Section 2

CONSOLE DEBUGGER

BUG-8

TABLE OF CONTENTS

INTRODUCTION

The Heath Console Debugger, BUG-8, is designed to allow you to enter and debug machine language programs from a console terminal. BUG-8 occupies the lowest 3K of RAM, starting at 040 100. A user program can be loaded into any free RAM (random access memory) locations, and can be manipulated via BUG-8.

BUG-8 contains facilities to perform the following nine major functions:

- Display the contents of a selected memory location.
- Alter the contents of a selected memory location.
- Display the contents of any 8080 register.
- Alter the contents of any 8080 register.
- Execute the user program a single instruction at a time.
- Execute the program.
- Insert breakpoints and execute the user program.
- Load user programs from tape storage.
- Dump user program to tape storage.

A number of features were designed into BUG-8 for your convenience. Memory locations and memory and register contents may be displayed as bytes or as words, in octal, decimal, or ASCII format. With these features, you can select the most familiar or desirable format. BUG-8 also contains a single instruction facility that permits you to execute your program a single instruction at a time. And for more advanced program analysis, a breakpointing feature is included that permits you to execute several instructions in a program and then return to control to BUG-8 for analysis and/or modification.

The standard Heath console driver is incorporated in BUG-8; therefore, it responds to all the normal console input and output control characters. For this reason BUG-8 includes error detection and command completion as outlined on Page 2-3 of the "Introduction" to this Software Reference Manual.

MEMORY COMMANDS

The memory commands permit you to display and alter the contents of indicated memory locations. The format for memory display commands is:

< FORMAT CONTROL > < range > < blank >

The form for the alter memory command is:

< FORMAT CONTROL > < range > = < value list >

Format control specifies that memory display/alteration is in word or byte format, and whether octal, decimal or ASCII notation is to be used. The range specifies the memory address or addresses to be displayed or altered, and the command is executed by the typing of a blank using the space bar on the console terminal.

The Format Control

The format control consists of two characters which specify the form of the values that are to be displayed and entered. The format control field may take on a number of different forms. They are:

FORMAT CONTROL	DESCRIPTION
< null > < null >	Display/alter octal integers, byte format.
F < null >	Display/alter as octal integers, word format.
< null > A	Display/alter as ASCII characters, byte format.
FA	Display/alter as ASCII characters, word format.
< null > D	Display/alter as decimal integers, byte format.
FD	Display/alter as decimal integers, word format.

WORD FORMAT (F)

If an F is specified as the first character of the format control field, it indicates that the values are to be displayed/altered as "full words." This is to say that memory locations are taken as two byte pairs. The second byte is considered to be the high order (most significant) byte and is displayed first. The first byte is considered to be the low order (least significant) byte and is displayed last.

BYTE FORMAT (NULL)

If an F is not specified, the first character is null, indicating that the values are to be displayed/altered as single bytes. You can create a NULL by not typing any character for the format control portion of the memory command.

OCTAL FORMAT (NULL)

If no option (a NULL) is specified as the second character of the format control field, the values to be displayed/altered are taken to be octal integers. The NULL was chosen to specify both byte format and octal notation, as byte octal is the most commonly used format. A blank separates each octal integer, or octal integer pair if the F is specified.

DECIMAL FORMAT (D)

If a D is specified as the second character of the format control field, the values to be displayed/altered are taken to be decimal integers. A blank separates each decimal integer, or decimal integer pair if the F is specified.

ASCII FORMAT (A)

If an A is specified as the second character of the format control field, the values to be displayed/altered are converted from/to eight bit representations of ASCII characters. A blank separates each character, or character pair if the F is specified.

Range

The range field consists of a beginning address and an ending address. You can specify addresses by using the appropriate offset octal integers; or you can use the NULL, #, and cnt (count) as indicated below.

RANGE FORM	DESCRIPTION
ADDR < null >	Range specifies the single memory location ADDR.
ADDR1-ADDR2	Range specifies the memory locations ADDR1 through ADDR2 inclusive.
ADDR/cnt	Range specifies cnt memory locations starting at location ADDR. NOTE: cnt is a decimal integer ≤ 255.
#-ADDR	Range specifies the memory locations starting at the beginning of the previous range and ending at ADDR.
#/cnt	Range specifies cnt memory locations starting at the beginning or the previous range. NOTE: cnt is a decimal integer ≤ 255.
< null>/cnt	Range specifies cnt memory locations starting at the address following the last address of the previous range. NOTE: cnt is a decimal integer ≤ 255.
< null > -ADDR	Range specifies memory locations starting at the address following the last address of the previous range and extending to memory location ADDR.

For example, to display memory location 000 043 through 000 047, BUG-8 simply requires the user to type 43-47 followed by a blank (a blank is generated by using the console terminal space bar). For example:

```
B: 43-47 100 112 107 114 100
B:

B: /4 303 053 040 365
B:
```

NOTE: In the first example, the contents of memory locations 000 043 through 000 047 are displayed on the first line in byte format octal. The next four bytes (locations 000 050 to 000 053) are displayed when the command /4 is typed. The contents of these next four bytes are displayed as soon as a blank is typed after the /4.

If the first address specified is greater than the second address specified, an error message is generated and only the contents of the first memory location are displayed. The form of the error message is:

```
LWA>FWA
```

For example:

```
B: 47-43 LWA>FWA
100
B:
```

NOTE: If you attempt to enter a numerical address which does not fit the offset octal format, BUG-8 rejects the improper entry and sounds the console terminal bell. For example, the number 067777 does not fit the offset octal format; therefore, BUG-8 does not allow the second 7 to be entered.

Displaying Memory Contents

To display the values in the specified range and in the specified format, type a blank following the format and range fields. BUG-8 immediately executes the command. In the following examples, the contents of a number of locations, 000 043 to 000 070 in the PAM-8 ROM, are displayed in octal byte format, in octal word format, in decimal byte format, in decimal word format, in ASCII byte format, and finally in ASCII word format. NOTE: When all the bytes or words in the specified range cannot be displayed on the line, a new line is started. BUG-8 supplies the starting address of the new line.

```
B: 43-70 100 112 107 114 100 303 053 040 365 257 303 143 002 303 056
000062 040 076 320 303 235 001 303

B: F43-70 112100 114107 303100 040053 257365 143303 303002 040056
000063 320076 235303 303001

B: D43-70 064 074 071 076 064 195 043 032 245 175 195 099 002 195
000061 046 032 062 208 195 157 001 195
B: FD43-70 19008 19527 49984 08235 45045 25539 49922 08238 53310 40387
000067 49921

B: A43-70 @ J G L @    +          C       .    >
B: FA43-70 J@ LG  @  +    C        .   >
```

Altering Memory — Decimal or Octal

To alter memory in decimal or octal formats, type an = after the format control and range fields. BUG-8 will then type the value of the first byte, or double byte if an F was used in the format control and follow this with a /. You can then type a new value if you want to change the contents of this location. If the contents of the location are not to be changed, or if sufficient new digits have been entered to complete the change, type a space or a carriage return.

If you type a space, BUG-8 offers the next byte (if there is one in the range) for alteration. If you type a carriage return, BUG-8 returns to the command mode.

In the following example, memory locations 60000 through 60031 are loaded with the octal values of the ASCII characters A through Z. NOTE: On the first three lines, the initial address is followed by the = sign, the current octal value in that memory location, and then a /. The octal value for the letter is entered following the slash. On the successive lines, a range of successive locations are opened and then changed to the sequentially ascending ASCII characters.

After the letters have been entered, the 26 memory locations are examined in byte format as ASCII characters. The 26 locations are then examined in word format as ASCII characters. Note that the second byte is treated as the most significant byte. Finally, the 26 locations are opened in byte octal format, using the # as the first address of the range.

```
B: 60000 = 000/101          (load A)
B: 60001 = 353/102          (load B)
B: 60002 = 341/103          (load C)
B: 60003/23=311/104 357/105 365/106 257/107 062/110 237/111 061/112
060012 303/113 274/114 061/115 315/116 230/117 062/120 012/121 376/122
042/123 302/124 135/125 057/126 120/127 131/130 023/131 046/132

B: A60000/26 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B: FA#/26 BA DC FE HG JI LK NM PO RQ TS VU XW ZY
B: #-60031 101 102 103 104 105 106 107 110 111 112 113 114 115 116
060016 117 120 121 122 123 124 125 126 127 130 131 132
B;
```

The BACK SPACE and RUBOUT keys are not effective when you are entering memory locations. Values placed in memory are taken as modulus 256 numbers (if they are entered in byte format) or as modulus 65,535 numbers (if they are entered in full word format). Thus, if you make a mistake, simply type the correct value with enough leading zeros to cause the bad digit to be eliminated. For example, if byte 70,000 is to be set to 123 and the mis-type 125 occurs, it may be correctly entered as:

```
B: 70000=111/125123
B: 70000 123
```

NOTE: Only the three least significant digits are accepted for this byte location.

Altering Memory — ASCII Format

To alter memory in ASCII format, type an = after the format control (A for ASCII) and range fields. The processing is similar to decimal or octal format memory alterations. The contents of the opened locations should then be followed by a /. You can then enter the replacement character (or two characters if the word format is used). However, the space and the carriage return are considered to be ASCII character values. To exit the command prematurely, use the ESCAPE or CONTROL-C key to avoid altering a location.

```
B: A70000=/A
B: A70001=>/B
B: A70002=/C
B: A70003-70031=2/D /E /F /G /H /I /J /K" /L /M /N /OW/P#/Q /RZ/S /T
070024 /U /V8/W /X /Y//Z
B: 67374-67377 076 000 303 122
B: A-70031 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B:
```

REGISTER COMMANDS

BUG-8 permits you to display the contents of all registers using octal, decimal, or ASCII, or to display the contents of individual registers using octal, decimal, or ASCII. In addition to displaying the contents of these registers, you can alter the various registers in any of the three modes. NOTE: If the F command is used in the format field, a register command is rejected, as register size is predetermined.

Displaying All Registers

To display the contents of all registers, enter a command of the form.

```
<FORMAT> <CNTRL-R>
```

BUG-8 displays the register contents in a specified format. NOTE: An M register is displayed in the all register command and can be specified in other commands. This register is the concatenation of the H and L registers. For example:

```
B: <CNTRL-R>
A=027 B=000 C=000 D=004 E=000 H=041 L=031 F=106 P=043324 M=041031
S=137377
B:

B: D<CNTRL-R>
A=023 B=000 C=000 D=004 E=000 H=033 L=025 F=070 P=09172 M=08473 S=24575
B:

B: A<CNTRL-R>
A= B= C= D= E= H=! L= F=F P=# M=! S=←
```

A Control R (CNTRL-R) should be typed after each command. However, no character is actually displayed. Also note that the ASCII display is not particularly meaningful unless printing ASCII characters are contained in the desired registers.

Displaying Individual Registers

To display the contents of any single register, use a command in the following format:

```
< FORMAT > REG < REG-NAME > < blank >
```

For example, to display the contents of register A, type:

```
B: REGA =101
B: DREGA =065
B: AREGA =A
B:
```

In the above example, the first line calls for the contents of register A to be displayed in octal format. In the second line, the contents of register A are displayed in the decimal format, and in the third line, the contents of register A are displayed in ASCII format. In the following example, the contents of the 16-bit register pair H and L, known as the M or memory register, are displayed in octal format.

```
B: REGM =041031
B: REGH =041
B: REGL =031
B:
```

Altering Register Contents

To alter the contents of a register, use a command in the following format:

```
< FORMAT > REG < REG-NAME > =
```

BUG-8 will then display the previous contents of the register (in the specified format octal, decimal or ASCII), followed by a /. It then accepts a new value if one is typed in. When you are using octal or decimal format, use a carriage return to close the entry or to skip the change. When you are using the ASCII format, type a single ASCII character to close the register. However, as the carriage return is a valid ASCII character, you must use ESCAPE or CONTROL-C to skip the change. The following examples demonstrate the altering of register contents.

```
B: REGA=102/103          (Change contents of A from 102₈ (ASCII B) to 103₈ (ASCII C).
B: DREGA=067/066         (Change contents of A from 67₁₀ (ASCII C) to 66₁₀ (ASCII B).
B: AREGA=B/C             (Change contents of A from ASCII B to ASCII C.)

B: REGA=103/ ↵           (A carriage return skips the change.)
B: AREGA=C/ < CNTRL-C >  (A CONTROL-C skips the change.)
B: AREGA=C               (The location is unaltered.)
```

NOTE: The last three are examples of skipping the change (leaving the location unaltered).

EXECUTION CONTROL

One of the primary functions of BUG-8 is execution control. It lets you step through the program, one or more instructions at a time, while examining register and memory contents. In addition, complete breakpointing is available, permitting you to execute a number of instructions and then return to BUG-8 control to examine register and memory contents. Execution control is divided into the areas of single stepping, breakpointing, and the GO command.

Single Stepping

The form of the single step command is:

```
STEP ADDR/CNT
```

where ADDR is an offset octal address (or a null) and "CNT" is a decimal step count, ≤ 255. If an address is not specified, BUG-8 starts stepping at the current PC-register address. When the instructions are completed, BUG-8 types the PC-register value and returns to the command mode. If an address is specified, BUG-8 starts stepping at the specified address and, when the instructions are completed, displays the terminating address value before returning to the command mode.

The following program increments the contents of memory location 055 100 each time the BC register pair is incremented from 000 000 to 027 000. This program is used to demonstrate a number of the execution control features of BUG-8.

ADDRESS	LABEL	INSTRUCTION	OCTAL	COMMENT
055 000	L1	LXI H	041	Point HL to 055 100.
055 001			100	
055 002			055	
055 003		MVI M	066	Load memory with 000.
055 004			000	
055 005	L2	INX B	003	Increment BC pair.
055 006		MOVA B	170	Load A with B.
055 007		CPI 027Q	376	Compare. Is B at 027 yet?
055 010			027	
055 011		JNZ L2	302	Jump back. Not 027 yet.
055 012			005	
055 013			055	
055 014		INR M	064	Passed 027. Increment memory

```
055 015          MOVA  M      176   Load A with memory.
055 016          CPI   377Q   376   Compare. Is memory at 377 yet?
055 017                       377
055 020          MVI   B      006   Reset B to 000.
055 021                       000
055 022          JNZ   L1     302   Jump back. Not at 377 yet.
055 023                       005
055 024                       055
055 025          RST2         327   Memory is 377. Back to BUG-8.
```

NOTE: The RST2 instruction is used to return this program to BUG-8. When BUG-8 encounters an RST2 instruction, it checks the breakpoint table. If there is nothing set, it returns to BUG-8.

For example, to load the above program using BUG-8,

```
B: 55000-55025=164/041 055/100 376/055 110/066 312/000 252/003 055/170
055007 376/376 103/027 312/302 334/005 055/055 376/064 102/176 312/376
020/377 056/006 376/000 114/302 312/005 101/055 056/327
B:
```

Set the front panel of the H8 to display the BC pair. Zero the BC pair and step through the first 6 program steps using BUG-8 as in the following example. The BC pair will momentarily display 000 001 as the steps are executed.

```
B: REGB=053/000
B: REGC=132/000
B: STEP 55000/6
-P=055005-
```

BUG-8 returns the value of the PC once the first six steps are executed.

Breakpointing

BUG-8 contains several commands to set, display, and clear breakpoints in your program. Breakpointing permits you to execute portions of a program once (or a number of times if the portion of a program is in a loop). Breakpointing is especially useful in de-bugging programs which have a tendency to destroy themselves or obliterate the cause of the problem in the process of complete execution.

SETTING BREAKPOINTS

The breakpoint command is used to set a breakpoint. The form of the breakpoint command is:

```
BKPT ADDR1/CNT1, . . . . . ,ADDRn/CNTn
```

BUG-8 allows up to 6 breakpoints. They are entered in the breakpoint table within BUG-8, replacing any previously defined breakpoints at those addresses. No more than six breakpoints may be entered in the breakpoint table.

The CNT field may be used to specify the breakpoint repeat count. It is a decimal number in the range of 1 to 255. Using the breakpoint count means the breakpoint does not cause control to return to the monitor mode until the breakpoint is executed CNT-1 times. Thus, you may execute a loop a number of times prior to returning to the command mode via a breakpoint instruction. As noted, the Breakpoint Instruction executes CNT-1 times, without recognizing the breakpoint. The last time through the loop, the instruction at the breakpoint address is not executed. The breakpoint returns control to BUG-8. NOTE: If CNT is not specified, the value 1 is assumed.

For example, the program of the previous example is run with the H8 front panel displaying memory location 055 100.

```
B: 55100=001/000
B: BKPT 55015/6
B: GO 55000
-P=055015-
B:
```

NOTE: 055 100 is incremented by 6.

```
B: 55100=006/000
B: BKPT 55015/6,55014/10,55022/30
B: GO 55000
-P=055015-
B: GO
-P=055014-
B: GO
-P=055022-
B:
```

DISPLAYING BREAKPOINTS

To display the current status of the breakpoint table, use the breakpoint display command. BUG-8 can display the contents of the breakpoint table. The form of the breakpoint command is:

```
BKPT DSPLY
```

BUG-8 provides a listing of the current breakpoints in the form:

```
BKPT DSPLY ADDR1/CNT1 ADDR2/CNT2 . . . . . ADDRn/CNTn
```

where ADDR is the address of the breakpoint instruction, and CNT are the loop counts remaining on the designated breakpoints. NOTE: When the breakpoint count is exhausted, it causes control to return to BUG-8. The breakpoint is removed from the breakpoint table, and nonexhausted breakpoints are saved. For example:

```
B: 55100=036/000
B: BKPT 55015/6,55014/10,55022/30
B: BKPT DSPLY 055015/006 055014/010 055022 030
B: GO 55000
-P=055015
B: BKPT DSPLY 055014/004 055022/025
B: GO
-P=055014-
B: BKPT DSPLY 055022/021
B: GO
-P=055022-
B: BKPT DSPLY
B:
```

CLEARING INDIVIDUAL BREAKPOINTS

To clear an individual breakpoint, use the command

```
CLEAR ADDR1, . . . ,ADDRn
```

where ADDR1, . . . ,ADDRn specifies the address of the breakpoint to be removed from the table.

CLEARING ALL BREAKPOINTS

To complete clear all breakpoints from the breakpoint table, use the breakpoint clear command

```
CLEAR ALL
```

For example:

```
-P=040261-
B: BKPT 55012/10,55014/15,55020/20,55022/200
B: BKPT DSPLY 055012/010 055014/015 055020/020 055022/200
B: CLEAR 55014,55022
B: BKPT DSPLY 055012/010 055020/020
B: CLEAR ALL
B: BKPT DSPLY
B:
```

EXEC

The EXEC (execute) command is a combination of the GO and BKPT commands. The form of the EXEC command is:

```
EXEC SADDR-ADDR1, . . . . ,ADDRn
```

where "SADDR" is the starting address for execution. If the starting address is omitted, execution starts at the current program counter register value. ADDR1 through ADDRn are the addresses of breakpoints to be set before execution. Thus, for example, to start at byte 055 000 and to execute to byte 055 015, the command is typed as:

```
B: EXEC 55000-55015
-P=055015-
B:
```

GO

Use the GO command to transfer control to your program. You can set breakpoints before via the BKPT command. The form of the GO command is:

```
GO [SADDR]
```

If you specify "SADDR," execution begins at this specified address. If you do not specify "SADDR," execution begins at the current value of the program counter register. For example, simple execution of the previous program is accomplished by

```
B:GO
-P=0550250
B:
```

TAPE HANDLING

BUG-8 offers three commands for tape handling. With these commands you can dump to a tape, load from a tape, or verify a tape.

DUMP

The DUMP command allows BUG-8 to write a file that is an image of the desired memory range. The operation is very much like the H8 front panel dump. The form of the DUMP command is:

```
DUMP ADDR1-ADDR2
```

ADDR1 and ADDR2 are specified in offset octal. Ready the tape transport before typing the carriage return after ADDR2. The contents of the Program Counter are also saved. Set the PC to the program entry point before you type a return.

LOAD

The LOAD command allows BUG-8 to read a file containing a memory image. The form of this command is:

```
LOAD
```

Once the load is complete, control is returned to BUG-8 for execution or other manipulations. When control is returned to BUG-8, the Program Counter is set to the entry point on the image tape. During a load a tape error may occur. These are explained in the "Tape Errors" section below.

VERIFY

The VERIFY command allows BUG-8 to verify a memory image file. The form of this command is:

```
VERIFY
```

BUG-8 reads the file without loading it and responds with

```
OK
```

if the CRC on the tape matches its computed CRC. If the CRCs do not match, a tape error is generated. (See "Tape Errors.")

Tape Errors

During a LOAD or VERIFY, one of two error message may be generated. A checksum error is generated if the computed CRC for the record in question does not match the CRC at the start of the record. The form of the checksum error message is

```
CHKSUM ERR-IGNORE?
```

A 'Y' in response to the question "ignore?" aborts the error message and the next consecutive record is accepted. NOTE: Ignoring the checksum error is not recommended unless there is no other way to recover the data. If a checksum error is flagged, the chances are very good the data in the designated record is faulty.

A sequence error (SEQ ERR) is generated if the file records are not in the proper sequence. For example, if two record numbers are not consecutive, an error is generated. The form of the sequence error is

```
SEQ ERR
```

Typing a space after the SEQ ERR message generates a tape error message and the entire file must be reread.

TERMINAL CONTROL CHARACTERS

BUG-8 recognizes a number of control characters. Some of these are valid when you are entering commands, and others are only valid when BUG-8 is printing information on the console terminal. You can make a control character by simultaneously depressing the appropriate key plus the CONTROL (CNTRL) key. This is similar to depressing the SHIFT key along with a letter.

Input Control Characters

The following characters are only valid when you are entering commands.

BACKSPACE (CONTROL-H)

The BACKSPACE character causes the **last character** on the line to be discarded. A backspace code is sent to the CRT terminal so it can perform a cursor backspace, but the backspace is ignored by teleprinters, and some other printing terminals. If you attempt to BACKSPACE into column zero, a bell code is echoed noting this illegal operation. NOTE: Backspace can be changed when BUG-8 is configured. See "Appendix A," or Page 3 in "Appendix B" in the "Introduction" to this Software Reference Manual.

RUBOUT

The RUBOUT key causes the **current line** to be discarded. A carriage return/line feed is sent to the terminal. Once the RUBOUT is typed, the entire line may be re-entered. NOTE: Rubout can be changed when BUG-8 is configured. See "Appendix A," or Page 2-20 in the "Introduction" to this Software Reference Manual.

Output Control Characters

The following characters are only valid during output.

OUTPUT SUSPENSION AND RESTORATION, CNTRL-S AND CNTRL-Q

Typing Control S during an output suspends the output to the console terminal and suspends program execution. This command is particularly useful when you are using a video terminal, since you can use the Control S or suspend feature each time a screen is nearly filled and information at the top will be lost due to scrolling.

Typing Control Q permits BUG-8 to continue execution and output information to the terminal. The Control Q cancels the Control S function.

THE DISCARD FLAG, CNTRL-O AND CNTRL-P

Typing Control O toggles the DISCARD FLAG. This stops the output on the terminal but program execution does not halt until the program terminates. Typing a Control P (or typing Control O again) clears the discard flag. Control O is often used to discard the remainder of long outputs. Control P clears the discard flag.

COMMAND COMPLETION

When BUG-8 is in the command mode, each terminal keystroke is considered for validity. If the character belongs to no possible command, it is refused and the bell code is echoed to the terminal. If the command syntax allows only one next character, BUG-8 supplies and prints this character for the user.

In addition to simple syntax checking, BUG-8 also processes command range expressions as they are being entered. Should the user enter a range expression referring to nonexistant memory, BUG-8 refuses further entry and echoes a bell code. Thus, should apparently valid characters be rejected by BUG-8, it indicates that the command range expression may be invalid.

APPENDIX A

Loading Procedures

Loading From the Software Distribution Tape

1. Load the tape in the reader.
2. Ready the tape transport.
3. Press LOAD on the H8 front panel.
4. A single beep indicates a successful load.
5. Press GO on the H8 front panel.
6. The console terminal will respond with:

```
HEATH/WINTEK TERMINAL DEBUGGER.
BUG-8 ISSUE # 02.01.00.
COPYRIGHT WINTEK CORP.,01/77
```

7. Configure BUG-8 as desired, answering the following questions. Prompt each question by typing its first character on the console terminal keyboard.

```
• AUTO NEW-LINE (Y/N)?
• BKSP = 00008/
• CONSOLE LENGTH = 00080/
• HIGH MEMORY = 16383/
• LOWER CASE (Y/N)?
• PAD = 4/
• RUBOUT = 00127/
• SAVE?
•
```

8. Before executing SAVE, have the transport ready at the DUMP port.
9. To use BUG-8 directly from the distribution tape, type the return key at any time rather than a question prompt key. BUG-8 responds:

```
HEATH BUG-8 # 02.01.00.
B:
```

BUG-8 is ready to use.

REQUIRED PATCHES FOR

HEATH H8 BUG - 8

Upgrading 02.01.00.
To 02.01.01.

THESE PATCHES MUST BE INSTALLED IN THE

PRODUCT EVERY TIME A LOAD IS MADE FROM

THE CONFIGURATION TAPE

NOTES: None.

USE: These patches cause BUG-8 to remove any outstanding breakpoints from the program when it is entered at the warm-start address (040 103).
The breakpoint table is cleared WITHOUT removing any patches from the program when BUG-8 is entered at the cold start address (040 100).

```
043377  061
047040  126
047042  012
050011  031 054
050017  031 054
054023  315 100 045 303 336 043 346 177
054033  303 225 052
054307  023 054
055222  337
057100  061
```

Loading From a Configured Tape

1. Load the tape in the tape transport.
2. Ready the tape transport.
3. Press LOAD on the H8 front panel.
4. A single beep indicates a successful load.
5. Press GO on the H8 front panel.
6. The console terminal should respond with:

```
HEATH BUG-8 # 02.01.00.
B:
```

BUG-8 is ready to use in the configured form.

APPENDIX B

BUG-8 Command Summary

Memory Commands

Memory display form:

```
FORMAT CONTROL    range  =
```

Memory Alter form:

```
FORMAT CONTROL    range  blank
```

FORMAT	RESULT
< null > < null >	byte octal
F < null >	word octal
< null > A	byte ASCII
FA	word ASCII
< null > D	byte decimal
FD	word decimal

Range

The range field consists of a beginning address and an ending address. You can specify addresses by using the appropriate offset octal integers; or you can use the NULL, #, and cnt (count) as indicated below.

RANGE FORM	DESCRIPTION
ADDR < null >	Range specifies the single memory location ADDR.

ADDR1-ADDR2	Range specifies the memory locations ADDR1 through ADDR2 inclusive.
ADDR/cnt	Range specifies cnt memory locations starting at location ADDR. NOTE: cnt is a decimal integer ≤ 255.
#-ADDR	Range specifies the memory locations starting at the beginning of the previous range and ending at ADDR.
#/cnt	Range specifies cnt memory locations starting at the beginning of the previous range. NOTE: cnt is a decimal integer ≤ 255.
< null >/cnt	Range specifies cnt memory locations starting at the address following the last address of the previous range. NOTE: cnt is a decimal integer ≤ 255.
< null > -ADDR	Range specifies memory locations starting at the address following the last address of the previous range and extending to memory location ADDR.

Register Commands

All registers:

```
FORMAT CNTRL-R
```

Single register:

```
REG REG-NAME blank
```

Altering register:

```
REG REG-NAME=
```

Execution Control

Single stepping:

```
STEP ADDR/CNT
```

Breakpointing:

```
BKPT ADDR1/cnt 1, . . . . . ,ADDRn/cntn     (n-6)
```

Breakpoint display:

```
BKPT DSPLY
```

Clearing breakpoints:

```
CLEAR ADDR1, . . . . . ,ADDRn
CLEAR ALL
```

GO:

```
GO (ADDR)    (Starts at PC value if ADDR is not specified)
```

Execute:

```
EXEC SADDR-ADDR1, . . . . . ,ADDRn    (combines GO AND BKPT)
```

Tape Handling

```
DUMP ADDR1-ADDR2 (CR)
LOAD (CR)
VERIFY (CR)
```

INDEX