Questdata
Volume 2 Issue #5 ©

CRT CONTROLLER TAMES 1861 VIDEO

By Stephen P. Clark

A CRT controller (CRTC) is a mechanism used to place characters on a display screen. Computer terminals, television typewriters (TVTs), and most specialized CRT devices employ some type of CRTC to produce messages on a screen.

This article describes two software routines which provide 1802/1861 microprocessor systems with the capabilities of a CRTC, including cursor control, line scrolling, reverse video, and character generation. The software occupies two pages of memory (including screen refresh routine), and displays four pages (1K bytes). It produces 21 lines of 16 characters on the screen. Hardware requirements are:

1. 1802 system with 1861 video display chip
2. 1 3/4 K of RAM (7 pages)

The routines may be called from application programs, may be combined with an input routine (provided below), or may be used to print prestored messages.

Functions of a CIRC

A CRTC must do three things:

1. Produce a video signal.
2. Generate characters.
3. Control the cursor.

The video signal must be produced by hardware (the 1861 chip in Elf systems), but the other two functions may be provided by hardware, software, or some combination. Single chip (hardware) CRTCs are available today, and have several advantages over the software approach: they leave the CPU free for other functions, they generally give higher resolution character sets, and they are easier to use with application programs. However, some hardware display systems do not allow for graphics, have fixed character sets, and completely isolate the display from the application program. Before purchasing a video display system (video board), it is important to decide what functions it will be required to perform. If character printing will be the main application, a serial interface to an RS-232 terminal will be sufficient. If graphics capability is required, you will need a bit-mapped display, or some other arrangement which gives access to the character set and/or display memory.

The routines given here simulate the operation of a terminal with the 1802/1861 hardware. Each routine will be discussed below, along with its function in a CRTC.

Character Generator, CGEN

The second function of a CRTC, character generation, is performed by the routine CGEN using the character pattern table CTABLE. Each character in CTABLE occupies three bytes. Each byte of the pattern is split into two four-bit halves and stacked in the display memory area, left half over right half. Characters are thus four bits wide and six bits high. Some clarity is lost with the four-bit width, since spacing must be included, but it is not that noticeable.

The ASCII characters space (hex 20) through upper-case Z (hex 5A) are included in CTABLE, three bytes each, thus CTABLE is 177 (decimal) bytes long. As an example, the character A (upper-case A) was designed in the following manner:

```
 .   0100 4
. .  1010 A byte 1 = 4A
...  1110 E
. .  1010 A byte 2 = EA
. .  1010 A
     0000 0 byte 3 = A0
```

Both CTABLE and CGEN reside on RAM page 2, and together they occupy the entire page, addresses

0200-02FF. The details of CTABLE may be obtained by following the reverse of the above procedure. When CGEN is entered, registers RC-RF have been set up by the calling routine. CGEN operates in the following manner (refer to the listing for CGEN):

CGEN-CONT: Sets up a mask of F0 or 0F to clear the display memory, depending on the switch in RF high. Stores the mask in WORK! (work area in the calling program; see CRTC 1.5 discussion).

NEXTC: Gets the next byte from CTABLE, breaks it into two four-bit patterns, left justifies them, and stores them in WORK2 and WORK3.

RJUST: If switch (RF high) is not zero, right justifies WORK2 and WORK3.

DISP: Takes a byte from the display area, ANDs it with WORK1 to clear it, ORs it with WORK2 to put the pattern in, then replaces it in the display. Then moves down eight bytes on the display and does the same operations with WORK3. Thus, two half-bytes are placed on the screen.

CHEK: Checks loop counter in RF low. If zero, exit. If not zero, move down eight bytes on the display and go to NEXTC.

CGEN must be called with the MARK/SEP method. Registers RC through RF are destroyed by CGEN (see further details below).

Cursor Control, CRTC 1.5

The cursor is more than a character on the screen (see reference 2). It is a whole subsystem for determining what happens on the display. The routine CRTC 1.5 performs the following actions listed as essential for cursor operation in reference 2:

1. Shows the operator the next position on the screen.

2. Updates and enters a new character when it arrives.

3. Advances one character after update.

4. Returns to home (upper left screen).

5. Returns to beginning of next line (carriage return).

6. Erases the display (fills with blanks).

[illegible] also performs these operations, listed as optional by reference 2:

7. Backspace.

8. Scrolls upwards.

9. Emphasizes parts of the display (reverse video). Refer to the CRTC 1.5 listing for the labels used in the detailed discussion below:

CRTC: Loads cursor locations from storage at HDISPL, LDISPL and SWITCH, and sets up registers RB through RF. No special setup is required for calling CRTC 1.5. It expects the character to be in the accumulator (D register) upon entry, and should be called with the MARK/SEP technique.

CKSP: Checks the character for one of the special characters:

CR(0D)-carriage return/line feed

DEL(7F)-clears the screen

ESC(1B)-reverse video

BS(08)-backspace, control-H

These characters were selected for ease of use with a particular ASCII keyboard, and can be easily changed.

CAP: Converts lower case to upper case.

VALID: Computes CTABLE address for valid characters. The formula is: (character minus 20 hex) times three. This is done by storing the subtraction in WORK1, shifting left (multiply by two) and adding the original. Then call CGEN. (Note: Some invalid characters will sneak by.)

RHERE: Return from CGEN call. Re-establishes registers, then begins cursor update.

UPSW-NUBYT: Updates the left/right switch, and checks to see if a new display byte is needed for the next cursor position. If a new display byte is needed, it also checks for end of line. Since all lines end in 7 or F, the last three bits may be examined for this.

CRET: Performs carriage return/line feed. The line feed is done first by adding 30 hex (six lines), then the return is accomplished by zeroing the lower three bits (all lines begin with 0 or 8 in the lower half-byte).

CURSET: Check for cursor off the screen. If too low, go to HOME. If not too high, go to RPLCUR. If too high, pass through SCROLL.

SCROLL: Moves the entire screen up 30 bytes (hex), clears the bottom line (see CLEAR), and sets the cursor on the bottom line at SETL. (Note: The routine does not scroll until there is a attempt to place the cursor on page seven.)

RPLCUR: Replaces the cursor and left/right switch at HDISPL, LDISPL, and SWITCH.

WRCUR: If CURON is not zero, this section writes the cursor on the screen. The @ (at sign, hex 40) in CTABLE locations 0260-0263 is used. It is set as an underline.

CLEAR: Clears the screen and homes the cursor. The screen is cleared by taking the first byte of the space from CTABLE, location 0200, and writing it over the entire display area. If you are using reverse video, the screen will be all white (see REVRS).

HOME: Sets the cursor to the top of the screen, location 0310 with SWITCH set to zero. There are 128 rows of dots on the screen, and we use 6 for each line of characters, giving 21 lines of characters with 2 rows of dots left over. These two rows are at the top, 0300-0307 and 0308-030F.

BKSP: Backspaces by resetting the switch and decrementing the cursor display location if necessary. This will not backspace from the beginning of a line. Control-H (hex 08) is used for backspace.

REVRS: Gives reverse video. This is accomplished by reversing the character set in CTABLE, rather than the display. This was done so the system can be reversed at any time, allowing individual lines or characters to be reversed for highlighting. Typing ESC (escape, hex 1B) will reverse the character set from whatever it was before, so once the screen is reversed, to set it back type ESC again. If you alternately reverse and clear ESC, DEL, ESC, DEL), you will see the screen go alternately light and dark.

CURON: Cursor on/off. This should be set when the program is entered, or by the monitor. If zero, no cursor will be written but all other functions will be the same.

HDISPL: Storage for the high byte of the current cursor (display) address. Should start at 03.

LDISPL: Low byte of the cursor address. Should start at 10.

SWITCH: Left/right cursor address. Determines which half byte of the cursor location to write to.

WORK1-WORK3:Work area used by the routines.

The routine CRTC 1.5 is located at the beginning of page one, locations 0100-01EE. It must be called with the MARK/SEP technique, with its address in one of the registers R3-RA inclusive, and the character in the accumulator (D register) in ASCII hex representation. The registers RB-RF will be destroyed after CRTC 1.5 and CGEN finish, so the contents of these in the calling program should be saved and restored if needed after the call. Between calls, however, RB-RF may be used for any purpose. Since scratch registers are often required, I suggest RB-RF be used for temporary storage, and R3-RA for (relatively) permanent values such as subroutine address, data pointers, etc. CRTC 1.5 initializes the registers RB-RF each time, and need not be saved by the calling program.

Display Refresh Routine, REFR

The screen is refreshed by the routine REFR. It is located on page one after CRTC 1.5, at addresses 01EF-01FF. This is the standard 1 K byte display routine for 1802/1861 systems; it requires that R2 point to a stack location, and R1 must contain the REFR address, 01EF. These must be set up by the user's program (see below). REFR sets R0 to point to 0300, the beginning of the display area.

Changing the Routines

These routines are designed to reside on pages one and two, and to display pages three through six. To move them, changes must be made to CRTC 1.5 and REFR. These locations require changing in CRTC 1.5:

Page where CRTC 1.5 resided: 0105

Page where CGEN/CTABLE are: 0112

Beginning of display: 016B,0174,01B6,01C7

End of display: 0170,0182,0187,0191,01C3

In REFR, change location F9 to beginning display page. Alternate character sets may be placed in CTABLE. CGEN is "pure" subroutine (does not modify itself), and no changes are required to move it.

If other special characters are desired, you will need to make room on the CRTC 1.5 page (page one) for the code. The REFR routine could be moved if desired.

Using the System

The keyboard read routine listed below demonstrates use of the system. My system has an ASCII keyboard at parallel input port 7 (6F instruction) attached to the EF3 line. To use the hex keyboard, change locations:

19 from 3E to 3F
1B from 6F to 6C
21 from 36 to 37

You may then type the ASCII code on the hex keyboard and press the input switch.

The instructions at 0000-0018 are used to set up the registers for display:

R0 = used by screen refresh
R1 = screen refresh routine, 01F1
R2 = stack pointer
R3 = main program counter (arbitrary)
RA = CRTC 1.5 address, 0103 (arbitrary)

Register RB-RF will be destroyed after calling CRTC 1.5, but may be used between calls, or saved and restored.

. Since the display refresh routine is located with CRTC 1.5, it need not be entered with the main program. The system should be called with the instructions at 001E-0020:

```
MARK
SEP X     Call
DEC 2     Return here from call
```

where X is some register R3-RA.

After entering the main program, type several characters in order to become familiar with the character set. Try alternating reverse and clear, ESC, DEL, ESC, DEL. You can draw block pictures by spacing over, reversing, spacing some more, then reversing again, and spacing to end of line.

Having 21 lines of 16 characters means you can get enough text on the screen to display several messages, or several names and addresses.

References

1. Haas, Bob, "Single Chip Video Controller," Byte, May, 1979, page 52.

2. Lancaster, Don, TV TYPEWRITER COOKBOOK, Howard W. Sams and Co., 1976, pages 104-105.

Registers Used

	Keyboard Read	CRTC 1.5	REFR	CGEN
X	2	E	2	E
P	3	A	1	B
0		DMA		
1	Interrupt	PC		
2	Stack	Stack		
3	PC			
A	Addr. of CRTC 1.5	PC		
B		CGEN		PC
C		Character		Character
D		Cursor Pointer		Pointer
E		Pointer		Pointer
F		Data		Data

Keyboard Read

This program reads an ASCII keyboard. See text for details.

```
ADDR CODE   LABEL   OPCODE OPERAND COMMENT

0000 3003   BEGIN   BR     SETUP   ..........First 3
                                   bytes open
0002 C4             NOP
0003 90     SETUP   GHI    0       Set up page pointers
0004 B2             PHI    2       Stack on page 0
0005 B3             PHI    3       MAIN on page 0
0006 F801           LDI    #01
0008 B1             PHI    1       REFR on page 1
0009 BA             PHI    A       CRTC 1.5 on page 1
000A F8F1           LDI    #REFR
000C A1             PLO    1       R1=REFR address
000D F8FE           LDI    #STACK
000F A2             PLO    2       R2=STACK
0010 F817           LDI    #MAIN
0012 A3             PLO    3       R3=MAIN address
0013 F803           LDI    #CRTC
0015 AA             PLO    A       RA=CRTC 1.5
                                   address
0016 D3             SEP    3       Change prog.
                                   counter
0017 E2     MAIN    SEX    2       Set x=R2
0018 69             INP    1       Turn TV on
0019 3D19   WAIT1   BN2    WAIT1   Wait for keyboard
001B 6D             INP    5       Read ASCII keyboard
001C 64             OUT    4       Send to LEDs
001D 22             DEC    2       Reset stack for
                                   OUT 4*
001E 79             MARK
001F DA             SEP    A       Call CRTC 1.5
0020 22             DEC    2       Return, decrement
                                   stack
0021 3521   WAIT2   B2     WAIT2   Wait for keyboard up
0023 3019           BR     WAIT1   Go get another input
```

*The OUT 4/DEC 2 send the input character to the hex display LEDs. This is for demonstration only, and not required by the CRTC 1.5 system. Remove these two instructions for most applications.

CRTC 1.5

Following is a listing of the subroutine CRTC 1.5. This routine generates the cursor, performs character conversion, and handles special characters. It' resides at the beginning of page one.

```
ADDR CODE   LABEL   OPCODE OPERAND COMMENT
0100 E2     EXITT   SEX    2       Return
0101 12             INC    2         to
0102 70             RET            Caller
0103 AC     CRTC    PLO    C       Save character
0104 F801           LDI    #01     Current page
0106 BE             PHI    E         to RE
0107 F8E9           LDI    #HDISPL Cursor address
                                   pointer
0109 AE             PLO    E         to RE
010A EE             SEX    E       Set X=RE
010B 72             LDXA           Cursor location
010C BD             PHI    D         to RD
010D 72             LDXA
010E AD             PLO    D
010F 72             LDXA           Switch (left/right)
0110 BF             PHI    F         to RF
0111 F802           LDI    #02     Character page
0113 BB             PHI    B         to RB
0114 BC             PHI    C         and RC
0115 F8B4           LDI    #CGEN   CGEN address
0117 AB             PLO    B         to RB
0118 8C     CKSP    GLO    C       Get saved character
0119 FB0D           XRI    #0D     Check for CR (return)
011B 325A           BZ     CRET
011D 8C             GLO    C
011E FB7F           XRI    #7F     Check for DEL (clear)
0120 32B5           BZ     Clear
0122 8C             GLO    C
0123 FB1B           XRI    #1B     Check for ESC
                                   (reverse)
0125 32D9           BZ     REVRS
0127 8C             GLO    C
0128 FB08           XRI    #08     Check for BS
                                   (backspace)
012A 32CD           BZ     BKSP
012C 8C             GLO    C
012D FD5A           SDI    #5A     5A-char. (Z)
012F 3335           BPZ    CAP     Already upper case,
                                   branch
0131 8C             GLO    C
0132 FF20           SMI    #20     Make upper case
0134 AC             PLO    C
0135 8C     CAP     GLO    C
0136 FF20           SMI    #20     Char.-20 (space)
0138 3B00           BM     EXITT   Invalid, go exit
013A 5E     VALID   STR    E       Save in WORK1
013B FE             SHL            Multiply times 2
013C F4             ADD            Plus original
                                   (= 3 times)
013D AC             PLO    C       Table address to RC
013E F803           LDI    #03     Loop counter
0140 AF             PLO    F         to RF
0141 79             MARK
0142 DB             SEP    B       Call CGEN
0143 22     RHERE   DEC    2       Return here
0144 F8E9           LDI    #HDISPL Reload RD, RE, RF
0146 AE             PLO    E
0147 72             LDXA           Reload cursor
0148 BD             PHI    D
0149 72             LDXA
014A AD             PLO    D
014B F0             LDX            Switch
014C FB01   UPSW    XRI    #01     Reset switch
014E 3A68           BNZ    CSM1    Same byte, go set
                                   cursor
```

```
ADDR CODE   LABEL   OPCODE OPERAND COMMENT
0150 8D     NUBYT   GLO    D       New byte
0151 FA07           ANI    #07     Take low three
                                   bits
0153 FB07           XRI    #07     Check for 111
0155 325A           BZ     CRET    Yes, do carriage
                                   return
0157 1D             INC    D       No, increment
                                   display
0158 3066           BR     CSM3    Go set cursor
015A 8D     CRET    GLO    D       Carriage return
015B FC30           ADI    #30     Next line, add 30
                                   hex
015D AD             PLO    D       for line feed
015E 9D             GHI    D
015F 7C00           ADCI   #00
0161 BD             PHI    D
0162 8D             GLO    D
0163 FAF8           ANI    #F8     Remove low 3 bits
0165 AD             PLO    D
0166 F800   CSM3    LDI    #00     Clear the switch
0168 BF     CSM1    PHI    F
0169 9D     CURSET  GHI    D       Set cursor
016A FF03           SMI    #03     Page-3
016C 3BC6           BM     HOME    Too low, go home
016E 9D             GHI    D
016F FD06           SDI    #06     6-page
0171 3399           BPZ    RPLCUR  On screen, go
                                   replace cursor
0173 F803   SCROLL  LDI    #03     Scroll up
0175 BD             PHI    D
0176 BF             PHI    F
0177 F810           LDI    #10     First line starts
0179 AD     SLIN2   PLO    D
017A FC30           ADI    #30     One line below
                                   first
017C AF             PLO    F       (RF greater than
                                   RD)
017D 4F     LOOP1   LDA    F       Load from RF,
                                   advance
017E 5D     LOOP2   STR    D       Replace to RD
017F 1D             INC    D       Advance RD
0180 9F             GHI    F       Check RF for end
0181 FD06           SDI    #06     6 - page
0183 337D           BPZ    LOOP1   Repeat if on screen
0185 9D             GHI    D       Check RD for end
0186 FD06           SDI    #06     6 - page
0188 3B90           BM     SETL    End of screen
018A F800           LDI    #00
018C AC             PLO    C
018D 0C             LDN    C       First byte of space
018E 307E           BR     LOOP2   Repeat, clear
                                   bottom
0190 F806   SETL    LDI    #06     Set bottom line
0192 BD             PHI    D
0193 F8D0           LDI    #D0
0195 AD     RPCM4   PLO    D
0196 F800   RPCM3   LDI    #00     Clear switch
0198 BF             PHI    F
0199 F8EB   RPLCUR  LDI    #SWITCH Replace cursor/
                                   switch
019B AE             PLO    E
019C 9F             GHI    F       Get switch in RF
019D 73             STXD           Replace switch in
                                   memory
019E 8D             GLO    D
019F 73             STXD           Replace cursor
01A0 9D             GHI    D
01A1 73             STXD
01A2 F0             LDX            Load CURON
01A3 3200           BZ     EXITT   Cursor off, exit
01A5 F860   WRCUR   LDI    #60     Cursor character
01A7 AC             PLO    C
01A8 F803           LDI    #03     Loop count
```

```
ADDR CODE    LABEL   OPCODE OPERAND COMMENT
01AA AF              PLO    F
01AB 9B              GHI    B
01AC BC              PHI    C       CGEN page
01AD F8EC            LDI    #WORK1  Point to WORK1
01AF AE              PLO    E
01B0 79              MARK
01B1 DB              SEP    B       Call CGEN for
                                    cursor
01B2 22      RHER2   DEC    2       Return here
01B3 3000            BR     EXITT   Exit
01B5 F803    CLEAR   LDI    #03     Clear screen
01B7 BD              PHI    D
01B8 F800            LDI    #00
01BA AD              PLO    D
01BB AC              PLO    C
01BC 0C              LDN    C       Load 1st byte of
                                    space
01BD AF              PLO    F       Save in RF
01BE 8F      LOOPC   GLO    F
01BF 5D              STR    D
01C0 1D              INC    D
01C1 9D              GHI    D
01C2 FD06            SDI    #06     6 - page
01C4 33BE            BPZ    LOOPC   Repeat
01C6 F803    HOME    LDI    #03     Home the cursor
01C8 BD              PHI    D
01C9 F810            LDI    #10     First line
01CB 3095            BR     RPCM4   Go place cursor
01CD 9F      BKSP    GHI    F       Backspace
01CE 3A96            BNZ    RPCM3   If not zero, go set
01D0 8D              GLO    D
01D1 FA07            ANI    #07     Check low 3 bits
01D3 3200            BZ     EXITT   Cannot bksp, from
                                    beginning
01D5 2D              DEC    D       Set back 1 byte
01D6 9F              GHI    F       Get switch again
01D7 304C            BR     UPSW    Go replace cursor
01D9 F8B0    REVRS   LDI    #B0     Reverse char. set
01DB AC              PLO    C
01DC EC              SEX    C
01DD F0      LOOPR   LDX
01DE FBFF            XRI    #FF     Complement
01E0 73              STXD           Replace, decrement
01E1 8C              GLO    C       Check for end
01E2 FC01            ADI    #01     Compensate for end
01E4 3ADD            BNZ    LOOPR   Repeat
01E6 3000            BR     EXITT   Exit
01E8 01      CURON   #01            Cursor on (0=off)
01E9 03      HDISPL  #03            Cursor address
01EA 10      LDISPL  #10
01EB 00      SWITCH  #00            Switch (0=left)
01EC 00      WORK1   #00            Work area
01ED 00      WORK2   #00
01EE 00      WORK3   #00
```

REFR

Following is a listing of the display refresh routine, REFR.

```
ADDR CODE    LABEL   OPCODE OPERAND COMMENT

01EF 72      EXITR   LDXA           Restore D
01F0 70              RET            Return to caller
01F1 C4      REFR    NOP            Timing no-op.
01F2 22              DEC    2       Decrement stack
                                    pointer
01F3 78              SAV            Save T Register
01F4 22              DEC    2       Decrement stack
01F5 52              STR    2       Save D on stack
01F6 E2              SEX    2
01F7 E2              SEX    2
```

```
ADDR CODE    LABEL   OPCODE OPERAND COMMENT
01F8 F803            LDI    #03     Display page
01FA B0              PHI    0       to R0
01FB F800            LDI    #00
01FD A0              PLO    0
01FE 30EF            BR EXITR       Exit
```

CTABLE

The character pattern table, CTABLE, is located at addresses 0200-02B0, and includes the characters space through upper-case Z. Each pattern occupies three bytes.

```
Addr Contents

0200 0000 0044 4040 AA00 00AE AEA0 EC6E 4092
0210 4810 0F99 F088 0000 4888 4042 2240 A4A0
0220 0004 E400 0004 8000 E000 0000 4012 4800
0230 EAAA E044 4440 E248 E0E2 62E0 8AE2 20E8
0240 62E0 48EA E0EA 2220 EA4A E0EA E220 0404
0250 0004 0480 2484 200E 0E00 8424 80EA 2020
0260 0000 F04A EAA0 EACA E068 8860 CAAA C0E8
0270 C8E0 E8C8 80E8 AAE0 AAEA A0E4 44E0 222A
0280 E0AA CAA0 8888 E0AE AAA0 9DB9 90EA AAE0
0290 EAE8 80EA AAF0 EACA 90E8 E2E0 E444 40AA
02A0 AAE0 AAAA 40AA AEA0 AA4A A0AA 4440 E248
02B0 E0
```

Characters designed by June Clark.

CGEN

This routine stacks the four-bit character patterns onto the display. It is located on memory page 2, following the character pattern table.

```
ADDR CODE    LABEL   OPCODE OPERAND COMMENT

02B1 E2      EXITC   SEX    2       Return
02B2 12              INC    2         to
02B3 70              RET            Caller
02B4 EE      CGEN    SEX    E
02B5 9F              GHI    F       Get Switch
02B6 32BC            BZ     SLEFT   Switch=left
02B8 F8F0    SRYT    LDI    #F0     Display clear
                                    right
02BA 30BE            BR     CONT
02BC F80F    SLEFT   LDO    #0F     Display clear left
02BE 5E      CONT    STR    E       Save clear in
                                    WORK1
02BF 0C      NEXTC   LDN    C       Load char. pattern
02C0 FAF0            ANI    #F0     Get left half
02C2 1E              INC    E       Point to WORK2
02C3 5E              STR    E       Save left half
02C4 4C              LDA    C       Load again,
                                    increment
02C5 FE              SHL            Shift left 4
02C6 FE              SHL
02C7 FE              SHL
02C8 FE              SHL
02C9 1E              INC    E       Point to WORK2
02CA 73              STXD           Save right half
02CB 9F              GHI    F       Get switch
02CC 32DB            BZ     DISP    If left, branch
02CE F0      RJUST   LDX            Justify right
02CF F6              SHR
02D0 F6              SHR
02D1 F6              SHR
02D2 F6              SHR
02D3 5E              STR    E       Replace WORK2
02D4 1E              INC    E       Point to WORK3
02D5 F0              LDX            Load WORK3
```

```
ADDR CODE     LABEL   OPCODE OPERAND COMMENT
02D6 F6               SHR
02D7 F6               SHR
02D8 F6               SHR
02D9 F6               SHR
02DA 73               STXD           Replace
02DB 2E       DISP    DEC    E       Point to WORK1
02DC 0D               LDN    D       Load byte from
                                     display
02DD F2               AND            Clear with WORK1
                                     mask
02DE 1E               INC    E       Point to WORK2
02DF F1               OR             Put char. in
02E0 5D               STR    D       Replace display
02E1 2E               DEC    E       Point to WORK1 again
02E2 1D               INC    D       Move display down 8
02E3 1D               INC    D
02E4 1D               INC    D
02E5 1D               INC    D
02E6 1D               INC    D
02E7 1D               INC    D
02E8 1D               INC    D
02E9 1D               INC    D
02EA 0D               LDN    D       Load byte from
                                     display
02EB F2               AND            Clear with WORK1
                                     mask
02EC 1E               INC    E       Point to WORK3
02ED 1E               INC    E
02EE F1               OR             Put char. in byte
02EF 5D               STR    D       Replace display
02F0 2E               DEC    E       Point to WORK1
02F1 2E               DEC    E
01F2 2F               DEC    F
02F3 8F       CHEK    GLO    F       Check end of loop
02F4 32B1             BZ     EXITC   Done
02F6 1D               INC    D       Move display down 8
02F7 1D               INC    D
02F8 1D               INC    D
02F9 1D               INC    D
02FA 1D               INC    D
02FB 1D               INC    D
02FC 1D               INC    D
02FD 1D               INC    D
02FE 30BF             BR     NEXTC   Repeat loop
```

```
0000 3003 C490 B2B3 F801 B1BA F8F1 A1F8 FEA2
0010 F817 A3F8 03AA D3E2 693D 196D 6422 79DA
0020 2235 2130 1900 0000 0000 0000 0000 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 0000 0000 0000 0000 0000 0000 0000 0000
0080 0000 0000 0000 0000 0000 0000 0000 0000
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
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 E212 70AC F801 BEF8 E9AE EE72 BD72 AD72
0110 BFF8 02BB BCF8 B4AB 8CFB 0D32 5A8C FB7F
0120 32B5 8CFB 1B32 D98C FB08 32CD 8CFD 5A33
0130 358C FF20 AC8C FF20 3B00 5EFE F4AC F803
0140 AF79 DB22 F8E9 AE72 BD72 ADF0 FB01 3A68
0150 8DFA 07FB 0732 5A1D 3066 8DFC 30AD 9D7C
0160 00BD 8DFA F8AD F800 BF9D FF03 3BC6 9DFD
0170 0633 99F8 03BD BFF8 10AD FC30 AF4F 5D1D
0180 9FFD 0633 7D9D FD06 3B90 F800 AC0C 307E
0190 F806 8DF8 D0AD F800 BFF8 EBAE 9F73 8D73
01A0 9D73 F032 00F8 60AC F803 AF9B BCF8 ECAE
01B0 79DB 2230 00F8 03BD F800 ADAC 0CAF 8F5D
01C0 1D9D FD06 33BE F803 BDF8 1030 959F 3A96
01D0 8DFA 0732 002D 9F30 4CF8 B0AC ECF0 FBFF
01E0 738C FC01 3ADD 3000 0103 1000 0000 0072
01F0 70C4 2278 2252 E2E2 F803 B0F8 00A0 30EF
0200 0000 0044 4040 AA00 00AE AEA0 EC6E 4092
0210 4810 0F99 F088 0000 4888 4042 2240 A4A0
0220 0004 E400 0004 8000 E000 0000 4012 4800
0230 EAAA E044 4440 E248 E0E2 62E0 8AE2 20E8
0240 62E0 48EA E0EA 2220 EA4A E0EA E220 0404
0250 0004 0480 2484 200E 0E00 8424 80EA 2020
0260 0000 F04A EAA0 EACA E068 8860 CAAA C0E8
0270 C8E0 E8C8 80E8 AAE0 AAEA A0E4 44E0 222A
0280 E0AA CAA0 8888 E0AE AAA0 9DB9 90EA AAE0
0290 EAE8 80EA AAF0 EACA 90E8 E2E0 E444 40AA
02A0 AAE0 AAAA 40AA AEA0 AA4A A0AA 4440 E248
02B0 E0E2 1270 EE9F 32BC F8F0 30BE F80F 5E0C
02C0 FAF0 1E5E 4CFE FEFE FE1E 739F 32DB F0F6
02D0 F6F6 F65E 1EF0 F6F6 F6F6 732E 0DF2 1EF1
02E0 5D2E 1D1D 1D1D 1D1D 1D1D 0DF2 1E1E F15D
02F0 2E2E 2F8F 32B1 1D1D 1D1D 1D1D 1D1D 30BF
```

BEAT THE MACHINE

by
Mike O'Rourke

I originally built this game as a digital project using discrete ICs. It took eleven ICs and although it had a variable delay it was not automatic. It also did not play the two notes at the end of the game. When I got my Super Elf I decided to program it to play my game. Using a microprocessor system you not only get a better more flexible game, but it takes only four ICs.

To play the game, load the program in at page 0. Press Run and the data displays will show 88. Press Input and whatever the computer displays you must press the corresponding keys. If you press the right keys and in time, the computer will award you one point. Your score will be displayed for three seconds then the 88 again. Press Input for another round. If you are incorrect you will be buzzed and the score will be displayed. Pressing the Input then starts a new game.

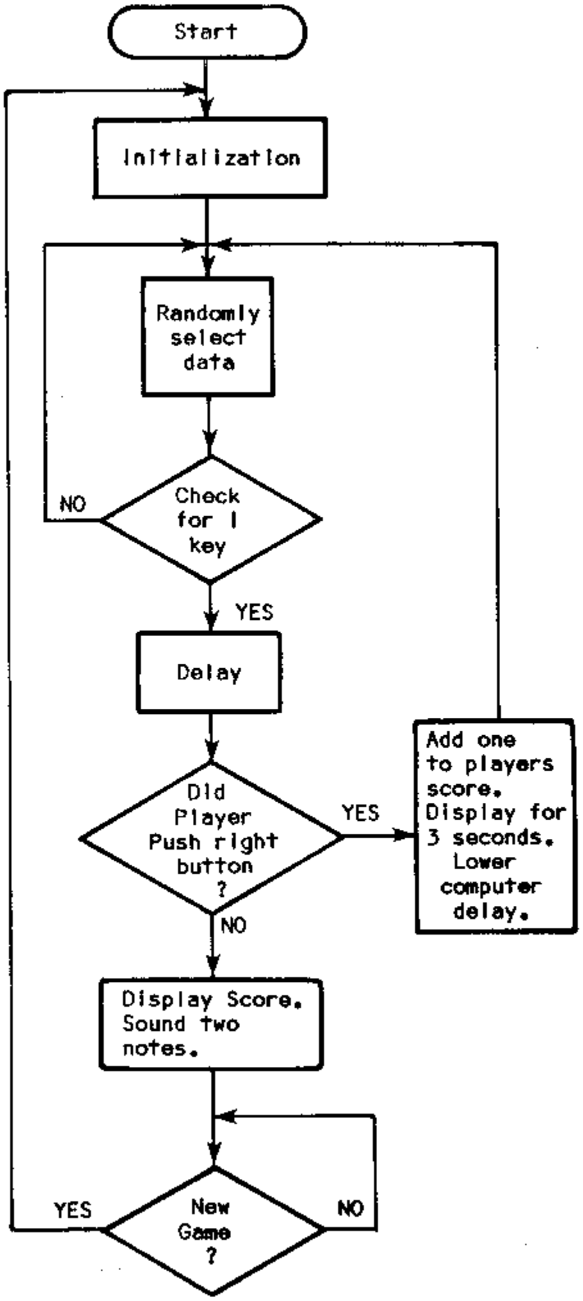
Incidentally, each time you are right the computer gets a little faster. If you are good try putting in combinations like DB, BD, etc. instead of AA, BB.

This program has been written so it will work on expanded as well as unexpanded memory.

```
Registers Used:
     X=6 or 0 or 3
     0=Entry PC
     2=Pointer to Player's input
     3=Pointer to Player's Score
     4=Pointer to Delay Value
     5=Delay Counter
     6=Pointer to Secret Data
     E=Delay Counter
     F=Tone Counter
```

Beat The Machine Listing

```
ADDR CODE                   COMMENT
0000 90 B2 B3 B4 B6         Initialize tops of registers.
0005 F8 90 A2
0008 F8 92 A4
000B F8 90 54               Initialize bottom of reg-
000E F8 93 A6               isters and memory locations.
0011 F8 91 A3
0014 90 53
0016 F8 AA 56               Store AA in memory.
0019 64 AA                  Display.
001B 37 32                  Check for I key.
001D F8 BB 56               Store BB in memory.
0020 64 BB                  Display.
0022 37 32                  Check for I key.
0024 F8 CC 56               Store CC in memory.
0027 64 CC                  Display.
0029 37 32                  Check for I key.
002B F8 DD 56               Store DD in memory.
002E 64 DD                  Display.
0030 3F 16                  Check for I key.
0032 37 32                  Wait for I key release.
0034 E2 7B
0036 04 B5 25 95            Load delay.
003A 3A 38 7A
003D 6C                     Input data.
003E E6 F5                  Check for match.
0040 32 79                  If a match go to 78.
0042 E3 64                  If not a match display score.
0044 7B F8 05 AF
0048 2F 8F 3A 48            Play first note.
004C 7A F8 04 AF
0050 2F 8F 3A 50
0054 F8 01 BE
0057 2E 9E 3A 44
005B 7B F8 FF AF            Play second note.
005F 2F 8F 3A 5F
0063 7A F8 FA AF
0067 2F 8F 3A 67
006B F8 01 BE
006E 2E 9E 3A 5B
0072 3F 72 37 74 E0 30 00 Restart.
0079 E3
007A F8 01 F4 53            Add one to players score.
007E 64 23                  Display.
0080 F8 FF AE BE            For three seconds.
0084 2E 9E 3A 84
0088 E4 F8 10 F5 54         Decrease computer delay.
008D E0
008E 30 16                  Go to beginning.
0090 Player Input
0091 Player score
0092 Computer delay
0093 Computer input
```


Start
Initialization
Randomly select data
NO
Check for I key
YES
Delay
Did Player Push right button ?
YES
Add one to players score. Display for 3 seconds. Lower computer delay.
NO
Display Score. Sound two notes.
YES
New Game ?
NO

```
0000 90B2 B3B4 B6F8 90A2 F892 A4F8 9054 F893
0010 A6F8 91A3 9053 F8AA 5664 AA37 32F8 BB56
0020 64BB 3732 F8CC 5664 CC37 32F8 DD56 64DD
0030 3F16 3732 E27B 04B5 2595 3A38 7A6C E6F5
0040 3279 E364 7BF8 05AF 2F8F 3A48 7AF8 04AF
0050 2F8F 3A50 F801 BE2E 9E3A 447B F8FF AF2F
0060 8F3A 5F7A F8FA AF2F 8F3A 67F8 01BE 2E9E
0070 3A5B 3F72 3774 E030 00E3 F801 F453 6423
0080 F8FF AEBE 2E9E 3A84 E4F8 10F5 54E0 3016
```

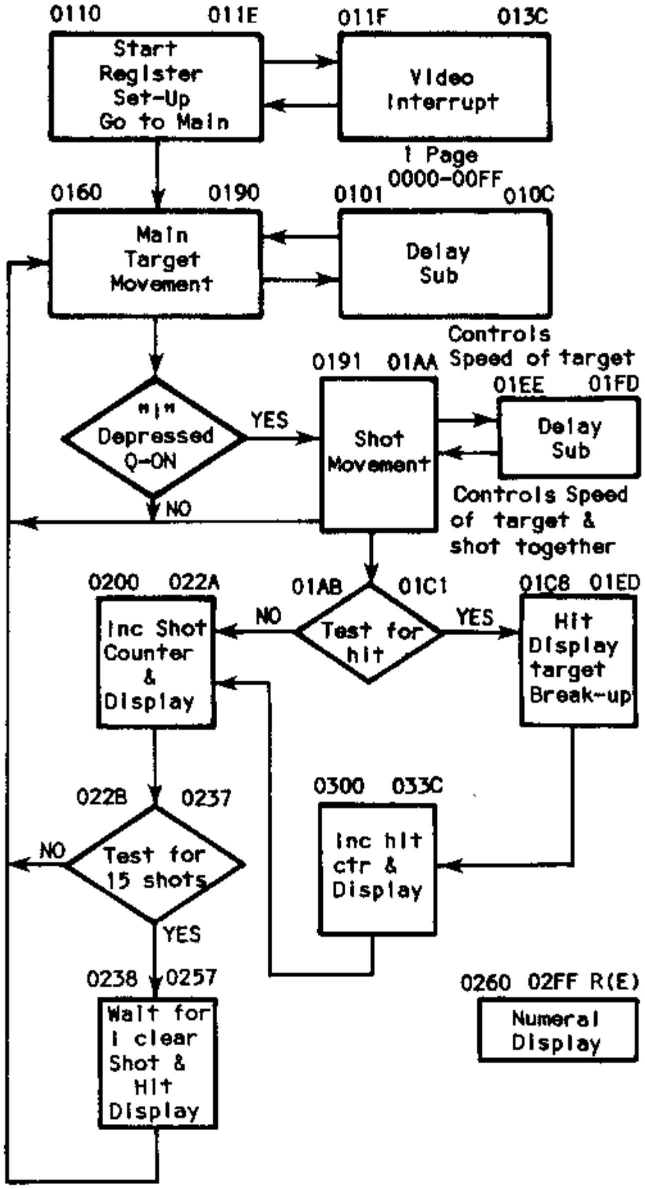
COSMAC COWBOY

by
Gary Gehlhoff

The Cosmac Cowboy Target Game displays a gun fighter and a repeating descending target as shown on page 8 of the program. When the input key is depressed a bullet travels from the end of the gun to the right, to location 00B7. If a hit is not detected the shot displays will be incremented and the target will continue to move. If a hit is detected (corresponding values in both the shot and target registers) the target is broken apart showing a hit and a 3 second delay is executed before incrementing the hit and shot display and resumption of the target movement. After 15 shots have been fired the program halts until the input is depressed. The score is then cleared and the target movement again resumes.

Two delay subroutines are required to keep the target moving at the same speed when the shot routine is both active and inactive. While the shot subroutine is active, control is shared among the shot routine, delay routine, and the main program.

This program is designed to operate from the monitor. At location 0340, or from 0000 via the "Q" button thru 0002.



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

Publisher ....Quest Electronics
Editor ....Paul Messinger
Assistant to Editor ...Jeanette Johnson
Associate Editor ....Allan Armstrong
Contributing Editors Ron Cenker
Van Baker
Art and Graphics ....Holly Olson
Proof Reading ....Judy Pitkin
Production ....John Larimer
Circulation ....Sue Orr

```
Registers Used:
     P=3
     X=A or 2
     0=DMA
     1=Interrupt
     2=SP
     3=PC
     4=Used
     5=Delay Counter
     6=Used
     7=Used
     8.1=temporary storage
     9=Used
     A=SP
     B=Used
     C.0=Used
     D=Used
     E=Used
     F=Used
```

SET-UP

```
ADDR CODE

0000 C0 01 10 00 00 00 00 00
0008 00 00 00 00 00 00 00 00
0010 FC 00 00 00 00 00 00 00
0018 C0 00 00 00 00 00 00 00
0020 FC CC FC FC 00 00 00 00
0028 0C FC CC 30 00 00 00 00
0030 FC CC FC 30 00 00 00 00
0038 00 00 00 00 00 00 00 00
0040 C3 30 00 00 00 00 00 00
0048 C3 00 00 00 00 00 00 00
0050 FF 33 F0 00 00 00 00 00
0058 C3 30 C0 00 00 00 00 00
0060 C3 30 C0 00 00 00 00 00
0068 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00
0078 3F 00 00 00 00 00 00 00
0080 FF C0 00 00 00 00 00 00
0088 3F 00 00 00 00 00 00 00
0090 3F 80 00 00 00 00 00 00
0098 3E 00 00 00 00 00 00 00
00A0 3F 00 00 00 00 00 00 00
00A8 0C 00 00 00 00 00 00 00
00B0 3F 0F 00 00 00 00 00 00
00B8 3F FC 00 00 00 00 00 00
00C0 3F 00 00 00 00 00 00 00
00C8 3F 00 00 00 00 00 00 00
00D0 3F 00 00 00 00 00 00 00
00D8 0C 00 00 00 00 00 00 00
00E0 0C 00 00 00 00 00 00 00
00E8 0C 00 00 00 00 00 00 00
00F0 0C 00 00 00 00 00 00 00
00F8 0F 00 00 00 00 00 00 00
```

TARGET ONLY - DELAY SUBROUTINE

```
ADDR CODE
0101 D3 B9 F8 CC A5 25 85
0108 3A 05 99 30 01
```

REGISTER SET-UP

```
ADDR CODE
0110 F8 01 B1 B2 B3 F8 3D A3
0118 F8 C6 A2 F8 21 A1 D3
```

REGISTER CHART

```
R(1)  Interrupt  0121
R(2)  Stack      01C7
R(3)  Main       013D
R(4)  Delay Counter in hit display
R(5)  Delay Counter in both delay subroutines
R(6)  Number of Shot positions 0006
R(7)  Starting Address of shot (00B2)
R(8)  Delay Subroutines
        a)  Target Only 0101, ENTRY = 0102
        b)  Target & Shot 01EE, ENTRY 1= 01F1,
            ENTRY 2 = 01EF
R(9)  Accumulator Saving & other counters
R(A)  Target starting address (0007)
R(B)  No of target positions (0020)
R(C)  R(C).0 shot display location
R(D)  Shot subroutine (0191), ENTRY = 0192
R(E)  Numeral display
R(F)  R(F).0 shot count R(F).1 Hit Count
```

VIDEO DISPLAY INTERRUPT

```
ADDR CODE
011F 72
0120 70 C4 22 78 22 52 F8 00
0128 B0 F8 00 A0 C4 C4 80 E2
0130 E2 20 A0 E2 20 A0 E2 20
0138 A0 3C 2E 30 1F
```

MAIN (TARGET SECTION)

```
ADDR CODE
013D 61 EA F8
0140 01 B4 B5 B8 BC BD F8 00
0148 B6 B7 BF AF BA BB F8 07
0150 AA F8 20 AB F8 92 AD F8
0158 02 A8 F8 B2 A7 F8 06 A6
0160 37 63 38 DD F8 03 5A D8
0168 8A FC 08 AA F8 03 F5 8A
0170 FF 08 AA D8 F8 00 5A D8
0178 8A FC 10 AA 37 81 31 81
0180 38 DD F8 03 5A D8 8B FF
0188 01 AB 3A 6F F8 00 5A 30
0190 4E
```

SHOT SECTION

```
ADDR CODE
0191 D3 7B F8 F1 A8 F8 80
0198 57 AE D8 D3 8E F6 3A 98
01A0 57 17 86 FF 01 A6 3A 91
01A8 F8 00 A8
```

HIT DETERMINATION

```
ADDR CODE
01AB 8A FF BF 32 C8
01B0 FF 08 32 C8 FC 10 32 C8
01B8 7A F8 B2 A7 F8 06 A6 C0
01C0 02 00
```

HIT DISPLAY (TARGET BREAK-UP)

```
ADDR CODE
01C8 8A FF 08 AA F8 0C 5A 8A
01D0 FC 08 AA F8 30 5A 8A FC
01D8 08 AA F8 0C 5A 7A F8 40
01E0 B4 24 94 3A E1 F8 B2 A7
01E8 F8 06 A6 C0 03 00
```

TARGET & SHOT - DELAY SUBROUTINE

```
ADDR CODE
01EE DD 38
01F0 D3 B9 F8 AA A5 25 85 3A
01F8 F5 99 31 EE 30 F0
```

```
SHOT NUMERAL & DISPLAY
ADDR CODE
0200 F8 00 BC F8 14 AC F8 02
0208 BE F8 60 AE 1F 8F FF 01
0210 A9 8E FC 0A AE 89 3A 0E
0218 F8 05 A9 4E 5C 1C 4E 5C
0220 1C 1C 1C 1C 1C 1C 1C 29
0228 89 3A 18 8F FF 0F 32 33
0230 C0 01 91 2E F8 00 BF AF

       SCORE CLEAR
ADDR CODE
0238 3F 38 37 3A F8 0B A9 F8
0240 14 AE F8 00 BE 5E 1E 5E
0248 1E 1E 1E 1E 1E 1E 1E 89
0250 FF 01 A9 3A 42 C0 01 91

    HIT COUNTER & DISPLAY
ADDR CODE
0300 F8 02 BE F8 60 AE F8 00
0308 BC F8 44 AC 9F FC 01 BF
0310 FF 01 B9 8E FC 0A AE 99
0318 3A 10 F8 05 A9 4E 5C 1C
0320 4E 5C 1C 1C 1C 1C 1C 1C
0328 1C 29 89 3A 1D 9F FF 0F
0330 32 35 C0 01 B8 2E F8 00
0338 BF AF C0 01 B8

ALTERNATE ENTRY POINT FOR SUPER MONITOR
0340 F8 01 B0 F8 10 A0 E3 70 00

          NUMERAL DISPLAY R(E)=0260
ZERO (0260)   003F  0033  0033  0033  003F
ONE  (026A)   003C  000C  000C  000C  003F
TWO  (0274)   003F  0003  003F  0030  003F
THREE(027E)   003F  0003  000F  0003  003F
FOUR (0288)   0033  0033  003F  0003  0003
FIVE (0292)   003F  0030  003F  0003  003F
SIX  (029C)   003F  0030  003F  0033  003F
SEVEN(02A6)   003F  0033  0003  0003  0003
EIGHT(0260)   003F  0033  003F  0033  003F
NINE (02BA)   003F  0033  003F  0003  0003
TEN  (02C4)   3C3F  0C33  0C33  0C33  3F3F
ELEVEN(02CE)  3C3C  0C0C  0C0C  0C0C  3F3F
TWELVE(02D8)  3C3F  0C03  0C3F  0C30  3F3F
THIRTEEN
      (02E2)  3C3F  0C03  0C0F  0C03  3F3F
FOURTEEN
      (02EC)  3C33  0C33  0C3F  0C03  3F03
FIFTEEN
      (02F6)  3C3F  0C30  0C3F  0C03  3F3F
```

```
0000 C001 1000 0000 0000 0000 0000 0000 0000
0010 FC00 0000 0000 0000 C000 0000 0000 0000
0020 FCCC FCFC 0000 0000 0CFC CC30 0000 0000
0030 FCCC FC30 0000 0000 0000 0000 0000 0000
0040 C330 0000 0000 0000 C300 0000 0000 0000
0050 FF33 F000 0000 0000 C330 C000 0000 0000
0060 C330 C000 0000 0000 0000 0000 0000 0000
0070 0000 0000 0000 0000 3F00 0000 0000 0000
0080 FFC0 0000 0000 0000 3F00 0000 0000 0000
0090 3F80 0000 0000 0000 3E00 0000 0000 0000
00A0 3F00 0000 0000 0000 0C00 0000 0000 0000
00B0 3F0F 0000 0000 0000 3FFC 0000 0000 0000
00C0 3F00 0000 0000 0000 3F00 0000 0000 0000
00D0 3F00 0000 0000 0000 0C00 0000 0000 0000
00E0 0C00 0000 0000 0000 0C00 0000 0000 0000
00F0 0C00 0000 0000 0000 0F00 0000 0000 0000
0100 00D3 B9F8 CCA5 2585 3A05 9930 01BB BCF8
0110 F801 B1B2 B3F8 3DA3 F8C6 A2F8 21A1 D372
0120 70C4 2278 2252 F800 B0F8 00A0 C4C4 80E2
0130 E220 A0E2 20A0 E220 A03C 2E30 1F61 EAF8
0140 01B4 B5B8 BCBD F800 B6B7 BFAF BABB F807
0150 AAF8 20AB F892 ADF8 02A8 F8B2 A7F8 06A6
0160 3763 38DD F803 5AD8 8AFC 08AA F803 5A8A
0170 FF08 AAD8 F800 5AD8 8AFC 10AA 3781 3181
0180 38DD F803 5AD8 8BFF 01AB 3A6F F800 5A30
0190 4ED3 7BF8 F1A8 F880 57AE D8D3 8EF6 3A98
01A0 5717 86FF 01A6 3A91 F800 A88A FFBF 32C8
01B0 FF08 32C8 FC10 32C8 7AF8 B2A7 F806 A6C0
01C0 0200 8A8A 8A60 188A 8AFF 08AA F80C 5A8A
01D0 FC08 AAF8 305A 8AFC 08AA F80C 5A7A F840
01E0 B424 943A E1F8 B2A7 F806 A6C0 0300 DD38
01F0 D3B9 F8AA A525 853A F599 31EE 30F0 1020
0200 F800 BCF8 14AC F802 BEF8 60AE 1F8F FF01
0210 A98E FC0A AE89 3A0E F805 A94E 5C1C 4E5C
0220 1C1C 1C1C 1C1C 1C29 893A 1B8F FF0F 3233
0230 C001 912E F800 BFAF 3F38 373A F80B A9F8
0240 14AE F800 BE5E 1E5E 1E1E 1E1E 1E1E 1E89
0250 FF01 A93A 42C0 0191 0000 0000 0000 0000
0260 003F 0033 0033 0033 003F 003C 000C 000C
0270 000C 003F 003F 0003 003F 0030 003F 003F
0280 0003 000F 0003 003F 0033 0033 003F 0003
0290 0003 003F 0030 003F 0003 003F 003F 0030
02A0 003F 0033 003F 003F 0033 0003 0003 0003
02B0 003F 0033 003F 0033 003F 003F 0033 003F
02C0 0003 0003 3C3F 0C33 0C33 0C33 3F3F 3C3C
02D0 0C0C 0C0C 0C0C 3F3F 3C3F 0C03 0C3F 0C30
02E0 3F3F 3C3F 0C03 0C0F 0C03 3F3F 3C33 0C33
02F0 0C3F 0C03 3F03 3C3F 0C30 0C3F 0C03 3F3F
0300 F802 BEF8 60AE F800 BCF8 44AC 9FFC 01BF
0310 FF01 B98E FC0A AE99 3A10 F805 A94E 5C1C
0320 4E5C 1C1C 1C1C 1C1C 1C29 893A 1D9F FF0F
0330 3235 C001 B82E F800 BFAF C001 B8
```

IMPROVING CHIP-8 GRAPHICS

by
David Crawford

The graphics system used by Chip-8 is perfectly adequate for many applications: games, numeric display, graphs... Many times, though, one would like to be able to have larger and finer graphics just for the sake of holding more information on the screen.

The Chip-8 graphics system uses a 64 by 32 format and uses 256 bytes of memory. The following modifications and additions to Chip-8 will improve the graphics capabilities of Chip-8 to a 64 by 64 format with the use of 512 bytes of memory. The additional memory needed is located in the page previous to the former display page.

Since the display now occupies the last two pages of memory, the Chip-8 workspace and variables must be relocated one page earlier at 0Z which is 05 for a 2k system or 0D for a 4k system. This is done immediately upon start up by changing locations 0000 through 000E to those in table A.

The display interrupt routine which controls the display must be modified. This program was relocated from the operating system ROM at address 8143 to RAM at 0Z73 through 0Z98. This code is shown in table B. Each of the 64 lines of 8 words is displayed twice, using 2 pages, 512 words, of memory. When the standard 4 by 5 block hex characters are displayed, they will appear the same width but half as tall.

The increase from 32 to 64 displayed lines necessitates modifying the DXYN command to allow Y to go from 31 to 63. To find space for the necessary modifications, the program jumps from page 00 to 0Z by switching the program counter. Make the modifications to the Chip-8 interpreter shown in table C and add the code shown in table D.

Since the erase subroutine at 00E0 only erases the first page of the display, a new machine language subroutine, located at 0Z60, was made to erase both pages. See table E.

```
The following are modifications to Chip-8.
     Table A
     Set pointer locations.
ADDR CODE              COMMENT
0000 91                get last page of memory.
0001 FF                subtract 1
0002 01
0003 BB                set high address of display
                       pointer.
0004 FF                subtract 1
0005 01
0006 B1                set high address of interrupt
                       pointer.
0007 B2                set high address of stack
                       pointer.
0008 B6                set high address of vX pointer.
0009 F8                set low address of stack
000A CF                pointer.
000B A2
000C F8                set low address of interrupt
000D 76                pointer.
000E A1
     Table C
     Modify show command DXYN
ADDR CODE              COMMENT
0070 F8                branch to area containing
                       modified show command.
0071 0Z--------------Z=5 for 2k systems
0072 BF                Z=D for 4k systems
0073 F8
0074 40
0075 AF
0076 DF                change program counter.
0077 00
0078 3B                test for overrun of display.
0079 B3
007A 9C
007B FC
007C 01
007D BC
007E FB
007F XX--------------08 for 2k, 10 for 4k
0080 32
0081 D9
0082 30
0083 B3
0084 00
0085 00
00D7 30                new location for overrun test.
00D8 78                branch to 0078.
     Table B
     interrupt service routine.
     Substituting for operating system code @ 8143
     through 816D.
ADDR CODE              COMMENT
0Z73 7A                tone off
0Z74 42
0Z75 70                return to main program.
entry from main
---6 22                store main program counter
0Z77 78                on stack.
0Z78 22
0Z79 52
0Z7A C4                synchronize timing.
0Z7B 19                increment random number.
0Z7C F8                set DMA pointer.
     Table B
ADDR CODE              COMMENT
0Z7D 00
0Z7E A0
0Z7F 9B
0Z80 B0
0Z81 E2
0Z82 E2                two page display program block.
0Z83 80
0Z84 E2                show the first 60 lines repeating
0Z85 20                each line twice until EF1 flag
0Z86 A0                goes high.
0Z87 E2
0Z88 3C
0Z89 83
0Z8A 80                show last 4 lines
0Z8B E2
0Z8C 20
0Z8D A0
0Z8E E2
0Z8F 34                service timer and tone.
0Z90 8A
0Z91 98
0Z92 32                test timer-go to tone if zero.
0Z93 98                 -else decrement timer.
0Z94 AB
0Z95 2B
0Z96 8B
0Z97 B8
0Z98 88
0Z99 32                test tone, if 0- go to
                       "tone off".
0Z9A 73                - else turn tone
0Z9B 7B                on and decrement
0Z9C 28                tone counter.
0Z9D 30                branch to
0Z9E 74                74.


     Table D
     Continue modifications to show command
ADDR CODE              COMMENT
0Z40 06                get X position
0Z41 FA                mask
0Z42 07                with 07
0Z43 BE                store bit position
0Z44 06                get X position
0Z45 FA                mask
0Z46 3F                with 63
0Z47 F6                divide by 8
0Z48 F6
0Z49 F6
0Z4A 22                store on stack
0Z4B 52                get y position
0Z4C 07
0Z4D FA                mask
0Z4E 3F                with 63
0Z4F FE                multiply by 8
0Z50 FE
0Z51 FE
0Z52 F1                combine to form address
0Z53 AC
```

```
0Z54 9B
0Z55 3B          if carry then address is on
0Z56 59          next page.
0Z57 FC
0Z58 01
0Z59 BC
0Z5A 94          return to 0086
0Z5B B3
0Z5C F8
0Z5D 86
0Z5E A3
0Z5F D3
```

Table E
Machine language subroutine to erase 2 pages of display.

```
ADDR CODE          COMMENT
0Z60 9B            set RF to point to start of
0Z61 BF            display page.
0Z62 94            get a 0
0Z63 AF
0Z64 5F            store 2 0's in memory
0Z65 1F
0Z66 5F
0Z67 1F            repeat
0Z68 94            get 0
0Z69 5F            store 3 0's
0Z6A 1F
0Z6B 5F
0Z6C 1F
0Z6D 5F
0Z6E 1F
0Z6F 8F
0Z70 3A            branch until address equals
0Z71 68            zero.
0Z72 D4            return
```

SOUNDS OF COSMAC

by
Mark Wendell

The COSMAC computer is truly a versatile instrument, and this program only further proves this point. I have written up a series of five programs that fit into the music algorithm on page 11 of QUESTDATA Vol. 1 Issue 7, but these are not music programs—they're..well..different. They demonstrate not only the flexibility of the machine, but also the flexibility of such a relatively simple program.

COSMAC FROG

If you enjoyed, back in QUESTDATA 7, a short program called the COSMAC Cricket (by yours truly), then this little routine ought to be of interest to you. It's called the Cosmac Frog. First, load the music algorithm, putting an 04 in location 10, and an 04 in location 3D (these provide tempo and the short interval between notes). Next, load the program in Table 1 starting at location 45. Check for errors and run. You may notice that this is actually a tonally lowered and slowed version of the Cricket.

SPECIAL NOTE— I have replaced the small speaker provided with my Elf with a larger, eight ohm speaker and the sound quality has improved ten-fold.

TELEPHONE

This program emulates the sound of an old-fashioned, crank telephone ring. Location 10 must be changed to 03, and location 3D to 05. Load the tone values and run.

R2 - D2

This program makes the sound of the all-too-popular robot, R2-D2. Change location 10 to 0A and location 3D to 03. Load the tone values and run (and check the cover of QD 5).

MYSTERY TUNE

I haven't named this one yet - but its interesting and leaves open a lot of room for experimentation. For best results, leave locations 10 and 3D the same as Program 3 - although play around with different speeds. Simply load and run.

LASER BLAST

This program creates quite a realistic laser blast when Input is pressed. In order to have the program enabled only when "I" is pressed, the following modifications have to be made: change location 02 to 47; at location 43, put in 37 04 30 & 43. Locations 10 and 3D must both be 01. Load the tone values, run, and press "I". Notice that the tone values begin at location 47.

You may find this form of experimentation both fun and rewarding. Try different patterns (I hope you've caught the patterns in Programs 4 + 5), and change the values in locations 10 and 3D - satisfaction guaranteed.

```
TABLE THREE
 R2-D2

ADDR CODE      ADDR CODE      ADDR CODE
0045 07 1B     005D 06 1B     0075 19 06
0047 04 33     005F 05 2B     0077 0B 3F
0049 05 2A     0061 04 3B     0079 04 4C
004B 09 14     0063 03 43     007B 03 4B
004D 03 47     0065 06 4C     007D 05 4C
004F 05 2D     0067 08 17     007F 00 00
0051 08 15     0069 06 19
0053 07 1B     006B 04 33
0055 0B 4B     006D 0B 4C
0057 05 4C     006F 06 1F
0059 0A 08     0071 09 14
005B 04 4C     0073 10 0A
```

```
TABLE ONE
COSMAC FROG

ADDR CODE
0045 03 47
0047 03 47
0049 03 47
004B 03 47
004D 03 47
004F 03 47
0051 03 47
0053 03 47
0055 03 47
0057 03 47
0059 03 43
005B 03 43
005D 03 43
005F 03 43
0061 03 43
0063 03 43
0065 03 43
0067 03 43
0069 03 43
006B 03 43
006D 03 43
006F 03 43
0071 B0 4C
0073 00 00
```

```
TABLE TWO
TELEPHONE

ADDR CODE
0045 04 0F
0047 04 08
repeat 4-byte sequence
above 30 times (who said
programming was fast?)
at the end of which, put
in the data:
00FF 4C 90 4C
     00 00
```

```
TABLE FOUR
MYSTERY TUNE

ADDR CODE
0045 01 01
0047 01 02
0049 01 03
004B 01 04
004D 01 05
004F 01 06
0051 01 07
0053 01 08
0055 01 09
0057 01 0A
0059 01 0B
005B 01 0C
005D 01 0D
005F 01 0E
0061 01 0F
0063 01 0E
0065 01 0D
0067 01 0C
0069 01 0B
006B 01 0A
006D 01 09
006F 01 08
0071 01 07
0073 01 06
0075 01 05
0077 01 04
0079 01 03
007B 01 02
007D 00 00
```

```
TABLE FIVE
LASER BLAST

ADDR CODE
0047 09 09
0049 0A 0A
004B 0B 0B
004D 0C 0C
004F 0D 0D
0051 0E 0E
0053 0F 0F
0055 10 10
0057 11 11
0059 12 12
005B 00 00
```

COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC

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

ADDRESS CORRECTION REQUESTED