Questdata
Volume 2 Issue # 10 ©

STIX

by
Vic Worthington

GAME DESCRIPTION AND OPERATING INSTRUCTIONS

In James F. Fixx's, "Games for the Super Intelligent", he describes a game called "Matches in Rows". This game goes by many names and there are several variations but usually the object of the game is to force your opponent to take the last stick.

STIX is an electronic adaptation of one version of this game. With STIX the object is still to "stick" your opponent with the last stick. However, your opponent is the computer.

The game begins with three rows of sticks displayed on the video monitor. ROW A contains 7 sticks, row B contains 5 sticks, and row C contains 3 sticks. On each turn a player must take at least 1 stick but cannot take more than 3 Sticks. Sticks can be taken from any row. The player taking the last stick loses.

After the program is loaded the computer is placed in the run mode. The video monitor displays the three rows of sticks and a question mark appears in the lower left corner of the screen. Either player may start the game.

If the Elf is to go first, enter FF through the hex keypad and press the I key. The question mark disappears and after a short delay the computer emits a beep sound and the Elf eliminates its choice of sticks.

If the person is first, enter 00 and press the I key. The question mark will be replaced with a P. Whenever the P is displayed the person's move can be entered through the hex keypad. For example; if you wish to take 2 from row A, press the 2 key and the A key. The hex display will show 2A. Press the I key and if 2 from A is a valid move (row A contains 2 or more sticks) the computer will take 2 sticks from row A. If the move is invalid (row A contains less than 2 sticks) the computer will squawk and a question mark replaces the P. The question mark will remain until a valid move has been entered. The computer will consider as invalid any take request greater than 3 or any row request other than A,B, or C. It should be noted that if the I key has not been pressed the keypad entry can be changed. This allows mistakes to be corrected.

The Elf and the person alternate turns until only one stick remains. If it is the Elf's turn, and only one stick remains, a W will appear to signal that the person has won. If it is the person's move, and only one stick remains, the progam waits for the person to take the last stick and then an L is displayed to signal that the person has lost. The W or the L will remain for a short time and then a new game will start. Pressing the I key when the W or the L is displayed will immediately start a new game. To start a new game at any other time; restart the program at location 00 00.

PROGRAM OPERATION

STIX is loaded into the first four pages of memory. The main program occupies all of page zero except the last thirty-one locations. These thirty-one locations contain the video interrupt routine. Page 3 is the display area and pages 1 and 2 contain the various sub routines and the data tables. Execution of the program begins at location 00 00.

It appears that the game is played on the video monitor, but actual play takes place in registers A and B. The video display merely reflects the current status of the A and B registers. High order register A contains the stick count for row A. Low register A contains the count for row B, and row C count is stored in high register B.

Each stick displayed is represented by a bit set in a row's respective register. For example; when row A displays seven sticks, high order register A contains 7F. If row A displays 5 sticks, high order register A contains 1F. To eliminate sticks the program simply shifts off to the right the desired number of bits. To take 2 sticks from row A, high register A is shifted right 2 times. The print routine is called and the display area is updated to show the new row count. Since time is cheaper than memory, the print routine doesn't determine which row was changed. It simply updates all three rows. If a bit is a zero, the routine clears that bit's respective position in the display area. If a bit is a one, the routine loads a stick in that bit's respective position in the display area. The print routine is also used to load the display area with sticks at the beginning of each game.

The person's move doesn't require any special programming. The person's move is entered through the hex keypad. The move is checked for validity, and if it is valid, the requested number of sticks (bits) are shifted from the selected row. The print routine is called and the display area is updated to show the results of the person's move.

After the person's move, the program checks to see if the person has won or lost. If a win or a lose is found the appropriate letter is displayed. If no win or lose is found the Elf takes a turn.

The Elf's turn is a little more complex. The 7-5-3 row arrangement presents 192 (8x6x4) possible configurations. There are 72 configurations, of the 192, for which no effective counter move exists. On each one of its turns the Elf will attempt to establish one of these 72 "HIT" positions. Once the Elf has left his opponent with one of these hits, no matter what move the opponents makes, the Elf can re-establish a new hit on each of its succeeding moves. If a hit is not found after trying all possible moves, one stick is taken from a randomly selected row.

It should be noted that when the Elf starts the game, the Elf will take one stick from a random row. This is the only time that the Elf does not look for a hit.

Play continues until a win or a lose is detected. After an appropriate delay to display the W or the L the program jumps to STPLA (00 2A). Here the A and B registers are loaded with the 7-5-3 stick arrangement, print is called, and the computer waits for the keypad to determine who is first.

```
STIX - Assignments

I/O Assignments

A. Input Key Status                 EF4
B. Hex Keypad Input                 INP 4
C. Hex Display Output               OUT 4
D. Video Display On                 OUT 1

I/O Instructions' Memory Locations

ØØ 29    61                         Turn on Video

ØØ 38    6C 64 --                   Read Keypad and Display
   3B    3F --                      Branch if 1 key is not pressed
   3D    37 --                      Branch until 1 key is released

ØØ BD    3F --                      See above
   BF    37 --

Ø2 CA    6C 64 --                   See above
   CD    3F --
   CF    37 --

Register Assignments

RØ DMA    XX XX
R1 VIDEO  ØØ E3 (VDO) --            Video Interrupt Routine Pointer
R2 STACK  Ø2 ØØ                     Stack Pointer
R3 STIX   ØØ Ø8 (MAIN)              Main Program Counter
R4 PRINT  Ø1 71 (PTR)               Print Routine Pointer
R5 STATS  Ø2 Ø6 (STA)               Status Routine Pointer
R6 PERSN  Ø2 C6 (PER)               Person's Move Routine Pointer
R7 ELFGO  Ø1 Ø1 (ELF)               Elf's Move Routine Pointer
R8 SUBRT  Ø2 XX                     General Sub Routine Pointer
R9 DAPT   Ø1 XX                     Data Area Pointer
RA RA     XX XX                     Row A and B Count
RB RB     XX XX                     Row C Count
RC CTR    XX XX                     General Counter
RD TEMP   Ø2 Ø1                     Temporary Storage Pointer
RE RTCK   XX XX                     Real Time Clock
RF LINK   Ø3 XX                     Display Area Pointer
```

```
XX  XX  XX  XX  XX  XX  XX
XX  XX  XX  XX  XX  XX  XX    XXXX
XX  XX  XX  XX  XX  XX  XX   XX  XX
XX  XX  XX  XX  XX  XX  XX   XX  XX
XX  XX  XX  XX  XX  XX  XX   XX  XX
XX  XX  XX  XX  XX  XX  XX   XXXXXX
XX  XX  XX  XX  XX  XX  XX   XX  XX
XX  XX  XX  XX  XX  XX  XX   XX  XX

        XX  XX  XX  XX  XX
        XX  XX  XX  XX  XX    XXXXX
        XX  XX  XX  XX  XX   XX  XX
        XX  XX  XX  XX  XX   XX  XX
        XX  XX  XX  XX  XX   XXXXX
        XX  XX  XX  XX  XX   XX  XX
        XX  XX  XX  XX  XX   XX  XX
        XX  XX  XX  XX  XX   XXXXX

 XXXX                   XX  XX  XX
XX  XX                  XX  XX  XX    XXXXX
   XX                   XX  XX  XX   XX
  XX                    XX  XX  XX   XX
 XX                     XX  XX  XX   XX
 XX                     XX  XX  XX   XX
                        XX  XX  XX   XX
 XX                     XX  XX  XX    XXXXX
```

VIDEO DISPLAY AT THE BEGINNING OF THE GAME

STIX - Page ØØ

MAIN Program (225) Execution begins at ØØ ØØ

```
ØØ INIT   F8 ØØ B1 B3          Initialize registers
Ø4        F8 Ø8 A.Ø(MAIN) A3
Ø7        D3
Ø8 MAIN   F8 Ø1 B4 B7 B9 AD
ØE        F8 Ø2 B2 B5 B6 BD A2
16        F8 33 A.Ø(VDO) A1    R1 is Video interrupt pointer
19        F8 71 A.Ø(PTR) A4    R4 is Print routine pointer
1C        F8 Ø6 A.Ø(STA) A5    R5 is Status routine pointer
1F        F8 C6 A.Ø(PER) A6    R6 is Person routine pointer
22        F8 Ø1 A.Ø(ELF) A7    R7 is Elf routine pointer
25        F8 Ø3 A.1(DSP) BF    RF is Display link
28        ED                   TEMP is stack
29        61                   Turn on Video
2A STPLA  F8 7F BA             Row A = 7
2D        F8 1F AA             Row B = 5
3Ø        F8 Ø7 BB             Row C = 3
33        D4                   Print
34        F8 A3 A.Ø(?) A9 D5   Status = ?
38 INA    6C 64 2D             If input is ØØ Person is first
3B        3F 3B INA            If input is FF Elf is first
3D        37 3D *
3F        32 9Ø PRFST
41        F8 AB A.Ø(BLK) A9 D5 Elf is first - blank status
45 RAND   8E FA Ø3 5D          Use Real Time Clock to
49        FB Ø1 32 65 CTK      Find random row
4D        FØ FB Ø2 32 5E BTK
52        FØ FB Ø3 3A 45 RAND  If RTCK not A, B, or C, try again
57 ATK    9A 32 45 RAND        Random row is A - if A = Ø try again
5A        F6 BA 3Ø 6A DELAY    A ≠ Ø so take 1 from row A
5E BTK    8A 32 45 RAND        Random row is B - if B = Ø try again
61        F6 AA 3Ø 6A DELAY    B ≠ Ø so take 1 from row B
65 CTK    9B 32 45 RAND        Random row is C - if C = Ø try again
68        F6 BB                C ≠ Ø so take 1 from row C
6A DELAY  F8 ØØ AE             Short Delay
6D TIM    8E FF 44 3A 6D TIM
72 BEEP   F8 35 AC             Make Beep
75 MORB   7B F8 55
78 ON     FF Ø1 3A 78 ON
7C        7A F8 15
7F OFF    FF Ø1 3A 7F OFF
93        2C 8C 3A 75 MORB
87        D4                   Print Elf's move
88 PAUSE  F8 ØØ AE             Short Pause
8B PTM    8E FF 22 3A 8B PTM
9Ø PRFST  F8 9B A.Ø(?) A9 D5   Status = P
94 PRMV   D6                   Input Person's move
95        31 C3 INVD           If Q set Person's move is invalid
97        D4                   Move valid - print Person's move
98 LWH    9A 5D 8A F4 5D 9B F4 Check for game over
9F        FF Ø1 32 AB WIN      Person has won
A3        3B B1 LOSE           Person has lost
A5        D7                   Game is not over - Elf takes turn
A6        39 45 RAND           Hit was not found - take random
A8        7A 3Ø 6A DELAY       Hit was found - print Elf's move
AB WIN    F8 8B A.Ø(W) A9 D5   Status = W
AF        3Ø B5 WAIT
B1 LOSE   F8 93 A.Ø(L) A9 D5   Status = L
B5 WAIT   F8 ØØ AE             Wait for time up or for I key
B8 WAT    8E FF FF 32 2A STPLA
BD        3F B8 WAT
BF        37 BF *
C1        3Ø 2A STPLA
C3 INVD   F8 A3 A.Ø(?) A9 D5   Status = ?
C7 ZAP    F8 Ø5 A9             Make Zap Sound
CA RPT    F8 28 AC
CD SET    7B
CE SIT    8C AB
DØ CTD    2B 8B 3A DØ CTD
D4        39 CD SET
D6        7A
D7        2C 8C 3A CE SIT
DB        29 89 3A CA RPT
DF        3Ø 94 PRMV
```

VIDEO Interrupt Routine (31)

```
E1 EXVDO  72 7Ø                This routine points DMA to
E3 VDO    22 78 22 52 C4 C4 C4 Display area
EA        F8 Ø3 A.1(DSP) BØ
ED        F8 ØØ A.Ø(DSP) AØ
FØ CTI    8Ø B2
F2       §E2 2Ø AØ
F5       §E2 2Ø AØ
F8       §E2 2Ø AØ
FB       §3C FØ CTI
FD        1E                   Bump Real Time Clock
FE        3Ø E1 EXVDO
```

STIX - Page Ø1

ELF Sub Routine (112)

```
ØØ EXELF  D3                   Return to Main program
Ø1 ELF    F8 89 A.Ø(HIT) A8    Point SUBRT to HIT routine
Ø4        7A
Ø5 AROW   9A 32 28 BROW        If row A is empty go to row B
Ø8        F6 BA D8             Take 1 from row A and check for hit
ØB        31 ØØ EXELF          If Q set hit was found - return Q set
ØD        9A 32 25 1TOA        No hit - if row A empty - restore A
1Ø        F6 BA D8             Take 1 more from A and check for hit
13        31 ØØ EXELF          If Q set hit was found - return Q set
15        9A 32 22 2TOA        No hit - if row A empty - restore A
18        F6 BA D8             Take 1 more from A and check for hit
1B        31 ØØ EXELF          If Q set hit was found - return Q set
1D        9A FE
1F 3TOA   F9 Ø1 FE             Put 1 in A
22 2TOA   F9 Ø1 FE             Put 1 in A
25 1TOA   F9 Ø1 BA             Put 1 in A
28 BROW   8A 32 4B CROW        If row B is empty go to row C
2B        F6 AA D8             Take 1 from row B and check for hit
2E        31 ØØ EXELF          If Q set hit was found - return Q set
3Ø        8A 32 48 1TOB        No hit - if row B empty - restore B
33        F6 AA D8             Take 1 more from B and check for hit
36        31 ØØ EXELF          If Q set hit was found - return Q set
38        8A 32 45 2TOB        No hit - if row B empty - restore B
3B        F6 AA D8             Take 1 more from B and check for hit
3E        31 ØØ EXELF          If Q set hit was found - return Q set
4Ø        8A FE
42 3TOB   F9 Ø1 FE             Put 1 in B
45 2TOB   F9 Ø1 FE             Put 1 in B
48 1TOB   F9 Ø1 AA             Put 1 in B
4B CROW   9B 32 ØØ EXELF       If row C is empty return Q not set
4E        F6 BB D8             Take 1 from row C and check for hit
51        31 ØØ EXELF          If Q set hit was found - return Q set
53        9B 32 6B 1TOC        No hit - if row C empty - restore C
56        F6 BB D8             Take 1 more from C and check for hit
59        31 ØØ EXELF          If Q set hit was found - return Q set
5B        9B 32 68 2TOC        No hit - if row C empty - restore C
5E        F6 BB D8             Take 1 more from C and check for hit
61        31 ØØ EXELF          If Q set hit was found - return Q set
63        9B FE
65 3TOC   F9 Ø1 FE             Put 1 in C
68 2TOC   F9 Ø1 FE             Put 1 in C
6B 1TOC   F9 Ø1 BB             Put 1 in C
6E        3Ø ØØ EXELF          Return with Q not set
```

PTR Sub Routine (27)

```
7Ø EXPTR  D3                   Return to Main program
71 PTR    F8 1A A.Ø(DOR) A8    Point SUBRT to DOR routine
74        F8 ØØ A.Ø(DSP) AF 5D Point LINK to row A
78        9A B8 D8             Print row A
7B        F8 6Ø A.Ø(ROB) AF 5D Point LINK to row B
7F        8A B8 D8             Print row B
82        F8 CØ A.Ø(ROC) AF 5D Point LINK to row C
86        9B B8 D8             Print row C
89        3Ø 7Ø EXPTR
```

STIX - Page Ø1

Data Table (112)

```
8B W      C3 C3 C3 C3 DB FF E7 C3
93 L      6Ø 6Ø 6Ø 6Ø 6Ø 6Ø 6Ø 7E
9B P      7E 66 66 66 7E 6Ø 6Ø 6Ø
A3 ?      C3 66 Ø6 ØC 18 18 ØØ 18
AB BLK    ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
B3 HITØ   1F ØF ØØ
B6        ØF 1F Ø1
B9        Ø7 7F Ø7
BC        Ø3 3F Ø3
BF        Ø1 ØF ØØ
C2        ØØ 1F Ø1
C5 HIT1   1F 1F Ø1
C8        ØF ØF ØØ
CB        Ø7 3F Ø3
CE        Ø3 7F Ø7
D1        Ø1 1F Ø1
D4        ØØ ØF ØØ
D7 HIT2   1F 7F Ø7
DA        ØF 3F Ø3
DD        Ø7 1F Ø1
EØ        Ø3 ØF ØØ
E3        Ø1 7F Ø7
E6        ØØ 3F Ø3
E9 HIT3   1F 3F Ø3
EC        ØF 7F Ø7
EF        Ø7 ØF ØØ
F2        Ø3 1F Ø1
F5        Ø1 3F Ø3
F8        ØØ 7F Ø7
```

```
STIX - Page Ø2

Temporary Storage (4)

ØØ TEMS   ØØ ØØ ØØ ØØ              This is Stack and Temp storage area

STA Sub Routine (21)

Ø4 EXSTA  ED                       TEMP is stack
Ø5        D3                       Return to Main program
Ø6 STA    F8 C0 A.Ø(ROC) AF        LINK points to status position
Ø9        E9                       DAPT is stack
ØA        F8 Ø8 AC                 CTR = 8
ØD AGN    72 5F 2C 8C              Store character pointed to by
11        32 04 EXSTA              DAPT in status position of
13        8F FC Ø8 AF 3Ø ØD AGN    Display area

DOR Sub Routine (54)

19 EXDOR  D4                       Return to PTR sub routine
1A DOR    F8 Ø7 A9 8B FE           Set counters for 7 sticks
1F CKBT   FE AB 33 33 DOSTK        If bit is 1 make a stick
23 BLANK  F8 Ø8 AC                 If bit is Ø blank a stick
26 CLR    F8 ØØ 5F 2C 8C
2B        32 43 SEVN
2D        8F FC Ø8 AF 3Ø 26 CLR
33 DOSTK  F8 Ø8 AC
36 STK    F8 C0 5F 2C 8C
3B        32 43 SEVN
3D        8F FC Ø6 AF 3Ø 36 STK
43 SEVN   29 89 32 19 EXDOR        If row is done return
47        ØD FC Ø1 5D AF
4C        8B 3Ø 1F CKBT            If not done point to next stick

TAK Sub Routine (56)

4F EXTAK  D6                       Return to PER sub routine
5Ø TAK    1D ØD FA FØ 5D           Mask for number to take
55        FB 3Ø 32 78 IS3          Take = 3
59        ØD FB 2Ø 32 6D IS2       Take = 2
5E        ØD FB 1Ø 3A 64 INV       If take ≠ 1 take is invalid
63 IS1    8B FF Ø1 3B 64 INV       If row ≠ 1 or more take is invalid
68        8B F6 AB 3Ø 4F EXTAK     Take 1 and return
6D IS2    8B FF Ø3 3B 64 INV       If row ≠ 2 or more take is invalid
72        8B F6 F6 AB              Take 2 and
76        3Ø 4F EXTAK              Return
78 IS3    8B FF Ø7 3B 64 INV       If row ≠ 3 or more take is invalid
7D        8B F6 F6 F6 AB           Take 3 and
82        3Ø 4F EXTAK              Return
84 INV    7B 3Ø 4F EXTAK           Set Q to signal invalid

HIT Sub Routine (62)

87 EXHIT  ED                       TEMP is stack
88        D7                       Return to ELF sub routine
89 HIT    9B FB Ø7 32 A4 EQ3       Row C = 3
8E        9B FB Ø3 32 A0 EQ2       Row C = 2
93        9B FB Ø1 32 9C EQ1       Row C = 1
98 EQØ    F8 B3 A.Ø(HITØ)          Hit list is Ø
9A        3Ø A6 SEST
9C EQ1    F8 C5 A.Ø(HIT1)          Hit list is 1
9E        3Ø A6 SEST
AØ EQ2    F8 D7 A.Ø(HIT2)          Hit list is 2
A2        3Ø A6 SEST
A4 EQ3    F8 E9 A.Ø(HIT3)          Hit list is 3
A6 SEST   A9                       Hit list to DAPT
A7        B9                       DAPT is stack
A8        F8 Ø6 AC                 CTR = 6
AB NTHT   8A F3 3A BC BNC          Row B does not compare
AF        6Ø 9A F3 32 B9 HITT      Row B and A compare - set hit
B4        6Ø 9A F3 3A B7 EXHIT     Row B and A do not compare
B9 HITT   7B 3Ø B7 EXHIT           Set Q and return
BC BNC    6Ø 6Ø 6Ø                 Bump stack to next set
BF        1C 8C 3A AB NTHT
C3        3Ø B7 EXHIT              All dons - hit not found

PER Sub Routine (59)

C5 EXPER  D3                       Return to Main program
C6 PER    F8 5Ø A.Ø(TAK) A8        Point SUBRT to TAK routine
C9        7B
CA KYB    6C 64 2D                 Input Person's move
CD        3F CA KYB
CF        37 CF *                  Temp = Row
D1        2D FA ØF 5D              Temp + 1 = Take
D5        7A
D6        FB ØA 32 E8 ROWA         Take from row A
DA        FØ FB ØB 32 FØ ROWB      Take from row B
DF        FØ FB ØC 32 F8 ROWC      Take from row C
E4        1D 7B 3Ø C5 EXPER        Row is invalid
E8 ROWA   9A AB D8                 Check row A for valid take
EB        31 C5 EXPER              Take invalid - do not save
ED        BA 3Ø C5 EXPER           Take valid - save take
FØ ROWB   8A AB D8                 Check row B for valid take
F3        31 C5 EXPER              Take invalid - do not save
F5        AA 3Ø C5 EXPER           Take valid - save take
F8 ROWC   9B AB D8                 Check row C for valid take
FB        31 C5 EXPER              Take invalid - do not save
FD        BB 3Ø C5 EXPER           Take valid - save take
```

```
STIX - Page Ø3

Display Area (256)

ØØ DSP    ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
Ø8        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 3C
1Ø        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 66
18        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 66
2Ø        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 66
28        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 7E
3Ø        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 66
38        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 66
4Ø        ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
48        ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
5Ø        ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
58        ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
6Ø ROB    ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
68        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 7C
7Ø        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 66
78        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 66
8Ø        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 7C
88        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 66
9Ø        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 66
98        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 7C
AØ        ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
A8        ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
BØ        ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
B8        ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
CØ ROC    ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
C8        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 3E
DØ        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 6Ø
D8        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 6Ø
EØ        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 6Ø
E8        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 6Ø
FØ        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 6Ø
F8        ØØ ØØ ØØ ØØ ØØ ØØ ØØ 3E
```

```
STIX - Hex Listing

Page ØØ

ØØ   FB ØØ B1 B3   F8 Ø8 A3 D3   FB Ø1 B4 B7   B9 AD F8 Ø2
1Ø   B2 B5 B6 B8   BD A2 F8 E3   A1 F8 71 A4   F8 Ø6 A5 F8
2Ø   C6 A6 F8 Ø1   A7 F8 Ø3 BF   ED 61 F8 7F   8A F8 1F AA
3Ø   F8 Ø7 B8 D4   F8 A3 A9 D5   6C 64 2D 3F   38 37 3D 32
4Ø   9Ø F8 AB A9   D5 8E FA Ø3   5D FB Ø1 32   65 FØ FB Ø2
5Ø   32 5E FØ FB   Ø3 3A 45 9A   32 45 F6 BA   3Ø 6A 8A 32
6Ø   45 F6 AA 3Ø   6A 9B 32 45   F6 BB F8 ØØ   AE 8E FF 44
7Ø   3A 6D F8 35   AC 7B F8 55   FF Ø1 3A 78   7A F8 15 FF
8Ø   Ø1 3A 7F 2C   8C 3A 75 D4   F8 ØØ AE 8E   FF 22 3A 8B
9Ø   F8 9B A9 D5   D6 31 C3 D4   9A 5D 8A F4   5D 9B F4 FF
AØ   Ø1 32 AB 3B   B1 D7 39 45   7A 3Ø 6A F8   8B A9 D5 3Ø
BØ   B5 F8 93 A9   D5 F8 ØØ AE   8E FF FF 32   2A 3F B8 37
CØ   BF 3Ø 2A F8   A3 A9 D5 F8   Ø3 A9 F8 28   AC 7B 8C AB
DØ   2B 8B 3A DØ   39 CD 7A 2C   8C 3A CE 29   89 3A CA 3Ø
EØ   94 72 7Ø 22   78 22 52 C4   C4 C4 F8 Ø3   BØ F8 ØØ AØ
FØ   8Ø E2 E2 2Ø   AØ E2 2Ø AØ   E2 2Ø AØ 3C   FØ 1E 3Ø E1

Page Ø1

ØØ   D3 F8 89 A8   7A 9A 32 28   F6 BA D8 31   ØØ 9A 32 25
1Ø   F6 BA D8 31   ØØ 9A 32 22   F6 BA D8 31   ØØ 9A FE F9
2Ø   Ø1 FE F9 Ø1   FE F9 Ø1 BA   8A 32 4B F6   AA D8 31 ØØ
3Ø   8A 32 48 F6   AA D8 31 ØØ   8A 32 45 F6   AA D8 31 ØØ
4Ø   8A FE F9 Ø1   FE F9 Ø1 FE   F9 Ø1 AA 9B   32 ØØ F6 BB
5Ø   D8 31 ØØ 9B   32 6B F6 BB   D8 31 ØØ 9B   32 68 F6 BB
6Ø   D8 31 ØØ 9B   FE F9 Ø1 FE   F9 Ø1 FE F9   Ø1 BB 3Ø ØØ
7Ø   D3 F8 1A A8   F8 ØØ AF 5D   9A AB D8 F8   6Ø AF 5D 8A
8Ø   AB D8 F8 C0   AF 5D 9B AB   D8 3Ø 7Ø C3   C3 C3 C3 DB
9Ø   FF E7 C3 6Ø   6Ø 6Ø 6Ø 6Ø   6Ø 6Ø 7E 7E   66 66 66 7E
AØ   6Ø 6Ø 6Ø 3C   66 Ø6 ØC 18   18 ØØ 18 ØØ   ØØ ØØ ØØ ØØ
BØ   ØØ ØØ ØØ 1F   ØF ØØ ØF 1F   Ø1 Ø7 7F Ø7   Ø3 3F Ø3 Ø1
CØ   ØF ØØ ØØ 1F   Ø1 1F 1F Ø1   ØF ØF ØØ Ø7   3F Ø3 Ø3 7F
DØ   Ø7 Ø1 1F Ø1   ØØ ØF ØØ 1F   7F Ø7 ØF 3F   Ø3 Ø7 1F Ø1
EØ   Ø3 ØF ØØ Ø1   7F Ø7 ØØ 3F   Ø3 1F 3F Ø3   ØF 7F Ø7 Ø7
FØ   ØF ØØ Ø3 1F   Ø1 Ø1 3F Ø3   ØØ 7F Ø7 ØØ   ØØ ØØ ØØ ØØ

Page Ø2

ØØ   ØØ ØØ ØØ ØØ   ED D3 F8 C0   AF B9 F8 Ø8   AC 72 5F 2C
1Ø   8C 32 Ø4 8F   FC Ø8 AF 3Ø   ØD D4 F8 Ø7   A9 8B FE FE
2Ø   AB 33 33 F8   Ø8 AC F8 ØØ   5F 2C 8C 32   43 8F FC Ø8
3Ø   AF 3Ø 26 F8   Ø8 AC F8 C0   5F 2C 8C 32   43 8F FC Ø8
4Ø   AF 3Ø 36 29   89 32 19 ØD   FC Ø1 5D AF   8B 3Ø 1F D6
5Ø   1D ØD FA FØ   5D FB 3Ø 32   78 ØD FB 2Ø   32 6D ØD FB
6Ø   1Ø 3A 84 8B   FF Ø1 3B 84   8B F6 AB 3Ø   4F 8B FF Ø3
7Ø   3B 84 8B F6   F6 AB 3Ø 4F   8B FF Ø7 3B   84 8B F6 F6
8Ø   F6 AB 3Ø 4F   7B 3Ø 4F ED   D7 9B FB Ø7   32 A4 9B FB
9Ø   Ø3 32 AØ 9B   FB Ø1 32 9C   F8 B3 3Ø A6   F8 C5 3Ø A6
AØ   F8 D7 3Ø A6   F8 E9 A9 B9   F8 Ø6 AC 8A   F3 3A BC 6Ø
BØ   9A F3 32 B9   6Ø 9A F3 3A   87 7B 3Ø 87   6Ø 6Ø 6Ø 1C
CØ   8C 3A AB 3Ø   87 D3 F8 5Ø   A8 7B 6C 64   2D 3F CA 37
DØ   CF 2D FA ØF   5D 7A FB ØA   32 E8 FØ F8   Ø8 32 FØ FØ
EØ   FB ØC 32 F8   1D 7B 3Ø C5   9A AB D8 31   C5 BA 3Ø C5
FØ   8A AB D8 31   C5 AA 3Ø C5   9B AB D8 31   C5 BB 3Ø C5
```

T V T - 4 K

by
David Crawford

Uses improved Chip-8 graphics system (64 by 64)

TVT 4K was designed for the RCA Cosmac Vip for any user wishing to have the capabilities of a cheap TVT when inputting information. It was designed for a system with 4k of memory and the use of an ASCII keyboard through a machine language sub-routine which you write yourself.

The character, after being picked up, is first checked to see if it is a command character. (see list of command functions). If the character is not a command character it is stored in screen memory and then converted to its 4 by 7 bit pattern which is displayed on screen. Since all characters are stored in memory the user is able to use these characters in his programming. (Certain command characters are not stored, see list of command functions). Variable C is not changed so it is possible for the user to detect certain characters and even write a high-level language.

The screen is divided into 8 lines of 13 characters per line, the last character of which is reserved for a return. Screen memory will hold all of the characters which are on screen plus two lines which are off screen, reserved for scrolling.

The cursor condition consisting of white on black or black on white characters is changed by certain command codes which are shown in the list of command functions.

The command characters for the TVT 4K are as follows:

Cursor controls:

tab	right
backspace	left
\	up
line feed	down
control R or @	reverses cursor condition white on black or black on white. @ is stored in screen memory control R isn't.
control E	erases screen and screen memory.
return	stored in screen memory brings cursor to next line.
space	stored in screen memory increments cursor by one.
delete	deletes previous character
backslash	deletes from cursor to beginning of line, excluding cursor position.
control T	cursor to beginning of screen,- excluding cursor position.
control W	deletes from cursor to end of screen, excluding cursor position.

Scrolling:

If cursor is at bottom of screen	On receipt of a line feed or at return, the last line in ASCII memory is brought into view. The top line is stored as the first line in ASCII memory (out of view). The cursor will always end up at the lower left-hand corner of the screen.
If cursor is at top of screen.	On receipt of a \ the first line in ASCII memory is moved into view. The bottom line is stored as the last line in ASCII memory (out of view). The cursor will always end up at the upper left corner of the screen.

Variable Map

0	scratch
9	scratch
A	X
B	Y
C	contains ASCII character
D	pointer in screen memory
E	cursor condition flag

Terms

screen memory	memory used to store ASCII characters that are being displayed on screen. (2 lines reserved for scrolling)
display memory	memory used by Chip-8 in its graphics mode. 512 bytes.
ASCII memory	memory used to store ASCII character patterns.

```
ADDR CODE                    COMMENT
0200 12 02                   Initialize
0202 6A 00                   X
0204 6B 00                   Y
0206 6D 0D                   memory pointer
0208 6E 00                   cursor flag
020A 0D 60                   erase display
020C 2D 08                   display cursor (X,Y)
020E 12 10                   goto 210

                             Wait for keyboard input
0210 6C 00                   VC contains ASCII
                             character
0212 AD FC                   I=0DFC
0214 0D 38-------------------check keyboard for input
0216 4C 00                   (user's subroutine)
0218 12 10                   no-branch to 210
021A 2A 84                   yes-gosub 0A84
021C 12 A0                   goto 2A0
                             (beginning of user
                             programs)
                                 .           .
```

Subroutine to display all characters in screen memory, excluding first and last lines.

```
ADDR CODE                    COMMENTS
0220 0D 60                   erase display
0222 6A 00                   X=00
0224 6B 00                   Y=00
0226 6D 0D                   pointer=0D
0228 22 90                   gosub 290(sub. to read
                             character
022A 40 CD                   from screen memory).
022C 12 46
022E 40 00                   check for nondisplay-
                             able characters
0230 12 46                   00, space, @, return
0232 40 20
0234 12 46
0236 40 40
0238 12 70
023A 39 00
023C 2D 08
023E AD F0                   I=DF0
0240 0D 10                   get ASCII character
                             pattern
0242 AD 30                   I=D30
0244 DA B7                   display character
0246 4A 37                   check if end of line
0248 12 50                   yes-goto 250
024A 7A 05                   no-increment X by 05
024C 7D 01                   increment pointer VD by
                             01
024E 12 28                   goto 228
0250 7D 02                   increment pointer VD by
                             02
0252 6A 00                   set X to 00
0254 4B 38                   check if bottom of screen
0256 12 5C                   yes-goto 25C
0258 7B 08                   no-Y+08
025A 12 28                   goto 228
025C 6A 00                   At bottom of screen so:
                             X=00
025E 3C 60                   check VC if 60
0260 12 68                   no-goto 268
0262 6D 1A                   yes-VD=1A
0264 6B 08                   Y=08
0266 12 6C                   goto 26C
0268 6D 5B                   ASCII char. not 60 so:
                             VD=5B
026A 6B 30                   Y=30
026C 2D 08                   display cursor
026E 00 EE                   return
```

Software switch used to reverse character condition. Used only in previous subroutine.

```
ADDR CODE                    COMMENT
0270 39 00                   V9 is character condi-
                             tion flag
0272 69 FF                   V9=FF
0274 79 01                   V9+01
0276 12 46
```

Program block to erase first or last line in screen memory.

```
ADDR CODE                    COMMENT
027A AA 00                   I=A00--entrance to
                             erase first line
027C 6D 00                   VD=00
027E 4D 0D                   check pointer VD if 0D
0280 12 20                   yes-goto 220(sub. to
                             display all ASCII char-
                             acters.)
0282 60 00                   no-V0=00
0284 F0 55                   V0=MI
0286 7D 01                   VD+01
0288 12 7E                   goto 27E
028A AA 76                   I=A76--entrance to
                             erase last line
028C 12 7C                   goto 27C
```

Subroutine to read a character from screen memory according to pointer VD. V0=char.(MI)

```
ADDR CODE                    COMMENT
0290 AA 00                   I=0A00
0292 FD 1E                   I=I+VD
0294 F0 65                   V0=MI
0296 00 EE                   return
```

Subroutine to write a character into screen memory according to pointer VD. char.(MI)=V0

```
ADDR CODE                    COMMENT
0298 AA 00                   I=0A00
029A FD 1E                   I=I+VD
029C F0 55                   MI=V0
029E 00 EE                   return
```

Beginning of user programming space.

```
ADDR CODE                    COMMENT
02A0-08FF
```

Temporarily:

```
ADDR CODE                    COMMENT
02A0 12 10                   goto 210
```

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

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

ASCII Character Patterns (0900-09FF)

The following character patterns are stored in memory in eight byte blocks, with each byte being split in half to represent different characters. For example: AF the A represents the high half-byte and the F representing the low half-byte.

Memory Address	Character H	Character L	Pattern
0900	space	@	00 00 00 00 00 00 00 00
0908	!	A	00 26 29 2F 09 29 00 00
0910	"	B	50 5E 05 07 05 0E 00 00
0918	#	C	00 67 F8 68 F8 67 00 00
0920	$	D	60 7E 85 65 15 EE 60 00
0928	%	E	00 9F 38 6E C8 9F 00 00
0930	&	F	00 0F 08 0E 08 08 00 00
0938	'	G	40 47 08 0B 09 07 00 00
0940	(	H	20 4A 4A 4E 4A 4A 20 00
0948	)	I	40 2E 24 24 24 2E 40 00
0950	*	J	00 01 51 29 59 06 00 00
0958	+	K	00 09 2A 7C 2A 09 00 00
0960	,	L	00 08 08 08 48 4E 80 00
0968	-	M	00 09 0F 7F 09 09 00 00
0970	.	N	00 09 0D 0F 0B 49 00 00
0978	/	O	00 16 39 69 9C 86 00 00
0980	0	P	00 FF 99 9F 98 F8 00 00
0988	1	Q	00 26 69 29 2B 76 00 00
0990	2	R	00 EF 19 6F 8A F9 00 00
0998	3	S	00 E7 18 76 11 EE 00 00
09A0	4	T	00 AE A4 F4 24 24 00 00
09A8	5	U	00 F9 89 F9 19 F6 00 00
09B0	6	V	00 FA 8A FA 9A F4 00 00
09B8	7	W	00 F9 99 1F 1F 19 00 00
09C0	8	X	00 6A 9A 65 9A 6A 00 00
09C8	9	Y	00 FA 9A F4 14 F4 00 00
09D0	:	Z	00 07 41 02 44 07 00 00
09D8	;	[	0E 08 48 08 48 48 8E 00
09E0	<	\	00 20 40 80 40 20 00 00
09E8	=	]	0E 02 F2 02 F2 02 0E 00
09F0	>	↑	06 49 20 10 20 40 00 00
09F8	?	_	00 60 90 20 00 20 00 00

0A00-0A83 screen memory

Control Character detection ladder. (ASCII character contained in VC)

ADDR	CODE	COMMENT
0A84	4C 09	check if VC=09 TAB
0A86	1A E4	yes-goto AE4
0A88	4C 08	no-check if VC=08 BACKSPACE
0A8A	1A F2	yes-goto AF2
0A8C	4C 60	no-check if VC=60 \
0A8E	1B 00	yes-goto B00
0A90	4C 0A	no-check if VC=0A LINE FEED
0A92	1B 10	yes-goto B10
0A94	4C 40	no-check if VC=40 @
0A96	1B 20	yes-goto B20
0A98	4C 12	no-check if VC=12 CONTROL R
0A9A	1B 28	yes-goto B28
0A9C	4C 05	no-check if VC=05 CONTROL E
0A9E	1B 30	yes-goto B30
0AA0	4C 0D	no-check if VC=0D RETURN
0AA2	1B 50	yes-goto B50
0AA4	4C 7F	no-check if VC=7F DELETE
0AA6	1B 70	yes-goto B70
0AA8	4C 5C	no-check if VC=5C
0AAA	1B 96	yes-goto B96
0AAC	4C 11	no-check if VC=11 CONTROL Q
0AAE	1B A0	yes-goto BA0
0AB0	4C 17	no-check if VC=17 CONTROL W
0AB2	1B B0	yes-goto BB0
		no-continue on to AB4

Program block to display character. Uses subroutine at AD6 to store character in screen memory.

ADDR	CODE	COMMENT
0AB4	4A 3C	check if end of line
0AB6	00 EE	no-return
0AB8	2A D6	yes-store character in screen memory
0ABA	AD F9	I=DF9(V9 in Chip-8)
0ABC	0D 10	MLS to get ASCII character pattern
0ABE	AD 30	I=D30(temporary storage for pattern)
0AC0	DA B7	erase old character
0AC2	4E 00	check cursor condition
0AC4	2D 08	0-turn on cursor;1-leave cursor off
0AC6	AD FC	I=DFC(VC in Chip-8)
0AC8	0D 10	MLS to get ASCII char. pattern
0ACA	AD 30	I=D30(temporary storage for pattern)
0ACC	DA B7	display new character
0ACE	7A 05	X+05
0AD0	2D 08	display cursor
0AD2	00 EE	return

Subroutine to pick up old character in screen memory and put in new one. V9=old, VC=new

ADDR	CODE	COMMENT
0AD6	22 90	sub.-pick up character
0AD8	89 00	V9=V0
0ADA	80 C0	V0=VC
0ADC	22 98	sub.-put in new character
0ADE	7D 01	VD+01
0AE0	00 EE	

The following program blocks contain programming for the different control keys.

TAB

ADDR	CODE	COMMENT
0AE4	4A 3C	check if end of line
0AE6	00 EE	yes-return
0AE8	2D 08	no-erase cursor
0AEA	7A 05	X+05
0AEC	2D 08	display cursor
0AEE	7D 01	VD+01
0AF0	00 EE	return

BACKSPACE

ADDR	CODE	COMMENT
0AF2	4A 00	check if beginning of line
0AF4	00 EE	yes-return
0AF6	2D 08	no-erase cursor
0AF8	7A FB	X-05
0AFA	2D 08	display cursor
0AFC	7D FF	VD-01
0AFE	00 EE	return
0B00	4B 00	check if top of screen
0B02	2B EA	yes-gosub BEA(scroll down)
0B04	2D 08	no-erase cursor
0B06	7B F8	VB-08
0B08	2D 08	display cursor
0B0A	7D F3	VD-0D
0B0C	00 EE	return

LINE FEED

ADDR	CODE	COMMENT
0B10	4B 38	check if bottom of screen
0B12	2B D4	yes-gosub BD4(scroll up)
0B14	2D 08	no-erase cursor
0B16	7B 08	VB+08
0B18	2D 08	display cursor
0B1A	7D 0D	VD+0D
0B1C	00 EE	return

@ or CONTROL R

ADDR	CODE	COMMENT
0B20	4A 3C	check if end of line
0B22	00 EE	yes-return
0B24	2A D6	no-store VC in screen memory
0B26	7D FF	VD-01
0B28	3E 00	check cursor flag--entrance for C.R.
0B2A	6E FF	1- VE=FF
0B2C	7E 01	0- VE+01
0B2E	1A E4	goto AE4 TAB

CONTROL E

ADDR	CODE	COMMENT
0B30	0D 60	erase display
0B32	6A 00	X=00
0B34	6B 00	Y=00
0B36	6E 00	VE=00
0B38	6D 00	VD=00
0B3A	60 00	V0=00
0B3C	AA 00	I=0A00
0B3E	4D 84	check if VD=84
0B40	1B 48	yes-goto B48
0B42	F0 55	no-MI=V0
0B44	7D 01	VD+01
0B46	1B 3E	goto B3E
0B48	6D 0D	VD=0D
0B4A	2D 08	display cursor
0B4C	00 EE	return

RETURN

ADDR	CODE	COMMENT
0B50	2A D6	store ASCII in screen memory
0B52	7D FF	VD-01
0B54	4B 38	check if bottom of screen
0B56	2B D4	yes-gosub BD4(scroll up)
0B58	2D 08	no-erase cursor
0B5A	7D 01	VD+01
0B5C	7B 08	VB+08
0B5E	4A 3C	check if A=3C
0B60	1B 68	yes-branch to B68
0B62	7A 05	no-VA+05
0B64	7D 01	VD+01
0B66	1B 5E	goto B5E
0B68	6A 00	X=00
0B6A	2D 08	display cursor
0B6C	00 EE	return

DELETE

ADDR	CODE	COMMENT
0B70	4A 00	check if beginning of line
0B72	00 EE	yes-return
0B74	2D 08	no-erase cursor
0B76	7D FF	VD-01
0B78	6C 00	VC=00
0B7A	2A D6	store character in screen memory
0B7C	7A F8	X-05
0B7E	AD F9	I=DF9
0B80	0D 10	MLS to get ASCII char. pattern

ADDR	CODE	COMMENT
0B82	49 0D	check if V9=0D
0B84	1B 8A	yes-goto B8A
0B86	AD 30	no-I=D30
0B88	DA B7	erase character
0B8A	2D 08	display cursor
0B8C	4F 00	check if flag VF=00
0B8E	1B 92	yes-goto B92
0B90	2D 08	no-erase cursor
0B92	7D FF	VD-01
0B94	00 EE	return
0B96	4A 00	check if beginning of line
0B98	00 EE	yes-return
0B9A	2B 70	no-gosub B70 DELETE
0B9C	1B 96	goto B96

CONTROL Q

ADDR	CODE	COMMENT
0BA0	2B 96	gosub B96
0BA2	4B 00	check if top of screen
0BA4	00 EE	yes-return
0BA6	2D 08	no-erase cursor
0BA8	7B F8	Y-08
0BAA	6A 41	X=41
0BAC	2D 08	display cursor
0BAE	1B A0	goto BA0

CONTROL W

ADDR	CODE	COMMENT
0BB0	3A 3C	check if end of line
0BB2	1B C8	no-goto BC8
0BB4	3B 38	yes-check if bottom of screen
0BB6	1B C0	no-goto BC0
0BB8	2D 08	yes-erase cursor
0BBA	6A 3C	X=3C
0BBC	2D 08	display cursor
0BBE	00 EE	return
0BC0	2D 08	erase cursor
0BC2	6A FB	X=FB
0BC4	7B 08	Y+08
0BC6	2D 08	display cursor
0BC8	2D 08	erase cursor
0BCA	7A 0A	X+0A
0BCC	7D 01	VD+01
0BCE	2B 78	gosub B78 DELETE
0BD0	1B B0	

Program block to scroll up screen memory.

ADDR	CODE	COMMENT
0BD4	2D 08	erase cursor
0BD6	6D FF	VD=FF
0BD8	7D 0E	VD+0E
0BDA	4D 84	check if VD=84
0BDC	12 8A	yes-goto 28A
0BDE	22 90	no-gosub 290(sub. to read ASCII char. from screen memory)
0BE0	7D F3	VD-0D
0BE2	22 98	gosub 298(sub. to write ASCII char. into screen memory)
0BE4	1B D8	goto BD8

Program block to scroll down screen memory.

ADDR	CODE	COMMENT
0BEA	2D 08	erase cursor
0BEC	6D 84	VD=84
0BEE	7D F2	VD-0E
0BF0	4D FF	check if VD=FF
0BF2	12 7A	yes-goto 27A
0BF4	22 90	no-gosub 290(sub. to read ASCII char. from screen memory)
0BF6	7D 0D	VD+00
0BF8	22 98	gosub 298(sub. to write ASCII char. into screen memory)
0BFA	1B EE	goto BEE

Subroutine to turn cursor on or off.

ADDR	CODE	COMMENT
0D00	F8 F8	cursor pattern
0D02	F8 F8	
0D04	F8 F8	
0D06	F8 00	
0D08	AD 00	I=D00--entrance to sub-routine
0D0A	DA B7	display cursor pattern at (A,B)
0D0C	00 EE	return

Note: page C (0C00-0CFF) has been left available for user programming in Chip-8 or machine language. Location 0200 has been left blank for the use of a jump instruction if needed.

Machine Language Subroutine to get ASCII character pattern.

ADDR	CODE	COMMENT
0D10	F8 09	ASCII character pattern page #
0D12	BC	
0D13	92	
0D14	BE	
0D15	F8 30	paragraph # for temporary storage of
0D17	AE	ASCII character pattern. Must be
0D18	0A	in page D regardless of the position
0D19	FE	of this subroutine.
0D1A	FE	
0D1B	FE	
0D1C	AC	
0D1D	4C	
0D1E	33 26	branch to: 26
0D20	FA	
0D21	0F	
0D22	FE	
0D23	FE	
0D24	FE	
0D25	FE	
0D26	FA	
0D27	F0	
0D28	5E	
0D29	1E	
0D2A	8E	
0D2B	FB 37	stopping point for temp. storage of ASCII char.
0D2D	3A 1D	branch to: 1D
0D2F	D4	return

Machine language subroutine you-write-yourself to see if there is an input from an ASCII keyboard. If there is, then the ASCII character is put into MI. I is already set. 0D38-0D3F

Annotated Bibliography Additions

by

John Guarini

The following article references are to be additions to the Annotated Bibliography published in Questdata Volume 2, Issue 4.

1. Bregoli, Larry "The MM57109 Number Cruncher", Kilobaud Magazine, #33, Pg. 38 (September 1979).

2. Cheairs, Steven L. "Nom Card for the 1802", Part I, Radio-Electronics, pg. 45 (December 1978).

3. Cheairs, Steven L. "Nom Card for the 1802", Part II, Radio-Electronics (January 1979).

4. Crawford, Tom "Tiny Basic Square Root Routine", Kilobaud/Microcomputing, #38, pg. 172 (February 1980).

5. Duntemann, Jeff "The Cosmac Doodler", Byte Magazine, Volume 5, #5, Pg. 214 (May 1980).

6. RCA Publication MPM-202. "Timesharing Manual for the RCA CDP 1802 Cosmac Microprocessor".

7. RCA Publication MPM-203. "Evaluation Kit Manual for the RCA CDP 1802 Cosmac Microprocessor".

8. RCA Publicaion MPM-206. "Binary Arithmetic Subroutines for RCA Cosmac Microprocessors".

9. RCA Publication MPM-208. "Operator Manual for the RCA Cosmac Development System".

10. Strope, Gerald "Machine-Language Techniques for the 1802", Kilobaud/Microcomputing, #46, Pg. 192 (October 1980).

NOTE: In the Annotated Bibliography printed in Questdata Volume 2, Issue 4, part of a line was deleted from one of the references. In the article reference by Paul Wasserman titled "A Floating Point Subroutine Package for the 1802", the last line should have read, "Typo: The second calling address in Fig. 3 should read 02E9". We apologize for this ommission.

THE HAMURABI GAME

```
10 REM                  HAMURABI GAME
20 REM
30 REM              (Adapted by Fred Hannan)
40 REM
50 REM         Hamurabi has its roots in several claimed creations but
60 REM         I believe the original version for microcomputers was
70 REM         written by CREATIVE COMPUTING.
80 REM         This version was adapted from many different versions
90 REM         published in several magazines and books.
100 REM
110 REM         The instructions included in the game explain the rules
120 REM         and objectives.
130 REM
140 DEFINT Z
150 CLS
160 INPUT "HAMURABI GAME - Do you need instructions (YES or NO)"I$
170 IF MID$(I$,1,1)="N" GOTO 340
180 IF MID$(I$,1,1)="n" GOTO 340
190 PRINT "HAMURABI - You are the much loved king of the ancient"
200 PRINT "kingdom of SUMERIA."
210 PRINT
220 PRINT "The object is to keep your kingdom growing by raising grain,"
230 PRINT "using the grain to feed your people and the strangers who"
240 PRINT "come to your city, keep enough grain for seed for next years"
250 PRINT "crop. In addition, you will buy land for planting crops in"
260 PRINT "times of plenty or sell your farm land in bad times to"
270 PRINT "get much needed grain."
280 PRINT TAB(10);"(The plague strikes occasionally)"
290 PRINT "You buy or sell land at its current value, each person"
300 PRINT "needs 20 bushels for food, each person can plant a"
310 PRINT "maximum of 10 acres, and each acre costs 1/2 bushel to plant."
320 PRINT "(If you want to quit, sell all of your land)"
330 PRINT : INPUT "Press 'RETURN' to begin your reign."B$
340 A=100:B=5:C=0:D=2800:E=200:F=3:G=3000:H=1000:J=1:L=1
350 CLS:P=0: PRINT "HAMURABI, I beg to report that in year ";L;":"
360 PRINT
370 IF C=1 GOTO 390
380 PRINT C;" people starved, and ";: GOTO 400
390 PRINT " 1 person starved, and ";
400 IF B=1 GOTO 420
410 PRINT B;" people came to the city. ": GOTO 430
420 PRINT "1 person came to the city."
430 IF J>0 GOTO 460
440 A=A-(A/2)
450 PRINT "The plague killed half the people."
460 PRINT "The population is now ";A;"."
470 PRINT "We harvested ";G;" bushels at ";F;" bushels per acre."
480 PRINT "Rats destroyed ";E;" bushels, leaving ";D;" bushels in storage."
490 PRINT "The city owns ";H;" acres of land."
500 K=16+RND(6): PRINT "Land is worth ";K;" bushels per acre."
510 PRINT : PRINT "HAMURABI . . ."
520 PRINT
530 INPUT "     Buy how many acres"I: IF I=0 GOTO 570
540 J=I*K: IF J<=D GOTO 560
550 GOSUB 910: GOTO 530
560 D=D-J:H=H+I
570 PRINT : PRINT "   *  You are buying ";I;" acres."
580 IF I>0 GOTO 670
590 PRINT
600 INPUT "     Sell how many acres"I: IF I=0 GOTO 640
610 IF I<H GOTO 640
620 IF I=H GOTO 950
630 GOSUB 910: GOTO 600
640 P=1: PRINT : PRINT "   *  You are selling ";I;" acres."
650 GOTO 660
660 H=H-I:D=D+K*I
670 REM
680 PRINT
690 INPUT "     How many bushels shall we distribute as food"I
700 IF I<D GOTO 720
710 GOSUB 910: GOTO 690
720 D=D-I:C=A-(I/20):B=0: IF C>0 GOTO 740
730 B=-C/2:C=0
740 PRINT
750 PRINT "   *  You are distributing ";I;" bushels."
760 PRINT
770 INPUT "     How many acres shall we plant "I
780 IF I>H GOTO 800
790 J=I/2: IF J<=D GOTO 810
800 GOSUB 910: GOTO 770
810 IF I>10*A GOTO 800
820 D=D-J:F=RND(5)+1:G=F*I
830 E=(D+G)*7/100
840 E=E*RND(2)
850 D=D-E+G:J=RND(11)-1
860 B=B+(5-F)*(D/600)+1
870 IF B<=50 GOTO 890
880 B=50
890 IF B<0 THENB=0
900 A=A+B-C:L=L+1: GOTO 350
910 PRINT
920 PRINT "--->  HAMURABI !! Think again -- you only have"
930 PRINT "--->  ";A;" people, ";H;" acres, and ";D;" bushels in storage."
940 RETURN
950 CLS: PRINT "You sold all of your land."
960 PRINT "THE GAME IS OVER - GOODBYE!"
970 CLS
```

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

PARTIAL DISPLAY SUBROUTINES

by
Ken Mantei

The graphics test program displaying the Enterprise Spaceship (Popular Electronics, pg. 42, 44, July 1977, and in Netronics Assembly Manual) also shows the program bytes. If the graphic-modifying MAIN subroutine is not needed, space becomes available for modified INTERRUPT subroutines which display only the graphic.

Running from 0050-00FF, the graphic consists of 22 8-byte rows. Each row is scanned 4 times for a total of 88 desired graphic scans. To blank the screen during the first 40 (=128-88) scans, two 8-byte rows of zeros must be scanned 20 (=14 in hex) times. These are conveniently supplied in the original programs at 0040-004F; they could however appear anywhere on the same page as the graphic.

In the Top Blank program, the INTERRUPT subroutine puts half the number of blank scans in an unused register which is decremented and tested during blank scans. When this register reaches zero, the program jumps to load the graphic display address into R0. Two zero rows seem to be necessary since the decrementing, comparing, and repointing of R0 takes more time than is available in one between-scan period.

To display the graphic with unused lines at the bottom blanked, it is necessary to place the graphic memory so that it can be immediately followed, on the same page, by an 8-byte row of zero bytes. The graphic, originally at 0050-00FF, must be moved to 0048-00F7. 00F8-00FF must be zeros to be repeatedly scanned during blanking.

Program Bottom Blank implements this. A minor complication arises, due to the fact that during the last four scans the video chip pulls the display status line low. In the orignial program, the 3C instruction sensed this only as the last 4 scans were completed. In Bottom Blank this "almost-done" signal is sensed with 3 scans still to go. So the FOUR subroutine is added to keep repointing R0 to zeros until the display status line again goes high, signaling that the display window is finally closed. Without FOUR, the bytes following the zero row get displayed.

Once the logic of these modifications becomes clear, one should be able to write interrupt routines to blank the screen at the top, bottom, or top and bottom. Of course, users with more than one page of memory can dedicate one page for display, blank anywhere easily with zeros, and use the standard interrupt routine.

Top Blank

```
ADDR CODE              LABEL        COMMENT
0000 90 B1 B2 B3 B4                 B4 unnecessary
                                    in this mod.
0005 F8 39* A3                      *Address changes
                                    from P.E.
0008 F8 3F* A2                      original Graphics
                                    Test Program
000B F8 11 A1 D3
000F 72 70             RETURN
0011 22 78 22 52       INTERRUPT
0015 C4 C4 C4

                                    Bytes from 0018
                                    to 003F are
                                    different from
                                    original Graphics
                                    Test Program
0018 F8 14 A5                       #Blank scans/2
                                    (in hex) to R5.0
001B F8 00 B0                       Address of 2 zero
                                    lines to R0
```

```
ADDR CODE              LABEL      COMMENT
001E F8 40 A0          BLANK
---- DMA zero row 1
0021 25 85 32 27

---- DMA zero row 2

0025 30 1E

---- DMA zero row 2
0027 F8 50 A0          PIC
002A E2                REFRESH
---- DMA Scan 1

002B E2 20 A0

---- DMA Scan 2

002E E2 20 A0
---- DMA Scan 3
0031 E2 20 A0
---- DMA Scan 4
0034 80 3C 2A
0037 30 0F
0039 E2 69 30 3B       MAIN

0030 XX XX XX

0040 00 00 00 etc.     ZERO

0050 7B DE 0B etc.     GRAPHIC
```

Countdown blank scans by 2 and jump on zero to PIC to display graphics beginning at 0050

Scan each 8-byte line four times as in original Graphics Test Program

Check for end of display window. Do-nothing MAIN loop

3D to 3F is stack area

Bytes 0040 - 00FF containing graphic same as original P.E. Spaceship Program

Bottom Blank

```
ADDR CODE              LABEL      COMMENT
0000 90 B1 B2 B3 B4

0005 F8 39* A3

0008 F8 3F* A2

000B F8 11 A1 D3
000F 72 70             RETURN
0011 22 78 22 52       INTERRUPT
0015 C4 C4 C4
0018 F8 00 B0
001B F8 48 A0 E2

001F 80                REFRESH
```

B4 unnecessary in this mod.

*Address changes from P.E. original Graphics Test Program

Spaceship graphic moved to begin at 0048 instead of 0050

```
ADDR CODE              LABEL      COMMENT
---- DMA scan 1
0020 E2 20 A0
---- DMA scan 2

0023 E2 20 A0

---- DMA scan 3

0026 E2 20 A0
---- DMA scan 4
0029 FB F0

002B 3A 1F

002D 80

---- DMA zero row

002E 20 A0             BLANK
0030 3C 2E

---- DMA zero row
0032 20 A0 34 32       FOUR

---- DMA zero row

0036 30 0F 00
0039 E2 69 30 3B       MAIN

003D xx xx xx

0048 7B DE DB etc.     GRAPHIC

00F8 00 00 00 etc.     ZERO
```

Scan each 8-byte line four times as in original Graphics Test Program

Makes D=0 if last line.

Get new display line if D not = 0

Save in D address of 8 zero bytes that must immediately follow last line of graphic.

Repoint to zero row until EF1 shows we're in last 4 scans.

Repoint to zero row until EF1 shows last 4 scans done.

Do-nothing MAIN loop.

3D to 3F is stack area.

40 to 47 is unused

Bytes 0048-00F7 contain graphic found as 0050-00FF in original.

Zero row for blanking.

COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC

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

ADDRESS CORRECTION REQUESTED

BULK RATE
U.S. Postage Paid
QUEST
Electronics
Permit No. 649
Santa Clara, CA