program, include the /I switch in the first linker command line.

2. The linker responds with:

   ```
   LIBRARY SEARCH:
   ```

   Type RCI$ (followed by a carriage return) to request the floating-point conversion routines and a null line (carriage return only) to end the library search list.

The resulting program contains all the necessary conversion routines.

3.4 ON-LINE DEBUGGING TECHNIQUE (ODT)

If you use assembly language routines with FORTRAN routines, you may sometimes need to use FDT and ODT at the same time. There is no interaction between the two debugging techniques except when ODT, which runs at a higher priority, occasionally interrupts the output of FDT or of the FORTRAN program. FDT does not use breakpoint or T-bit traps.

3.5 EXECUTION SPEED

FDT uses the FORTRAN internal statement number traceback feature to gain control at the beginning of a FORTRAN program and to execute PAUSE and STEP commands. Therefore, FDT adds some overhead to the execution of each FORTRAN statement and slightly reduces the execution speed of any FORTRAN program.

The amount of FDT overhead time in a main FORTRAN program is difficult to reduce. However, you can get fully debugged, time-critical subroutines to run at full speed. You should compile these routines with the traceback feature disabled. You cannot use FDT within these routines. However, you can use the entry-pause feature of FDT for these routines because the entry pause does not require the traceback feature. If you issue a STEP command immediately before a subroutine compiled without internal statement numbers, control proceeds to the next executable FORTRAN statement in a routine with the traceback feature enabled.

Execution speed with FDT is affected by FDT pauses. Each active pause slows the execution of all FORTRAN statements. It particularly slows those with internal statement numbers greater than that of the statement where the pause occurs. The program runs more quickly if you reset pauses that are no longer necessary. Entry pauses (see Section 2.9) do not require overhead for traceback and hence are more time efficient than other PAUSE commands.

APPENDIX A

FDT COMMAND SUMMARY

Command	Parameters	Description
ACCEPT	loc=value	Assign value as the new contents of loc.
	'text'	Print text on terminal.
	loc	Accept a value from the terminal and assign it to loc.
CONTINUE	[ntimes]	Resume FORTRAN execution.
DIMENSION	name(i,j,...)[,loc]	Associate a location specification and a subscript list with a name.
ERASE	name[,name2,...]	Remove name associations.
GOTO	label	Unconditional branch command changes execution sequence within an FDT macro.
IF	loc<rel>value;FDT command	Execute the FDT command only if the condition is true.
MACRO	m(FDT commands)	Define FDT macro m.
	m	Execute FDT macro m.
	m()	Delete FDT macro m.
NAME	name[,loc]	Associate a location specification with a name.
PAUSE	proc,isn [AFTER ntimes] [MACRO m]	Create an FDT pause at internal statement number isn of procedure proc.
RESET	proc,isn	Remove an FDT pause.
START		Begin execution of main program.
STEP	[n]	Resume execution and execute n statements.

FDT COMMAND SUMMARY

Command	Parameters	Description
STOP		Return to operating system.
TYPE	loc 'text'	Print value or text on the terminal.
WATCH	loc	Cause an FDT pause when the contents of the specified location change.
WHAT		Print FDT status.

APPENDIX B

FDT LOCATION SPECIFICATION FORMATS

Format	Meaning
xxx	Location offset in octal bytes
name	Named location
name+xxx	Relative addressing
name(i,j,k,...)	Subscripted name
.MAIN.	Base address of the FORTRAN main program (if the main program was not named in a PROGRAM statement)
.BCOM.	Base address of FORTRAN blank common
.ABS.	The zero address of memory

APPENDIX C

FDT MODES

Mode	FORTRAN Type	Description
I	INTEGER*2	16-bit value displayed in decimal
J	INTEGER*4	32 bits, first 16 displayed in decimal
L	LOGICAL*4	32 bits, displayed as T or F
M	LOGICAL*1	8 bits, displayed as T or F
E	REAL*4	32 bits, scientific notation
D	REAL*8	64 bits, scientific notation
C	COMPLEX	64 bits, real and imaginary parts
B	BYTE	8 bits, displayed in decimal
R	----	16 bits, displayed as 3 RAD50 characters
O	----	16 bits, displayed in octal
An	----	A string of n ASCII characters (1 <= n <= 255)
Z	----	ASCIZ string (as used in the FORTRAN string handling package).

Any mode in the above table can be preceded by the letter "P" to indicate that the associated location represents a FORTRAN parameter variable.

APPENDIX D

FDT ERROR MESSAGES

Message	Description of Error
?BAD DIM	An invalid dimension limit was specified.
?BAD LOC	The location specification is not aligned correctly according to its mode or it references a location outside the program bounds.
?BAD MACRO	An attempt was made to redefine a macro while it was executing.
?BAD SUBS	Subscripts are in an invalid format or cause overflow.
FDT START FAIL	Invalid main program, bad start address, or internal statement numbers not enabled in main program.
?FORMAT	The input constant is not in the expected mode.
?LABEL	The label specified does not exist.
?MACRO #	The macro number is not in the range 0-7.
?NO CONVERSION	Floating-point formats are not available with current FORTRAN program (see Section 3.3).
?NO ROOM	Not enough memory space to define a new PAUSE, MACRO, or NAME.
?ONLY IN MACRO	The operation attempted is valid only within an FDT macro.
?PAUSE NOT FOUND	An attempt was made to remove a PAUSE that is not active.
%SUBSCR OUT OF BOUNDS	The specified subscripts exceed the declared dimensions. (Warning message only.)
?UNDEFINED	Syntax error or undefined name.

INDEX

Abbreviations, 1-3, 2-20
.ABS., 3-1, 3-3
Absolute address, 3-1
ACCEPT, 1-3, 2-2, 2-10, 3-4
Address,
  absolute, 3-1
  base, 1-4, 2-6, 3-2, 3-3, 3-4
  indirect, 3-4
  relative, 3-1, 3-3
/An, 2-3, 2-6, 2-21
Array,
  virtual, 2-6
Array element, 1-6, 1-9, 3-4
ASCII mode, 2-6
ASCIZ mode, 1-10, 2-21
Automatic FDT pause, 1-1, 1-9

Base address, 1-4, 2-6, 3-2, 3-3, 3-4
.BCOM., 1-5, 3-3
Blank common, 1-5, 3-3
BOUNDS,
  %SUBSCR OUT OF, 1-6

Code,
  FDT mode, 1-7, 1-8
  inline, 1-1
  mode, 1-7
  threaded, 1-1
Command types,
  FDT, 1-2
Commands,
  FDT control, 1-3
  information transfer, 1-3, 1-4, 2-21
  program control, 1-2
Commas, 2-2, 2-20
Common,
  blank, 1-5, 3-3
  named, 1-5, 3-1, 3-3
Conditional transfer of control, 2-9
Constant,
  string, 2-2
CONTINUE, 1-2, 1-11, 2-5, 2-14, 2-15
Control commands,
  FDT, 1-3
Convention,
  mode, 2-3
Conventions, 2-2
  syntax, 1-3, 1-4
Conversion,
  format, 3-4
Count,
  execution, 2-5, 2-15, 2-17, 2-22
CTRL/C, 2-8
Current procedure, 1-4, 1-7, 1-10

Data type, 1-4, 1-7, 1-8, 1-9
Debugging procedure, 1-1, 1-2
Defining a label, 2-8
Defining a macro, 2-10
Deleting a macro, 2-11
DIMENSION, 1-3, 1-7, 1-8, 1-9, 2-6
Dimensions, See Subscripts, number of, 1-6
Displacement, 1-6, 3-3, 3-4
Dummy variable, 1-7

Element,
  array, 1-6, 1-9, 3-4
Entry pause, 1-9, 2-15, 2-22, 3-5
Entry point, 1-9, 2-15
ERASE, 1-3, 1-7, 2-7
Error message, 1-4, 1-10, 2-14, 2-17
Executable statement, 2-15, 2-20
Executing a macro, 2-11
Execution count, 2-5, 2-15, 2-17, 2-22
Execution speed, 3-5

FAIL,
  FDT START, 1-2
FB,
  RT-11, 3-1
FDT, 1-1
FDT addressing, 1-4, 1-5, 1-6, 3-2, 3-3, 3-4

FDT command types, 1-2
FDT control commands, 1-3
FDT macro, 2-2, 2-3, 2-10, 2-11, 2-14
FDT mode, 2-21
FDT mode code, 1-7, 1-8
FDT pause, 1-1, 1-2, 1-4, 1-9, 1-10, 2-13, 2-21, 2-22
FDT START FAIL, 1-2
Foreground/background, RT-11, 3-1
Format conversion, 3-4
Format conversion routines, 3-4
FORTRAN PAUSE, 1-9
FRUN, 3-1

GOTO, 1-3, 2-8, 2-10

IF, 1-3, 2-9, 2-10, 3-4
IF,
  logical, 1-11, 2-9, 2-18
Implicit macro, 2-11
Indirect address, 3-4
Indirect addressing, 3-4
Information transfer commands, 1-3, 1-4, 2-21
Inline code, 1-1
Internal statement number, 1-1, 1-10, 2-5, 2-15, 2-22, 3-5

?LABEL, 2-8
Label,
  defining a, 2-8
  numeric, 2-8
Library, 2-13, 3-4
Link, 3-4
Link map, 1-1, 3-1
Literal,
  text, 2-2
Literal text, 2-20
Location,
  named, 1-4, 1-5
  offset, 1-5
  relative, 1-4
  subscripted name, 1-4, 1-6
Location relative, 1-6
Location specification, 1-4, 2-12, 3-2
Logical IF, 1-11, 2-9, 2-18
Logical relation, 2-9
Loops, 2-8

MACRO, 1-3, 2-10
Macro,
  defining a, 2-10
  deleting a, 2-11
  executing a, 2-11
  implicit, 2-11
Main program name, 1-2
.MAIN., 3-3
Map,
  link, 1-1, 3-1
  storage, 1-1, 1-5, 1-8, 2-6, 3-2
Message,
  error, 1-4, 1-10, 2-14, 2-17
Mode, 1-7, 2-3, 2-9, 2-12, 2-20
Mode,
  ASCII, 2-6
  ASCIZ, 1-10, 2-21
  step, 2-18
Mode code, 1-7
Mode convention, 2-3
Module conversion, 2-4, 2-9

NAME, 1-3, 1-5, 1-7, 1-8, 1-9, 2-12, 3-1, 3-3
Named common, 1-5, 3-1, 3-3
Named location, 1-4, 1-5
?NO CONVERSION, 2-4
?NO ROOM, 2-15
Number of dimensions, 1-6
Numeric label, 2-8

ODT, 3-4
Offset, 1-5, 2-6
Offset location, 1-5
Overhead, 3-5

/P option, 3-1
Parameter, 1-4, 1-8, 3-4
Pause, 1-9

PAUSE, 1-2, 2-5, 2-11, 2-13, 2-15, 2-21, 3-5
Pause,
  automatic FDT, 1-1, 1-9
  entry, 1-9, 2-15, 2-22, 3-5
  FDT, 1-1, 1-2, 1-4, 1-9, 1-10, 2-13, 2-21, 2-22
  statement, 1-9, 2-15
  step, 1-9
  watch, 1-9, 2-21
Procedure,
  current, 1-4, 1-7, 1-10
Program control commands, 1-2
Program name,
  main, 1-2
Prompt, 1-2, 1-3, 2-2, 2-10

RAD50, 1-7, 2-3
RCI$, 3-4
REENTER, 1-9, 2-8
Relation,
  logical, 2-9
Relative,
  location, 1-6
Relative address, 3-1, 3-3
Relative location, 1-4
RESET, 1-2, 2-16
RESUME, 3-1
RSTS/E, 1-1
RT-11, 1-1
RT-11 FB, 3-1
RT-11 SJ, 1-10
RUN, 2-8

Single job,
  RT-11, 1-10
Specification,
  location, 1-4, 2-12, 3-2
Speed,
  execution, 3-5
START, 1-2, 1-9, 1-11, 2-8, 2-11, 2-17, 2-21
START FAIL,
  FDT, 1-2
Statement,
  executable, 2-15, 2-20
Statement number,
  internal, 1-1, 1-10, 2-5, 2-15, 2-22, 3-5
Statement pause, 1-9, 2-15
STEP, 1-2, 1-9, 1-11, 2-11, 2-18, 2-21, 3-5
Step mode, 2-18
Step pause, 1-9
STOP, 1-2, 2-19
Storage map, 1-1, 1-5, 1-8, 2-6, 3-2
String constant, 2-2
%SUBSCR OUT OF BOUNDS, 1-6
Subscript, 1-6, 2-9, 3-3
Subscripted name location, 1-4, 1-6
Syntax conventions, 1-3, 1-4

Text,
  literal, 2-20
Text literal, 2-2
Threaded code, 1-1
Transfer commands,
  information, 1-3, 1-4, 2-21
Transfer of control,
  conditional, 2-9
  unconditional, 2-8
Two's complement binary, 3-3
TYPE, 1-3, 2-2, 2-20, 3-4

Unconditional transfer of control, 2-8
?UNDEFINED, 1-4, 1-10, 2-6

Variable,
  dummy, 1-7
Virtual array, 2-6

WATCH, 1-2, 1-10, 2-21
Watch pause, 1-9, 2-21
WHAT, 1-3, 2-15, 2-22

/Z, 1-10, 2-3, 2-21

#, 2-3
%SUBSCR OUT OF BOUNDS, 1-6
.ABS., 3-1, 3-3
.BCOM., 1-5, 3-3
.MAIN., 3-3
/An, 2-6, 2-21
/P option, 3-1
/Z, 1-10, 2-3, 2-21
?LABEL, 2-8
?NO CONVERSION, 2-4
?NO ROOM, 2-15
?UNDEFINED, 1-4, 1-10, 2-6

FORTRAN
Debugging Technique
AA-H069A-TC

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____ Telephone_____

Street_____

City_____ State_____ Zip Code or Country_____

Please cut along this line.

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 152
MARLBOROUGH, MA
01752

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Documentation
200 Forest Street MR1-2/E37
Marlborough, Massachusetts 01752

digital equipment corporation

Printed in U.S.A.