DATA GENERAL CORPORATION
Southboro,
Massachusetts 01772
(617) 485-9100

# NOVA DEBUGGER MANUAL

Tapes: 091-000003-00 (high core)
091-000010-00 (low core)
091-000011-00 (8K high core)
091-000015-00 (16K high core)

June, 1969

093-000020-00

TABLE OF CONTENTS

## 1. Introduction

The NOVA Debugger is a program used to interface with user routines as an aid in debugging. It provides for up to four active breakpoints within the user's routines. The accumulators, Carry, and memory can be examined and modified from the teletype after a breakpoint has occurred. The machine state can be monitored during execution of a routine using simple commands to the Debugger from the teletype. The Debugger interfaces with any type routine, including those using the NOVA interrupt structure. The Debugger can also be used to punch ranges of memory in binary format acceptable as input to the Binary Loader.

## 2. Commands

### 2.1 Format

Commands to the Debugger are of the general form:

(argument) command

Commands are single teletype codes, usually letters. The argument may be null, a single digit, an address, or an expression. Expressions are of the form:

Octal number ± octal number ± ...

where "+" causes octal addition and "_" octal subtraction.

A carriage return will be indicated by the symbol "↵". A line feed will be indicated by the symbol "↓".

### 2.2 Descriptions

#### 2.2.1 Examine and Modify Commands

Various commands are available which enable the user to examine and change memory locations, accumulators, and certain Debugger registers. If we denote all of these as "registers", a register is said to be "open" after it has been examined. The option is available to simply "close" the register or to type an expression which is used to replace the current contents of the register before closing it.

The accumulators can be examined by the command,

nA

If "n" is not specified, the contents of all four accumulators will be typed out. To examine individual accumulators, "n" specifies the specific accumulator. For example, if "1A" is typed, the Debugger will respond /DDDDDD, where the D's rep-resent the octal digits of the contents of AC1. AC1 is now opened and may be modified or simply closed. A ↵ closes the register. The contents may be modified by typing an expression followed by a ↵ . Expressions may contain the special symbol "$" which has the meaning "the current contents of." For example, if AC2 contains 17, after

```
2A/ØØØØ17   $+3-1Ø
```

AC2 will contain 12.

Carry can be examined and modified in a similar manner by the command "C".

The status of Interrupt Enable can be examined by the command "I". Bit 15 will be set if interrupts were enabled when the Debugger was entered because of a breakpoint, reset otherwise. The remainder of the register will be zero.

The Debugger tests the teletype busy flags upon entry and will wait to take control until they have been cleared. The status of the teletype done flags can be examined by the command "T". Bit 14 represents TTO Done and bit 15 represents TTI Done. Each bit will be set if the respective Done flip-flop is set. The remainder of the register will be zero.

Any memory location can be opened by typing

```
adr/
```

where "adr" is the octal address of the memory location. The contents of the location will be printed and can be modified by typing an expression and closing the register. "$" can be used in the expression as well as second symbol, ".". A period has the meaning "address of the register most recently closed." For example, if "53/ ØØ4777↵" was the last command and response, the command "./" will cause "ØØ4777" to be printed again. In addition to using a carriage return to close the register, two additional codes can be used. A line feed closes the register and opens the succeeding location. A "↑" closes the register and opens the preceding memory location. The following command sequence illustrates these options.

```
1Ø71/ Ø176ØØ ↓
1Ø72/ 1ØØØØØ ↑
1Ø71/ Ø176ØØ ↵
```

To modify memory without examining, the following should be given:

```
adr!
```

Location "adr" is now open for modification. Adjacent registers can be open using "↑" or line feed.

### 2.2.2 Search Memory Commands

Three commands are associated with the search of memory over a specified range for a given configuration. The search command itself is of the form

adr1, adr2S

where "adr1" is the address where the search is to be started and "adr2" is the last address to be examined. If no address arguments are given, the search begins at Ø and ends at 77777. If only one address argument is given, the search begins at Ø and ends at the specified address. Two registers are used by the Debugger to determine how the search is performed. The first is a 16-bit mask register which is examined and modified exactly as an accumulator. The mask register is opened by the command "M". The second register contains a comparison word which is opened by the command "W". If "adr" is the current address being examined, then a match is said to occur if contents (adr)$\wedge$ M=W. A match causes the Debugger to print the memory location and its contents.

A common method of examing all words in a given address range is a search with M and W = Ø, since contents (adr) $\wedge$ M = Ø = W in all cases. To search for all ISZ instructions, one would set M= 174ØØØ and W= 01ØØØØ.

### 2.2.3 Breakpoint Commands

Breakpoints are the key elements of the Debugger. A breakpoint is an instruction address at which the user would like to stop execution and examine the current state of his routine. NOVA Debugger provides for up to four distinct breakpoints.

The addresses of the current breakpoints will be typed out by using the command "B". To change or activate a given breakpoint, the format is

adrB

where "adr" is the octal address desired for the breakpoint. The Debugger will assign this address to one of the four breakpoints (numbered BØ to B3), unless all are active. In the latter case, the Debugger will respond with "?" to indicate all breakpoints are in use. The command "D" can be used to deactivate all breakpoints. Alternatively, the command

nD

can be used to selectively deactivate breakpoint n ($\emptyset \leq n \leq 3$).

Upon execution, if the instruction at the breakpoint is encountered, execution halts before the breakpoint instruction is executed and control is transferred to the Debugger. Indication to the teletype is of the form adrBn where "adr" is the breakpoint address and "n" is the breakpoint number. The contents of all accumulators will be printed. The user may further examine the current machine state by using any of the Debugger commands. If the user should examine any breakpoint address, he will find the proper contents. The Debugger,upon entry, replaces the active breakpoint with their original contents, leaving breakpoints invisible to the user. However, if the user routine executes a HALT or transfers wildly, the breakpoint can be seen by examining the address from the console. Breakpoint n will appear as the instruction.

```
JMP  @1n.
```

The Debugger uses locations 1Ø to 13 of page Ø to store address entry points to itself.

Some caution should be used in the placement of breakpoints. If an arbitrarily complex subroutine could be transferred to at the breakpoint, the user should have no difficulty (since this is actually what happens). The following restrictions should, however, be noted.

1. The breakpoint should never be placed at a data word (these words are never executed).

2. The breakpoint should never be placed at an instruction which is modified during execution.

3. If the teletype busy flags are set upon a breakpoint entry, the Debugger will wait until they have been cleared before taking control. If TTI Busy is set and a tape is not mounted in the reader or a key is not depressed, the Debugger will loop continuously.

4. The breakpoint should not be placed at a point where interrupts cannot be held off for a long period of time, since the Debugger executes an INTDS upon entry.

5. The breakpoint should not be placed in an interrupt routine after a INTEN instruction, since another interrupt may occur and destroy location Ø before the Debugger gains control.

6. The breakpoint cannot be placed on any instruction that enables or disables interrupts, e.g. INTDS.

### 2.2.4 Run Commands

The command issued to cause the Debugger to transfer control to the user is

adrR.

If "adr" is specified, the Debugger will transfer to that address. If no "adr" is specified, the Debugger will transfer to an address the user must initialize in a starting location register. This register is opened and modified by the command "L".

To return control after a breakpoint, the command

nP

is given. If "n" is given, execution will procede from the breakpoint and control returned to the Debugger from that breakpoint only after "n" times through it. For example, 3P will cause the breakpoint instruction to be executed three times before a trap finally occurs. Note that 1P has the same effect as P. There are four count registers, each of which is associated with a breakpoint. These registers can be examined and modified by the command

nN

where "n" is the breakpoint number. These registers contain the procede count used in conjunction with the "P" command. The user may modify the procede count for any breakpoint other than the one which he intends to procede from. In other words, "nP" unconditionally overwrites the count register for the current breakpoint.

### 2.2.5 Punch Commands

The Debugger provides the user with the facility for punching binary tapes from memory. In this mode it performs a function identical to the Binary Punch Program (see program description 093-000001). A punch device register is defined to determine the output device. This register is opened with the command "H". If the output device is to be the teletype punch, the register should be set to Ø. If the output device is to be high speed punch, the register should be set to 1.

To punch blank leader on the output device, the command

nF

should be given. This will cause "n" inches of leader to be punched.

Ranges of memory can be punched in binary format by the command

```
adr1, adr2P
```

Memory from "adr1" to "adr2" inclusive will be punched.

An End Block (also known as a Start Block) can be punched using the command

```
ardE.
```

If no "adr" is given, the Loader will HALT after reading the block. If "adr" is given, the Loader will transfer control to this location after reading the block.

### 2.2.6 <u>Special Commands</u>

One special command is defined. Its form is:

```
exp=
```

The expression represented by "exp" is evaluated by the Debugger and its value is printed. For Example,

```
       7= returns 7
561-472= returns 67
  47+62=  returns 131
       .= returns the address of the last
          closed memory register
       $= returns the contents of the last
          closed register
```

## 3. OPERATING PROCEDURE

Two binary tapes of the Debugger will be provided as part of the standard NOVA software package. The Debugger is loaded using the Binary Loader and following the standard loading procedure. One binary tape will load into locations 4ØØ to 1777. The second binary tape will load into locations 62ØØ to 7577. The Debugger also requires locations 1Ø to 13 of page zero.

If control is lost during the testing process, the Debugger can be restarted at 4ØØ (or 62ØØ). Restarting will cause the contents of the accumulators to be printed and the I (interrupt) and T (teletype) registers to be cleared. All other Debugger registers and the breakpoint locations will remain unchanged.

## APPENDIX A

### Command Summary

| Command | Effect |
|---|---|
| A | Examine the contents of all accumulators |
| nA | Open accumulator n |
| B | Examine all breakpoint locations |
| adrB | Place a breakpoint at adr |
| C | Open the Carry register |
| D | Deactivate all breakpoints |
| nD | Deactivate breakpoint n |
| E | Punch an End Block to HALT |
| adrE | Punch an End Block to transfer control to adr |
| nF | Punch n inches of tape leader |
| H | Open punch device register (TTO = Ø, PTP = 1) |
| I | Open interrupt enable register (Enabled = 1, Disabled = Ø) |
| L | Open starting location register |
| M | Open mask register |
| nN | Open count register for breakpoint n |
| P | Proceed from breakpoint |
| nP | Proceed from breakpoint and break again the nth time it is encountered |
| n,mP | Punch memory in binary format from address n to address m inclusive |

DATA GENERAL
CORPORATION

| Command | Effect |
|---|---|
| R | Begin execution at the address specified in the L register |
| adrR | Begin execution at adr |
| S | Search memory from ØØØØØ to 77777 inclusive |
| adrS | Search memory from ØØØØØ to adr inclusive |
| n,mS | Search memory from address n to address m inclusive |
| T | Open teletype Done flags register (TTO Done = bit 14, TTI Done = Bit 15) |
| W | Open word register |
| ↓ | Close current register (if open) and open next register |
| ↵ | Close current register |
| ↑ | Close current register (if open) and open preceding register |
| adr/ | Open register at adr (Set "." = adr, "$" = contents (adr)) |
| adr! | Open register at adr without printing its contents |
| exp= | Evaluate exp and give the octal result |