



# THE BUG

A Debugging Program

By Michael H. Rivner  
(C) Copyright 1981 M.H. Rivner

Published by

SENSIBLE SOFTWARE, Inc.  
6619 Perham Drive  
West Bloomfield, Michigan 48033  
313-399-8877



**SENSIBLE SOFTWARE**

APPLE is a registered trademark of APPLE Computer Company

The author wishes to acknowledge the help of his wife Roberta Rivner, who typed this documentation and put up with me during the many hours and nights of work that I ignored her in order to write this program.

## TABLE OF CONTENTS

|                                                        |       |
|--------------------------------------------------------|-------|
| GETTING STARTED .....                                  | 1-1   |
| BOOTING THE BUG .....                                  | 1-2   |
| DISPLAY COMMANDS .....                                 | 2-1   |
| The "D" Command .....                                  | 2-1   |
| The "P" Command .....                                  | 2-1   |
| The "Z" Command .....                                  | 2-2   |
| The "L" Command .....                                  | 2-3   |
| The "I" Command .....                                  | 2-3   |
| The "N" Command .....                                  | 2-3   |
| The "Q" Command .....                                  | 2-3   |
| The "C" Command .....                                  | 2-4   |
| The "W" Command .....                                  | 2-4   |
| The "CTRL-T" Command .....                             | 2-4   |
| The "CTRL-G" Command .....                             | 2-4   |
| The "CTRL-F" Command .....                             | 2-4   |
| The "CTRL-S" Command .....                             | 2-5   |
| The "CTRL-N" Command .....                             | 2-5   |
| The "&" Command .....                                  | 3-1   |
| MODIFICATION COMMANDS .....                            | 3-1   |
| The "R" Command .....                                  | 3-1   |
| The "e" Command .....                                  | 3-2   |
| The "CTRL-W" Command .....                             | 3-3   |
| The "CTRL-P" Command .....                             | 3-3   |
| The "I" Command .....                                  | 4-1   |
| BREAKPOINT COMMANDS .....                              | 4-1   |
| The "B" Command .....                                  | 4-2   |
| The "T" Command .....                                  | 5-1   |
| EXECUTION COMMANDS .....                               | 5-1   |
| The "G" Command .....                                  | 5-2   |
| The "F" Command .....                                  | 5-3   |
| The "S" Command .....                                  | 5-3   |
| The "J" Command .....                                  | 6-1   |
| COMMANDS THAT EXIT OR RESTART THE BUG .....            | 6-1   |
| Reset .....                                            | 6-1   |
| The "E" Command .....                                  | 6-1   |
| The "M" Command .....                                  | 6-2   |
| The "CTRL-Q" Command .....                             | 6-2   |
| The "&" Command .....                                  | 6-2   |
| The "CTRL-Y" Command .....                             | 7-1   |
| ENTERING MULTIPLE COMMANDS .....                       | 8-1   |
| SPECIAL CONSIDERATIONS OF RUNNING THE BUG .....        | 9-1   |
| MEMORY LOCATIONS USED BY THE BUG DURING THE BOOT ..... | 10-1  |
| COMMAND EXAMPLES .....                                 | 10-1  |
| GENERAL COMMANDS .....                                 | 10-6  |
| USE OF THE "P" AND "Z" COMMANDS .....                  | 10-7  |
| USE OF THE BREAKPOINT COMMANDS .....                   | 10-9  |
| USE OF THE "F" COMMAND .....                           | 10-11 |
| USE OF THE "J" COMMAND .....                           | 10-16 |
| USE OF THE "B" COMMAND .....                           |       |
| APPENDIX I - COMMAND SUMMARY                           |       |
| APPENDIX II - THE BUG LOCATIONS                        |       |



## NOTICE

### OWNER'S MANUAL AND PROGRAM

(C) COPYRIGHT 1981 BY

M. H. RIVNER

This manual and the computer programs on the accompanying floppy disk that are described by this manual are copyrighted and contain proprietary information belonging to the author.

This manual may not be copied, photocopied, reproduced, translated or reduced to machine readable form, in whole or in part, without the prior written consent of the author.

The accompanying floppy disk may not be duplicated, in whole or in part, for any purpose. No copies of the floppy disks or this manual or the listings of the programs on the floppy disk may be sold or given to any person or other entity.

### LIMITATIONS OF WARRANTY AND LIABILITY

The author, or any dealer distributing this product, makes no warranty, expressed or implied, with respect to this manual, the related floppy disks and any other related items, their quality, performance, merchantability, or fitness for any particular use. It is solely the purchaser's responsibility to determine their suitability for any particular purpose.

The distributors and author of this program will in no event be held liable for direct, indirect or incidental damages resulting from any defect or omission in this manual, the floppy disks, or other related items and processes, including but not limited to any interruption of service, loss of business or anticipatory profit, or other consequential damages.

This statement of limited liability is in lieu of all other warranties or guarantees, express or implied, including warranties or merchantability and fitness for a particular purpose. The author neither assumes nor authorizes any other person to assume for him any other warranty or liability in connection with the sale of this product.

The author reserves the right to make changes in both the manual and accompanying programs at any time, without notice and with no responsibility to provide these changes to purchasers of earlier versions of this product.

NOTE: APPLE, APPLE ][, APPLE ][ PLUS, and APPLESOFT are registered trademarks of Apple Computer Inc.

## THE BUG---A DEBUGGING PROGRAM

by M.H. Rivner

(C) Copyright 1981 M.H. Rivner

THE BUG is a programming aid designed to be used as a tool by both the experienced and novice assembly language programmer. THE BUG provides the user with the capacity to control the execution of his program. Each instruction is displayed as it is executed. Program execution may be halted at any time with the use of the escape key. In addition, memory or register breakpoints may be set. These cause execution to stop when either the program counter is equal to one of eight user specified addresses or the 6502 registers equal values previously specified by the user. While program execution is halted, the user may make use of a large number of register or memory modification and display commands provided to the user by THE BUG. Breakpoints may at this time be redefined. In addition to the large variety of commands contained in THE BUG, all the APPLE II monitor commands are available. Program execution may then be restarted at either the next instruction or any instruction specified by the user.

THE BUG also provides the user with the ability to run parts of his program at full speed. This allows the user to concentrate only on those program sections that he desires to examine. Time is not wasted examining other parts of the program. Yet all the conditions, memory locations, and registers are correctly set because previous sections of code have been run. THE BUG provides two methods for doing this; one is a direct jump to a memory location and the other is full speed execution when the stack pointer is pushed below a certain value by a JSR instruction. This second method allows subroutines to execute at full speed returning the user to the debugging mode on return from the subroutine. The user's time is not spent debugging subroutines that he knows work.

The programmer will find that THE BUG will increase his efficiency by decreasing the amount of time spent debugging assembly language programs. The beginning programmer will find THE BUG to be a useful tool for learning assembly language programming.



## GETTING STARTED

BOOTING THE BUG

THE BUG is distributed on a standard 5 1/4" magnetic disk designed to run on the APPLE ][ disk drive. In order to boot THE BUG, DOS 3.3 must already be loaded in memory below \$C000 and HIMEM must be set correctly. To boot THE BUG first bring up DOS in the standard manner. The disk containing THE BUG is then placed in an APPLE ][ disk drive connected to any slot. THE BUG must be booted off Drive #1 of that slot. The DOS command "PR#(slot#)" is typed. The sign on message will then appear. To allow the user maximum flexibility, THE BUG is relocatable to any page boundary (at or above \$1000). The user must specify which page boundary THE BUG is to be loaded at by responding to the sign on question, "ENTER PAGE BOUNDARY IN HEX?" with either the 2 most significant hexadecimal digits of the address THE BUG is to reside or just a carriage return. The user must have reserved \$D00 memory addresses above the address entered as THE BUG requires slightly more than 3K of memory. The user must be careful that the chosen memory address does not interfere with DOS, as THE BUG does not check the validity of the address entered. If the user desires that THE BUG be located below HIMEM then all he need enter is a carriage return. If this option is used, THE BUG will load below HIMEM and then reset HIMEM to protect itself.

Two examples will now be used to illustrate the loading of THE BUG. In these examples user input will be bold faced, characters displayed in inverse video will be underlined.

Example #1--In this case an APPLE ][ PLUS computer is used. DOS 3.3 has already been loaded into memory. The disk containing THE BUG has been placed in SLOT #6, Drive #1. In this case the user specifies that THE BUG should be loaded at HIMEM. After THE BUG is loaded HIMEM is reset to protect THE BUG. The C/R symbol specifies the carriage return key. All user input lines will be terminated with a carriage return.

**] PR#6**

(the screen clears)

THE BUG--VER-4B.2--8/09/81

(C) COPYRIGHT 1981 M.H. RIVNER

ENTER PAGE BOUNDARY IN HEX? C/R

Example #2--In this case an APPLE ][ is used. The user specifies that THE BUG should be loaded beginning at \$7000. The memory from \$7000-\$7D00 will be utilized by THE BUG and will not be available to the user. HIMEM will not be changed by THE BUG. In this example DOS 3.3 has already been booted and the disk containing THE BUG has been placed in a disk drive attached to Slot #4, Drive #1.

**> PR#4**

(the screen clears)

THE BUG--VER-4B.2--8/09/81

(C) COPYRIGHT 1981 M.H. RIVNER

ENTER PAGE BOUNDARY IN HEX? 70

After THE BUG is loaded into memory, the loader program will check whether game paddles are connected to the system. If game paddles are present the user will be asked whether he desires to use them. The question will appear as follows:

## GAME PADDLE?

A "YES" response or just a carriage return will allow the user to use the game paddle functions in THE BUG to be described below. A "NO" response will cause THE BUG to ignore the game paddles. This allows the user the ability to debug programs that make use of the game paddles without THE BUG interfering with their function. If no game paddles are connected to the system this question will not be asked and THE BUG will act as if a no response was given. If the game paddles are used the pushbutton on game paddle zero will have the same effect as the escape key, i.e. it will stop user program execution when the "G" command is active. The "Q" command allows the user the ability to use the potentiometer on game paddle zero to control the speed of program execution. If the game paddle option is not selected at this point, hitting the pushbutton on game paddle zero will have no effect on program execution. The "Q" command in this case will cause program execution and instruction display to be slow enough so the user will be able to read the display but the user will not be able to control this speed.

Next the screen will be divided into two sections by a dotted line. The upper part of the screen will be reserved for user program output while THE BUG will use the lower portion of the screen for its display. An inverse asterisk will be displayed indicating that THE BUG is waiting for a command.

## GETTING STARTED

There are five basic groups of commands. These are:

- 1) Display Commands which either display information or change the way THE BUG'S display functions work.
- 2) Modification Commands which allow the user to change the contents of registers or memory.
- 3) Breakpoint Commands which allow the user to set conditions that cause the user program to halt.
- 4) Execution Commands which initiate user program execution.
- 5) Commands that Restart or Exit THE BUG that allow the user to exit THE BUG and execute either monitor, BASIC or DOS commands. Other commands in this group allow the user to return to THE BUG or exit permanently.

Each command consists of a one letter mnemonic followed by operands that may either be required or optional. All numbers entered will be interpreted as hexadecimal numbers. THE BUG's command interpreter ignores spaces. In many of the examples used in this manual spaces are used to improve their clarity. Appendix I contains a summary of THE BUG's commands. A detailed description of each command follows.

## DISPLAY COMMANDS

The display commands either display information to the user directly or change the way THE BUG displays information during program execution.

- 1) The "D" Command--Format D (no parms)

The "D" or "display registers" command will display the contents of the 6502 registers and the program counter.

### Example

\* D

PC=8942 A=42 X=00 Y=03 P=31 SP=FF

- 2) The "P" Command--Format P (optional address)

The "P" or "print memory" command, followed by an optional hexadecimal address, prints the contents of memory starting from the current address pointer. This may be set by the "P" command to be the address entered. If no address is entered the current address pointer set by the previous command is used. The "P" command will print up to eight locations in memory for each command entry.

### Example

\* PC35

0C35 - 00 45 4E

\* P

0C38 - 54 45 52 20 59 4F 55 52

Note in the first example only three memory locations are printed. The P command prints from the current address to the next address that is divisible by 8. In the second example no address was specified, in that case the memory dump started from the current address pointer which was set by the previous command.

- 3) The "Z" Command--Format Z (optional address)

The "Z" or "extended memory dump command" will display the contents of up to 120 contiguous memory locations at one time. The optional address specifies where the memory dump is to start. If no address is specified, then the dump begins at the current address pointer that was set by the previous command. To the right of the hexadecimal values contained at these locations will be their ASCII conversion.



# DISPLAY COMMANDS

ASCII characters with the most significant bit set will be displayed normally. True ASCII characters (MSB cleared) will be displayed in inverse video, with the exception of the space.

## Example

### \* ZB10

```

OB10- D4 C5 D8 D4 A0 D7 C9 D4 TEXT WIT
OB18- C8 A0 CD D3 C2 A0 D3 C5 H MSB SE
OB20- D4 0A C9 D3 A0 C4 C9 D3 T IS DIS
OB28- D0 CC C1 D9 C5 C4 A0 C9 PLAYED I
OB30- CE A0 CE CF D2 CD C1 CC N NORMAL
OB38- A0 D6 C9 C4 C5 CF AE A0 VIDEO.
OB40- 54 45 58 54 20 57 49 54 TEXT WIT
OB48- 48 20 4D 53 42 20 43 4C H MSB CL
OB50- 45 41 52 45 44 20 49 53 EARED IS
OB58- 20 44 49 53 50 4C 41 59 DISPLAY
OB60- 45 44 20 49 4E 20 49 4E ED IN IN
OB68- 56 45 52 53 45 20 56 49 VERSE VI
OB70- 44 45 4F 20 42 55 54 20 DEO BUT
OB78- 53 50 41 43 45 53 20 41 SPACES A
OB80- 52 45 20 4E 4F 54 2E 20 RE NOT.D

```

If the user wished to continue the memory dump from this point i.e. \$B88 only a "Z" need be entered as the first "Z" command set the next address pointer.

## 4) The "L" Command--Format L (optional address)

The "L" or "listing" command, disassembles ten instructions beginning at the current address pointer. The address from where the disassembly begins may be specified in this command. The "G" command will set the next address pointer so that an "L" command will start the disassembly at the next executable instruction.

## Example

### \* LB00

```

OB00- A0 20 LDY #20
OB02- A2 14 LDX #14
OB04- 88 DEY
OB05- D0 FD BNE $0B04
OB07- CA DEX
OB08- D0 FA BNE $0B04
OB0A- 60 RTS
OB0B- 00 BRK
OB0C- 00 BRK
OB0D- 00 BRK

```

# DISPLAY COMMANDS

Large portions of code may be disassembled by entering more than one "L" command on a line. For each "L" entered ten instructions will be disassembled.

## 5) The "I" Command--Format I

The "I" or "instruction display" command places THE BUG in the standard display mode. When execution is initiated with the "G" command all instructions will be disassembled before the instruction is performed. This display will be very fast, so fast that the user will not be able to read the display.

## 6) The "N" Command--Format N

The "N" or "No instruction display" command places THE BUG in the no display mode. When execution is initiated with the "G" command, no display will occur. This results in faster execution times. When execution is halted the 6502 registers and program counter will still be displayed.

## 7) The "Q" Command--Format Q

The "Q" or "Controlled instruction display" command places THE BUG in the slow speed display mode. When execution is initiated with the "G" command, the display will be at a slower speed. If the user had selected the game paddles at boot up time then the speed of the display and program execution would depend on the potentiometer on game paddle zero. By turning game paddle zero the user may slow down or speed up program execution. This allows sections of code that are of interest to be examined closely while other sections are sped over by turning the game paddle dial. If the game paddles have not been selected at boot up time, then a preset speed is used. This speed is slow enough that the user will be able to read the display. The user has no way to change the speed of the display in this mode but can speed up the display by entering another display mode with the "I" or "N" commands.

## 8) The "C" Command--Format C

The "C" or "printer on" command will Toggle (activate/deactivate) a printer connected to slot #1. If slot #1 is empty this command will have no effect. This command will have no effect on printers connected to other slots. When this command is active THE BUG's output will be directed to the printer allowing the user to obtain a hard copy. Since THE BUG has an output routine separate from that of the user program, this command will have no effect on user program output.



DISPLAY COMMANDS

9) The "W" Command--Format W

The "W" or "breakpoint display" command displays all active and register breakpoints set by the user. This command allows the user to easily keep track of all active breakpoints.

Example

\* W

BRKPT #1 B7B5

BRKPT #3 897F

REGS

A = 34

S = FF

Only those breakpoints that are set will be displayed.

10) The "CTRL-T" Command--Format ^T

The "^T" or "text mode" command places the screen in text mode. This is equivalent to the instruction "LDA #1". Note that after execution of each instruction in either the "G" or "S" modes the screen is automatically placed in text mode on the primary page.

11) The "CTRL-G" Command--Format ^G

The "^G" or "graphic mode" command places the video display in the graphics mode. This instruction allows the user to see the displays that his program plotted on the video display. This instruction is equivalent to "LDA #0". Note that after the execution of each instruction in either the "G" or "S" modes the video screen is reset to the text mode. This allows the user the ability to see the video display.

12) The "CTRL-F" Command--Format ^F

The "^F" or "primary page" command sets the APPLE's video display to display the primary text or video page. This instruction is equivalent to "LDA \$C054".

13) The "CTRL-S" Command--Format ^S

The "^S" or "secondary page" command sets the APPLE's video display to display the secondary text or video page. This instruction is equivalent to "LDA \$C055".

The "CTRL-N" Command--Format ^N

The "^N" or "normalize display" command sets the video display to text mode and displays the primary page. This is equivalent to "LDA \$C051, LDA \$C054".

The "E" Command--Format E

The "E" command when entered in THE BUG displays the values swapped to memory locations \$3C-\$3D. If a "P" command immediately follows the "E" the values swapped to locations \$1E-\$1F will be printed.

## MODIFICATION COMMANDS

These commands allow the user to change the contents of either memory locations or registers. These commands also give the user access to the language card if one is installed.

The "R" Command--Format R (reg): (val)

The "R" or "register modification" command allows the user to modify the 6502 registers. The "R" is followed by a letter designating which register is to be modified (A=accumulator, X=x register, Y=y register, S=stack pointer, PC=program counter). This is followed by a colon, then the decimal value to be placed in that register.

### Example

• D

PC=B000 A=00 X=00 Y=00 P=B1 SP=FF

• RA:5

(this command placed \$5 in the accumulator)

• RY:A

(this command placed \$A in the y register)

• D

PC=B000 A=05 X=00 Y=0A P=B1 SP=FFD

The "M" Command--Format @ (optional loc):(byte or word val list)

The "M" or "memory modify" command allows the user to change the contents of up to eight memory locations. Directly following the command mnemonic is the address to be modified in hexadecimal. If this is left out the next address location set by the previous command will be used. Following the address is a colon, this colon must be present even if the address is implied. Following this is a value list. This is a list of either byte or two byte word values which is terminated by either a carriage return or period. Each value on this list must be separated from the previous one by either a space or a comma. When two byte word values are entered, THE BUG automatically inverts the high order and low order bytes, so the value will be placed in memory in the proper 6502 format.

# MODIFICATION COMMANDS

# MODIFICATION COMMANDS

## Example

```
* @A00:8901,7E,47,76,4626
-
* PA00
0A00- 01 89 7E 47 76 26 46 00

* @A10:A0,445
-
* @:FF
-
* PA10
0A10- A0 45 04 FF 00 00 00 00D
```

Note that in the first example seven bytes are placed in memory beginning at \$A00. In this example the two byte values \$8901 and \$4626 are converted to the proper 6502 format, i.e. low byte first, high byte second. The "P" command then displays these changes. Next more bytes are entered beginning at \$A10. In the next line the "@" command is used with an implied address, in this case \$A13. The colon must be present even when the address is implied. The final "P" command displays these memory changes.

## 3) The "CTRL-W" Command--Format W

The "CTRL-W" or "language card write enable" command resets the language card so the user may modify memory locations on it. Once this command is issued memory addresses may be modified with the "@" command. This command is equivalent to "LDA \$C083, LDA \$C083". Please note that this command and the "CTRL-P" command do not modify the DOS pointers that point to the current version of BASIC. These commands may switch the user to a different version of BASIC because the language card may be turned on. This could produce unpredictable results if BASIC functions are executed before switching the language card to the proper memory space. The correct memory space can be addressed by addressing certain memory addresses as documented in the documentation accompanying the language card. Single memory addresses may be addressed with the "@" command. This command will have the same effect on similar extended memory cards manufactured by firms other than APPLE.

## 4) The "CTRL-P" Command--Format P

The "CTRL-P" or "language card write protect enable" command will prevent modification of the memory contained on the language card. This command is equivalent to "LDA \$C080".

## 5) The "I" Command--Format I

The "I" or "reset stack pointer" command will reset the stack pointer back to \$FF.



### BREAKPOINT COMMANDS

These commands allow the user to set conditions that will cause execution of the user program to halt.

- 1) The "B" Command--Format B (opt address): (opt BKPT #)

The "B" or "set memory breakpoint" command allows the user to set memory breakpoints in ROM or RAM. Program execution initiated with the "G" command will halt, when these memory addresses are executed. THE BUG allows a maximum of eight memory breakpoints. Following the "B" mnemonic the hexadecimal breakpoint address is entered. This is followed by an optional colon and breakpoint number. If breakpoints are entered without a breakpoint number they will be automatically assigned to the lowest free breakpoint number. When a ninth breakpoint is entered in this manner, THE BUG wraps around and overwrites the first breakpoint value. Breakpoints can be removed or cleared, increasing the execution speed, by entering "B" command alone without operands or followed by a colon and a breakpoint number. In the first case, all memory breakpoints are removed, leaving all other break conditions as they were. The second method just removes the individual breakpoint.

#### Example

\* B

\* B B0A

\* B EOF

\* B 700:6

\* W

BRKPT # 1 0B0A

BRKPT # 2 0EOF

BRKPT # 6 0700

\* B:2

\* W

BRKPT # 1 0B0A

BRKPT # 6 0700

## BREAKPOINT COMMANDS

In the first line all breakpoints are cleared with a single "B" command. The next commands set breakpoints at memory locations \$B0A, \$E0F and \$700. The first two breakpoints are entered as breakpoints one and two. The third breakpoint is breakpoint #6 because it was specified in the command. If no breakpoint number was entered it would have been breakpoint #3. The "W" command on the following line displays the breakpoints set. Next breakpoint #2 is removed by the command "B:2" and the breakpoints are again displayed.

2) The "T" Command--Format T (reg) (><=) (val) or TP, (S1,S0,V1,V0,B1,B0,D1,D0,I1,I0,Z1,Z0,C1,C0) (carriage ret or period)

The "T" or "register test" command allows the user to set or clear register conditions that will cause the program to stop execution. The command is followed by a letter designating the register to test (A=accumulator, X=x register, Y=y register, S=stack pointer, and P=status register). The status register is set differently than the other registers and will be discussed separately. The other registers are followed by one or more conditional operators H=<>J. Following the comparison symbols a hexadecimal byte is entered. Break conditions for each register must be entered individually using a separate "T" command for each register condition set. One or more registers may be set in this manner.

Status conditions are entered differently from those of the other registers. To set this register a "TP" is entered followed by a flag symbol and either a "1" for set or an "0" for clear. One or all status flags may be entered in a single "TP" command, each flag and condition is separated from the previous one by a comma. The entire list must be terminated with either a period or carriage return. Each time the "TP" command is entered the flags set by previous "TP" commands are cleared. Valid status register flag symbols are "Z" for zero, "V" for overflow, "C" for carry, "D" for decimal, "I" for interrupt, "B" for soft break, and "N" for sign flag. Status register flags not entered are not tested for.

Program execution initiated by the "G" command will halt when all the register conditions set with the "T" command are satisfied. The conditions set can be examined with the "W" command.

Register breakpoints may be cleared by entering the letter "T" followed by the register the user wants to clear. All register breakpoints may be cleared by entering the "-" command with no operands. Any time the "T" command is entered it updates the previous conditions set for that register with the new ones entered.

Example

```
* T
-
* TA=4F
-
* TS>FD
-
* TP,C1,Z0
-
* W

BRKPT # 1 0B0A
BRKPT # 6 0700
```

## REGS

```
A=4F
P C1 Z0
S>FD
```

```
* B
-
* TS
-
* TP,Z1
-
* TY>=3
```

## \* W

## REGS

```
A=4F
Y=>03
P Z1
```

## \* T

## \* W

In this example, all register breakpoints are cleared with a "T" command followed by no operands. Next THE BUG is set so that a breakpoint will occur if the accumulator equals \$4F, the stack pointer is greater than \$FD, the carry flag is set and the zero flag is cleared. If all these conditions are met, a breakpoint will occur when program execution is initiated by the "G" command. Flags and registers not specified in the "T" command have no effect. The "W" command displays all breakpoints set. Note the memory breakpoints set in a previous example are displayed as well since they have not been cleared.

## BREAKPOINT COMMANDS

Memory breakpoints are cleared by the "B" command in the second command line. The "T" commands that follow set up THE BUG so that a breakpoint will occur if the accumulator is equal to \$4F, the Y register is greater than or equal to 3 and the zero flag is set. The "T" command in the next to last command line clears all register breakpoints previously set. When no breakpoints are set the "W" command has no effect.

## EXECUTION COMMANDS

This group of commands control execution of the user's program.

- 1) The "G" Command--Format G (opt add):(opt cnt)

The "G" or "go" command initiates the debugger. The user's program will be run under the control of THE BUG. Execution will continue until one of the following conditions are satisfied, the loop count is zero, a BRK instruction is encountered, or a RTS instruction results in a stack overflow. Execution will be conditionally stopped if breakpoint conditions previously specified by the user with either the "B" or "T" commands are satisfied. Execution initiated with the "G" command may be halted by hitting the escape key. If the game paddle option is active pressing the pushbutton on game paddle zero will have the same effect as hitting the escape key.

When execution is halted the program counter location along with the values of each of the 6502 registers are displayed by THE BUG.

Disassembly and instruction interpretation are time consuming tasks. In many cases the user will desire to accelerate the execution of his program. This may be done by entering the "N" command to inhibit THE BUG'S instruction display. Speed may also be increased by the "J" command which allows subroutines which push the stack below a preset level to run at full speed. The user may also make use of the "F" command which will speed up execution by running sections of code at full speed. When any of these options are used, either less information is displayed or less control is given to the user. This may be necessary because some operations must run at full speed to work correctly.

The "G" command is followed by an optional address. If an address is entered, it is used as the starting point for program execution. If this is omitted execution will start from the address set by the previous "G" command, i.e. the next executable instruction. The "G" command can also be followed by a colon and a step count. This step count determines the number of instructions which are to be executed (maximum of 255 or \$FF) before stopping. If no step count is entered execution continues until either the escape key is pressed or one of the other conditions mentioned above is met. Examples of the use of this command will be given in the command examples section of this manual.



## 2) The "F" Command--Format F (add):(opt cnt)

The "F" or "full speed execution" command allows the user to execute part of his program at full speed. In this way sections of code that have been previously tested and are known work may be ignored allowing the user to concentrate only on sections of his program that need debugging. Full speed execution starts with the next executable instruction that was set with a previous "G" command. Full speed execution continues until the program counter is set to equal the address specified in the command. During full speed execution register or memory breakpoints are not checked for. In addition hitting either the escape key or the pushbutton on game paddle zero have no effect.

The full speed execution mode is entered with the "F" command followed by a required hexadecimal address. In order for this command to operate properly this address must be in RAM. If a ROM address is specified an error message will be generated and the command will not run. The address specified must be executed by the user's program or control will not return to THE BUG. If this occurs the user may use the reset key to regain control. With the Autostart ROM hitting the reset key will return control to THE BUG. With the monitor ROM hitting the reset key will put the user in the Apple monitor. In order to return to THE BUG a monitor "G" command to the sixth address in THE BUG's memory space must be executed i.e. \$8905 in a 48K Apple with THE BUG loaded at HIMEM. In order for the "F" command to work three memory locations starting with the address specified in the "F" command are modified. If the user's program modifies or references these locations before control is returned to THE BUG unpredictable results will occur. When control returns to THE BUG the original contents of these locations are restored.

An optional loop count may be used with this command. If used it is separated from the address by a colon. The loop count specifies the number of times the instruction at the entered address will be executed before control will return to THE BUG. Each time this instruction is executed the conditions set with the "T" command are checked for, if satisfied program execution will halt.

Upon exit from the Full speed mode the next executable instruction is modified to be the current contents of the program counter. Then control is returned to THE BUG and the program counter and the contents of the 6502 registers are displayed.

## 3) The "S" Command--Format S:(opt cnt)

The "S" or "single step" command allows the user to execute his program one step at a time. The same options that control the display with the "G" command are used with this command. The contents of the program counter and the 6502 registers are displayed following instruction execution. An optional count parameter may be entered, if it is this specifies the number of instructions to be executed. The register values will be displayed for each instruction executed. If no step count is entered only a single instruction will be executed. When execution is started with the "S" command the user will be unable to stop execution with the escape key or game paddle. Memory or register breakpoints have no effect in this mode.

## 4) The "J" Command--Format J:(val)

The "J" or "conditional stack full speed mode" command allows the user to specify a value which will be checked each time a JSR instruction is executed. If the stack pointer is pushed below this value THE BUG will enter the full speed mode. Control will return to THE BUG when a RTS instruction restores the stack pointer to the entered value. Since the "J" command modifies memory like the "F" command the same restrictions that apply to the "F" command also apply to the "J" command. The memory location following the JSR instruction that resulted in full speed execution, must contain the instruction to be executed following return from the subroutine. In other words, subroutines that modify their return address will not operate with the "J" command. In order to enable the "J" mode, the command is followed by a colon and a hexadecimal value. To disable the "J" mode, the command with no operands is entered.

## COMMANDS THAT EXIT OR RESTART THE BUG

### 1) Reset

In an Apple with an Autostart ROM, hitting the reset key will return the user to the coldstart routine of THE BUG. The display will be reset and the stack pointer will be reset to \$FF. In addition a number of internal pointers in THE BUG will be reset. This command will not work if the user modified the values contained at locations \$3F2-\$3F4. The memory locations at \$3F2-\$3F3 point to THE BUG's coldstart routine, when THE BUG is installed. If the Monitor ROM is used, the reset key will place the user in the Apple monitor. THE BUG's reset routine may be executed by using the monitor "G" command to execute the instructions starting at the sixth memory location in THE BUG. In a 48K Apple with THE BUG at HIMEM this location is \$8905. The command "8905G" will return the user to THE BUG.

### 2) The "E" Command--Format E

The "E" command returns the user to BASIC. While in BASIC the user may execute any BASIC or DOS command. If either the FP or INT command is entered HIMEM will be reset and THE BUG will not be protected. In these cases the user will have to manually reset these locations if he desires. From Applesoft, THE BUG may be reentered with the "G" command. From either BASIC, THE BUG may be reentered with a call to locations 1013 or 1016. These commands are "CALL 1013" or "Call 1016". These locations point to THE BUG's warmstart routine. If these entry points are used, the user is returned to THE BUG with all conditions unchanged. If these locations are modified these commands will not work.

### 3) The "M" Command--Format M

The "M" command transfers the user to the Apple monitor. The user may then execute any monitor command. The "CTRL-Y" monitor command returns the user to THE BUG's warmstart routine. The user may return to THE BUG with the use of either of the following commands: "3F5G" or "3F8G".



## COMMANDS THAT EXIT OR RESTART THE BUG

### 4) The "CTRL-Q" Command--Format Q

The "CTRL-Q" or "quit" command removes THE BUG from memory. With the Monitor ROM, the monitor is entered. With the Autostart ROM, DOS is rebooted if the appropriate disk is in Drive #1. If the disk containing THE BUG is still in the disk drive, THE BUG will be restarted. If the HIMEM default location is used, then each time that THE BUG is loaded, HIMEM will be reset to a value \$D00 lower than previously, unless HIMEM is reset by the user or another program. Note that THE BUG expects the value at HIMEM to be correct when it is loaded.

### 5) The "&" Command--Format &

The "&" command when entered in Applesoft does an unconditional jump to THE BUG'S warmstart routine. This is equivalent to the BASIC instruction "CALL 1013". This instruction will not work correctly if the contents of \$3F5-3F7 are modified.

### 6) The "CTRL-Y" Command--Format ^Y

The "CTRL-Y" Command when entered in the monitor preforms an unconditional jump to THE BUG'S warmstart routine. This is equivalent to the monitor instruction "3F8G". This instruction will not work correctly if the contents of \$3F8-3FA are modified.

## ENTERING MULTIPLE COMMANDS

In order to allow the user maximum flexibility and ease when using THE BUG, he may type multiple commands on a single line. If this is done, the user must be careful to follow the proper command syntax. Each command must be terminated by either a comma or carriage return. In the case of the "q" and "tp" commands, since the value lists of these commands contain commas these commands are terminated by a period. In certain instances no command delineators are necessary. The user must be very careful if command delineators are not used, as THE BUG may misinterpret his command. This is allowed in instances where the command mnemonic cannot be mistaken as a hexadecimal digit or a valid part of the previous command. Any spaces typed in are ignored by the command interpreter, in this documentation spaces are sometimes used to separate the address from the command for clarity. These spaces are optional.

### Example

\* B A02,B 300,W

In this instance each command was followed by a comma.

\* TY=>3A,TX<FE,TP,C0,Z1.W

In this instance the "TY" and "TX" are both delineated with a comma, the "TP" command is delineated with a period from the final "W" command. If a comma was used in this example instead of a period an error would have resulted.

\* LB90LLLLLL

In this case a command delineator is not necessary as "L" cannot be mistaken for a hexadecimal digit.



### SPECIAL CONSIDERATIONS OF RUNNING THE BUG

Because of the internal structure of the 6502 microprocessor, base page locations make more efficient use of processor instructions due to increases in program execution speed and to decreases in program size. Base page locations, therefore are used to enhance the execution of THE BUG itself. In order to make THE BUG as transparent to the user as possible, all but three of the base page locations used by THE BUG are swapped, or saved in another location when the user's program is executing. Because of the memory swapping examination of certain locations will not produce the expected values. A list of the locations swapped appears in Appendix II. Locations \$3C to \$45 are used by THE BUG as address registers. These locations are swapped so the user need not worry about modifying these locations with his program. The "&" command is provided to allow the user to examine the values these locations contain during program execution. The values placed in locations \$3C-\$3D can be displayed by entering the "&" command. If this command is immediately followed by the "p" command the values placed in locations \$3E-\$45 can be displayed.

There are three base page locations which may not be used by the user as they are used by THE BUG and not swapped. These are \$3A, \$3B, and \$2F.

THE BUG also has its own set of input and output hooks. The user's set are swapped out and placed in the first five locations of THE BUG. If the default relocation value was chosen in a 48K APPLE, then the input and output hooks would be placed at \$8900 to \$8904. These locations correspond to locations \$35 to \$39. Beware when program execution is stopped and not continued while in DOS. DOS switches the input and output hooks while it is executing. If the user does not manually reset the input and output hooks to their original value DOS I/O is lost until the hooks stored at \$8900 to \$8904 are restored. The normal value of these hooks in a 48K APPLE starting at \$8901 is \$BD, \$9E, \$81, and \$9E.

If the user goes from THE BUG to BASIC or the monitor when his program is halted in the DOS I/O routines problems could arise. The problems occur because the DOS input and output routines save the value of the registers and stack pointer. If program execution is suspended while executing the DOS input-output routines entering the monitor or BASIC will change the values DOS saves for the registers and the stack pointers. If program execution is then restarted incorrect values for the registers and the stack pointers will result leading to fatal errors. As long as the user does not leave THE BUG when his program is halted in the DOS I/O routines he need not worry about the above error as THE BUG's I/O routines bypass the DOS I/O routines. During other times the user is free to exit THE BUG with both the "E" or "M" commands without worrying about the above error.

## SPECIAL CONSIDERATIONS

THE BUG maintains two stacks, it's stack and the user's stack. Because of this, these values will not be in their expected locations. These values may be found at the relocation address plus \$C00. THE BUG will execute instructions that reference locations \$100 to \$1FF correctly provided indirect addressing is not used.

### Example

```
TSX          ;THE VAL OF THE SP IS
             ;IN THE X REG
LDA $100,X   ;THE BUG WILL MODIFY
             ;THIS INSTRUCTION
             ;SO THE CORRECT VALUE IS USED

LDA #$10
LDY #$00
STA $FF
STY $FE      ;$FE-FF CONTAIN $100
TXA
TAY
LDA ($FE),Y  ;IN THIS CASE THE BUG WOULD NOT
             ;MODIFY THE INSTRUCTION
             ;AND THE INCORRECT
             ;VALUE WOULD BE USED

RTS
```

Since the stack is rarely addressed in this manner, this limitation should not be a problem to most users.

Programs that make use of interrupt driven routines will be difficult to debug with THE BUG since when an interrupt occurs THE BUG will lose control of program execution. This should not be a problem to most users since interrupt driven routines are rarely used with the APPLE II computer. There are no interrupt driven routines in the Monitor ROM, the Autostart ROM, APPLESOFT, Integer BASIC or DOS.

THE BUG also swaps out the line input buffer, but this swap is transparent to the user. THE BUG uses the input area, but swaps it out before it uses the area for input and immediately swaps the user input back in afterward. If the user were to display the contents of the display buffer with THE BUG, the correct contents of the buffer would be displayed and would not be corrupted through the use of THE BUG. The maximum size of THE BUG's input buffer is 39 characters. It would be unlikely for the user to enter a command line longer than that. If the user were to go to the monitor, then the input buffer would be corrupted through the input of the monitor commands.

When either the "E" or "M" commands are used the values at locations \$36-\$39 are replaced with the values found at a displacement of \$7C4-\$7C7 from the beginning of the program. The locations of other base page values can be found in Appendix II.

## MEMORY LOCATIONS USED BY THE BUG DURING THE BOOT

Memory locations \$200-\$DFF are used by the loader program contained in THE BUG's boot. Data located at these memory locations will be destroyed when THE BUG is booted. All other memory locations with the exception of the memory used by the main program will remain intact when THE BUG is booted. The programs residing at \$200-\$DFF are used only during THE BUG's boot up process. Once THE BUG is up this memory location may be used by the user without interfering with the operation of THE BUG.

## COMMAND EXAMPLES

Two demonstration programs were loaded into memory at \$B00 - \$CB4 along with THE BUG. These will be used to demonstrate THE BUG so the user may become familiar with the commands available with THE BUG.

The first program is a simple timing loop. The listings will contain comments in this documentation for the benefit of the user. Obviously the "L" command cannot generate comments and these will not appear on the display. The following command will list the first program.

\* L B00

|       |    |    |     |        |                 |
|-------|----|----|-----|--------|-----------------|
| 0B00- | A0 | 20 | LDY | #\$20  | ; X&Y REG RESET |
| 0B00- | A2 | 14 | LDX | #\$14  |                 |
| 0B04- | 88 |    | DEY |        | ;COUNTING LOOPS |
| 0B05- | D0 | FD | BNE | \$0B04 |                 |
| 0B07- | CA |    | DEX |        |                 |
| 0B08- | D0 | FA | BNE | \$0B04 |                 |
| 0B0A- | 60 |    | RTS |        |                 |
| 0B0B- | 00 |    | BRK |        | ; NOT PART OF   |
| 0B0C- | 00 |    | BRK |        | ; PROGRAM       |
| 0B0D- | 00 |    | BRK |        |                 |

Execution of this program may be initiated with the following command:

\* G B00

Note that each instruction is displayed as it is executed. The display scrolls very rapidly making it difficult for the user to examine the program. When the RTS instruction at \$B0A is executed the user will be returned to THE BUG's monitor. It has taken a considerable amount of time for this short program to execute. The reason for this is because THE BUG executes many lines of code for each instruction it examines for the user. This is necessary for THE BUG to be able to perform its functions. We will later show the user how program execution may be sped up. If the user desires to stop program execution before the RTS instruction is encountered he may either type the escape key or hit the pushbutton on game paddle zero if the game paddle option is active. When execution is halted by any of these methods the program counter plus the 6502 registers are displayed.



# COMMAND EXAMPLES

If the user desires the display to be slower so that it could be read the following command may be entered:

```
* Q
_
* G B00
```

Each instruction is again displayed to the user but the speed of the display depends on whether the game paddle option is active. If it is the speed of the display may be varied by turning the potentiometer on game paddle zero. This allows the program sections of interest to be examined closely while other code is passed over quickly. If the game paddle option is inactive the speed of the display is slower than the standard display speed but this slower speed cannot be varied.

Program execution may be terminated with either the escape key or pushbutton. Again the contents of the program counter and the 6502 registers are displayed.

Instruction execution can be sped up by entering the "N" command. When this command is active the instructions will not be disassembled and displayed as they are executed.

```
* N
_
* GB00
```

This command will not display each instruction but will display the values of the program counter and the 6502 registers when user program execution is halted. Note that it takes considerably less time for these instructions to be executed under this mode.

An optional instruction counter may be used with the "G" command.

```
* I
_
* G B00:5
```

This set of instructions first resets THE BUG to the standard display mode and then executes 5 instructions starting at \$B00. The instruction counter may be any number from \$1 to \$FF. Execution may still be halted prematurely by any of the methods described above.

The next instruction may be executed with the following command:

```
* G:1
```

# COMMAND EXAMPLES

Note when there is no address entered the next executable instruction is executed by the "G" command. The same result will be accomplished with the following command:

```
* S
```

The "S" command may be used to execute more than one instruction.

```
* S:A
```

In this example 10 instructions are executed. Following the disassembled code for each instruction is a listing of both the program counter and the 6502 registers. Hitting the escape key will not halt execution when it is initiated with the "S" command. In addition memory and register breakpoints will not terminate program execution. A starting address cannot be specified by the "S" command but must have been previously specified by a "G" command.

The "N" and "Q" commands have the same effects with the "S" command as they do with the "G" command. With the "N" command the instructions are not disassembled but the value of the program counter and the 6502 registers are displayed following the execution of each instruction.

```
* N
_
* S:10
```

In this example 16 instructions are executed. Since the "N" command was issued only the value of the program counter and the 6502 registers are displayed after each instruction is executed.

```
* I
```

Reenters the standard display mode. If the user has a printer attached to slot #1, a listing of his THE BUG operations may be obtained with the "C" command.

```
* C
_
* G:12
```

This command directs THE BUG's output to the printer. Depending on the printer interface used output may be displayed on the screen as well. All output is directed to the printer until a second "C" command is entered. The second command executes \$12 instructions.

```
* C
```

At this point the printer is disenabled.

## COMMAND EXAMPLES

THE BUG allows the user to modify the value of his registers. Program execution is restarted with the following command:

```
* G B00:3
```

Three instructions are executed. The program counter is now equal to \$B05. The Y reg = \$1F and the X reg = \$14. We will now modify the program so that the X reg = 2 and the Y reg = 3.

```
* RX:2
```

```
* RY:3
```

```
* D
```

The first two command place values in the X and Y registers. The "D" command displays the values of the program counter and the 6502 registers so we can verify the changes that we have just made.

Execution is now restarted with the "G" command.

```
* G
```

Program execution is faster now because the outer loop of the program executes only two times.

In this section we will demonstrate how THE BUG may be used to modify a program. We will show two methods: one which can be used by the person who has the mini assembler and the other by a person without the mini assembler.

The monitor is entered with the "M" command.

```
* M
```

Next the mini assembler is entered.

```
* F666G
```

Now the user will modify his program.

```
I B00:LDX #$14
```

```
0B00- A2 14 LDX #$14
```

```
I LDY #$20
```

```
0B02- A0 20 LDY #$20
```

```
I B08:BNE $B02
```

```
0B08- D0 F8 BNE $0B02
```

```
I $^Y
```

The "CTRL-Y" instruction returned the user to THE BUG. The modified program can now be examined with the "L" command.

```
* LB00
```

```
0B00- A2 14 LDX #$14
```

```
0B02- A0 20 LDY #$20
```

```
0B04- 88 DEY
```

```
0B05- D0 FD BNE $0B04
```

```
0B07- CA DEX
```

```
0B08- D0 F8 BNE $0B02
```

```
0B0A- 60 RTS
```

```
0B0B- 00 BRK
```

```
0B0C- 00 BRK
```

```
0B0D- 00 BRK
```

Users who do not have the mini assembler may still change their program but this requires the user to hand assemble his program. The following command lines will modify the demo program.

```
* @B00:A2,14,A0,20
```

```
* @B09:F8
```

```
* LB00
```

The above commands made the same changes that were made with the mini assembler but the user needed to look up op-codes and displacements. The new program may now be saved with a BSAVE command or executed from THE BUG.

If we wanted to save this program we would enter the following sequence of commands. Before we do this the disk containing THE BUG should be removed from the disk drive and a standard DOS disk inserted.

```
* E
```

```
] SAVE BUG.DEMO.I,A$B00,L$B
```

```
] CALL 1013
```

The same sequence could have been used to enter any DOS command.

This modified program may be run with the following command:

```
* GB00
```

This program takes considerably less time to execute than the previous one since the Y register is not reset to \$FF each time the instruction at \$B08 occurs.

USE OF THE P AND Z COMMANDS:

The P command will display the hexadecimal value of up to 8 memory locations. The following command will print the values at locations \$B15 - \$B17.

\* PB15

0B15- D7 C9 D4

Locations \$B18 - \$B1F can be displayed with the following command:

\* P

0B18- C8 A0 CD D3 C2 A0 D3 C5

The Z command may be used to display a large block of text and its ASCII translation.

\* ZB10

|       |                         |          |
|-------|-------------------------|----------|
| 0B10- | D4 C5 D8 D4 A0 D7 C9 D4 | TEXT WIT |
| 0B18- | C8 A0 CD D3 C2 A0 D3 C5 | H MSB SE |
| 0B20- | D4 0A C9 D3 A0 C4 C9 D3 | T IS DIS |
| 0B28- | D0 CC C1 D9 C5 C4 A0 C9 | PLAYED I |
| 0B30- | CE A0 CE CF D2 CD C1 CC | N NORMAL |
| 0B38- | A0 D6 C9 C4 C5 CF AE A0 | VIDEO.   |
| 0B40- | 54 45 58 54 20 57 49 54 | TEXT WIT |
| 0B48- | 48 20 4D 53 42 20 43 4C | H MSB CL |
| 0B50- | 45 41 52 45 44 20 49 53 | EARED IS |
| 0B58- | 20 44 49 53 50 4C 41 59 | DISPLAY  |
| 0B60- | 45 44 20 49 4E 20 49 4E | ED IN IN |
| 0B68- | 56 45 52 53 45 20 56 49 | VERSE VI |
| 0B70- | 44 45 4F 20 42 55 54 20 | DEO BUT  |
| 0B78- | 53 50 41 43 45 53 20 41 | SPACES A |
| 0B80- | 52 45 20 4E 4F 54 2E 20 | RE NOT.  |

Note that the text from \$B10 - \$B3F is in normal video as these values all have their MSB set. The text from \$B40 - \$B87 is in inverse video since these values have their MSB cleared.

USE OF THE BREAKPOINT COMMAND:

If we desired to stop program execution at location \$B07 we could set a breakpoint.

\* B

\* B B07

The first "B" command clears all register breakpoints and the second one sets a breakpoint at \$B07.

\* GB00

Restarts the program, the first loop will execute and when the instruction at \$B07 is encountered, execution will stop.

Execution may be restarted by entering the "G" command.

\* G

Execution will continue until the instruction at \$B07 is again reached.

If execution is initiated with the "S" command the program will not halt at memory breakpoints, since breakpoints have no effect on execution initiated with the "S" command. Hitting the escape key or the pushbutton on game paddle #0 will also not halt the program.

\* S:7F

With the above instruction program execution will continue until \$7F instructions are executed. A BRK instruction or a RTS instruction which results in stack overflow will also halt program execution.

All breakpoints that are set may be displayed with the following command:

\* W



The "T" command will now be used to set up a register breakpoint that will cause program execution to halt when the Y register equals 6.

\* TY=6

\* B

\* G

The above command sets the register breakpoint. Previously set memory breakpoints are cleared by the "B" command. Execution will stop at \$B05. Note that the Y register is equal to 6.

\* G

A "G" command entered now will only execute a single instruction since the next instruction does not change the Y register and it still satisfies the breakpoint conditions.

\* G

This "G" command will again execute many instructions as the next instruction modifies the Y register. Execution will continue until the Y register again equals 6.

We can now set up THE BUG so that execution will halt only when both the Y register equals 6 and the X register equals 5.

\* RX:10

\* TX=5

\* G

The first instruction resets the X register to \$10. Next a breakpoint is set for the X register. A breakpoint is already set for the Y register. Execution is then initiated with the "G" command. Execution continues until \$B05. Note that the conditions set for the X and Y registers are both satisfied. Both of these conditions must be satisfied or execution will continue. As many register conditions may be set as the user desires, in order for execution to halt all set conditions must be satisfied. Registers that are not set are not tested for. A display of the current breakpoint conditions can be obtained with the "W" command.

\* W

We can remove the X register breakpoint with the following command:

\* TX

Now execution again will depend only on the Y register. All register breakpoints are cleared with this command.

\* T

THE BUG can be set so a breakpoint will occur everytime the Z flag is set i.e. a result equals 0.

\* TP,Z1

\* G

Execution will continue until \$B05, note that the Y register was decremented to zero by the previous instruction, setting the Z flag. THE BUG can be made so that execution will stop when the sign flag is cleared i.e., an instruction results in a positive number.

\* TP,S0

\* W

Note that only the sign flag is active as the command that set the sign flag breakpoint cleared the Z flag breakpoint. Both may be set with the following command:

\* TP,S0,Z1

The register breakpoint combination may be displayed with the "W" command.

\* W

We will now clear all register breakpoint conditions.

\* T

#### USE OF THE F COMMAND:

THE BUG gives the user the ability to execute his program at full speed. The following command will execute this program at full speed.

\* GB00:1

\* FB0A

Note it takes an extremely short time for this program to execute. The initial "G" instruction sets up the next instruction register. This value is the same as the value of the program counter displayed with the "D" command. The "F" command cannot set this address but starts execution at the next executable instruction and continues until the location specified in the command is reached. The next executable instruction is then reset to be the address specified in the "F" command i.e. \$B0A.

An optional loop count can be specified with this command.

\* GB00:1

\* FB08:2

In this case execution starts at \$B00 and continues until the instruction at \$B08 is executed two times.

The register breakpoints may be checked with the "F" command. The only time they are checked is when the instruction specified is executed.

\* TX=8

\* FB08:F

In this case the instruction at \$B08 is not executed 15 times since the X register is equal to 8 the 10th time the instruction is executed. The X register is only checked when the instruction at \$B08 is executed. Note that the starting address in this instruction has been set by the previous instruction to be \$B08.

Next we will demonstrate the results of specifying an address that will never be executed.

\* T

\* GB00:1

\* FB0B

Location \$B0B is never executed, as a result control will not return to THE BUG. When the RTS occurs, stack overflow will occur, taking the user to the monitor. The user can return to THE BUG by hitting the reset key if he has an autostart ROM. With the monitor ROM he must execute the 6th Loc in THE BUG. The address of this location can be found at locations \$3F2 - \$3F3. In a 48K Apple with THE BUG located at HIMEM this location is \$8905. The monitor command "8905G" will return the user to THE BUG.

### USE OF THE J COMMAND:

The second program located between \$B90 - \$CB3 is somewhat more complex and it will be used to demonstrate the "J" command. This program basically asks the user a few questions to which the user will respond to. Then the bell will be rung 16 times. Finally the speaker will be clicked 256 times, the interval between the clicks will depend on the setting of game paddle zero. The following command will list the program.

\* LB90,L,L,L,L,L,L,L,L

|      |     |     |    |      |          |  |                    |
|------|-----|-----|----|------|----------|--|--------------------|
| OB90 | -A0 | 05  |    | LDY  | #\$05    |  | ; 5 SPACES         |
| OB92 | -20 | BE  | FD | JSR  | \$FD8E   |  | ;CROUT             |
| OB95 | -88 |     |    | DEY  |          |  |                    |
| OB96 | -D0 | FA  |    | BNE  | \$0B92   |  |                    |
| OB98 | -B9 | 3B0 |    | CLDA | \$0C3B,Y |  | ;LOC OF QUEST #1   |
| OB9B | -08 |     |    | PHP  |          |  | ;SAVE STATUS       |
| OB9C | -09 | 80  |    | ORA  | #\$80    |  | ;SET MSB           |
| OB9E | -20 | F0  | FD | JSR  | \$FDF0   |  | ;COUT1             |
| OBAB | -CB |     |    | INY  |          |  | ;NEXT CHARACTER    |
| OBAB | -28 |     |    | PLP  |          |  | ;STOP IF NEG       |
| OBAB | -10 | F3  |    | BPL  | \$0B98   |  | ;LOOP              |
| OBAB | -20 | 6F  | FD | JSR  | \$FD6F   |  | ;GETLIN            |
| OBAB | -20 | 8E  | FD | JSR  | \$FD8E   |  | ;CROUT             |
| OBAB | -A0 | 00  |    | LDY  | #\$00    |  | ;RES CHAR PNTR     |
| OBAD | -B9 | 54  | OC | LDA  | \$0C54,Y |  | ;LOC OF NXT OUTPUT |
| OBBO | -08 |     |    | PHP  |          |  | ;SAVE STA          |
| OBBI | -09 | 80  |    | ORA  | #\$80    |  |                    |
| OBBI | -20 | F0  | FD | JSR  | \$FDF0   |  | ;OUTPUT CHAR       |
| OBBI | -C8 |     |    | INY  |          |  | ;NEXT              |
| OBBI | -28 |     |    | PLP  |          |  | ;STOP IF NEG       |
| OBBI | -10 | F3  |    | BPL  | \$0BAD   |  | ;PRINT LOOP        |
| OBBA | -A0 | 00  |    | LDY  | #\$00    |  |                    |
| OBBC | -B9 | 00  | 20 | LDA  | \$200,Y  |  | ;INPUT BUF         |
| OBBC | -C9 | 8D  |    | CMP  | #\$8D    |  | ;END AT CR         |
| OBBI | -F0 | 06  |    | BEQ  | \$0BC9   |  |                    |
| OBBI | -20 | F0  | FD | JSR  | \$FDF0   |  | ;OUTPUT CHAR       |
| OBBI | -C8 |     |    | INY  |          |  | ;NEXT CHAR         |
| OBBI | -10 | F3  |    | BPL  | \$00BC   |  | ;PRINT LOOP        |
| OBBI | -A0 | 00  |    | LDY  | #\$00    |  |                    |
| OBBI | -B9 | 57  | OC | LDA  | \$0C57,Y |  | ;NEXT STRING       |
| OBBI | -08 |     |    | PHP  |          |  | ;SAVE STA          |
| OBBI | -09 | 80  |    | ORA  | #\$80    |  | ;SET MSB           |
| OBBI | -20 | F0  | FD | JSR  | \$FDF0   |  | ;OUTPUT CHAR       |
| OBBI | -C8 |     |    | INY  |          |  | ;NEXT CHAR         |
| OBBI | -28 |     |    | PLP  |          |  |                    |
| OBBI | -10 | F3  |    | BPL  | \$0BCB   |  | ;PRINT LOOP        |
| OBBI | -20 | 8E  | FD | JSR  | \$FD8E   |  | ;CR                |
| OBBI | -A0 | 00  |    | LDY  | #\$00    |  |                    |
| OBBI | -B9 | 69  | OC | LDA  | \$0C69,Y |  | ;NEXT STRING       |
| OBBI | -08 |     |    | PHP  |          |  | ;SAVE STA          |
| OBBI | -09 | 80  |    | ORA  | #\$80    |  | ;SET MSB           |
| OBBI | -20 | F0  | FD | JSR  | \$FDF0   |  | ;OUTPUT CHAR       |



# COMMAND EXAMPLES

```

OBE6 -C8      INY
OBE7 -28      PLP
OBE8 -10 F3    BPL $0BDD      ;PRINT LOOP
OBEA -8A      TXA            ;X # OF CHAR
OBEB -20 DA FD JSR $FD8E      ;CR
OBEF -A0 00    LDA #$00      ;256 CLICKS
OBF0 -B9 76 0C LDA $0C76,Y    ;SAVE
OBF3 -08      PHP            ;MEMORY
OBF4 -09 80    ORA #$80      ;END AFTER 250
OBF6 -20 F0 FD JSR $FDF0      ;GAME PADDLE #0
OBF9 -C8      INY            ;READ GAME PADDLE
OBFA -28      PLP            ;PADDLE VAL
OBFB -10 F3    BPL $0BFD      ;IN ACCM
OBFD -20 8E FD JSR $FD8E      ;WAIT LOOP
OC00 -A0 10    LDY #$10      ;CLICK SPEAKER
OC02 -20 3A FF JSR $FF3A      ;LOOP
OC05 -88      DEY
OC06 -D0 FA    BNE $0C02
OC08 -A0 00    LDY #$00
OC0A -B9 8F 0C LDA $0C8F,Y    ;NEXT STRING
OC0D -08      PLP
OC0E -09 80    ORA #$80
OC10 -20 F0 FD JSR $FDF0      ;OUTPUT CHAR
OC13 -C8      INY
OC14 -28      PLP
OC15 -10 F3    BPL $0C0A      ;PRINT LOOP
OC17 -20 8E FD JSR $FD8E      ;CR
OC1A -A9 FF    LDA #$FF      ;256 CLICKS
OC1C -8D 3A 0C STA $0C3A      ;SAVE
OC1F -CE 3A 0C DEC $0C3A      ;MEMORY
OC22 -F0 10    BEQ $0C34      ;END AFTER 250
OC24 -A2 00    LDX #$000      ;GAME PADDLE #0
OC26 -20 1E FB JSR $FB1E      ;READ GAME PADDLE
OC29 -C8      INY            ;PADDLE VAL
OC2A -98      TYA            ;IN ACCM
OC2B -20 A8 FC JSR $FCA8      ;WAIT LOOP
OC2E -AD 30 C0 LDA $C030      ;CLICK SPEAKER
OC31 -4C 1F 0C JMP $0C1F      ;LOOP
OC34 -60      RTS

```

There are more elegant methods for performing these functions but this program is a nice demonstration. The variable and string storage sections of this program can be displayed by the following command.

\* ZC3A

We can now run this section of code with the following command:

\* GB90

# COMMAND EXAMPLES

Note this program runs very slowly. When input is requested the user must pause between entering each character to give the program a chance to catch up. The bell does not sound like a bell and the clicks are very far apart. The game paddle input part of this program also is not read in correctly. Execution can be sped up with the "N" command.

\* N

\* GB90

Execution is still very slow. Many of the functions used in this program must execute at full speed to work correctly. The "J" command is designed for this purpose. We can now set a value for the stack so that when a JSR instruction pushes the stack below that level execution will enter the full speed mode to return THE BUG when a RTS instruction restores the stack pointer to the calling level. In the case of this program we can set this value to \$FF as all subroutines called by this program are located in the monitor. These subroutines are known to work correctly and user needn't spend time debugging them. The following instructions causes THE BUG to enter the full speed mode when the stack is pushed below \$FF by a JSR instruction.

\* I

\* !

\* J:FF

The "I" instruction places THE BUG back in the standard display mode. The "I" instruction resets the stack pointer to \$FF. The "J" instruction enables the conditional full speed execution mode. Execution is initiated with the following instruction.

\* GB90

Now execution occurs much faster. String input occurs at normal speed. The bell sounds correctly. The clicks still occur very far apart but the user is able to hear the effect of turning the game paddle. Only the instructions in the main program are disassembled, these instructions are also checked by THE BUG for any conditions set. All subroutines are run at full speed, their instructions not disassembled nor are conditions set by the user checked by THE BUG.

Speed will be increased further by using the "N" command in conjunction with the "J" command.

\* N

\* G B90

The clicking sounds are much faster and the user can tell a real difference in the speed when he turns the game paddle.



A great difference in speed can again be noted with the following commands:

\* G B90:1

\* F C34

Now the clicks are very close together and with the use of the game paddle can be made into a single tone. The program was run at full speed.

We are now going to modify the program so game paddle #1 is used to control the speed of the clicks instead of game paddle #0.

First we list the instructions that need to be changed with the "L" command.

\* LC24S

```
0C24 -A2 00      LDX  #$00
0C26 -20 1E FB   JSR  $FB1E
```

Ten instructions are listed but we are only interested in the first instruction which places a zero in the X-register. The value in the X register determines which game paddle is to be read when the routine at \$FB1E (PREAD) is executed. If the X register were equal to 1 then game paddle #1 would be read. We therefore want to change the value at \$C25 to 1.

\* @C25:1

\* LC24S

```
0C24 -A2 01      LDX  #$01
0C26 -20 1E FB   JSR  $FB1E
```

We will now run the program with the "Q" command active. The "Q" command resets the display so that the instructions are again disassembled. The previously entered "J" command is still active.

\* Q

\* GB90

The speed of program execution is controlled by game paddle #0 if the game paddle option is active. The speed of the clicks are controlled by game paddle #1. User's who have game paddles will be able to vary the execution speed of their program so that sections of interest can be slowed down for careful observation while other sections may be sped by.

The "J" command will not work if the jump to full speed execution occurs in ROM. The following commands will illustrate this.

\* I

\* BB90

\* GB90

\* J:FD

\* G

The "I" command returns THE BUG to the standard display mode. The "J" command resets THE BUG so that full speed execution will now occur when a JSR instruction pushes the stack pointer below \$FD. Note execution will halt at \$FDF9 since the JSR at this instruction would push the stack pointer below \$FD. Since this instruction is in ROM full speed execution cannot occur since Locations \$FDFC - \$FDFF would need to be modified. An error message "ADR NOT IN RAM" is displayed and control returns to THE BUG. If user desires to reset J to \$FF he must first reset the stack to \$FF. The following commands will accomplish this.

\* J

\* TS=FF

\* G

\* T

\* J:FF

\* G

The initial "J" command disables the conditional full speed mode. The "T" command sets up a register breakpoint so execution will halt when the stack pointer equals \$FF. Next the "G" initiates execution from where the previous "G" command stopped. Execution halts when the stack pointer equals \$FF. Now the register breakpoint is removed with another "T" command. The "J" command is entered and then execution is restarted with the "G" command.

USE OF THE B COMMAND

The final section will demonstrate the "B" command. We are going to enter 8 breakpoints that will divide this program into 8 sections.

\* B BAB  
\_

\* B B98  
\_

\* B:l  
\_

\* B BBA  
\_

\* B BC9  
\_

\* B BD8  
\_

\* B BFE  
\_

\* B C00  
\_

\* B C08  
\_

\* W  
\_

|       |    |       |
|-------|----|-------|
| BRKPT | #1 | 0BBA  |
| BRKPT | #2 | 0BAB  |
| BRKPT | #3 | 0B98  |
| BRKPT | #4 | 0BC9  |
| BRKPT | #5 | 0BD8  |
| BRKPT | #6 | 0BFE  |
| BRKPT | #7 | 0C00  |
| BRKPT | #8 | 0C08D |

This demonstrates how breakpoints are entered. Remember BRKPT #1 was previously set by us to be \$B98. The next 2 breakpoints entered filled the 2 lowest free locations. Then BRKPT #1 was erased with the "B:l" command therefore the next breakpoint entered was placed in the lowest unfilled locations i.e. breakpoint #1. The other breakpoints filled locations 4-8 in sequential order.

Execution is now started with the "G" command.

\* G B90  
\_

Execution is stopped at \$B98 because BRKPT #3 was satisfied. Execution is now restarted with the "G" command.

\* F BA5  
\_

\* G  
\_

Execution is now stopped at \$BAB, the next breakpoint encountered. We now want to enter another breakpoint with the "B" command.

\* B C1A  
\_

\* G  
\_

Note execution does not stop at loc \$BBA but rather continues to \$BC9 since BRKPT #1 which used to contain \$BBA was changed to \$C1A by the "B" command above since the 9th breakpoint entered overwrites the 1st breakpoint if no breakpoints are free. If another breakpoint were now entered it would overwrite breakpoint #2.

I hope these examples will help the user become familiar with the program. As can be seen from these examples many commands can be added together, giving the user alot of debugging power. I have tried as best as I can to explain this program. However it is impossible to foresee all possible questions that might arise. As a result, I will be happy to answer any questions about the program provided the user submits his questions in writing to the publisher SENSIBLE SOFTWARE, Inc. with a self addressed stamped envelope. If you have any suggestions on how to improve this program or have found any Bugs in it, I would like to hear from you as well.

## APPENDIX I: COMMAND SUMMARY

## COMMAND FORMAT

## DESCRIPTION

## DISPLAY COMMANDS

D ..... Displays registers  
 P(opt add) ..... Print memory command  
 Z(opt add) ..... Print Extended memory  
 L(opt add) ..... Disassemble  
 I ..... Instruction display  
 N ..... No instruction display  
 Q ..... Controlled instruction display  
 C ..... Printer enable/disable  
 & ..... Display base page vals \$3C-\$3D  
       (If a "P" command follows  
       the "&" command \$3E-\$45  
       values will be displayed)  
 W ..... Display breakpoints  
 ^T ..... Places screen in Text Mode  
 ^G ..... Places screen in Graphics Mode  
 ^F ..... Places screen in Primary Page  
 ^S ..... Places screen in Secondary Page  
 ^N ..... Places screen in Primary Page  
           and Text Mode

## MODIFICATION COMMANDS

@(opt add):(val list) ..... Modify memory command  
       (CR or period)  
 R(reg):(val) ..... Modify register  
 ^W ..... Disable language card write  
       protect (same as LDA \$C083)  
 ^P ..... Enable language card write  
       protect (same as LDA \$C080,  
       LDA \$C080)  
 I ..... Reset stack pointer to \$FF

## BREAKPOINT COMMANDS

B(opt add):(#) ..... Set memory breakpoint  
 T(reg)(=<>)(val) ..... Set register breakpoint  
 TP, (Z1,Z0,V1,V0,C1,C0, ..... Set status register breakpoint  
       D1,D0,I1,I0,B1,B0,  
       N1,N0) (CR or Period)

## COMMAND FORMAT

## DESCRIPTION

## EXECUTION COMMANDS

G(opt add):(opt cnt) ..... Execution command  
 F(add):(opt cnt) ..... Full speed execution  
 S:(opt cnt) ..... Single step command  
 J:(val) ..... Conditional stack full speed  
                   command

## COMMANDS TO EXIT AND REENTER THE BUG

Reset key ..... Return to THE BUG--COLDSTART  
 E ..... Return to BASIC  
 M ..... Return to Monitor  
 Call 1013 ..... Return to THE BUG--WARMSTART  
 Call 1016 ..... Return to THE BUG--WARMSTART  
 ^Y ..... Return to THE BUG from the  
           monitor--WARMSTART  
 ^Q ..... Quit THE BUG  
 & ..... Return to THE BUG from  
           Applesoft--WARMSTART

## KEY:

(opt add)--an optional Hexadecimal address between \$0-\$FFFF  
 (add)--a Hexadecimal address between \$0-\$FFFF  
 (val list)--a list of hexadecimal values between \$0-\$FFFF each  
 value must be separated by either a space or a comma

(CR or period)--indicates that command must be terminated with  
 either a CR or .

(reg)--either an A,X,Y,S,P

(#)--a hexadecimal value between \$1-\$8

(=<>)--one or any combination of the following symbols =<>

(Z1,Z0,V1,V0,C1,C0,D1,D0,I1,I0,B1,B0,N1,N0)--one or any

combination of the bracketed values separated by commas

(opt cnt)--a hexadecimal value between \$1-\$FF

(val)--a hexadecimal number between \$0-\$FF



# APPENDIX II--THE BUG LOCATIONS

| LOC         | NAME               | DISPLACEMENT | VAL @ HIMEM   |
|-------------|--------------------|--------------|---------------|
| \$20        | WNDLEFT            | \$7D0        | \$90D0        |
| \$21        | WNDWDTH            | \$7D1        | \$90D1        |
| \$22        | WNDTOP             | \$7D2        | \$90D2        |
| \$23        | WNCB'TM            | \$7D3        | \$90D3        |
| \$24        | CH                 | \$7D4        | \$90D4        |
| \$25        | CV                 | \$7D5        | \$90D5        |
| \$28        | BASL               | \$7D6        | \$90D6        |
| \$29        | BASH               | \$7D7        | \$90D7        |
| \$2A        | BAS2L              | \$7D8        | \$90D8        |
| \$2B        | BAS2H              | \$7D9        | \$90D9        |
| \$2F        | LENGTH             | ----         | -----         |
| \$33        | PROMPT             | \$7C2--BUG'S | \$90C2        |
|             |                    | \$7C3--PROGS | \$90C3        |
| \$35        | ----               | \$0          | \$8900        |
| \$36-\$37   | CSWL-CSWH (User's) | \$1-\$2      | \$8901-\$8902 |
| \$38-\$39   | KSWL-KSWH (User's) | \$3-\$4      | \$8903-\$8904 |
| ^8\$36-\$37 | CSWL-CSWH (Mon)    | \$7DA-\$7DB  | \$90DA-\$90DB |
| \$38-\$39   | KSWL-KSWH (Mon)    | \$7DC-\$7DD  | \$90DC-\$90DD |
| \$3A-\$3B   | PCL-PCH            | ----         | -----         |
| \$3C-\$3D   | A1L-A1H            | \$7FE-\$7FF  | \$90FE-\$90FF |
| \$3E-\$3F   | A2L-A2H            | \$800-\$801  | \$9100-\$9101 |
| \$40-\$41   | A3L-A3H            | \$802-\$803  | \$9102-\$9103 |
| \$42-\$43   | A4L-A4H            | \$804-\$805  | \$9104-\$9105 |
| \$44-\$45   | A5L-A5H            | \$806-\$807  | \$9106-\$9107 |
| \$4C-\$4D   | HIMENI             | ----         | -----         |
| \$73-\$74   | HIMENA             | ----         | -----         |
| \$100-\$1FF | STACK              | \$C00-\$CFF  | \$9500-\$95FF |
| \$200-\$225 | INPUT BUF          | \$B77-\$B9E  | \$9477-\$949E |
| ---         | COLDSTART ROUTINE  | \$0005       | \$8905        |
| ---         | WARMSTART ROUTINE  | \$0089       | \$8989        |

# THE BUG

## COMMAND SUMMARY

### COMMAND FORMAT

### DESCRIPTION

#### DISPLAY COMMANDS

|                  |                                                                                                                         |
|------------------|-------------------------------------------------------------------------------------------------------------------------|
| D .....          | Displays registers                                                                                                      |
| P(opt add) ..... | Print memory command                                                                                                    |
| Z(opt add) ..... | Print Extended memory                                                                                                   |
| L(opt add) ..... | Disassemble                                                                                                             |
| I .....          | Instruction display                                                                                                     |
| N .....          | No instruction display                                                                                                  |
| Q .....          | Controlled instruction display                                                                                          |
| C .....          | Printer enable/disable                                                                                                  |
| & .....          | Display base page vals \$3C-\$3D<br>(If a "P" command follows<br>the "&" command \$3E-\$45<br>values will be displayed) |
| W .....          | Display breakpoints                                                                                                     |
| ^T .....         | Places screen in Text Mode                                                                                              |
| ^G .....         | Places screen in Graphics Mode                                                                                          |
| ^F .....         | Places screen in Primary Page                                                                                           |
| ^S .....         | Places screen in Secondary Page                                                                                         |
| ^N .....         | Places screen in Primary Page<br>and Text Mode                                                                          |

#### MODIFICATION COMMANDS

|                             |                                                                           |
|-----------------------------|---------------------------------------------------------------------------|
| @(opt add):(val list) ..... | Modify memory command<br>(CR or period)                                   |
| R(reg):(val) .....          | Modify register                                                           |
| ^W .....                    | Disable language card write<br>protect (same as LDA \$C083)               |
| ^P .....                    | Enable language card write<br>protect (same as LDA \$C080,<br>LDA \$C080) |
| ! .....                     | Reset stack pointer to \$FF                                               |

#### BREAKPOINT COMMANDS

|                                                                      |                               |
|----------------------------------------------------------------------|-------------------------------|
| B(opt add):(\$)                                                      | Set memory breakpoint         |
| T(reg)(=<>)(val)                                                     | Set register breakpoint       |
| TP,(Z1,Z0,V1,V0,C1,C0,<br>D1,D0,I1,I0,B1,B0,<br>N1,N0)(CR or Period) | Set status register breakpoin |



# THE BUG

## COMMAND FORMAT

## DESCRIPTION

### EXECUTION COMMANDS

G(opt add):(opt cnt) ..... Execution command  
 F(add):(opt cnt) ..... Full speed execution  
 S:(opt cnt) ..... Single step command  
 J:(val) ..... Conditional stack full speed command

### COMMANDS TO EXIT AND REENTER THE BUG

Reset key ..... Return to THE BUG--COLDSTART  
 E ..... Return to BASIC  
 M ..... Return to Monitor  
 Call 1013 ..... Return to THE BUG--WARMSTART  
 Call 1016 ..... Return to THE BUG--WARMSTART  
 ^Y ..... Return to THE BUG from the monitor--WARMSTART  
 ^Q ..... Quit THE BUG  
 & ..... Return to THE BUG from Applesoft--WARMSTART

### KEY:

(opt add)--an optional Hexadecimal address between \$0-\$FFFF  
 (add)--a Hexadecimal address between \$0-\$FFFF  
 (val list)--a list of hexadecimal values between \$0-\$FFFF each value must be separated by either a space or a comma  
 (CR or period)--indicates that command must be terminated with either a CR or .  
 (reg)--either an A,X,Y,S,P  
 (#)--a hexadecimal value between \$1-\$8  
 (= <>)--one or any combination of the following symbols = <>  
 (Z1,Z0,V1,V0,C1,C0,D1,D0,I1,I0,B1,B0,N1,N0)--one or any combination of the bracketed values separated by commas  
 (opt cnt)--a hexadecimal value between \$1-\$FF  
 (val)--a hexadecimal number between \$0-\$FF

COPYRIGHT ©1981 by  
 SENSIBLE SOFTWARE, INC.  
 All Rights Reserved



**SENSIBLE SOFTWARE**

6619 Perham Drive  
 West Bloomfield, Michigan 48033  
 (313) 399-8877



04/10/82

#### THE BUG Version 4D.1

This addendum will describe the changes implemented with version 4D.1. The major enhancements of this revision are as follows:

- 1) THE BUG now has a checksum command (^K) that allows the user to insure that THE BUG's memory image is intact.
- 2) THE BUG has been implemented to optionally run in Bank 1 of the language card or RAM expansion card (which is usually unused).
- 3) THE BUG is now supplied as 6 standard unprotected DOS 3.3 files. This allows the user to back up THE BUG for archival purposes.
- 4) DOS commands may now be entered from THE BUG with the ^D command.
- 5) The display has been modified to display the ASCII value of memory locations accessed by THE BUG.
- 6) The ^Q command has been modified so that it no longer reboots the system. This command now resets the page 3 hooks to their previous values and jumps to the DOS coldstart location \$3D3.
- 7) A bug in the previous version was corrected so the display of certain swapped page 0 locations is now correct.

#### The New Disk Format

THE BUG is now supplied as six standard unprotected DOS 3.3 binary files. This was done to facilitate the use of THE BUG. THE BUG consist of the following files.

B 003 BUG.DEMO.OBJ  
B 006 THE BUG  
B 003 LCBUGLD.OBJ  
R 023 RBUG2  
R 003 LCBUG  
B 003 LCBUG2D

The files THE BUG and RBUG2 are always needed in order to run THE BUG. THE BUG is a program that loads RBUG2 into memory and then jumps to it. RBUG2 is the relocatable main code of THE BUG. BUG.DEMO.OBJ contains the demonstration program and is only needed if the demonstration program is run. LCBUGLD.OBJ is used to load RBUG2 on to the language card. LCBUG is needed if the bank switch buffer is to be located between \$1000-\$BFFF. LCBUG2 is needed if the bank switch buffer is located on page 2.

To execute THE BUG the following command is entered:

#### ] BRUN THE BUG

The standard display will be shown and the user will be asked to enter a page boundary in hex. If the user enters a carriage return THE BUG will be located below HIMEM and will reset HIMEM to protect itself. If the user enters a number, THE BUG will be relocated to that page of memory. THE BUG is \$DFF bytes long. If the user enters an "L", THE BUG will be loaded on to Bank 1 of the language card between \$D000-\$DDFF.

#### ENTER PAGE BOUNDARY IN HEX? L

This area of the language card is not used by either APPLESOFT or INTEGER BASIC as they use Bank 2 of the language card; therefore, the user may use either of these two programs without destroying THE BUG. A small program of \$FF bytes in length is located below HIMEM. HIMEM is reset to protect this program. This program is used by the "E", "M", "^D", and "^Q" commands to reset the language card so that the correct bank will be on when these commands are executed. The "^Y", "&", or "RESET" commands will return the user to THE BUG. Note that a jump to location \$D005 will not coldstart THE BUG. "CALL 1013" will return the user to THE BUG from INTEGER BASIC. Note that the language card version of THE BUG cannot be used to debug a program located between \$D000-\$DFFF. If the user desires to debug a program at these locations, he should load THE BUG in the lower 48K of memory. When THE BUG is located in the language card the "^W" and "^P" commands should not be used.

The \$FF byte bank switch program may be loaded anywhere in memory above \$1000, by entering the page address of the location it is to be loaded at along with an "L".

#### ENTER PAGE BOUNDARY IN HEX? 20L

This command will place THE BUG in the language card and place the bank switch program at \$2000-\$2070 in memory. HIMEM will not be altered.

The Bank Switch program may optionally be loaded in the upper part of the text buffer (\$287-\$2FB). This is usually a safe location of memory as only an input line of greater than 130 characters long will destroy this region. This mode is entered by typing "LX" to the opening question.

#### ENTER PAGE BOUNDARY IN HEX? LX

The user is now given the option of whether or not to load the demonstration program. This program will be loaded only if the user answers "Y" to the question "LOAD DEMO PROGRAM". If just a carriage return or a "N" is entered this program will not be loaded.

Note that only the file "THE BUG" may be run by the "BRUN" command. All the other files are called by this main loader program and will not run by themselves.

#### The New Display

In the current version of THE BUG an additional line has been added to the register display. This line displays the value of each bit in the status register and displays the address accessed by the last instruction executed and the contents of this address at the completion of the instruction. In addition, the display has been modified to display the ASCII representation of each accessed memory location.

\* D

PC=8942 A=42 X=00 Y=03 P=31 S=FF  
C=1 Z=0 I=0 D=0 B=1 V=0 S=0 (0876)=A2 "

In this example the status register values are displayed. In addition the memory location addressed by the last instruction executed is displayed. In this case the address was \$876. In this case the value at location \$876 was equal to \$A2 whose ASCII representation is a ". If the last instruction did not reference a memory location no address will be displayed. In other words an instruction such as

\* S

OB00- A2 14 LDX #14  
PC=OB02 A=0 X=14 Y=0 P=30 S=FF  
C=0 Z=0 I=0 D=0 B=1 V=0 S=0

will not produce an address display. If an address located between \$C000-\$C0FF is referenced, its contents will not be displayed. Since, in most instances accessing these addresses performs functions, it was felt best not to access them a second time. If an address in the stack is referenced (\$100-\$1FF) it will be modified to reference the location it was relocated to.

#### The S Command

The operation of the "S" command has been changed in this modification. If the S command is entered without operands it will still cause only the next instruction to be executed. If on the other hand, the optional count is specified its function differs. Unlike version 4B.2, in version 4D.1, execution initiated with the S command will be terminated by all the conditions that terminate execution initiated by the G command. In other words execution will be terminated by 1) hitting the escape key, 2) pushing the pushbutton on game paddle 0, or 3) satisfying either a memory or register breakpoint condition. If the count is a positive number (\$00-\$7F) this number will specify the number of instructions to be executed. If the count is a negative number (\$80-\$FF) execution will continue until the user



terminates execution with either the escape key or pushbutton. A breakpoint condition is satisfied. As before the values of the registers are displayed after each instruction executed in this mode.

#### The CTRL-K Command

Format--^K

This command checks the integrity of THE BUG to make sure that its memory image is still intact. This command compares the checksum of the current memory image with the one initially loaded into memory by the loader program. This command performs two types of checksums to increase the accuracy of this check. It is possible to alter the memory image so this algorithm will not detect an error but this is a very unlikely occurrence. If no error is detected, the message "Good" will be displayed. If an error is detected the program will halt and display the message "ERR CKSM". The user should reload THE BUG if this error occurs as the current memory image is unreliable. This command will only check the main program and not the small \$FF byte Bank Switch program located elsewhere in memory.

#### The CTRL-D Command

Format--^D (DOS command) CR

The user may now enter DOS commands without exiting THE BUG. The user may do this by typing a "^D" followed by a valid DOS command. This command string must be terminated with a carriage return.

#### \* ^D CATALOG

This command will print the catalog. After the catalog is printed, the user will be returned to THE BUG. If a DOS error occurs the user will be left in the currently active version of BASIC. If this happens, the user may return to THE BUG with the reset command or "R".

#### The CTRL-Q Command

Format--^Q

The ^Q command is still entered to exit THE BUG. This command no longer reboots the system. Instead the page 3 vectors are restored to what their values were before THE BUG was run and a DOS coldstart is performed.

#### CORRECTIONS TO CURRENT MANUAL

- Page 2-2 line 21 -- ARE NOT.D  
ignore "D" at the end of the line
- Page 3-2 line 8 -- 0A10- A0 45 FF 00 00 00 00D  
ignore "D" at the end of the line
- Page 4-2 line 22 -- H=<>J  
this should read {=<>}
- Page 10-5 line 29 -- | SAVE BUG.DEMO.I,ASB00,LSB  
should read | BSAVE BUG.DEMO.I,ASB00,LSB
- Page 10-14 line 14 -- \* LC24S  
should read \* LC24
- Page 10-14 line 16 -- \* \$FB1ED  
should read \* \$FB1E
- Page 10-14 line 24 -- \* LC24S  
should read \* LC24

#### THE BUG Addresses

Since the program has been modified the addresses found in appendix II of the original manual are no longer valid. Below is a new appendix II to replace the former.

# APPENDIX II--THE BUG LOCATIONS

| LOC  | NAME     | DISPL | HIMEM  | LOC | LCARD  | LOC |
|------|----------|-------|--------|-----|--------|-----|
| \$20 | WNDLEFT  | \$8A1 | \$90A1 |     | \$D8A1 |     |
| \$21 | WNDWDTH  | \$8A2 | \$90A2 |     | \$D8A2 |     |
| \$22 | WNDTOP   | \$8A3 | \$90A3 |     | \$D8A3 |     |
| \$23 | WNCBDM   | \$8A4 | \$90A4 |     | \$D8A4 |     |
| \$24 | CH       | \$8A5 | \$90A5 |     | \$D8A5 |     |
| \$25 | CV       | \$8A6 | \$90A6 |     | \$D8A6 |     |
| \$28 | BASL     | \$8A7 | \$90A7 |     | \$D8A7 |     |
| \$29 | BASH     | \$8A8 | \$90A8 |     | \$D8A8 |     |
| \$2A | BAS2L    | \$8A9 | \$90A9 |     | \$D8A9 |     |
| \$2B | BAS2H    | \$8AA | \$90AA |     | \$D8AA |     |
| \$2F | LENGTH   | ----  | -----  |     | -----  |     |
| \$33 | PROMPT   |       |        |     |        |     |
|      | (BUG'S)  | \$893 | \$9093 |     | \$D893 |     |
|      | (PROGS)  | \$894 | \$9094 |     | \$D894 |     |
| \$35 | ----     | \$000 | \$8800 |     | \$D000 |     |
| \$36 | CSWL     |       |        |     |        |     |
|      | (USER'S) | \$001 | \$8801 |     | \$D001 |     |
|      | (MON'S)  | \$8AB | \$90AB |     | \$D8AB |     |
| \$37 | CSWH     |       |        |     |        |     |
|      | (USER'S) | \$002 | \$8802 |     | \$D002 |     |
|      | (MON'S)  | \$8AC | \$90AC |     | \$D8AC |     |
| \$38 | KSWL     |       |        |     |        |     |
|      | (USER'S) | \$003 | \$8803 |     | \$D803 |     |
|      | (MON'S)  | \$8AD | \$90AD |     | \$D8AD |     |
| \$39 | KSWH     |       |        |     |        |     |
|      | (USER'S) | \$004 | \$8804 |     | \$D004 |     |
|      | (MON'S)  | \$8AE | \$90AE |     | \$D8AE |     |

# APPENDIX II -- CONTINUED

| LOC       | NAME        | DISPL | HIMEM      | LOC | LCARD  | LOC |
|-----------|-------------|-------|------------|-----|--------|-----|
| \$3A-\$3B | PCL-PCH     | ----  | -----      |     | -----  |     |
| \$3C      | A1L         | \$8D2 | \$90D2     |     | \$D8D2 |     |
| \$3D      | A1H         | \$8D3 | \$90D3     |     | \$D8D3 |     |
| \$3E      | A2L         | \$8D4 | \$90D4     |     | \$D8D4 |     |
| \$3F      | A2H         | \$8D5 | \$90D5     |     | \$D8D5 |     |
| \$40      | A3L         | \$8D6 | \$90D6     |     | \$D8D6 |     |
| \$41      | A3H         | \$8D7 | \$90D7     |     | \$D8D7 |     |
| \$42      | A4L         | \$8D8 | \$90D8     |     | \$D8D8 |     |
| \$43      | A4H         | \$8D9 | \$90D9     |     | \$D8D9 |     |
| \$44      | A5L         | \$8DA | \$90DA     |     | \$D8DA |     |
| \$45      | A5H         | \$8DB | \$90DB     |     | \$D8DB |     |
| \$4C-\$4D | HIMENI      | ----  | -----      |     | -----  |     |
| \$73-\$74 | HIMENA      | ----  | -----      |     | -----  |     |
| \$100     | STACK BEG   | \$D00 | \$9500     |     | \$DD00 |     |
| \$1FF     | STACK END   | \$DFF | \$95FF     |     | \$DDFF |     |
| \$200     | INPUT BUF   |       |            |     |        |     |
|           | (BEG)       | \$CC1 | \$94C1     |     | \$DCC1 |     |
|           | (END)       | \$CE9 | \$94E9     |     | \$DCE9 |     |
| ---       | COLDST ROUT |       |            |     |        |     |
|           | (NORM)      | \$005 | \$8805     |     |        |     |
|           | (LX)        | \$2B9 | (absolute) |     |        |     |
|           | (L)         | \$032 | \$9532     |     |        |     |
| ---       | WARMST ROUT |       |            |     |        |     |
|           | (NORM)      | \$07F | \$887F     |     |        |     |
|           | (LX)        | \$2C9 | (absolute) |     |        |     |
|           | (L)         | \$043 | \$9543     |     |        |     |