

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 program 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.

STIX - Page 00

MAIN Program (225)

Execution begins at 00 00

```

00 INIT F8 00 B1 B3 Initialize registers
04 F8 00 A.0(MAIN) A3
07 D3
08 MAIN F8 01 B4 B7 B9 AD
0E F8 02 B2 B5 B6 B8 BD A2
16 F8 03 A.0(VDO) A1 R1 is Video interrupt pointer
19 F8 04 A.0(PTR) A4 R4 is Print routine pointer
1C F8 05 A.0(STA) A5 R5 is Status routine pointer
1F F8 06 A.0(FER) A6 R6 is Person routine pointer
22 F8 07 A.0(ELF) A7 R7 is Elf routine pointer
25 F8 08 A.1(DSP) B8 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
30 F8 07 BB Row C = 3
33 D4 Print
34 F8 A3 A.0(?) A9 D5 Status = ?
38 INA 6C 04 2D If input is 00 Person is first
3B F8 3E INA If input is FF Elf is first
3D F8 00 *
3F 32 00 PREFST
41 F8 AB A.0(BLK) A9 D5 Elf is first - blank status
45 RAND 0E FA 03 5D Use Real Time Clock to
49 F8 01 32 65 CTX Find random row
4D F8 FB 02 5E BTK
52 F8 FB 03 3A 4E RAND
57 ATK 9A 32 45 RAND If RTCK not A, B, or C, try again
5A F8 BA 30 6A DELAY Random row is A - if A = 0 try again
5E BTK 8A 32 45 RAND A ≠ 0 so take 1 from row A
61 F8 6A 30 6A DELAY Random row is B - if B = 0 try again
65 CTX 9B 32 45 RAND B ≠ 0 so take 1 from row B
68 F8 BB 4E RAND Random row is C - if C = 0 try again
6A DELAY F8 00 AE C ≠ 0 so take 1 from row C
6D TIM 8E FF 44 3A 6D TIM Short Delay
72 BEEP F8 35 AC Make Beep
75 MORB 7B FB 55
78 ON FF 01 3A 7B ON
7C 7A FB 15
7F OFF FF 01 3A 7F OFF
93 2C 8C 3A 7E MORB
97 D4
98 PAUSE F8 00 AE
9B PTM 8E FF 22 3A 8B PTM
9D PREFST F8 2B A.0(?) A9 D5
9A PRMV D6
95 31 C3 INVD
97 D4
98 LWH 9A 5D 8A F4 5D 9B F4
9F FF 01 32 AB WIN
A3 3B 81 LOSE
A5 D7
A6 39 45 RAND
A8 7A 30 6A DELAY
AB WIN F8 8B A.0(W) A9 D5
AF 30 8E WAIT
B1 LOSE F8 03 A.0(L) A9 D5
B5 WAIT F8 00 AE
B8 WAT 8E FF FF 32 2A STPLA
BD 3F 8B WAT
BF 37 8F *
C1 30 2A STPLA
C3 INVD F8 A3 A.0(?) A9 D5
C7 ZAP F8 03 A9
CA RPT F8 2B AC
CD SET 7B
CE SIT 8C AB
D0 CTD 2B 8B 3A D0 CTD
D4 39 CD SET
D6 7A
D7 2C 8C 3A CE SIT
D8 29 89 3A CA RPT
D9 30 94 PRMV

```

VIDEO Interrupt Routine (31)

```

E1 EXVDO 72 70 This routine points DMA to
E3 VDO 22 78 22 52 C4 C4 C4 Display area
EA F8 03 A.1(DSP) B0
ED F8 00 A.0(DSP) A0
F0 CTI 80 E2
F2 1E2 20 A0
F5 1E2 20 A0
F8 1E2 20 A0
FB 1E2 20 CTI
FD 1E Bump Real Time Clock
FE 30 E1 EXVDO

```

STIX - Page 01

ELF Sub Routine (112)

```

00 EXELF D3
01 ELF F8 89 A.0(HIT) A8 Return to Main program
04 7A Point SUBRT to HIT routine
05 AROW 9A 32 28 BROW If row A is empty go to row B
08 F6 BA 08 Take 1 from row A and check for hit
0B 31 00 EXELF If Q set hit was found - return Q set
0D 9A 32 25 1TOA No hit - if row A empty - restore A
10 F6 BA 08 Take 1 more from A and check for hit
13 31 00 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 08 Take 1 more from A and check for hit
1B 31 00 EXELF If Q set hit was found - return Q set
1D 9A FE
1F 3TOA F9 01 FE Put 1 in A
22 2TOA F9 01 FE Put 1 in A
25 1TOA F9 01 BA Put 1 in A
28 BROW 8A 32 4B CROW If row B is empty go to row C
2B F6 AA 08 Take 1 from row B and check for hit
2E 31 00 EXELF If Q set hit was found - return Q set
30 8A 32 46 1TOB No hit - if row B empty - restore B
33 F6 AA 08 Take 1 more from B and check for hit
36 31 00 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 08 Take 1 more from B and check for hit
3E 31 00 EXELF If Q set hit was found - return Q set
40 8A FE
42 3TOB F9 01 FE Put 1 in B
45 2TOB F9 01 FE Put 1 in B
48 1TOB F9 01 AA Put 1 in B
4B CROW 9B 32 00 EXELF If row C is empty return Q not set
4E F6 BB 08 Take 1 from row C and check for hit
51 31 00 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 08 Take 1 more from C and check for hit
59 31 00 EXELF If Q set hit was found - return Q set
5B 9B 32 66 2TOC No hit - if row C empty - restore C
5E F6 BB 08 Take 1 more from C and check for hit
61 31 00 EXELF If Q set hit was found - return Q set
63 9B FE
65 3TOC F9 01 FE Put 1 in C
68 2TOC F9 01 FE Put 1 in C
6B 1TOC F9 01 BB Put 1 in C
6E 30 00 EXELF Return with Q not set

```

PTR Sub Routine (27)

```

70 EXPTR D3 Return to Main program
71 PTR F8 1A A.0(DOR) A8 Point SUBRT to DOR routine
74 F8 00 A.0(DSP) AF 5D Point LINK to row A
78 9A AB 08 Print row A
7B F8 60 A.0(ROB) AF 5D Point LINK to row B
7F 8A AB 08 Print row B
82 F8 00 A.0(ROC) AF 5D Point LINK to row C
86 9B AB 08 Print row C
89 30 70 EXPTR

```

STIX - Page 01

Data Table (112)

```

8B W C3 C3 C3 C3 DE FF E7 C3
93 L 60 60 60 60 60 60 7E
9B P 7E 66 66 66 7E 60 60 60
A3 ? C3 66 66 66 18 18 00 18
AB BLK 00 00 00 00 00 00 00 00
B3 HIT0 1F 0F 00
B6 0F 1F 01
B9 07 7F 07
BC 03 3F 03
BF 01 0F 00
C2 00 1F 01
C5 HIT1 1F 1F 01
C8 0F 0F 00
CB 07 3F 03
CE 03 7F 07
D1 01 1F 01
D4 00 0F 00
D7 HIT2 1F 7F 07
DA 0F 3F 03
DD 07 1F 01
E0 03 0F 00
E3 01 7F 07
E6 00 3F 03
E9 HIT3 1F 3F 03
EC 0F 7F 07
EF 07 0F 00
F2 03 1F 01
F5 01 3F 03
F8 00 7F 07

```

STIX - Page 02

Temporary Storage (4)

00 TEMS 00 00 00 00

STA Sub Routine (21)

```

04 EXSTA ED
05 D3
06 STA FB C0 A.0(ROC) AF
09 E9
0A FB 08 AC
0D AGN 72 5F 2C 8C
11 32 04 EXSTA
13 8F FC 08 AF 30 00 AGN

```

DOR Sub Routine (54)

```

19 EXDOR D4
1A DOR FB 07 A9 8B FE
1F CKBT FE AB 33 33 DOSTK
23 BLANK FB 08 AC
26 CLR FB 00 5F 2C 8C
2B 32 43 SEVN
2D 8F FC 08 AF 30 26 CLR
33 DOSTK FB 08 AC
36 STK FB C0 5F 2C 8C
3B 32 43 SEVN
3D 8F FC 08 AF 30 36 STK
43 SEVN 29 09 32 19 EXDOR
47 0D FC 01 5D AF
4C 8B 30 1F CKBT

```

TAK Sub Routine (56)

```

4F EXTAK D6
50 TAK 1D 0D FA F0 5D
55 FB 30 32 78 IS3
59 0D FB 20 32 6D IS2
5E 0D FB 10 3A 04 INV
63 IS1 8B FF 01 3B 04 INV
68 8B FF AB 30 4F EXTAK
6D IS2 8B FF 03 3B 04 INV
72 8B FF F6 AB
76 30 4F EXTAK
78 IS3 8B FF 07 3B 04 INV
7D 8B FF F6 FC AB
82 30 4F EXTAK
84 INV 7B 30 4F EXTAK

```

HIT Sub Routine (62)

```

87 EXHIT ED
88 D7
89 HIT 9B FB 07 32 A4 EQ3
8E 9B FB 03 32 10 EQ2
93 9B FB 01 32 2C EQ1
98 EQ0 FB B3 A.0(HIT0)
9A 30 AC SEST
9C EQ1 FB C5 A.0(HIT1)
9E 30 AC SEST
A0 EQ2 FB D7 A.0(HIT2)
A2 30 AC SEST
A4 EQ3 FB E9 A.0(HIT3)
A6 SEST A9
A7 E9
A8 FB 06 AC
AB NTHT 8A F3 3A BC BNC
AP 60 9A F3 32 B9 HITT
B4 60 9A F3 3A B7 EXHIT
B9 HITT 7B 30 87 EXHIT
BC BNC 60 60 60
BF 1C 8C 3A AB NTHT
C3 30 87 EXHIT

```

PER Sub Routine (59)

```

C5 EXPER D3
C6 PER FB 20 A.0(TAK) AB
C9 7B
CA KYB 6C 64 2D
CD 3F CA KYB
CF 37 CF *
D1 2D FA 0F 5D
D5 7A
D6 FB 0A 32 E8 ROWA
DA F0 FB 0B 32 F0 ROWB
DF F0 FB 0C 32 F0 ROWC
E4 1D 7B 30 C5 EXPER
E8 ROWA 9A AB D8
EB 31 C5 EXPER
ED BA 30 C5 EXPER
F0 ROWB 8A AB D8
F3 31 C5 EXPER
F5 AA 30 C5 EXPER
FB ROWC 9B AB D8
FD 31 C5 EXPER

```

This is Stack and Temp storage area

TEMP is stack
Return to Main program
LINK points to status position
DAPT is stack
CTR = 8
Store character pointed to by
DAPT in status position of
Display area

Return to PTR sub routine
Set counters for 7 sticks
If bit is 1 make a stick
If bit is 0 blank a stick

If row is done return
If not done point to next stick

Return to PER sub routine
Mask for number to take
Take = 3
Take = 2
If take # 1 take is invalid
If row # 1 or more take is invalid
Take 1 and return
If row # 2 or more take is invalid
Take 2 and
Return
If row # 3 or more take is invalid
Take 3 and
Return
Set Q to signal invalid

TEMP is stack
Return to ELF sub routine
Row C = 3
Row C = 2
Row C = 1
Hit list is 0
Hit list is 1
Hit list is 2
Hit list is 3
Hit list to DAPT
DAPT is stack
CTR = 6
Row B does not compare
Row B and A compare - set hit
Row B and A do not compare
Set Q and return
Bump stack to next set
All done - hit not found

Return to Main program
Point SUBRT to TAK routine

Input Person's move
Temp = Row
Temp + 1 = Take
Take from row A
Take from row B
Take from row C
Row is invalid
Check row A for valid take
Take invalid - do not save
Take valid - save take
Check row B for valid take
Take invalid - do not save
Take valid - save take
Check row C for valid take
Take invalid - do not save
Take valid - save take

STIX - Page 03

Display Area (256)

```

00 DSP 00 00 00 00 00 00 00 00
03 00 00 00 00 00 00 00 3C
10 00 00 00 00 00 00 00 66
18 00 00 00 00 00 00 00 66
20 00 00 00 00 00 00 00 66
28 00 00 00 00 00 00 00 78
30 00 00 00 00 00 00 00 66
38 00 00 00 00 00 00 00 66
40 00 00 00 00 00 00 00 00
48 00 00 00 00 00 00 00 00
50 00 00 00 00 00 00 00 00
58 00 00 00 00 00 00 00 00
60 ROB 00 00 00 00 00 00 00 00
68 00 00 00 00 00 00 00 7C
70 00 00 00 00 00 00 00 66
78 00 00 00 00 00 00 00 66
80 00 00 00 00 00 00 00 7C
88 00 00 00 00 00 00 00 66
90 00 00 00 00 00 00 00 66
98 00 00 00 00 00 00 00 7C
A0 00 00 00 00 00 00 00 00
A8 00 00 00 00 00 00 00 00
B0 00 00 00 00 00 00 00 00
B8 00 00 00 00 00 00 00 00
C0 ROC 00 00 00 00 00 00 00 00
C8 00 00 00 00 00 00 00 3E
D0 00 00 00 00 00 00 00 00
D8 00 00 00 00 00 00 00 00
E0 00 00 00 00 00 00 00 00
E8 00 00 00 00 00 00 00 00
F0 00 00 00 00 00 00 00 00
F8 00 00 00 00 00 00 00 3E

```

STIX - Hex Listing

Page 00

```

00 FB 00 B1 B3 FB 00 A3 D3 FB 01 B4 B7 B9 AD F8 02
10 B2 B5 B6 B8 BD A2 F8 B3 A1 F8 71 A4 FB 06 A5 FB
20 C6 A6 F8 01 A7 F8 03 BF ED 61 F8 7F BA FB 1F AA
30 FB 07 B8 D4 F8 A3 A9 D5 6C 64 2D 3F 38 37 3D 32
40 90 F8 AB A9 D5 6E FA 03 5D FB 01 32 65 F0 FB 02
50 32 5E F0 FB 03 3A 45 9A 32 45 F6 BA 30 6A BA 32
60 45 F6 AA 30 6A 9B 32 45 F6 BB FB 00 AE 8N FF 44
70 3A 6D FB 35 AC 7B F8 55 FF 01 3A 78 F8 15 FF
80 01 3A 7F 2C 0C 3A 75-D4 F8 00 AE 8E FF 22 3A BB
90 FB 9B A9 D5 D6 31 C3 D4 9A 5D 8A F4 5D 9B F4 FF
A0 01 3A 7F 2C 0C 3A 75-D4 F8 00 AE 8E FF 22 3A BB
B0 01 3A 7F 2C 0C 3A 75-D4 F8 00 AE 8E FF 22 3A BB
C0 01 3A 7F 2C 0C 3A 75-D4 F8 00 AE 8E FF 22 3A BB
D0 01 3A 7F 2C 0C 3A 75-D4 F8 00 AE 8E FF 22 3A BB
E0 01 3A 7F 2C 0C 3A 75-D4 F8 00 AE 8E FF 22 3A BB
F0 01 3A 7F 2C 0C 3A 75-D4 F8 00 AE 8E FF 22 3A BB

```

Page 01

```

00 D3 F8 89 AB 7A 9A 32 28 F6 BA D8 31 00 9A 32 25
10 F6 BA D8 31 00 9A 32 22 F6 BA D8 31 00 9A 32 25
20 01 FE F9 01 FE F9 01 BA 8A 32 45 F6 AA D8 31 00
30 8A 32 45 F6 AA D8 31 00 8A 32 45 F6 AA D8 31 00
40 8A 32 45 F6 AA D8 31 00 8A 32 45 F6 AA D8 31 00
50 D8 31 00 9B 32 65 F6 BB D8 31 00 9B 32 65 F6 BB
60 D8 31 00 9B 32 65 F6 BB D8 31 00 9B 32 65 F6 BB
70 D5 F8 1A AB F8 00 AF 5D 9A AB D8 F8 00 AF 5D
80 AB D8 F8 C0 AF 5D 9B AB D8 30 70 C3 C3 C3 DE
90 FF E7 C3 60 60 60 60 60 60 60 60 60 60 60 60
A0 60 60 60 3C 66 06 0C 18 18 00 18 00 00 00 00
B0 00 00 00 1F 0F 00 0F 1F 01 07 7F 07 07 07 07
C0 0F 00 00 1F 01 1F 01 0F 0F 0F 07 3F 03 03 03
D0 07 01 1F 01 00 0F 00 1F 7F 07 0F 0F 0F 0F 0F
E0 05 0F 00 01 7F 07 0F 3F 03 1F 3F 03 03 03 03
F0 0F 00 03 1F 01 01 3F 03 00 7F 07 00 00 00 00

```

Page 02

```

00 00 00 00 00 ED D3 F8 C0 AF B9 F8 00 AC 72 5F 2C
10 8C 32 04 8F FC 08 AF 30 0D D4 F8 07 A9 8B FE FE
20 AB 33 33 FB 08 AC F8 00 5F 2C 8C 32 43 8F FC 08
30 AF 30 26 FB 08 AC F8 00 5F 2C 8C 32 43 8F FC 08
40 AF 30 36 29 89 32 19 0D FC 01 5D AF 8B 30 1F D6
50 1D 0D FA F0 5D FB 30 32 78 0D FB 20 32 6D 32 6D
60 10 3A 84 8B FF 01 3B 84 8B FF 01 3B 84 8B FF 01
70 3B 84 8B FF 01 3B 84 8B FF 01 3B 84 8B FF 01
80 F6 AB 30 4F 7B 30 4F ED D7 9B FB 07 32 A4 EQ3
90 03 32 A0 9B FB 01 32 9C FB 03 30 A6 FB 06 AC BA
A0 F8 D7 30 A6 F8 E9 A9 B9 F8 06 AC BA F3 3A BC BNC
B0 9A F3 32 B9 60 9A F3 3A B7 EXHIT 7B 30 87 EXHIT
C0 8C 3A AB 30 87 D3 F8 50 32 E8 ROWA 9A AB D8
D0 CF 2D FA 0F 5D 7A FB 0A 32 E8 ROWB 8A AB D8
E0 FB 0C 32 FB 1D 7B 30 C5 EXPER 31 C5 EXPER
F0 8A AB D8 31 C5 AA 30 C5 EXPER 31 C5 EXPER

```

TVT - 4K

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.
cursor to beginning of screen, - excluding cursor position.
deletes from cursor to end of screen, excluding cursor position.

control W

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 00	memory pointer
0208	6E 00	cursor flag
020A	00 60	erase display
020C	2D 08	display cursor (X,Y)
020E	12 10	goto 210
0210	6C 00	Wait for keyboard input VC contains ASCII character
0212	AD FC	I=ODFC
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 00	pointer=0D
0228	22 90	gosub 290(sub. to read character from screen memory).
022A	40 CD	
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	0A 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 00	check pointer VD if 00
0280	12 20	yes-goto 220(sub. to display all ASCII char- acters.)
0282	60 00	no-V0=00
0284	F0 55	V0=M1
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.(M1)

ADDR	CODE	COMMENT
0290	AA 00	I=0A00
0292	FD 1E	I=I+VD
0294	F0 65	V0=M1
0296	00 EE	return

Subroutine to write a character into
screen memory according to pointer VD.
char.(M1)=V0

ADDR	CODE	COMMENT
0298	AA 00	I=0A00
029A	FD 1E	I=I+VD
029C	F0 55	M1=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

PublisherQuest Electronics
EditorPaul Messinger
Proof ReadingJudy Pitkin
ProductionJohn Larimer

The contents of this publication are
copyright and shall not be reproduced without
permission of QUESTDATA. Permission is granted
to quote short sections of articles when used in
reviews of this publication. QUESTDATA welcomes
contributions from its readers. Manuscripts
will be returned only when accompanied by a self
addressed stamped envelope. Articles or
programs submitted will appear with the authors
name unless the contributor wishes otherwise.
Payment is at the rate of \$15 per published
page. QUESTDATA exists for the purpose of
exchanging information about the RCA 1802
microcomputer.

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	Pattern
	H L	
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 08 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

0A00-0AB3 screen memory

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

ADDR CODE	COMMENT
0AB4 4C 09	check if VC=09 TAB
0AB6 1A E4	yes-goto AE4
0AB8 4C 08	no-check if VC=08
	BACKSPACE
0AB8 1A F2	yes-goto AF2
0ABC 4C 60	no-check if VC=60 \
0ABE 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

0AAE 1B A0
0AB0 4C 17

0AB2 1B B0

CONTROL Q
yes-goto BA0
no-check if VC=17
CONTROL W
yes-goto BB0
no-continue on to AB4

Program block to display character. Uses sub-routine 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
	l=DF9(V9 in Chip-8)
0ABA AD F9	MLS to get ASCII character pattern
0ABC 0D 10	l=D30(temporary storage for pattern)
0ABE AD 30	erase old character
0AC0 DA B7	check cursor condition
0AC2 4E 00	0=turn on cursor;1=leave cursor off
0AC4 2D 08	l=DFC(VC in Chip-8)
0AC6 AD FC	MLS to get ASCII char. pattern
0AC8 0D 10	l=D30(temporary storage for pattern)
0ACA AD 30	display new character
0ACC DA B7	X+05
0ACE 7A 05	display cursor
0AD0 2D 08	return
0AD2 00 EE	

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.

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
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 04	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

0 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 FB	X-05
0B7E	AD F9	I=DF9
0B80	0D 10	MLS to get ASCII char.
		pattern

ADDR	CODE	COMMENT
0B82	49 0D	check if VD=0D
0B84	1B 8A	yes-goto B8A
0B86	AD 30	no-I=030
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 2BA
0BDE	22 90	no-gosub 290(sub. to
		read ASCII char. from
		screen memory)
		VD-0D
0BE0	7D F3	gosub 298(sub. to write
0BE2	22 98	ASCII char. into screen
		memory)
		goto BD8
0BE4	1B D8	

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=000--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 ASCII character pattern. Must be in page D regardless of the position of this subroutine.
0D17 AE	
0D18 0A	
0D19 FE	
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. branch to: 1D
0D2D 3A 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 M1. 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 Publication MPM-206. "Binary Arithmetic Subroutines for RCA Cosmac Microprocessors".
9. RCA Publication MPM-208. "Operator Manual for the RCA Cosmac Development System".
10. Strobe, 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 omission.

THE HAMURABI GAME

10 REM

20 REM

30 REM

40 REM

50 REM

60 REM

70 REM

80 REM

90 REM

100 REM

110 REM

120 REM

130 REM

140

DEFIN

Z

150

CLS

160

INPUT

"HAMURABI

GAME - Do

you need

instructions

(YES or

NO)"

170

IF

MID\$(1

5,1,1)

="N"

GOTO

340

180

IF

MID\$(1

5,1,1)

="n"

GOTO

340

190

PRINT

"HAMURABI

- You

are the

much

loved

king of

the

ancient"

200

PRINT

"kingdom

of

SUMERIA."

210

PRINT

"The

object

is to

keep your kingdom growing by raising grain."

220 PRINT "using the grain to feed your people and the strangers who"

230 PRINT "come to your city, keep enough grain for seed for next years"

240 PRINT "crop. In addition, you will buy land for planting crops in"

250 PRINT "times of plenty or sell your farm land in bad times to"

260 PRINT "get much needed grain."

270 PRINT TAB(10);"(The plague strikes occasionally)"

280 PRINT "You buy or sell land at its current value, each person"

290 PRINT "needs 20 bushels for food, each acre costs 1/2 bushel to plant."

300 PRINT "maximum of 10 acres, and each acre costs 1/2 bushel to plant."

310 PRINT "(If you want to quit, sell all of your land)"

320 PRINT : INPUT "Press RETURN to begin your reign."\$

330 A=100:B=5:C=0:D=2800:E=200:F=3:G=3000:H=1000:J=1:L=1

340 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 ";L;": acres of land."

HAMURABI GAME

(Adapted by Fred Hannan)

Hamurabi has its roots in several claimed creations but

I believe the original version for microcomputers was

written by CREATIVE COMPUTING.

This version was adapted from many different versions

published in several magazines and books.

The instructions included in the game explain the rules

and objectives.

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<=0 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=I: 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<0 GOTO 720

710 GOSUB 910: GOTO 690

720 D=D-I:C=C-(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<=0 GOTO 810

800 GOSUB 910: GOTO 770

810 IF I>10*A GOTO 900

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 THEN B=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

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

QUESTDATA

P.O. Box 4430

Santa Clara, CA 95054

A 12 issue subscription to QUESTDATA, the publication devoted entirely to the COSMAC 1802 is \$12.

(Add \$6.00 for airmail postage to all foreign countries except Canada and Mexico.)

Your comments are always welcome and appreciated. We want to be your 1802's best friend.

Payment.

☐ Check or Money Order Enclosed

Made payable to Quest Electronics

☐ Master Charge No. _____

☐ Bank Americard No. _____

☐ Visa Card No. _____

Expiration Date: _____

Signature _____

NAME _____

ADDRESS _____

CITY _____

STATE _____

ZIP _____

☐ Renewal ☐ New Subscription

ADDR CODE	LABEL	COMMENT	ADDR CODE	LABEL	COMMENT
001E F8 40 A0	BLANK		---- DMA scan 1		
---- DMA zero row 1			0020 E2 20 A0		
0021 25 85 32 27		Countdown blank scans by 2 and jump on zero to PIC to display graphics beginning at 0050	---- DMA scan 2		Scan each 8-byte line four times as in original Graphics Test Program
---- DMA zero row 2			0023 E2 20 A0		
0025 30 1E			---- DMA scan 3		
---- DMA zero row 2			0026 E2 20 A0		
0027 F8 50 A0	PIC		---- DMA scan 4		
002A E2	REFRESH		0029 FB F0		Makes D=0 if last line.
---- DMA Scan 1		Scan each 8-byte line four times as in original Graphics Test Program	002B 3A 1F		Get new display line if D not = 0
002B E2 20 A0			002D 80		Save in D address of 8 zero bytes that must immediately follow last line of graphic.
---- DMA Scan 2			---- DMA zero row		Repoint to zero row until EFI shows we're in last 4 scans.
002E E2 20 A0			002E 20 A0	BLANK	
---- DMA Scan 3			0030 3C 2E		
0031 E2 20 A0			---- DMA zero row		
---- DMA Scan 4			0032 20 A0 34 32	FOUR	Repoint to zero row until EFI shows last 4 scans done.
0034 80 3C 2A		Check for end of display window. Do-nothing MAIN loop	---- DMA zero row		
0037 30 0F			0036 30 0F 00		
0039 E2 69 30 3B	MAIN	3D to 3F is stack area	0039 E2 69 30 3B	MAIN	Do-nothing MAIN loop.
003D XX XX XX		Bytes 0040 - 00FF containing graphic same as original P.E. Spaceship Program	003D xx xx xx		3D to 3F is stack area.
0040 00 00 00 etc.	ZERO		0048 7B DE DB etc.	GRAPHIC	40 to 47 is unused Bytes 0048-00F7 contain graphic found as 0050-00FF in original.
0050 7B DE 0B etc.	GRAPHIC		00F8 00 00 00 etc.	ZERO	Zero row for blanking.
Bottom 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					
0018 F8 00 B0					
001B F8 4B A0 E2		Spaceship graphic moved to begin at 0048 instead of 0050			
001F 80	REFRESH				

COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB

22

QUESTDATA

P.O. Box 4430

Santa Clara, CA 95054

ADDRESS CORRECTION REQUESTED**BULK RATE**

U.S. Postage Paid

QUEST
ElectronicsPermit No. 549
Santa Clara, CA