Questdata
Volume 2 Issue # 9 ©

BLACKBOARD AND DOODLES

by
Jack Krammer

The simplicity and low cost of the COSMAC hardware to provide video graphics allows a means of showing off your computer, when asked the proverbial question. "But what does it do?" The other aspect of personal satisfaction is a demonstration in which the person who asked the question can get involved with the computer demonstration. To this end I wrote the following programs BLACKBOARD and DOODLES.

The BLACKBOARD program, when run, provides the player with a clean screen, and in the lower right hand corner of the display area a blinking cursor. The blinking cursor is used as chalk to write or draw images, to use as an eraser to allow selective removal of an image, or to allow movement of the "chalk" to another part of the display without leaving a trace.

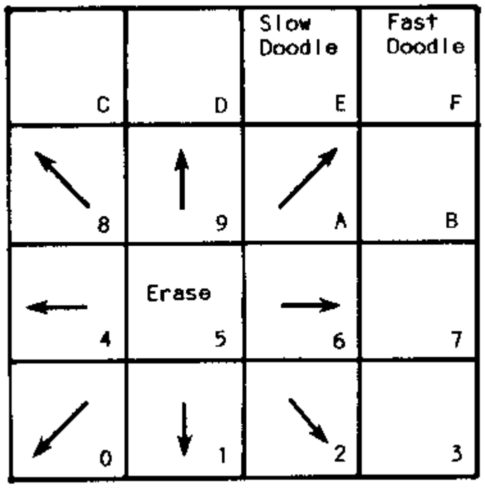

Figure 1 Elf II Keyboard

Motion of the blinking cursor is controlled by the data stored in the keyboard, and movement continues as long as the "INPUT" switch is pressed. If the high order bits in the keyboard re "5" the blinking cursor acts like an eraser, nd its movement leaves no trace. The low order bits, 0,1,2,4,6,8,9 and A, determine direction.

The program requires 1K of memory, and was written for the ELF II keyboard, as shown in Fig.1 which depicts the relationship between key and function. Fig.2 shows the SUPER ELF keyboard, and the necessary program changes to provide for logical chalk movement versus key placement. Motion can be in any of eight directions. When the chalk attempts to extend the limits of the display area it disappears and reappears at the opposite display edge as if the edges were actually wrapped around. Example; the chalk moving UP will disappear at the top and reappear at the bottom, on the same vertical line and continuing to move UP. What confused me during the program development was when the chalk moved left (or right), when it reappeared on the opposite side, it was not on the same horizontal line. This necessitated the subroutines left (or right) wrap around correction.

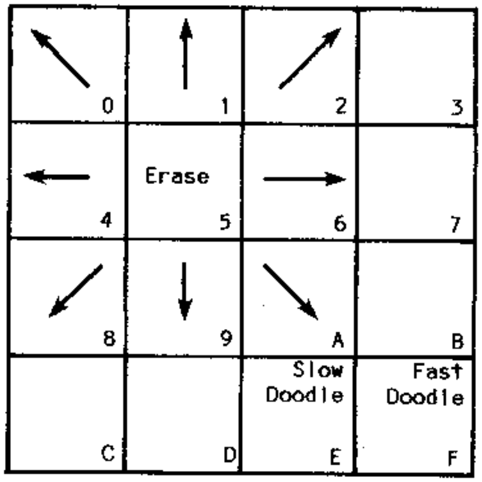

Figure 2 Super Elf Keyboard

Address	Was	Should be
0103	30	60
0107	40	48
010B	50	70
0117	60	30
011B	48	40
011F	70	50

Required Program Changes

The subroutine "Blinking Cursor" uses Wait For (DMA) Interrupt (00) op code to provide the delay required for blinking and also determines the rate of the cursor movement. To speed up the blinking and movement rate, put a short branch to jump over some of the interrupt code. This subroutine also reads the keyboard, and tests the high order bits to see if the number "5" was pressed for the "do not write" function.

To enter the Slow or Fast Doodle programs, press E or F respectively, and sit back and watch. Once in either doodle program there is no escape back to Blackboard or the other Doodle program. It is necessary to restart the program, but if a pattern is desired to be saved from Doodle for Backboard, then put 20 at address 0001 instead of 12.

During the Doodle programs it is possible to stop the action by pressing the "INPUT" switch. I recommend, for either Doodle program, a simple pattern be drawn in the lower right of the screen. After you are familiar with what is happening you can try expanding your experimentation.

Another change that is possible, is to draw against a white background. To accomplish this you change the 00 at 001A, in the "Clear Display" routine, to an FF. This will cause all display page memory locations, except 0300, to be loaded with all 1's which will provide a white screen. The missing 1's in 0300 can be filled in with the chalk. To write on this white board it is necessary to press "5", as the high order bits, and then a direction key. The trace will be black as it erases the white.

Registers Used:

```
X = 2 or 4 or 1 or 9
P = 3
0 = DMA Pointer
1 = Pointer for Clear Display/interrupt
2 = Stack Pointer
3 = Program counter
4 = Pointer to 02F0
8 = Old Display Pointer
9 = New Display Pointer
A = Seed Pointer
```

```
ADDR CODE    LABEL   OPCODE OPERAND COMMENT
0000 F8 12           LDI    12      (or 20 for no
                                     clear).
0002 A0              PLO    0       jump if P=0
0003 93              GHI    3       Initialize
0004 B0              PHI    0       RO.
0005 E3              SEX    3       Let X=P.
0006 70              RET            Disable interrupts
                                    and run.
0007 00              X=0,P=0        Data for above.
```

"Clear Display"

```
ADDR CODE    LABEL   OPCODE OPERAND COMMENT
0012 F8 03 B1                       03FF-->R(1) Top
                                    address of
0015 F8 FF A1                       Display page
0018 E1                             R(1) data pointer
0019 F8 00 73                       (Loop)00-->D-->
                                    M(R(1)); R(1)-1
001C 81                             R(1).0-->D
001D 3A 19                          Return to loop If
                                    D not equal to 0.
001F 73                             (D=0)-->M(R(1))
```

"INITIALIZE"

```
0020 F8 00 B1                       Required for dis-
                                    play refresh
0023 F8 50 A1                       Required for dis-
                                    play refresh
0026 F8 02 B4                       Required for read-
                                    Ing keyboard
0029 F8 F0 A4                       Required for read-
                                    ing keyboard
002C 30 30
002E C4 C4
0030 F8 00 B3                       R(3)=006C=Main
                                    Program Counter
0033 F8 6C A3                       R(3)=006C=Main
                                    Program Counter
0036 F8 03 B8 B9                    R(8)=03FF=Old dis-
                                    play location
003A F8 FF A8 A9                    R(9)=03FF=New dis-
                                    play location
003E F8 02 BA                       R(A)=02FF=(Temp)
                                    location of seed
0041 F8 FF AA                       R(A)=02FF=(Temp)
                                    location of seed
0044 F8 01 5A                       Seed 01-->D-->
                                    M(R(A))=02FF
0047 F8 01 B2                       R(2)=01FF=Top of
                                    Stack
004A F8 FF A2                       R(2)=01FF=Top of
                                    Stack
004D D3                             Program Counter=
                                    R(3)=006C
004E 72 70
0050 22 78
0052 22 52
0054 C4 C4 C4
0057 F8 03 B0                       R(0)=0300 Display
                                    Page
005A F8 00 A0                       R(0)=0300 Display
                                    Page
005D 80 E2
005F E2 20 A0
0062 E2 20 A0
0065 E2 20 A0
0068 3C 5D
006A 30 4E
```

"MAIN PROGRAM"

```
006C E2 69                          R(2)Data pointer,
                                    Turn on TV chip
006E 30 70                          Short branch to
                                    "Blinking Cursor"
```

```
"Blinking Cursor"  (R(8) old display location, R(9)
                   new display location, R(A)=02FF=
                   Seed bit storage)
0070 F8 03 B8 B9                    Reestablish 03 in
                                    R(8).1 and R(9).1
0074 E9 0A F1 59                    M(R(A))-->D or
                                    M(R(9))-->D-->
                                    M(R(9))
0078 89 A8                          R(9).0-->D-->
                                    R(8).0
```

```
ADDR CODE      LABEL   OPCODE OPERAND COMMENT
007A 00 00 00 00 00                   1/6 second delay
                                      interrupt
007F 00 00 00 00 00                   1/6 second delay
                                      interrupt
0084 FA 08 F7 59                      M(R(8))-->D-
                                      M(R(A))-->D-->
                                      M(R(9))
0088 00 00 00 00 00                   1/6 Second delay
                                      interrupt
008D 00 00 00 00 00                   1/6 Second delay
                                      interrupt
0092 E4 6C                            Keyboard to
                                      [M(R(4)) and] D
0094 FA F0                            D and F0-->D(Save
                                      Most significant
                                      4 bits)
0096 FF 50                            D-50-->D Test if
                                      "5" depressed
0098 32 A0                            Short branch to
                                      "Read Input"
009A E9 0A F1 59                      M(R(A))-->D or
                                      M(R(9))-->D-->
                                      M(R(9))
009E 30 A0                            Short branch to
                                      "read input"
"Read Input"
00A0 E4
00A1 3F 70                            "Input" SW
                                      depressed?  No, go
                                      to "Blinking
                                      Cursor"
00A3 6C                               Keyboard to
                                      [M(R(4)) and] D
00A4 C0 01 00                         Go to M(0100)
                                      "Test Data"

"Test Data" (Keyboard)

0100 FA 0F                            D and 0F-->D Save
                                      least significant
                                      4 bits
0102 32 30                            D=0(key 0 pressed)
                                      Go to "left down"
0104 FF 01                            D-01-->D
0106 32 40                            D=0(Key 1 pressed)
                                      Go to "DOWN"
0108 FF 01                            D-01-->D
010A 32 50                            D=0(Key 2 pressed)
                                      Go to "right down"
010C FF 02                            D-02-->D
010E 32 80                            D=0(Key 4 pressed)
                                      Go to "Left"
0110 FF 02                            D-02-->D
0112 32 A0                            D=0(Key 6 pressed)
                                      Go to "Right"
0114 FF 02                            D-02-->D
0116 32 60                            D=0(Key 8 pressed)
                                      Go to "Left Up"
0118 FF 01                            D-01-->D
011A 32 48                            D=0(Key 9 pressed)
                                      Go to "UP"
011C FF 01                            D-01-->D
011E 32 70                            D=0(key A pressed)
                                      Go to "Right Up"
0120 FF 04                            D-04-->D
0122 32 F0                            D=0(Key E pressed)
                                      Go to (long jump)
                                      slow doodle
0124 FF 01                            D-01-->D
0126 32 F3                            D=0(Key F pressed)
                                      Go to (long jump)
                                      fast doodle
0128 C0 00 70                         Long branch to
                                      "Blinking Cursor"
```

```
"LEFT DOWN"

ADDR CODE      LABEL   OPCODE OPERAND COMMENT
0130 0A                               M(R(A))-->D
0131 FF                               Shift D left,
                                      MSB(D)-->DF,
                                      0-->LSB
0132 3B 41                            DF=0?  Short
                                      branch to "DOWN"
0134 F8 01 5A                         (Seed) 01-->D-->
                                      M(R(A))
0137 88 FC 07 A9                      R(8).0-->D+07-->
                                      D-->R(9).0
013B 30 28                            Go to "Blinking
                                      Cursor" (Via
                                      M(0128))

"DOWN"

0140 0A                               M(R(A)-->D "Left
                                      down" or "right
                                      down" shifted
                                      input
0141 5A                               D-->M(R(A))
0142 88 FC 08 A9                      R(8).0-->D+08-->
                                      D-->R(9).0
0146 30 28                            Go to "Blinking
                                      Cursor" (Via
                                      M(0128))

"UP"

0148 0A                               M(R(A))-->D
0149 5A                               D-->M(R(A)) "Left
                                      up" or "right
                                      down" shifted
                                      input
014A 88 FF 08 A9                      R(8).0-->D-08-->
                                      D-->R(9).0
014E 30 28                            Go to "Blinking
                                      Cursor" (Via
                                      M(0128))

"RIGHT DOWN"

0150 0A                               M(R(A))-->D
0151 F6                               Shift D right
                                      LSB(D)-->DF, 0-->
                                      MSB(D)
0152 3B 41                            DF=0?  Short
                                      branch to "DOWN"
0154 F8 80 5A                         (Seed) 80-->D-->
                                      M(R(A))
0157 88 FC 09 A9                      R(8).0-->D+09-->
                                      D-->R(9).0
015B 30 28                            Go to "Blinking
                                      cursor" (via
                                      M(0128))

"LEFT UP"

0160 0A                               M(R(A))-->D
0161 FE                               Shift D left,
                                      MSB(D)-->DF,
                                      0-->MSB(D)
0162 3B 49                            DF=0?  Short
                                      branch to "UP"
0164 F8 01 5A                         (Seed)01-->D-->
                                      M(R(A))
0167 88 FF 09 A9                      R(8).0-->D-09-->
                                      D-->R(9).0
016B 30 28                            Go to "Blinking
                                      cursor" (Via
                                      M(0128)).
```

```
"RIGHT UP"

ADDR CODE     LABEL   OPCODE OPERAND COMMENT
0170 0A                              M(R(A))-->D
0171 F6                              Shift D right,
                                     LSB(D)-->DF, 0-->
                                     MSB(D)
0172 3B 49                           DF=0?  Short
                                     branch to "UP"
0174 F8 80 5A                        (Seed)80-->D-->
                                     M(R(A))
0177 88 FF 07 A9                     R(8).0-->D-07-->
                                     D-->R(9).0
017B 30 28                           Go to "Blinking
                                     Cursor" (Via
                                     M(0128))

"LEFT"

0180 0A                              M(R(A))-->D
0181 FE                              Shift D Left,
                                     MSB(D)-->DF,
                                     0-->LSB(D)
0182 33 87                           DF=1?
0184 5A                              D-->M(R(A))
0185 30 28                           Go to "Blinking
                                     cursor" (Via
                                     M(0128))
0187 F8 01 5A                        (Seed)01-->D-->
                                     M(R(A))
018A 88 FF 01 A9                     R(8).0-->D-01-->
                                     R(9).0
018E 30 B0                           To "Left Wrap
                                     around" correction
"RIGHT"

01A0 0A                              M(R(A))-->D
01A1 F6                              Shift D right,
                                     LSB(D)-->DF,
                                     0-->MSB(D)
01A2 33 A7                           DF=1?
01A4 5A                              D-->M(R(A))
01A5 30 28                           Go to "Blinking
                                     cursor" (via
                                     M(0128))
01A7 F8 80 5A                        (Seed)80-->D-->
                                     M(R(A))
01AA 88 FC 01 A9                     R(8).0-->D-01-->
                                     D-->R(9).0
01AE 30 D0                           To "Right wrap
                                     around" correction
"LEFT WRAP AROUND"

01B0 88 FA 0F 32 BF                  R(8).0-->D and
                                     0F-->D, Short
                                     branch if D=0
01B5 FF 08                           D-08-->D
01B7 32 BF                           Short branch if
                                     D=0
01B9 88 FF 01 A9                     R(B).0-->D-01-->
                                     D-->R(9).0
01BD 30 28                           Go to "Blinking
                                     cursor" (Via
                                     M(0128))
01BF 88 FC 07 A9                     R(8).0-->D+07-->
                                     D-->R(9).0
01C3 30 28                           Go to "Blinking
                                     Cursor" (Via
                                     M(0128))

"RIGHT WRAP AROUND"

01D0 88 FA 0F FF 07                  R(8).0-->D and 0F
                                     -->D-07-->D
01D5 32 E1                           Short branch if
                                     D=0
01D7 FF 08                           D-08-->D
01D9 32 E1                           Short branch if
                                     D=0
```

```
ADDR CODE     LABEL   OPCODE OPERAND COMMENT
01DB 88 FC 01 A9                     R(8).0-->D+01-->
                                     D-->R(9).0
01DF 30 28                           Go to "Blinking
                                     Cursor" (Via
                                     M(0128))
01E1 88 FF 07 A9                     R(8).0-->D-07-->
                                     D-->R(9).0
01E5 30 28                           Go to "Blinking
                                     Cursor" (Via
                                     M(0128))

(To Doodle Programs)

01F0 C0 02 20                        To slow doodle
01F3 C0 02 40                        To fast doodle

"SLOW DOODLE"

0220 F8 07 A8                        07-->D-->R(8).0
                                     Start at right top
                                     of display
0223 88       LOOP                   R(8).0-->D
0224 FF 09 A9                        D-09-->D-->R(9).0
0227 F8 03 B8 B9                     Reestablish high
                                     order address of
                                     display
022B 08 5A                           M(R(8))-->D-->
                                     M(R(A))Temporary
                                            Storage
022D 09 EA F3 59                     M(R(9))-->D x or
                                     M(R(A))-->D-->
                                     M(R(9))
0231 18                              R(8)+1
0232 00                              Wait for interrupt
                                     (Delay)
0233 3F 23                           "Input" SW not
                                     pressed cont. loop
0235 30 33                           Else wait here
                                     till "Input" SW
                                     released

"FAST DOODLE"

0240 F8 07 A8                        07-->D-->R(8).0
                                     Start at right
                                     top of display
0243 88       LOOP                   R(8).0-->D
0244 FF 09 A9                        D-09-->D-->
                                     R(9).0
0247 F8 03 B8 B9                     Reestablish high
                                     order address of
                                     display
024B 08 5A                           M(R(8))-->D-->
                                     M(R(A))Temporary
                                            Storage
024D 09 EA F3 59                     M(R(9))-->D x or
                                     M(R(A))-->D-->
                                     M(R(9))
0251 18                              R(8)+1
0252 88 FA FF                        R(8).0-->D and
                                     FF-->D
0255 3A 43                           Go to M(0243) if
                                     D not zero
0257 00 00 00 00 00                  Wait for interrupt
                                     1/3 second delay
025C 00 00 00 00 00                  Wait for interrupt
                                     1/3 second delay
0261 00 00 00 00 00                  Wait for interrupt
                                     1/3 second delay
0266 00 00 00 00 00                  Wait for interrupt
                                     1/3 second delay
026B 3F 43                           "Input" SW not
                                     pressed cont. loop
026D 00                              Else wait here
                                     till
026E 30 6B                           "Input" SW
                                     released
```

```
0000 F812 A093 B0E3 7000 0000 0000 0000 0000
0010 0000 F803 B1F8 FFA1 E1F8 0073 813A 1973
0020 F800 B1F8 50A1 F802 B4F8 F0A4 3030 C4C4
0030 F800 B3F8 6CA3 F803 B8B9 F8FF A8A9 F802
0040 BAF8 FFAA F801 5AF8 01B2 F8FF A2D3 7270
0050 2278 2252 C4C4 C4F8 03B0 F800 A080 E2E2
0060 20A0 E220 A0E2 20A0 3C5D 304E E269 3070
0070 F803 B8B9 E90A F159 89A8 0000 0000 0000
0080 0000 0000 EA08 F759 0000 0000 0000 0000
0090 0000 E46C FAF0 FF50 32A0 E90A F159 30A0
00A0 E43F 706C C001 0000 0000 0000 0000 0000
00B0 0000 0000 0000 0000 0000 0000 0000 0000
00C0 0000 0000 0000 0000 0000 0000 0000 0000
00D0 0000 0000 0000 0000 0000 0000 0000 0000
00E0 0000 0000 0000 0000 0000 0000 0000 0000
00F0 0000 0000 0000 0000 0000 0000 0000 0000
0100 FA0F 3260 FF01 3248 FF01 3270 FF02 3280
0110 FF02 32A0 FF02 3230 FF01 3240 FF01 3250
0120 FF04 32F0 FF01 32F3 C000 7000 0000 0000
```

```
0130 0AFE 3B41 F801 5A88 FC07 A930 2800 0000
0140 0A5A 88FC 08A9 3028 0A5A 88FF 08A9 3028
0150 0AF6 3B41 F880 5A88 FC09 A930 2800 0000
0160 0AFE 3B49 F801 5A88 FF09 A930 2800 0000
0170 0AF6 3B49 F880 5A88 FF07 A930 2800 0000
0180 0AFE 3387 5A30 28F8 015A 88FF 01A9 30B0
0190 0000 0000 0000 0000 0000 0000 0000 0000
01A0 0AF6 33A7 5A30 28F8 805A 88FC 01A9 30D0
01B0 88FA 0F32 BFFF 0832 BF88 FF01 A930 2888
01C0 FC07 A930 2800 0000 0000 0000 0000 0000
01D0 88FA 0FFF 0732 E1FF 0832 E188 FC01 A930
01E0 2888 FF07 A930 2800 0000 0000 0000 0000
01F0 C002 20C0 0240 0000 0000 0000 0000 A3FF
0200 0000 0000 0000 0000 0000 0000 0000 0000
0210 0000 0000 0000 0000 0000 0000 0000 0000
0220 F807 A888 FF09 A9F8 03B8 B908 5A09 EAF3
0230 5918 003F 2330 3300 0000 0000 0000 0000
0240 F807 A888 FF09 A9F8 03B8 B908 5A09 EAF3
0250 5918 88FA FF3A 4300 0000 0000 0000 0000
0260 0000 0000 0000 0000 0000 003F 4300 30FF
```

TINY BASIC SCREEN CLEAR

by

Paul W. Morris

The following program uses a list command in program sequence to simulate a screen clear command, which Tiny Basic doesn't currently have.

To accomplish this task, use a line number outside of the normal program line sequence (In the example, line #10000). Also added in the example program is a small delay loop (at line #5000) to give a person time to read the screen.

Now, any time in the program a person can jump to the delay loop, then list line #10000, which will clear the screen and continue with the next line number.

Enclosed as an example is Patrick Taylor's program "A Free Education on Slot Machines", taken from Questdata Vol. I, Issue 11, pg. 10.

The only problem encountered, is that when listing the entire program on a Video Monitor, line 10000 will clear the screen. Line 10000 has no effect on a TTY Listing (If the TTY has no fast forward capability). Otherwise, a page eject will occur on many other printers.

```
" A Free Education on Slot Machines"
      by Patick Taylor

5 GOSUB 5000
6 LIST 10000
10 PR "$1 SLOT MACHINE"
20 PR "PAYOFF IS $6 FOR"
30 PR "3 OF A KIND. ALL"
40 PR "OTHERS LOSE."
50 PR "NUMBER OF DOLS."
60 PR "TO START";
70 INPUT M
80 LET X=RND(345)
85 GOSUB 5000
86 LIST 10000
90 PR "DO YOU WISH TO "
92 PR "PLAY (1 YES, 0 NO)"
93 PR "YOU HAVE ";M;" DOLLAR(S)"
100 INPUT A
110 IF A=0 THEN GOTO 410
120 LET C=0
130 LET L=0
140 LET O=0
150 LET I=0
155 IF I=3 THEN GOTO 270
160 LET N=RND(3)+1
170 IF N=1 THEN GOTO 180
172 IF N=2 THEN GOTO 210
174 IF N=3 THEN GOTO 240
180 PR "*CH*";
190 LET C=C+1
200 GOTO 260
210 PR "*LM*";
220 LET L=L+1
230 GOTO 260
240 PR "*OR*";
250 LET O=O+1
260 LET I=I+1
265 GOTO 155
270 IF L=3 THEN GOTO 350
280 IF C=3 THEN GOTO 350
290 IF O=3 THEN GOTO 350
300 PR " TOO BAD, YOU LOSE"
310 LET M=M-1
320 PR
330 IF M=0 THEN GOTO 400
340 GOTO 380
350 PR "   YOU WON $6"
360 LET M=M+6
370 PR
380 GOSUB 5000
385 LIST 10000
390 GOTO 90
400 LIST 10000
401 PR "NO MORE MONEY"
410 PR "SORRY 'BOUT THAT"
420 END
5000 L=15
5001 L=L-1
5002 IF L>0 GOTO 5001
5003 RETURN
10000 CONTROL L
```

STORAGE TRANSFER

by
Dave Taylor

This program is very useful when temporary storage of programs is required to free a memory location for other programming, but cassette storage is not needed or desired. The storage portion of the program allows data to be moved to a new location while retaining the data at the original location. The transfer portion permits data from two memory locations to be exchanged, thus, several different programs can be loaded in the memory at different locations, but all with the same starting address (i.e. M 0000) and be recalled to that location anytime they are desired. As the data is moved from one location, the data stored at the second location is moved to the original location to keep it from being destroyed.

As a practical example, I have an ELF which has been modified to run COSMAC VIP programming which always start on page 0200 following an interpreter program. With the data transfer program, I am able to store several different VIP programs in open RAM locations and recall them to page 0200 while the original contents of page 0200 are stored at the "From" RAM location. In this manner, several different VIP programs can be called without having to manually reload the memory each time.

To use the program, enter the program at 0000, press and release reset and go and enter the control code to select the mode desired. A 01 control code will enable the memory storage portion of the program, while a 02 code will call the memory transfer (exchange) portion. This is followed by the High and Low starting addresses of the read "From" and read "To" memory locations. After entering the addresses, enter the address (low order only) of the last data byte to be read from the "From" location (maximum of FF for a one page read cycle). If an incorrect control code is entered, the Q light will come on for the duration of the delay subroutine, then turn off.

As a quick test, enter the program starting at M 0000, press and release reset and go and load the following information into the stack, pressing and releasing the input switch after each entry:

01	(selects the data storage mode)
00	(high order "From" address)
00	(low order "From" address)
01	(high order "To" address)
00	(low order "To" address)
62	(address of last "From" data byte to be stored)

When the input is released following the "Last Byte" entry, the program will begin to run. The first data byte located at the "From" address will be displayed then loaded to the first address of the "To" location. Both registers will be incremented, and the next byte fetched, displayed and loaded until the contents of the last "From" address are loaded to the appropriate "To" memory location. When the storage is completed, the Q light will come on and the program goes to a stop loop. At this time all data from M 0000 through M 0062 should have been loaded to M 0100 through M 0162.

After this is completed, the storage mode can be tested. Press and release reset and go and enter the following data, pressing the Input switch each time:

02	(selects the data transfer mode)
01	(High order "From" address)
00	(Low order "From" address)
00	(High order "To" address)
78	(Low order "To" address)
62	(address of last "From" data byte to be transferred)

When the input is released, the display will indicate the address location of each byte as it is exchanged. On completion, the Q light comes on and the program goes to the stop loop.

At this time, all data from M 0100 to M 0162 should have been loaded to M 0078 through M 00BA and the data formerly present at M 0078 through M 00BA should now appear at M0100 to M 0162.

The delay byte at M 0057 can be varied from 01 for a quick storage/transfer (approximately 2.5 seconds with a 3.579545/2 clock) to FF for a display slow enough to allow the bytes to be written down while the program is progressing.

Registers Used:

```
     X=2
     P=0
     0=PC
     1=Counter
     2=Stack Pointer
     3=Pointer
     4=Pointer
     5=Pointer
```

```
ADDR CODE           COMMENTS

0000 90 B1 B2 B3    Initialize registers to R0.1
0004 F8 06 A1       Set counter to read six bytes
0007 F8 6A A2       Stack
000A F8 51 A3       Points to location where "Last
                    Byte" entry is stored
000D E2             R 2=X
000E 22             Decrement R2
000F 3F 0F 37 11    Wait for input switch on/off
0013 6C 64 22       Read switches, load byte to
                    stack and display entry
0016 21             Decrement R1 (counter)
0017 81 3A 0E       Load counter to D, loop to M
                    (000E) if D not 00, decrement
                    stack and wait for next byte
                    until counter is 00
001A 72 53          Get first byte from stack
                    (last one entered), load to M
                    (R 3), Stack + 1
001C 72 A4          Get next byte from stack and
                    load to R4.0 (low order "To"
                    address) Stack + 1
001E 72 B4          Get next byte from stack and
                    load to R4.1 (high order "To"
                    address) Stack + 1
0020 72 A5          Get next byte from stack and
                    load to R5.0 (low order "From
                    address) Stack + 1
0022 72 B5          Get next byte from stack and
                    load to R5.1 (high order "From"
                    address) Stack + 1
0024 F8 4F A3       Points to subroutine for delay
                    and register increment
0027 72             Get last byte from stack (first
                    one entered) and load to D.
0028 FD 02          Subtract 02 from D
002A 32 3B          If D equals 00, branch to
                    transfer program, if not,
                    continue
002C FD 01          Subtract 01 from D
002E 32 33          If D now equals 00, branch to
                    storage program, if not, continue
0030 7B D3 00       Incorrect control code was
                    entered, turn Q on, branch to
                    subroutine to turn it off after
                    delay and stop
0033 E5             R 5=X (start of storage program)
0034 64 25          Get M (R X) and display, R X is
                    decremented to cancel increment
                    caused by "64" instruction
0036 F0 54          Load M (R X) to D, Load D to M
                    (R 4)
0038 D3             Call subroutine to delay display
                    byte and increment registers 4
                    and 5
0039 30 33          Return from subroutine, branch
                    to start of storage program and
                    continue load
003B E5             R 5=X (start of transfer program)
003C F0 A6          Load M (R X) to D, Load D R6.0
                    until data from R4 has been moved
003E 04 55          Load M (R 4) to D, load D to
                    M (R 5)
```

```
ADDR CODE           COMMENTS
0040 86 54          Load R6.0 to D, load D to M (R 4)
                    now transfer of one set of bytes
                    is complete
0042 D3             Call Subroutine
0043 E2 85 52       Return from subroutine, R 2=X,
                    R5.0 to D, D to M (R 2)
0046 64 22          Get M (R X) and display, R 2-1
0048 F0 FC 01 52    Get M (R X), load to D, add 01
                    to D, and store at M (R 2)
004C 30 3B          Branch to start of transfer
                    program, reset X to R 5 and
                    continue loading
004E D0             Exit fom subroutine
004F 85 FD xx       Load R5,0 to D and subtract from
                    xx (last data byte to be loaded)
0052 32 60          Branch to Stop Loop if D equals
                    00, if not, continue
0054 14 15          Increment R 4 and R 5
0056 F8 10 B1       Load Delay byte to R1.1
0059 21 91          Decrement R 2 and load it to D
005B 3A 59          If D not equal to 00, branch and
                    continue decrement
005D 7A             Delay loop complete, if Q is on,
                    turn it off
005E 30 4E          Branch to subroutine exit
0060 7B             Turn Q on
0061 30 61          Stop loop
0063 xx             Not used.  This is to allow use
                    of Long Branch instructions, etc
                    if stop loop is not desired.
0064 - 0069         Stack
006A xx             Work area for address counter
```

```
0000 90B1 B2B3 F806 A1F8 6AA2 F851 A3E2 223F
0010 0F37 116C 6422 2181 3A0E 7253 72A4 72B4
0020 72A5 72B5 F84F A372 FD02 323B FD01 3233
0030 7BD3 00E5 6425 F054 D330 33E5 F0A6 0455
0040 8654 D3E2 8552 6422 F0FC 0152 3038 D085
0050 FDFF 3260 1415 F810 B121 913A 597A 304E
0060 7B30 61
```

NOTE FROM EDITOR:

The use of Option 03 on the Super Monitor (Block Move) will accomplish the same task.

QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

Publisher ....Quest Electronics
Editor ....Paul Messinger
Proof Reading ....Judy Pitkin
Production ....John Larimer

QUESTDATA welcomes contributions from its readers. Manuscripts will be returned only when accompanied by a self addressed stamped envelope. Articles or programs submitted will appear with the authors name unless the contributor wishes otherwise. Payment is at the rate of $15 per published page. QUESTDATA exists for the purpose of exchanging information about the RCA 1802 microcomputer.

ADDRESS SEARCH

by
David Cartier

In the process of creating a floating point BASIC for my ELF computer I needed to search for addresses, etc. The one byte routine in my monitor was not adequate so I designed a two byte search routine. When placed at any point in memory it will search the whole memory for the bytes and give the high then the low address of the last byte. Since I store the bytes to be searched for at the end of the routine this gives an automatic default location for the search.

I included three options. Anything other than an FF or EE will continue the search so all locations where the bytes appear can easily be found. This is handy when you change a subroutine location using the general subroutine call/return in the 1802, for example, D41234 which is a call of a subroutine at 1234.

The next option is FF. This gets you out of the routine when the appropriate two byte address (your choice is put at location PP48 and PP49 where PP is the page number).

The last option is EE. This restarts the search.

The operation is as follows. Jump to the search routine start address. Input the high part of the address to start the search, then the low part. Next input the first search byte then the second. The ELF display will show the high part of the address where the bytes were found. Pushing input gives a display of the low part of the address.

```
            Registers Used:

                 X = 2
                 P = 0
                 0 = PC
                 2 = Storage Pointer
                 E = Used
                 F.0 = Used
```

```
ADDR CODE           COMMENT
0000 F8
0001 PP             Note(1)
0002 B2             Set Up
0003 F8             Storage
0004 51             Pointer
0005 A2
0006 F8             4 Time
0007 04             Loop
0008 AF
0009 E2
000A 7B
000B 3F             Wait for
000C 0B             Input
000D 7A
000E 37
000F 0E
0010 6C              Input
0011 64             Byte
0012 2F
0013 8F
0014 3A
0015 0A
0016 22             Put
0017 02             Bytes
0018 AF             In
0019 22             Registers
001A 02
001B BF
001C 22
001D 02
001E AE
001F 22
0020 02
0021 BE
0022 EE
0023 2E
0024 1E
0025 9F             Check
0026 F3             For a
0027 3A             match
0028 24
0029 1E
002A 8F
002B F3
002C 3A
002D 25
002E E2
002F 9E
0030 52             Output
0031 64             High part
0032 7B             of address
0033 3F
0034 33
0035 37
0036 35
0037 22
0038 8E
0039 52
003A 7B
003B 64             Output
003C 22             low part
003D 3F             of address
003E 3D
003F 7A
0040 37             Wait for
0041 40             input
0042 6C             input
0043 64             continue
0044 22             code
0045 FB
0046 FF             Note(2)
0047 C2
0048 HH
0049 LL
004A FB
```

```
ADDR CODE           COMMENT
004B 11             Restart
004C 32
004D 00
004E EE             Continue
004F 30             storage
0050 25
0051 X
0052 X
0053 X              Note(1)
0054 X              PP =
0055 X              Page Number
                    of
                    search
                    routine

                    Note(2)
                    user
                    address
                    HH =
                    High
                    Part
                    LL =
                    Low
                    Part
0070 E2             Debug
0071 22             routine
0072 52
0073 64
0074 7B
0075 3F
0076 75
0077 7B
0078 C0
0079 01
007A 0D
```

```
0000 F800 B2F8 51A2 F804 AFE2 7B3F 0B7A 370E
0010 6C64 2F8F 3A0A 2202 AF22 02BF 2202 AE22
0020 02BE EE2E 1E9F F33A 241E 8FF3 3A25 E29E
0030 5264 7B3F 3337 3522 8E52 7B64 223F 3D7A
0040 3740 6C64 22FB FFC2 0000 FB11 3200 EE30
0050 25B0 2020 2000 0000 0000 0000 0000 0000
0060 0000 0000 0000 0000 0000 0000 0000 0000
0070 E222 5264 7B3F 757B C001 0D
```

MODS TO BIORHYTHM

by
Gary Gehlhoff

Here is a modified version of the Biorhythm program by Gary Gehlhoff published in Questdata Volume 2, Issue 4. This version will allow you to list biorhythm data for any number of days you wish. The listing consists of the original Biorhythm program with the modifications implemented.

NOTE: In the first check example, published with the original program, the number of days should have read -38 instead of .37.

```
10 REM BIORHYTHM PROGRAM
20 PR "BIORHYTHM"
30 PR
40 PR "BIRTH DATE"
50 PR "-----MONTH-";
60 INPUT M
70 PR "-------DAY-";
80 INPUT D
90 PR "------YEAR-";
100 INPUT Y
110 PR
120 PR "TODAY'S DATE"
130 PR "-----MONTH-";
140 INPUT N
150 PR "--------DAY-";
160 INPUT E
170 PR "------YEAR-";
180 INPUT Z
185 PR "NUMBER OF DAYS TO BE CALCULATED";
187 INPUT G
188 G=G+1
190 L=(Z-Y)/4
200 B=(Z-Y)*365
210 U=0
220 F=0
230 W=0
240 O=M-N
250 REM
260 IF O>0 GOTO 431
270 IF M=N GOTO 415
280 GOSUB 1000
290 W=W+C
400 IF M=12 THEN M=0
405 M=M+1
410 GOTO 270
415 F=E-D
420 A=E+B+W+F
430 GOTO 595
431 N=N-1
432 M=M-1
433 IF N=0 THEN N=12
434 IF M=0 THEN M=12
440 IF M=N GOTO 590
450 GOSUB 1000
460 W=W-C
570 M=M-1
580 GOTO 434
590 F=E-D
595 PR "I","S","P","C"
600 A=L+B+W+F
610 P=A-A/23*23
620 S=A-A/28*28
630 I=A-A/33*33
640 REM - ASSUME AVERAGE PERSON
650 IF I<9 THEN J=I*3
660 IF I>8 IF I<25 THEN J=49-3*I
670 IF I>24 THEN J=3*I-99
680 IF S<8 THEN T=S*4
690 IF S>7 IF S<22 THEN T=56-4*S
700 IF S>21 THEN T=4*S-112
710 IF P<6 THEN Q=P*2
720 IF P>5 IF P<18 THEN Q=23-2*P
730 IF P>17 THEN Q=2*P-46
750 G=G-1
760 IF G=0 THEN END
770 A=A+1
900 PR J,T,Q,J+T+Q
940 GOTO 610
1000 C=31
1010 IF M=4 THEN C=30
1020 IF M=6 THEN C=30
1030 IF M=9 THEN C=30
1040 IF M=11 THEN C=30
1050 IF M=2 THEN C=28
1060 RETURN
```

AUTO TELEPHONE DIALER

by
Stephen Rarick

In this program the Q line is toggled on and off according to the values of telephone digits stored in memory (from location 80 on). The Q line can then be buffered and used to drive a relay connected serially across the red wire in the phone line, or a solenoid may be used to toggle the cradle switch on the phone. These methods have been described in detail in Radio-Electronics and other publications, so I will leave the reader to experiment here.

To use the program, load the numbers (via monitor, etc.) from locations 80 to C8. Example: For a number such as 664-9341 load (starting at location 80) 06 06 04 09 03 04 01 00 with 01 at 86 being the end, or last digit of the number. Location 87 is 00, to signal the end of data and therefore the dialing sequence. As written, the ten numbers may be less but not more than 7 digits. Every number must end with 00 in the last memory space. The number at the last location must be checked, because the number of digits is not counted, therefore it may keep reading data to the end of the program (or a 00 byte is read). I realize the program could store more numbers with a little more efficient use of memory, or a load sequence could be added directly in the program, but I'll leave that to someone else.

To use, clear memory, then load the program starting at location 0000. Store the numbers as described with number 1 at location 80, ending with number 10 at location C8. The delay values in the subroutines, at locations E1 and F1 respectively may be experimented with for best on and off times for dialing. Mine works with the values shown, but who knows?

After loading turn run on. Press button one thru nine to select stored telephone number. Press input button, digits stored will be shown on hex display as they are pulsing relay or solenoid. When finished press input again for same number, or press different 0-9 button for different telephone number. Don't forget protection diode across relay or solenoid to protect Q line and 1802.

```
Registers Used:

     X=3 or 4
     P=0
     0=Program Counter
     1=00E0 = Delay Subroutine
     2=00F0 = Long Delay Subroutine
     3=006F = Scratchpad
     4=0070 = Index Pointer
     5=Number
```

```
ADDR CODE             COMMENT
0000 90               Load 00
0001 B1               In registers
0002 B2               1,2,3,4,5
0003 B3
0004 B4
0005 B5
0006 F8               Initialize subroutine
0007 E0               delay locations, two
0008 A1               delays one slower than
0009 F8               the other.  Slower one provides
000A F0               break between digits
000B A2               dialed
000C F8               Scratch location for
000D 6F               digit 0-9 to call up
000E A3               number desired to call.
000F E3               X=3
0010 3F               Wait for input button
0011 10
0012 37
0013 12
0014 6C               Input numerical value from
0015 FA               keyboard (00=No.1, 09 = No. 10)
0016 0F               and against 0F, save low 4 bits
0017 F9               or against 70, gives 70 thru 79
0018 70               (other values non-valid) address
0019 A4               table of number locations
001A E4               X=4
001B F0               Load via X, put in low reg. 4
001C A4               now address of telephone data
001D F0               Load via X
001E A5               put in reg. 5 (low)
001F 64               output to display
0020 32               If data is 00, end of
0021 00               telephone number, start over
0022 7A               Q off
0023 D1               short delay subroutine
0024 7B               Q on
0025 D1               Short delay subroutine
0026 7A               Q off
0027 25               Decimal value in low register 5
0028 85               get low register 5
0029 3A               Q on off routine if
002A 23               data = telephone digit not 00
002B D2               Longer subroutine, end
002C 30               of one digit being dialed.
002D 1D               Back for next digit till done
002E 00               (Scratch at 6F, 2E till
                      6F unused)
0070 80               Pointer table
0071 88               to addresses of
0072 90               separate numbers to
0073 98               be dialed.  Memory
0074 A0               allocated spaced for
0075 A8               seven digits, plus
0076 B0               end signal (00)
0077 B8               could be expanded for
0078 C0               larger telephone numbers
0079 C8
007A 00               End of table
0080 - 00C8           Numbers stored here in form 01
                      to 0A.  A zero in telephone
                      dialing is actually 10 pulses,
                      hence use 0A Hex for a zero in
                      the telephone number.  Space for
                      ten 7 digit numbers plus stop
                      bytes.
```

```
ADDR CODE      COMMENT
00DF D0        Set PC to Reg. 0 (return to main)
00E0 F8        Shorter delay
00E1 08        Subroutine.
00E2 B7        Loops till register 7
00E3 27        (High) = 00 then
00E4 97        back to main
00E5 3A
00E6 E3
00E7 30
00E8 DF
00E9 00

00EF D0        PC back to Register 0
00F0 F8        Return to main program
00F1 30        Longer delay routine,
00F2 B7        Same operation as
00F3 27        first.  Used to
00F4 97        separate digit values
00F5 3A        being dialed.
00F6 F3
00F7 30
00F8 EF
00F9 00
```

```
0000 90B1 B2B3 B4B5 F8E0 A1F8 F0A2 F86F A3E3
0010 3F10 3712 6CFA 0FF9 70A4 E4F0 A4F0 A564
0020 3200 7AD1 7BD1 7A25 853A 23D2 301D 0000
0030 0000 0000 0000 0000 0000 0000 0000 0000
0040 0000 0000 0000 0000 0000 0000 0000 0000
0050 0000 0000 0000 0000 0000 0000 0000 0000
0060 0000 0000 0000 0000 0000 0000 0000 0000
0070 8088 9098 A0A8 B0B8 C0C8 0005 0604 0305
0080 0102 0304 0506 0700 010A 010A 010A 0100
0090 0000 0000 0000 0000 0000 0000 0000 0000
00A0 0000 0000 0000 0000 0000 0000 0000 0000
00B0 0000 0000 0000 0000 0000 0000 0000 0000
00C0 0000 0000 0000 0000 0000 0000 0000 0000
)0D0 0000 0000 0000 0000 0000 0000 0000 00D0
00E0 F808 B727 973A E330 DF00 0000 0000 00D0
00F0 F830 B727 973A F330 EF00
```

ANIMATION PROGRAM MODS

by
Jeff Jones

Here are a few suggestions for improving the animation program in Popular Electronics Magazine, July 1977 issue. First, by changing the contents of address 0043 to 11 you get an extra line to animate which results in a display location starting at 0078 and ending at 00FF.

Second, by reversing the starship pattern the ship flies forward instead of backward. To do this modification, implement the listing below.

```
ADDR CODE
0078 00 00 00 00 00 00 00 00
0080 07 E0 00 00 00 00 00 00
0088 FF FF FF FF 00 00 00 00
0090 80 00 00 00 80 00 60 00
0098 40 00 00 00 80 07 FE 00
00A0 7F FF FF 3F 07 FC 03 FE
00A8 00 00 01 20 08 00 F0 02
00B0 00 00 01 20 07 FC 03 FE
00B8 00 00 01 20 02 0B FC 00
00C0 00 00 01 20 04 10 F0 00
00C8 00 00 01 20 09 E0 00 00
00D0 00 00 0F 3F FE 18 00 00
00D8 00 00 08 00 00 0C 00 00
00E0 00 00 08 00 3F CE 00 00
00E8 00 00 0F FC 00 0C 00 00
00F0 00 00 00 03 F0 18 00 00
00F8 00 00 00 00 0F E0 00 00
```

PROGRAM 1 MONITOR

by
David Taylor

Program 1 is a modification to the PIXIE graphics main program which will permit the reading, writing or execution of a program at any memory location while being displayed on a video monitor. This is accomplished by using an expanded version of the Popular Electronics "ETOPS" system which provides for high and low byte memory addressing.

To examine a memory location, enter 01 and turn the Run switch on, 01 will be displayed. Enter the high order address, followed by the low order byte, depressing the input switch each time. When the input switch is pressed again, the memory byte at the selected location will be displayed. Continue to press the input switch to examine the remainder of the memory.

To change a memory location, enter 02, turn the run switch on and enter the high and low order address bytes, pressing the input switch each time. The "Q" light will come on after the low order byte is entered. Enter the new byte and press the input switch, the byte will be displayed and stored at the designated location. Entering 02, turning the run switch on, and then the address 00 1A will permit the display of any page of an expanded memory to be displayed on the video monitor when it's high order address is entered at this location.

To execute a program, enter 00, turn the run switch on, and enter the high and low order memory address of the selected program. Program execution will begin after the low order address is entered. Registers 1 thru 5 cannot be used on any executed program since these are required for the interrupt and expanded ETOPS control.

Register Used:

0=DMA
1=Interrupt Service
2=SP
3=PC
4=Scratch pad pointer
5=Input

```
ADDR CODE                          ADDR CODE
0000 90 B1 B2 B3                   0039 37 39
0004 B4 B5                         003B 32 61
0006 F8 2E A3                      003D F6 33 4A
0009 F8 6F A2                      0040 6C 64 B5
000C F8 12 A1                      0043 3F 43
000F D3                            0045 37 45
0010 72                            0047 7B
0011 70                            0048 30 51
0012 22 78                         004A 6C 64 B5
0014 22 52                         004D 3F 4D
0016 C4 C4 C4                      004F 37 4F
0019 F8 00 B0                      0051 6C 64 24 A4
001C F8 00 A0                      0055 95 B4
001F 80 E2                         0057 3F 57
0021 E2 20 A0                      0059 37 59
0024 E2 20 A0                      005B 39 5E
0027 E2 20 A0                      005D 6C
002A 3C 1F                         005E 64
002C 30 10                         005F 30 57
002E E2 69                         0061 6C 64 B5
0030 F8 6C A4                      0064 3F 64
0033 E4                            0066 37 66
0034 6C 64 24                      0068 6C 25 A5
0037 3F 37                         006B D5
                                   006C 00

0000 90B1 B2B3 B49E F82E A3F8 6FA2 F812 A1D3
0010 7270 2278 2252 C4C4 C4F8 00B0 F800 A080
0020 E2E2 20A0 E220 A0E2 20A0 3C1F 3010 E269
0030 F86C A4E4 6C64 243F 3737 3932 61F6 334A
0040 6C64 B53F 4337 457B 3051 6C64 B53F 4D37
0050 4F6C 6424 A495 B43F 5737 5939 5E6C 6430
0060 576C 64B5 3F64 3766 6C25 A5D5 00
```

21 QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

[redacted]