CFD 21-DEC-72 17:57 13445

NDDT Symbolic Debugger User'S Guide

This is a preliminary edition of the soon to be released NDDT debugging system. Your comments and suggestions are appreciated. 4

Updated to Jan 2, 1973

5a
5b

CONTENTS

5b7

INTRODUCTION 5c

A new version of the NDDT system is being released in stages. 5c1

NDDT is a symbolic debugging routine for LLO programs written for the NLS system. It is designed to be easy to use, relieving the programmer from the typical "where am I now" syndrome of some debuggers. This document provides a description of the features currently implemented and supposedly working as advertised. This document will be kept current and should serve as a NDDT user's guide. Several additional features are implemented in various stages of development, but users are cautioned against "playing" with them until their descriptions appear in this document. Your comments and suggestions are of course encouraged and most welcome. 5c2

5c3

NDDT COMMANDS 5 D

All commands to NDDT consist of single upper or lower case characters or sequences of single characters. 5d1

Most commands are terminated with a CA or CR which may appear interchangably. NDDT echos the full text of each command allowing the user to verify his command state. A command may be aborted prior to completion by typing in CD or RUBOUT. 5d2

5d3

KEYWORD commands 5d4

Each keyword command defines a parse submode for NDDT and establishes the legal context for subsequent command input. 5d4a

5d4b

BREAKPOINT commands 5d4c

The breakpoint commands allow the setting, clearing, and printing of breakpoints. Breakpoints may be unconditional, or may be conditioned by a relational expression or boolean procedure invocation. Breakpoints are numbered by NDDT beginning with zero and they may be referenced by this number or by their breakpoint address. 5d4c1

The formal syntax of the command is: 5d4c2

B[REAKPOINT] 5d4c2a

C[LEAR] 5d4c2a1

A[LL] CA - clear all breakpoints now in effect 5d4c2a1a

num CA - clear only the indicated breakpoint 5d4c2a1b

P[RINT] 5d4c2a2

CA - print all breakpoints now in effect 5d4c2a2a

num CA - print only the indicated breakpoint 5d4c2a2b

S[ET] <addrexp> 5d4c2a3

CA - set unconditional bkpt 5d4c2a3a

C. 5d4c2a3b

C[ALL] <procname> CA 5d4c2a3b1

- the named procedure is executed at the breakpoint address, and the breakpoint is taken if the procedure returns TRUE. 5d4c2a3b1a

R[EPLACES BREAKPOINT:] num CA 5d4c2a3b2

- the specified breakpoint replaces a previous breakpoint. 5d4c2a3b2a

T[EST] <addrexp> <relationop> <constexp>
CA 5d4c2a3b3

- the contents of the cell identified by the address expression is compared to the constant expression when control passes through the breakpoint location. If the relation is TRUE then the breakpoint is executed. 5d4c2a3b3a

The relational opeator may be any one of the following: (=,#,<,>,<=,>=). 5d4c2a3b3b

5d4c2a3b3c

CONTINUE command 5d4d

The CONTINUE command is used to proceed from a breakpoint and to return to NLS after entering NDDT via the Control-H trap. 5d4d1

The syntax is: 5d4d2

C[ONTINUE] CA - continue interrupted execution 5d4d2a

5d4d2b

DEFINE commands 5d4e

The define command is used to create a new DDT symbol, or alter the value of any previously defined DDT symbol, including those created both by users and compilers. The command has two mode flavors, NEW and OLD for creating new symbols and redefining old symbols respectively. 5d4e1

Note that the mark stack must not be empty in order for there to be space for new symbols to be defined. If there are no entries in the mark stack, the user may define one using the MARK command (i,e, M[ARK SYMBOL TABLE] S[ET AT] <newblockname> CA). 5d4e2

Symbols defined with this command have a global scope, and may be used to satisfy external references for L10 user program compilations. 5d4e3

The syntax of the command is: 5d4e4

D[EFINE] 5d4e4a

N[EW] <symbol> CA <enterexp> CA 5d4e4a1

O[LD] <symbol> CA <enterexp> CA 5d4e4a2

5d4e4a3

FIND command 5d4f

The FIND commad provides a mechanism for bounded and masked word content searches. A word value, mask value, and address limits are specified, and the specified range is searched for a masked equality with the specified value. The address and contents of each word which match are printed out. If a mask is not specified, then a mask of 36M (all ones) is assumed. 5d4f1

After entering all of the requested information, an additional CA is required for validation of the command. 5d4f2

The syntax is: 5d4f3

F[IND CONTENT:] <enterexp> CA [MASKED BY:] <enterexp> CA 5d4f3a

[LOWER ADDRESS:] <addrexp> CA [UPPER ADDRESS:] <addrexp> CA 5d4f3b

[OK] CA 5d4f3c

The syntax of the <enterexp> includes PDP10 instruction forms as well as all address expressions. 5d4f4

5d4f5

GOTO command 5d4g

The GOTO command provides the capability for unconditional transfer to a specified location. Note that it is the users responsibility to provide a return path to the system (BREAKPOINT) if desired, as the GOTO mechanism is a uncontrolled transfer. 5d4g1

G[O TO] <addrexp> CA - jump to the specified location 5d4g1a

MARK commands 5d4h

The MARK commands allows the user to control the searching of the DDT symbol table. A symbol table mark stack is maintained by NDDT and it controls the lookup process for identifying symbols used by NDDT. Entries in this stack are names of symbol table blocks corresponding to LIO FILES or user PROGRAMS. When a user program is compiled, its name defines a block in the ddt symbol table, and this name is automatically placed at the top of the mark stack. Local as well as global symbols are known for blocks in the mark stack only. 5d4h1

The MARK command allows blocks to be added to and deleted from the mark stack, and its current contents may be printed out. 5d4h2

The syntax of the command is as follows: 5d4h3

M[ARK SYMBOL TABLE:] 5d4h3a

C[LEAR BLOCK] <blockname> CA - remove block from stack 5d4h3a1

P[RINT CONTENTS OF STACK] CA - prints current contents 5d4h3a2

S[ET AT:] <blockname> CA - add block name to stack 5d4h3a3

5d4h3a4

PROCEDURE commands 5d4i

The PROCEDURE command allows a specified procedure to replace an existing one or allows a specified procedure to be called. 5d4i1

A backup facility is provided to re-institute the original procedure in place of any replaced. 5d4i2

The command syntax looks like: 5d4i3

P[ROCEDURE] 5d4i3a

B[ACK UP TO] <procname> CA - re-institute the named proc. 5d4i3a1

C[all] <procname> 5d4i3a2

CA - call named routine - no arguments 5d4i3a2a

(expl,...) CA - call the proc with an argument list 5d4i3a2b

- the names procedure is called, passing a parameter list if specified, and the return value is typed out. A single character string literal may be passed to a procedure by enclosing the string text in double quotes (" lit "). 5d4i3a2b1

R[EPLACE PROCEDURE] <oldprocname> [BY] <newprocname> CA 5d4i3a3

- the <oldprocedure> is replaced by a <newprocedure> so that any calls on the old one go instead to the new one. The old procedure name is saved in a list and may be restored by the B[ACKUP TO] option. 5d4i3a3a

5d4i3a3b

RECORD command 5d4j

The RECORD command is used to position the internal NDDT record pointer to some L10 record definition. Subsequent SHOW RECORD commands utilize the current record definition as set by this command. 5d4j1

The syntax of this command is: 5d4j2

R[ECORD POINTER SET TO:] name CA 5d4j2a

where Name is the name of some L10 RECORD. Note that it may be necessary to use the MARK command to make local records known to the NDDT system. 5d4j2b

5d4j2c

SHOW commands 5d4k

The SHOW commands open a cell for inspection and/or modification by NDDT. At any time a single address value is known as the current location, and the show

command or any of its subcommands manipulate the current location. 5d4k1

The mode specification on this command refers to the mode of the printed output, and if set, remains in effect until changed by a new mode specification. The type specification is used to determine how the current location is to be used for generating the printed output. If the assignment option (←) is used, the the current contents of the entity are printed and then replaced by the new value. 5d4k2

The syntax of the SHOW command is: 5d4k3

S[HOW] <type> <showbody> 5d4k3a

<type> - if given, sets the current entity type 5d4k3a1

L[OCATION] - type is a memory location 5d4k3a1a

R[ECORD] - type is a L10 record 5d4k3a1b

S[TRING] - type is a NLS string 5d4k3a1c

<showbody> - main body of the command 5d4k3a2

<saddrexp> <terminator> 5d4k3a2a

<reladdrexp> 5d4k3a2b

<terminator> 5d4k 3a2c

5d4k3a3

where the preceeding syntactic forms are defined as: 5d4k3b

<saddrexp> - sets the current loc. to <saddrexp> 5d4k3c

::= <addrexp> ! <saddrexp> . <L10recordfield> 5d4k3c1

<reladdrexp> - relative addressing, sets current loc. 5d4k3d

↑ - show preceeding entity 5d4k3d1

LF - show next entity 5d4k3d2

N[EXT] - show next entity 5d4k3d3

TAB - set current loc to address field of current loc. 5d4k3d4

<terminator> - terminates the show command 5d4k3e

CA - show current location in current mode 5d4k3e1

<modespec> - if given, sets new mode specification 5d4k3e2

C. 5d4k3e2a

N[UMERIC] - set mode to numeric 5d4k3e2a1

S[YMBOLIC] - set mode to symbolic 5d4k3e2a2

T[EXT] - set mode to text mode 5d4k3e2a3

← <enterexp> - <addrexp> or PDP10 instruction 5d4k3e3

- changes the contents of the current location. 5d4k3e3a

- if the current location is a string, then a literal string is typed in, which replaces the current contents of the string. 5d4k3e3a1

5d4k3e4

Modifying fields within records 5d4k4

All record fields are treated as unsigned integer values, and the contents of any record field may be inspected/modified using the qualified form of the <saddrexp> with the SHOW LOCATION command. 5d4k4a

Example: 5d4k4b

Assume the existence of the following L10 record template for the definition PDP10 instruction format: 5d4k4c

(inst) RECORD % PDP10 instruction format % 5d4k4c1

```
        addr10[18],        % address field %          5d4k4c1a
        index10[4],        % index reg %              5d4k4c1b
        indir10[1],        % indirect bit %           5d4k4c1c
        accum10[4],        % accumulator reg %        5d4k4c1d
        opcod10[9];        % instruction opcode %     5d4k4c1e
```

Suppose we wish to change the accumulator field of an instruction at location TEST+31 from A1 to A3. The following NDDT command could be uset (the info echoed by NDDT is in brackets): 5d4k4d

```
S[HOW] L[OCATION] TEST+31.accum10← [A1 ←] A3 CA
                                                  5d4k4d1
```

5d4k4d2

TRACE command 5d4l

The TRACE command is used to record each time control passes through a particular location. Every time control passes through an active trace location, that fact is noted by typing out the address of the location being traced. The syntax of the command is: 5d4l1

```
T[RACE LOCATION] <addrexp> CA                     5d4l1a
```

5d4l1b

VALUE command 5d4m

The VALUE command is used to show the value of an address expression. The current typeout mode may also be set by this command. 5d4m1

The syntax is: 5d4m2

```
V[ALUE OF] <addrexp>                              5d4m2a
```

CA - print value in current typeout mode 5d4?M2a1

<modespec> CA - change typeout mode, then print value 5d4m2a2

5d4m2a3

special character commands 5d5

The special character commands are intended to allow high user interaction rates for commonly used functions by compressing the command syntax to a single character. The = command is used to type out the contents of the current location in numeric mode without changing the current mode specification, and the other commands are really subcommands of the SHOW command and are processed exactly as if they had been preceeded by the command S[HOW]. 5d5a

The commands are: 5d5a1

=	Show current location in numeric typout	5d5a1a
←	Assign a value to current location	5d5a1b
↑	Show previous location	5d5a1c
LF	Show next location	5d5a1d
TAB	Show location addressed by current location	5d5a1e

5d5a1f

Built in NDDT symbols 5e

A certain set of symbols are built in to NDDT and are recognized in address expressions. These symbols and there significance are explained below. 5e1

NDDT symbols 5e2

M	frame base register	5e2a
S	frame top register	5e2b
RP	record pointer register	5e2c
A1	register A1	5e2d
A2	register A2	5e2e
A3	register A3	5e2f
A4	register A4	5e2g

R1	register R1	5e2h
R2	register R2	5e2i
R3	register R3	5e2j
R4	register R4	5e2k
FRAME	current stack frame description	5e2l
F	current stack frame description	5e2m
RET	current stack frame return address	5e2n
SIG	current stack frame signal location	5e2o
MARK	current stack frame mark location	5e2p
PARMS	current stack frame parameters	5e2q
P	current stack frame parameters	5e2r
LOCALS	current stack frame LOCALS	5e2s
L	same as LOCALS	5e2t
TOP	top stack frame in procedure stack	5e2u
BASE	first stack frame in procedure stack	5e2v

5e2w

Using NDDT from DNLS/TNLS 5f

NDDT is entered via a control-h interrupt in both DNLS and TNLS. The commands for enabling this interrupt mechanism are slightly different in each system. They are: 5f1

From DNLS: G/O TO/ N/DDT ARM CONTROL-H/ CA 5f1a

From TNLS: G/O TO/ D/DT/ CR 5f1b

After enabling the control-h mechanism, NDDT is initially entered by typing in control-h at any time. NDDT will respond with its command herald character (>), indicating that NDDT is ready to accept a command. Control remanins in NDDT until the user leaves via the CONTINUE or GO TO commands. The CONTINUE command is normally given after

setting breakpoints, replacing procedures, etc, and if NDDT was entered via a control-h interrupt, control will return to the DNLS/TNLS system. 5f2

NDDT address expressions 5g

The formal BNF syntax for address expressions is given below. Operators are evaluated left to right with no hierarchy, space is equivalent to the addition operator (+). 5g1

5g2

<addrexp> ::= <term> ! <addrexp> <op> <term> 5g3

<term> ::= <symbol> ! <constant> 5g4

<constant> ::= <normalconst> ! <halfwordconst> 5g5

<normalconst> ::= <octal digit> ! <normalconst> <octal digit> 5g6

! - <normalconst> 5g7

<halfworoconst> ::= <normalconst>,,<normalconst> 5g8

<octal digit> ::= 0 ! 1 ! 2 ! 3 ! 4 ! 5 ! 6 ! 7 5g9

<op> ::= SP ! + ! - ! * ! / 5g10

5g11

NDDT symbol conflicts 5h

When user programs are being compiled and debugged, it is often convenient to be able to replace an existing routine by a new version of some the same routine. Some method of name qualification is required to distinguish between new and old occurrences of the same symbol. An escape mechanism is built into NDDT which allows this. Any symbol preceeded by the character semi-colon (;) is assumed to be refer to an old occurrence of the symbol. The semi-colon has the effect of disabling the symbol table marking mechanism for the following symbol, causing it to be identified in the "old" section of the symbol table. 5h1

Example use of the symbol escape code 5h2

Suppose an existing routine named TEST is to be replaced by a new version of the same routine for chechout purposes. The NDDT PROCEDURE REPLACE command can be used as follows: 5h2a

```
P[ROCEDURE] R[EPLACE ] ;TEST [BY] TEST CA          5h2a1
```

In this example the existing procedure named TEST is replaced by a new version having the same name. 5h2b