Data General

SOFTWARE

DEBUG II
User's Manual

093–000020–03

Ordering No. 093-000020

Printed in the United States of America
Rev. 03, January 1975

NOTICE

Original Release - June, 1969
First Revision - January, 1972
Second Revision - January, 1973
Third Revision - January, 1975

This revision of the Debug II User's Manual, 093-000020-03, supersedes 093-000020-02. It is a minor revision, correcting omissions in examples and in operation using the binary relocatable version of Debug II. The list of pages changed is given at the back of the manual.

INTRODUCTION

Data General Corporation's Debug II is a program that interfaces with user routines as an aid in debugging. It provides for four active breakpoints within the user's routines. The accumulators, Carry and memory locations can be examined and modified from the teletypewriter after a breakpoint has occurred.

The machine state can be monitored during execution of a routine using simple commands to Debug II from the teletypewriter. Debug II interfaces with any type of routine, including those that use the DGC interrupt structure.

In addition, Debug II can be used to punch ranges of memory in binary format acceptable as input to the Binary Loader and to perform desk-calculator type expression evaluations as the teletypewriter.

TABLE OF CONTENTS

FORMAT OF COMMANDS

Commands to Debug II are of the general form:

```
[argument] command
```

where: argument is null, a single digit, an address or an expression.

command is a single Teletype*® code.

Argument expressions are of the form:

```
octal no. ± octal no. ± ...
```

where "+" causes octal addition and "-" causes octal subtraction. An octal number may be replaced by $, a special symbol meaning "current contents of the most recently opened register".

In the text following, a carriage return will be indicated by the symbol) and a line feed will be indicated by the symbol ↓ .

EXAMINING AND MODIFYING SPECIAL REGISTERS

Users can examine and modify registers, where "registers" denote both memory locations, accumulators, Carry and certain Debug II registers.

A register is said to be "open" after it has been examined. The options are either to simply "close" the register or to alter the current contents of the open register by typing new octal contents before closing it.

Accumulator Commands

Accumulators can be examined by the command:

```
nA
```

where n specifies the accumulator (0-3). Debug II responds by typing a slash followed by the octal contents of the accumulator. The accumulator is now open and may be modified by typing an octal expression or may be closed without alteration.

A) closes the open register. To modify the contents, type the new octal contents followed by) .

*Teletype is a registered trademark of Teletype Corporation, Skokie, Illinois.

Accumulator Commands (Continued)

For example:

ØA/ ØØØØ12)	User opens ACØ. Debug II prints the current contents (ØØØØ12), and user closes the register.
2A/ ØØØØ17 $+3-1Ø)	User opens AC2. Debug II prints the current contents (ØØØØ17). User modifies the contents to ØØØØ12 and closes the accumulator.

The command:

A

causes the contents of all four accumulators to be printed out. However, the nA command must be given if the user wishes to modify the contents of an accumulator.

Other Special Register Commands

Debug II permits the examination and modification of a number of other special registers:

The command:

C

causes the opening and printing of the contents of the Carry register. If Carry was set when Debug II was entered, the contents are ØØØØØ1; otherwise, the contents are zeroed.

The command:

I

causes the opening and the printing of the contents of the Interrupt Enable register. Bit 15 will be set if interrupts were enabled when Debug II was entered because of a breakpoint; otherwise, bit 15 is reset.

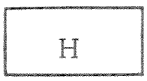
The command:

T

Other Special Register Commands (Continued)

causes the opening and printing of the contents of the TTY Done register. Bits 14 and 15 are significant. If bit 14 is set, the respective TTO Done flip-flop is set. If bit 15 is set, the respective TTI Done flip-flop is set. The remainder of register bits will be zero.

The command:



causes the opening and printing of the contents of the Punch Device Register. The register is discussed in the section on punch commands.

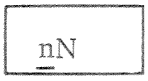
The command:

causes the opening and printing of the contents of the Starting Location register. The register is discussed in the section on run commands.

The commands:

cause the opening and printing of the contents of the Word and Mask registers respectively. These registers are discussed in the section on search commands.

The command:



causes the opening and printing of the contents of Breakpoint Count register n (n = Ø, 1, 2, or 3). The command is discussed in the section on run commands.

EXAMINING AND MODIFYING MEMORY REGISTERS

Any memory location can be opened and examined by typing:

```
adr/
```

where: adr is the octal address of the desired location,
or "$" for the contents of the current location,
or "." for the address of the most recently opened memory register.

Memory locations are modified in the same manner as are special registers, by typing the desired octal expression following the printout of the current contents.

A memory register is closed by typing). In addition to simply closing the modified or unmodified register, the user can close the current register and open the succeeding register by striking a line feed (↓), or he can close the current register and open the preceding register by striking ↑ (Shift N keys).

A memory location can be opened without its current contents being printed by the command:

```
adr!
```

Location adr is now open for modification.

When succeeding or preceding locations are opened, using ↓ or ↑, the contents will be printed only if the original command was:

```
adr/
```

If the original command was:

```
adr!
```

the contents of the adjacent location being opened will not be printed.

```
ØØ1Ø61/ Ø176ØØ )
.! 1ØØØØØ )
```

←no modification.
←open previously closed memory register (1Ø61) to alter contents.

```
ØØ1271/ Ø1755Ø ↓
ØØ1272/ Ø16635 ↓
ØØ1273/ Ø16331   ØØ6331 )
```

←User opens 1271 and two succeeding registers, changing the contents of register 1273.

```
ØØØ766! Ø176ØØ ↑
ØØØ765! 1ØØØØØ )
```

←User opens without examining registers 766 snd 765 and changes the contents of both.

```
$/ Ø16635 )
```

←User examines contents of current register.

MEMORY SEARCH COMMANDS

The search command is used to search all memory or portions of memory for given contents. The format of the memory search command is:

```
[adr1] [,adr2] S
```

where: $\underline{adr}_1$ is the optional octal address of the first location to be searched, if both arguments are given.

$\underline{adr}_2$ is the optional octal address of the last location to be searched.

When the command S is given without arguments, the search begins at location Ø and terminates at 77777. If only a single address is given, the search starts at location Ø, and terminates at the address given by the argument.

The contents are searched for a match with contents of the W (word) and M (mask) registers. The W register contains the contents to be searched for. The mask register contains a mask that is ANDed with the location's contents such that a

match occurs if:

$$W = (adr) \wedge M$$

where ( ) means "contents of".

The word and mask registers are opened by giving the commands:

W	and	M

The response from Debug II is the same as for an accumulator. Debug II will print a slash followed by the current octal contents of the register. The user may then change the contents if desired.

```
M /ØØØØØØ )          ← mask register contains Ø
W /Ø1ØØØØ Ø )        ← word register is set to Ø
4Ø,71ØS              ← user obtains printout of all locations
                       from 4Ø through 71Ø.
```

```
M /ØØØØØØ 174ØØØ )   ← set M to 174ØØØ
W /ØØØØØØ Ø1ØØØØ )   ← set W to Ø1ØØØØ
S                    ← search all of core for all ISZ instructions
```

```
M /174ØØØ 177777     ← set M to 177777 (all ones)
W /Ø1ØØØØ 132765     ← set W register to word desired
777S                 ← search from location Ø through location
                       777 for 132675. Debug II will output to the
                       console all locations that match 132675.
```

BREAKPOINT COMMANDS

Breakpoints are key elements of the debugger. They permit a user to execute a small portion of his program and then check the program status. A breakpoint is an instruction address at which the user can stop program execution, reenter the debugger and use the Debug II commands to check the current state of his program. The user can set up to four breakpoints, using Debug II.

The command to set a breakpoint has the format:

```
adrB
```

where: adr is the octal location at which the breakpoint will be set.

Debug II assigns the address to one of the four breakpoints (numbered Ø to 3), unless all breakpoints are active. If all breakpoints are in use, Debug II prints out:

```
?
```

To determine what breakpoints are in use and their addresses, the command:

```
B
```

is given. Debug II responds by typing the breakpoint number and the location at which the breakpoint is set.

To deactivate all breakpoints, the command:

```
D
```

can be given. To deactivate a single breakpoint, the command:

```
nD
```

is given, where n is the number of the breakpoint (Ø through 3).

When the user transfers from the debugger to his program for execution (see next section, Run Commands), the program will execute until an instruction at which a breakpoint is set is encountered. Execution halts before the breakpoint instruction

is executed, and control transfers to Debug II. Debug II prints out:

adrBn

on the Teletype, where:

adr is the breakpoint address and
n is the breakpoint number.

Debug II then prints the contents of all accumulators. The user can examine further the current machine state using any of the Debug II commands.

If the user examines the breakpoint address he will find the proper contents of his program, since Debug II replaces the active breakpoint upon debugger entry with the original contents of the address, leaving the breakpoints invisible to the user. However, if the user routine executes a HALT or transfers wildly, the breakpoints can be seen by examining the addresses from the control console.

The breakpoint will appear as the instruction:

JMP @1n

where: n is the number of the breakpoint; Debug II uses locations 1Ø through 13 of page Ø to store address entry points to itself.

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 what actually happens). The following restrictions should, however, be noted:

1. The breakpoint should never be placed at a data word (since these words are never executed.)
2. The breakpoint should never be placed at an instruction that is modified during execution.
3. If the Teletype busy flags are set upon breakpoint entry, Debug II 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, Debug II 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 Debug II executes an INTDS upon entry.

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

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

Examples of setting and using breakpoints are included in the next section, Run Commands.

RUN COMMANDS

Ths user can transfer control from Debug II to the user routine with the command:

```
[adr] R
```

where: adr is an optional location to which control is transferred.

If adr is not specified, Debug II will transfer to an address which the user must initialize in the starting location register. This register is opened for modification by the command:

```
L
```

To return control to the user routine after a breakpoint, the user can give the command:

```
[n] P
```

where: n is an unsigned integer indicating the number of times the breakpoint instruction is to be encountered before another break occurs returning control to Debug II.

If n is not given, a trap will occur the first time the breakpoint is encountered.

Note that the count for the P (or Proceed) command applies only to the breakpoint that causes transfer to Debug II. It is possible to set the break proceed counts for the other breakpoints using the breakpoint count registers. There are four breakpoint count registers, opened using the command:

RUN COMMANDS (Continued)

nN

where: n is the number (Ø through 3) of the breakpoint.

The contents of each breakpoint count register indicate the number of times the instruction having that breakpoint can be executed before a transfer to Debug II occurs. The user can set or modify the proceed count for any breakpoint except the one he intends to proceed from. The argument to the P command unconditionally overwrites any count in the count register for that breakpoint.

1557B L / ØØ1ØØØ) R	←user sets breakpoint at 1577 and checks contents of L register, starting execution at that address.
ØØ1557 B3 ØØØ1ØØ ØØØØ4Ø ØØØØ11 ØØØØ17	←Debug II prints out breakpoint and contents of accumulators at the break.
ØØ123Ø / Ø15731 ↓ ØØ1231/ ØØ4Ø2Ø ØØ4Ø35)	←user examines locations 1230 and 1231, changing contents of 1231.
3D P	←user deletes breakpoint 3 but proceeds with execution from the instruction that contained that breakpoint.
3134 B2 ØØØ241 ØØ1115 ØØØ777 ØØØØ25	←Control is returned to Debug II at another breakpoint.
1N 000020/ 1) 15P	←User changes the break proceed counter for breakpoint 1, and proceeds with execution from breakpoint 2, allowing 14 passes through breakpoint 2.

PUNCH COMMANDS

Debug II allows the user to punch binary tapes of ranges of memory. In this mode it performs a function identical to the Binary Punch Program (document 093-000001).

PUNCH COMMANDS (Continued)

The user selects the output device to be used for punching by setting the contents of the punch device register. The punch device register is opened for examination and modification by 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 the high speed punch, the register should be set to 1. To punch blank leader or trailer tape on the output device, the user gives the command:

```
nF
```

where: n is the number of inches (in octal) of leader to be punched.

The command determining the range of memory to be punched is:

```
adr1, adr2 P
```

where: adr_1 is the first location to be punched and

adr_2 is the last location to be punched.

adr_1 must be less than or equal to adr_2.

An End Block (also called a Start Block) can be punched using the command:

```
[adr]E
```

where: the optional argument adr is the address to which the Binary Loader will transfer control after reading the block.

If no adr is given, the Binary Loader will halt after reading the block and the user can manually insert the starting address.

A problem arises when using Debug II in punch mode with the Teletype as the output device. Since the Teletype is used both for command input and for output, if the punch is left in the "ON" position while typing Debug II commands, these characters will be punched as part of the output.

PUNCH COMMANDS (Continued)

Debug II resolves the problem in the following way. After the punch command is given, Debug II will HALT with the Carry light on. This signals the user to turn the punch ON and press CONTINUE on the console, causing the output to be punched. When the punching operation is complete, Debug II will again HALT but with the Carry light off. This signals the user to turn the Teletype punch OFF and press CONTINUE on the console. Additional Debug II commands can then be given. The signaling process by Debug II will be repeated as many times as is necessary.

EXPRESSION EVALUATION

Debug II can be used to perform simple desk calculator operations by using the special command:

expression=

where: expression follows the format for argument expressions:

octal no. ± octal no. ± . . .

and octal numbers may be replaced by either the special character $ (contents of the current location) or the special character . (most recently closed location.)

Debug II will respond with the result of evaluation in octal immediately following the = sign.

561 -472+3 - 5 = 65	
.= ØØ7567	← if 7567 was the address of the most recently closed memory register
.+$ = 007777	← if 7567 was the address of the most recently closed memory register and if the contents of the most recently closed register (not necessarily memory) were 210.

OPERATING PROCEDURE

There are two absolute binary tapes of Debug II provided as part of the standard software package. Debug II is loaded using the Binary Loader, following standard loading procedure (See Binary Loader manual, 093-000003.)

One binary tape will load into locations 4ØØ to 1777; the other loads into locations 62ØØ to 7577. Debug II also requires locations 1Ø to 13 of page zero.

A relocatable binary tape of Debug II is provided to users who wish to use relocatable loading of Debug II. The starting address specified on the relocatable binary tape is 264_8 + start of module. Thus, the programmer must check the loaded address in his listing and modify as required.

If control is lost during the testing process, Debug II can be restarted at the starting location of the particular version being used (4ØØ or 62ØØ). Restarting will cause the contents of the accumulators to be printed and the I (Interrupt Enable) and T (TTY Done) registers to be cleared. All other Debug II registers and the breakpoint locations will remain unchanged.

COMMAND SUMMARY

Special Register Commands

A	Print the contents of all accumulators.
nA	Open and print the contents of accumulator n. (n = 0 - 3)
C	Open and print contents of Carry register. (Carry set = 1; Carry reset = 0.)
H	Open and print contents of the Punch Device register. (TTP = 0; PTP = 1).
I	Open and print contents of the Interrupt Enable register. (Enabled = 1; disabled = 0).
L	Open and print contents of the Starting Location register. (See Execute Commands.)
M	Open and print contents of the Mask register. (See Search Memory Commands.)
nN	Open and print contents of Breakpoint Count register n. (n = 0 - 3).
T	Open and print contents of TTY Done register. (If bit 14 is set, TTI done; if bit 15 set, TTO done.)
W	Open and print contents of Word register. (See Search Memory Commands.)
)	Close an open special register.

Memory Register Commands

adr/	Open and print contents of memory location adr.
adr!	Open memory location adr.
)	Close an open memory register.
↓	Close an open memory register and open the succeeding register.
↑	Close an open memory register and open the preceding register.

Search Memory Commands

n, mS	Search contents of memory from address n to address m inclusive, matching contents of the location ANDed with contents of the Mask register to the contents of the Word register.

Breakpoint Commands

B	Print out all breakpoints and their contents.
adrB	Set a breakpoint at location adr of the user program.
D	Delete all breakpoints.
nD	Delete breakpoint n. (n = 0 - 3).

Execute Commands

R	Execute user program from location given in the Starting Location register.
adrR	Execute user program from location adr.
[n]P	Proceed with execution of the user program from the instruction containing the last breakpoint. Execution will halt when the breakpoint is again encountered unless the integer argument n is given, indicating the number of times the breakpoint may be encountered before execution halts.

Punch Commands

nF	Punch blank leader or trailer tape.
adr_1,adr_2P	Punch range of memory from adr_1 to adr_2.
[adr]E	Punch an end block.

Expression Evaluation

exp=	Evaluate expression exp and print the resultant octal value.

Changes from Revision 2 to Revision 3 of 093-000020, Debug II User's Manual

Page	Nature of Change
i	NOVA has been changed to DGC.
5, 10	Addresses in the memory register examples are printed out in full before printing contents.
6	The result of the memory search in the example will cause a console printout of any matches found.
13	The required procedure for using the binary relocatable tape of Debug II is included.

DataGeneral

PROGRAMMING DOCUMENTATION
REMARKS FORM

Document Title	Document No.	Tape No.

SPECIFIC COMMENTS: List specific comments. Reference page numbers when applicable. Label each comment as an addition, deletion, change or error if applicable.

GENERAL COMMENTS: Also, suggestions for improvement of the Publication.

FROM:

Name	Title	Date

Company Name

Address (No. & Street)	City	State	Zip Code

Form No. 10-24-004

FOLD DOWN FIRST FOLD DOWN

FIRST
CLASS
PERMIT
No. 26
Southboro
Mass. 01772

BUSINESS REPLY MAIL

No Postage Necessary If Mailed In The United States

Postage will be paid by:

Data General Corporation

Southboro, Massachusetts 01772

ATTENTION: Programming Documentation

FOLD UP SECOND FOLD UP

STAPLE