BREAKPOINT ROUTINE FOR 6502s

John Zeigler
8 Seaview Dr., Pittsburg CA 94565
(415) 894-3661

[This routine was distributed at the Homebrew Computer Club meeting, March 17, 1976. It is reprinted with the author's permission.]

This routine is entered via a software breakpoint. It is entered when the processor encounters a 00 op-code. Upon entering, the program counter is printed, followed by the active flags, accumulator, X index register, & index register, and stack pointer, terminated by a carriage return and line feed. It then waits for the user to type in a new op-code. Upon receiving that op-code, the original 00 code is replaced with the op-code that was input, the stack is returned to pre-interrupt status, and execution of the original program continues from the breakpoint.

To use this routine, it is necessary to load the interrupt vector, FFFE and FFFF, with 64 and 02, respectively, and place the 00 breakpoint op-code in the desired location. The following storage is required: 0000-0007, 0200-02E3, FFFE-FFFF. Note: This routine calls subroutines located in the TIM Monitor.

```
                     BUG PRØGRAM LISTING          VERSIØN 1

0200  85 07        NEG    STA 07        ;SAVE MØDIFIED P STATUS
0202  A9 4E               LDA #$4E      ;LØAD A WITH 'N'
0204  20 C6 72            JSR WRT       ;TYPE 'N'
0207  A5 07               LDA 07        ;RESTØRE MØDIFIED P
0209  4C 7F 02            JMP V         ;RETURN TØ PRØG. V
020C  85 07        ØVERFL STA 07        ;SAVE MØDIFIED P
020E  A9 56               LDA #$56      ;LØAD A WITH 'V'
0210  20 C6 72            JSR WRT       ;TYPE 'V'
0213  A5 07               LDA 07        ;RESTØRE MØDIFIED P
0215  4C 82 02            JMP B         ;RETURN TØ PRØG. B
0218  85 07        BRK    STA 07        ;SAVE MØDIFIED P  .
021A  A9 42               LDA #$42      ;LØAD A WITH 'B'
021C  20 C6 72            JSR WRT       ;TYPE 'B'
021F  A5 07               LDA 07        ;RESTØRE MØDIFIED P
0221  4C 86 02            JMP D         ;RETURN TØ PRØGRAM D
0224  85 07        DEC    STA 07        ;SAVE MØDIFIED P
0226  A9 44               LDA #$44      ;LØAD A WITH 'D'
0228  20 C6 72            JSR WRT       ;TYPE 'D'
022B  A5 07               LDA 07        ;RESTØRE MØDIFIED P
022D  4C 89 02            JMP I         ;RETURN TØ PRØGRAM I
0230  85 07        IRQDIS STA 07        ;SAVE MØDIFIED P
0232  A9 49               LDA #$49      ;LØAD A WITH 'I'
```

```
0234  20 C6 72          JSR WRT      ;TYPE 'I'
0237  A5 07             LDA 07       ;RESTØRE MØDIFIED P
0239  4C 8C 02          JMP Z        ;RETURN TØ PRØGRAM Z
023C  85 07      ZERØ   STA 07       ;SAVE MØDIFIED P
023E  A9 5A             LDA #$5A     ;LØAD A WITH 'Z'
0240  20 C6 72          JSR WRT      ;TYPE 'Z'
0243  A5 07             LDA 07       ;RESTØRE MØDIFIED P
0245  4C 8F 02          JMP C        ;RETURN TØ PRØGRAM C
0248  85 07      CARRY  STA 07       ;SAVE MØDIFIED P
024A  A9 43             LDA #$43     ;LØAD A WITH 'C'
024C  20 C6 72          JSR WRT      ;TYPE 'C'
024F  A5 07             LDA 07       ;RESTØRE MØDIFIED P
0251  4C 92 02          JMP CØNT     ;RETURN TØ PRØGRAM CØNT
0254  85 00             STA 00       ;SAVE A IN 00
0256  86 01             STX 01       ;SAVE X IN 01
0258  84 02             STY 02       ;SAVE Y IN 02
025A  68                PLA          ;PULL P ØT A
025B  85 03             STA 03       ;SAVE P IN 03
025D  68                PLA          ;PULL PCL TØ A
025E  85 04             STA 04       ;SAVE PCL IN 04
0260  68                PLA          ;PULL PCH TØ A
0261  85 05             STA 05       ;SAVE PCH IN 05
0263  BA                TSX          ;MØVE S TØ X
0264  86 06             STA 06       ;SAVE S IN 06
0266  D8                CLD          ;NØT DECIMAL MØDE
0267  20 8A 72          JSR CRLF     ;DØ A CRLF
026A  20 CF 02          JSR MØDPC    ;CØRRECT PCL & PCH
026D  A5 05             LDA 05       ;LØAD A WITH PCH
026F  20 B1 72          JSR WRØB     ;TYPE PCH IN HEX
0272  A5 04             LDA 04       ;LØAD A WITH PCL
0274  20 B1 72          JSR WRØB     ;TYPE PCL IN HEX
0277  20 77 73          JSR SPACE    ;SPACE 1 CHARACTER
027A  A5 03             LDA 03       ;LØAD A WITH P
027C  2A                RØL A        ;RØTATE N FLAG TØ CARRY
027D  B0 81             BCS NEG      ;BRANCH IF N FLAG SET
027F  2A         V      RØL A        ;RØTATE V FLAG TØ CARRY
0280  B0 8A             BCS ØVERFL   ;BRANCH IF V FLAG SET
0282  2A         B      RØL A        ;RØTATE PAST UNUSED BIT
0283  2A                RØL A        ;RØTATE B FLAG TØ CARRY
0284  B0 92             BCS BRK      ;BRANCH IF B FLAG SET
0286  2A         D      RØL A        ;RØTATE D FLAG TØ CARRY
0287  B0 9B             BCS DEC      ;BRANCH IF D FLAG SET
0289  2A         I      RØL A        ;RØTATE I FLAG TØ CARRY
```

```
028A  B0 A4               BCS IRQDIS     ;BRANCH IF I FLAG SET
028C  2A          Z       RØL A          ;RØTATE Z FLAG TØ CARRY
028D  B0 AD               BCS ZERØ       ;BRANCH IF Z FLAG SET
028F  2A          C       RØL A          ;RØTATE C FLAG TØ CARRY
0290  B0 B6               BCS CARRY      ;BRANCH IF C FLAG SET
0292  20 77 73    CØNT    JSR SPACE      ;SPACE 1 CHARACTER
0295  A5 00               LDA 00         ;GET A
0297  20 B1 72            JSR WRØB       ;TYPE A
029A  20 77 73            JSR SPACE      ;SPACE 1 CHARACTER
029D  A5 01               LDA 01         ;GET X
029F  20 B1 72            JSR WRØB       ;TYPE X
02A2  20 77 73            JSR SPACE      ;SPACE 1 CHARACTER
02A5  A5 02               LDA 02         ;GET Y
02A7  20 B1 72            JSR WRØB       ;TYPE Y
02AA  20 77 73            JSR SPACE      ;TYPE SPACE
02AD  A5 06               LDA 06         ;GET S
02AF  20 B1 72            JSR WRØB       ;TYPE S
02B2  20 8A 72            JSR CRLF       ;DØ A CRLF
02B5  20 B3 73            JSR RDHEX      ;READ VALID ØPCØDE
02B8  A2 00               LDX #$00       ;PREPARE TØ LØAD ØPCØDE
02BA  81 04               STA (04,X)     ;STØRE CØRRECT ØPCØDE
02BC  A6 06               LDX 06         ;GET S
02BE  9A                  TXS            ;RESTØRE STACK PØINTER
02BF  A5 05               LDA 05         ;GET PCH
02C1  48                  PHA            ;RESTØRE PCH TØ STACK
02C2  A5 04               LDA 04         ;GET PCL
02C4  48                  PHA            ;RESTØRE PCL TØ STACK
02C5  A5 03               LDA 03         ;GET P
02C7  48                  PHA            ;RESTØRE P TØ STACK
02C8  A4 02               LDY 02         ;RESTØRE Y
02CA  A6 01               LDX 01         ;RESTØRE X
02CC  A5 00               LDA 00         ;RESTØRE A
02CE  40                  RTI            ;RETURN TØ PRØGRAM
02CF  A5 04       MØDPC   LDA 04         ;LØAD PCL IN A
02D1  F0 07               BEQ ALTER1     ;BRANCH IF PCL = 0
02D3  C6 04       ALT1    DEC 04         ;SET PCL = PCL-1
02D5  F0 08               BEQ ALTER2     ;BRANCH IF PCL = 0
02D7  C6 04       ALT2    DEC 04         ;SET PCL = PCL-2
02D9  60                  RTS            ;RETURN FRØM SUBRØUTINE
02DA  C6 05       ALTER1  DEC 05         ;SET PCH = PCH-1
02DC  4C D3 02            JMP ALT1       ;JUMP TØ ALT1
02DF  C6 05       ALTER2  DEC 05         ;SET PCH = PCH-1
02E1  4C D7 02            JMP ALT2       ;JUMP TØ ALT2
                          END
```

000, the problem is probably that there was no memory at that address.

If the WRITE data is zero and the read data NOT 377, the problem probably is that the memory slice was protected.

Elsewise, you may "trap" a bad memory slice by zeroing the last four locations (after making a note of them!) and doing the following changes:

-Change the START data to be your LAST memory location (like 017-377 in a 4K memory)

-Change the END data to be the beginning memory address to be tested (as in 000-050)

-Change location 000-024 to a DCX H (053).

. . . And run the program again. It will now stop, hopefully returning a new value in the ERROR locations. From these two addresses (the old address pointer you wrote down from locations 046 & 047, plus the new address pointer currently there) indicating between what two addresses (inclusive) that bad memory was found, you may have an indication that one of your IC's was bad (for instance, one bit would never go off: WRITE 000, READ 001 – it will usually be detected on WRITE 000.) and the memory pointers' difference will most likely be 1024. The memory slice and bit that is bad will indicate, with the help of a schematic, a bad memory chip.

Richard A. Kaapke 4485 Vision Dr., Apt. 9
KLUGES, Inc. San Diego, CA 92121

(All rights released to *DDJ* from Kaapke's Little Used but Greatly Esoteric Software, Incorporated)

REPRINTED *ALPHANUMERIC MUSIC WITH AMPLITUDE CONTROL* INCLUDES CORRECTIONS FROM AUTHOR

Malcom Wright has sent in three letters containing corrections and additions to his booklet on Altair computer music generation [see *DDJ*, Vol. 1, No. 5, for notes on this publication]. These letters (dated April 13, May 17, and June 7) have been included in the most recent reprint of the booklet, available from the PCC Bookstore.

COMPUTING CAREERS FOR DEAF PEOPLE

Proceedings of the 1975 ACM Conference on Computing Careers for Deaf People have been published by the Association for Computing Machinery. The Conference, sponsored by the ACM Special Interest Group on Computers and the Physically Handicapped (SIGCAPH), was held last April in the Washington, D.C. area. It featured 30 presentations (including 11 by deaf professionals) covering such topics as educational opportunities, special training programs, placement problems and solutions, federal legislation, on-the-job problems and solutions, and success factors.

Industry and government employers will find that these proceedings provide sound input to their plans for compliance with the requirements of the Rehabilitation Act of 1973 (Public Law 93-112).

The proceedings contain 125 pages and cost $6.75 for ACM Members and $9.00 for non-members. They are available, prepaid, from:

ACM Order Department
P. O. Box 12105
Church Street Station
New York, NY 10249

ERRATA TO ZEIGLER'S 6502 "BUG PROGRAM"

The March issue of *Dr. Dobb's Journal* [Vol. 1, No. 3] contained a "Breakpoint Routine for 6502's" submitted by John Zeigler. The final paragraph of the documentary text contains an error. It *should* read:

". . . it is necessary to load the interrupt vector, FFFE and FFFF, with *54* amd 02, respectively . . . "

PROPOSAL FOR *HANDY* SOFTWARE, WITH EXAMPLE

A STRING OUTPUT SUBROUTINE FOR THE 6502

Dear Jim: August 10, 1976

I have been noticing that in the *Journal*, the main subject has been large programs (BASIC's, monitors, text editors). I agree there is a need for large programs such as these, but I believe you should also concentrate on HANDY (Helpful Algorithms for Novice Do-It-Yourselfers) programs to save bytes in space-limited systems. I enclose my example: a string immediate-output subroutine for 6502-based systems. This routine saves pointers, loops, etc. normally used for string output by sequentially outputting the ASCII characters represented in hex in the bytes immediately following the Jump to Subroutine. After reading a terminating character (null), it returns to the instruction following the end of the string. No string addresses or lengths are needed.

The subroutine uses 40 (hex) contiguous bytes for program and intermediate storage and 2 zero-page bytes for indirect addressing. Calling the routine affects none of the registers, nor the stack. It has been implimented on an Apple Computer and copies of the program are being delivered to the Homebrew library and the CCC repository.

Chris Espinosa

```
LOC.  OBJECT CODE   SOURCE STATEMENT

                    ;;; STRINGOUT: HANDY STRING OUTPUT ROUTINE
                    ;;; DEVELOPED FOR THE MOS 6502 BY C. ESPINOSA
                    ;;;   PUBLIC DOMAIN    9/11/1976
                    ;
                          ORG $400
                    AKEEP EQU $043D
                    YKEEP EQU $043E
                    KKEEP EQU $043F
                    OUT   EQU $EFFF
                    LO    EQU $FE
                    HI    EQU $FF
                    LO    DPZ
                    HI    DPZ
0400  8D  3D  04    BEGIN STA AKEEP
0403  8C  3E  04          STY YKEEP    SAVE REGISTERS
0406  68                  PLA          GET RETURN ADDRESS
0407  85  FE              STA LO
0409  68                  PLA
040A  85  FF              STA HI
040C  A0  01              LDY #1       SET UP INDEX
040E  B1  FE        NEXT  LDA (LO),Y     GET NEXT CHAR
0410  F0  07              BEQ EXIT       END IF 00
0412  C8                  INY
0413  20  FF  EF          JSR OUT     OUTPUT IT
0416  4C  0E  04          JMP NEXT
0419  8C  3F  04    EXIT  STY KKEEP
041C  A5  FE              LDA LO
041E  38                  SEC
041F  6D  3F  04          ADC KKEEP  ADD STR. LEN.
0422  85  FE              STA LO     TO RETURN ADDRESS
0424  A5  FF              LDA HI
0426  69  00              ADC #00    CARRY
0428  85  FF              STA HI
042A  AD  3D  04          LDA AKEEP  RESTORE REGS
042D  AC  3E  04          LDY YKEEP
0430  6C  FE  00          JMP (LO)   RETURN
```