OSI-tems

GREAT MOMENTS IN OSI HISTORY

(by M. Bassman)

When wandering through our beloved 8K BASIC-in-ROM I discovered some routines for those of you trying to write assembler or machine language programs.

The first is one that gets a character from the keyboard and deposits it in the accumulator. The address is FEED, HEX. It can be used in BASIC like this:

```
10 POKE 11,0:POKE 12,253
20 X=USR(X):CH$=CHR$(PEEK(531))
```

(Credit for the BASIC lines that handle these locations belongs to D. Schwartz)

The second is the CRT driver. It takes a character from the accumulator and puts it on the screen. It takes care of scrolling and CRLF after 24 characters (or whatever $15_{(10)}$ contains). It is useless in BASIC but in assembler it is a great help. The address is $BF2D_{(16)}$.

The following programs will take a character from the keyboard and print it on the screen (WOW!).

BASIC

```
10 POKE 11,0:POKE 12,253
20 X=USR(X):C$=CHR$(PEEK(531))
30 PRINT C$;:GOTO 20
```

ASSEMBLER

```
FIRST STX #00
      JSR $FEED
      JSR $B2FD
      JMP FIRST
```

MACHINE LANGUAGE

```
3ØØ A2
3Ø1 Ø6
3Ø2 2Ø
3Ø3 ED
3Ø4 FE
3Ø5 2Ø
3Ø6 2D
3Ø7 BF
3Ø8 4C
3Ø9 ØØ
31Ø Ø3
```

After entering, begin with .0300G

Various Other Stuff

Location 11 & 12 (Decimal) is where the USR statement goes to, stored with lo byte, hi byte format.

Location 15 is LINE LENGTH COUNTER. Try POKE 15,23 or POKE 15,22 if you lose characters on the right side of your screen. POKE 15,0 doubles spaces text, and POKE 15,1 produces strange effects. Return to normal with POKE 15,72.

Location 518 is BAUD RATE CONTROL, with POKE 518,255 being slowest and POKE 518,0 fastest.

LOCATION 57100 Decimal is an easier way to use the keyboard scan. This scans for control keys. To find out which values they produce, try:

```
10 PRINT PEEK(57100):GOTO 10
```

while pressing down control keys.

A ROM BASIC routine is located at ABF5 Hex. It will look at the next position in your current BASIC program (like after your USR(X)), checks for a '(' then a value then a ')'. If not all of these it produces a syntax error.

For example, program (A) will produce a syntax error but program (B) will not.

(A)
```
10 POKE 11,245:POKE 12,171
20 X=USR(Y)
30 END
```

(B)
```
10 POKE 11,245:POKE 12,171
20 X=USR(Y)(1234)
30 END
```

If this has a practical value apart from the challenge of discovery it has yet to be known. If you do find a use, tell us about it.

AC$\emptyset$C Hex is the syntax error printer. Thus,

```
10 POKE 11,12:POKE 12,172
20 X=USR(X)
```

will produce a syntax error even though there is nothing wrong with the program.

Now here's the clincher. Ever use <BREAK> to get out of a program and then find you can't warm start? (Losing a valuable program, of course). Well, there is a way to (in most cases) temporarily get a program back long enough to put it on cassette.

If you find you can't warm start, do this:

1. Go into the monitor and copy on a piece of paper the contents of 7B,7C Hex and 301,302 Hex.

2. Cold Start, but enter your memory size rather than hitting return. (4046) for a 4K machine and 8192 for an 8K machine). If you hit return here, your program is zapped forever.

3. Go right back to the monitor. Replace 7B,7C and 301,302 with the values you had copied down before.

4. Warm Start, do a "LIST", then immediately go to the monitor and replace 7B,7C Hex with the contents of AA,AB Hex, respectively. Warm Start.

5. Save your program on cassettte, then you are safe to do as you please.

PROGRAM OF THE MONTH--SKEET SHOOT

Variable List

Z = Number of Aliens (Difficulty Factor).
BL = Bullets left
LG = Location of Gun
LB = Location of Bullet
D = Direction of Alien
S = Score

10 - 95	=	Initialize game, input difficulty factor.
100 - 130	=	Move left
140 - 155	=	Move right
160 - 200	=	Fire routine. Keeps bullet moving until it hits something.
210 - 240	=	Explosion.
250 - 270	=	If object is a wall, replace blown-away section.
280	=	POKE score.
300 - 330	=	POKE number of bullets remaining.
500 - 650	=	Determine a random direction for alien. Check to see if it is already occupied. POKE alien out of old location into new one.
1000-1140	=	Win/Lose message. Input to try again.

SKEETSHOOT is an example of a game containing all the standard functions incorporated into a real-time video game. A player moves his gun left or right at the bottom of the screen, firing up at aliens dodging his bullets. A player enters the number of alien targets he may fire at. Therefor, the lower the number of targets, the harder the game. Included is a section by section description of the major parts of the program.

```
OK
LIST

 10 FORK=53248TO54272:POKEK,32:NEXT
 11 PRINT"      SKEET!!!":PRINT"BY MIKE BASSMAN":FORK=1TO10:PRINT:NEXT
 12 INPUT"DIFFICULTY";Z
 13 FORK=1TO30:PRINT:NEXT
 15 IFZ<1ORZ>5THEN10
 20 FORK=53412TO53435:POKEK,187:POKEK+23*32,187:NEXT
 30 FORK=53412TO54148STEP32:POKEK,187:POKEK+23,187:NEXT
 40 DIM LT(Z)
 50 FORK=1TOZ:LT(K)=INT(RND(1)*20+53605)
 60 NEXTK
 70 POKE54117,32
 75 BL=Z*4
 80 LG=54095:POKELG,29
 90 POKE530,1
 95 P=57088:POKEP,127
 100 IFPEEK(P)<>127THEN140
 110 IFPEEK(LG-1)<>32THEN500
 115 POKELG,32
 120 LG=LG-1:POKELG,29
 130 GOTO500
 140 IFPEEK(P)<>191THEN160
 150 IFPEEK(LG+1)<>32THEN500
 155 POKELG,32:LG=LG+2:GOTO120
 160 IFPEEK(P)<>223THEN500
 165 IFBL=0THEN1100
 167 BL=BL-1:GOSUB300
 170 LB=LG
 175 LB=LB-32
 180 IFPEEK(LB)<>32THEN210
 190 POKELB,149:POKELB+32,32:POKELG,29
 200 GOTO175
 210 FC=PEEK(LB)
 215 POKELB+32,32
 220 FORK=1TO10
 230 POKELB,232:FORG=1TO10:NEXTG:POKELB,233:FORG=1TO10:NEXTG
 240 NEXTK:POKELB,32
 250 IFFC=187THENPOKELB,187
 260 IFFC=4THENS=S+1
 270 IFS=Z*2THEN1000
 280 POKE53382,ASC(MID$(STR$(S),2,1))
 290 GOTO500
 300 SL=53399
 310 FORK=1TOLEN(STR$(BL)):POKESL,ASC(MID$(STR$(BL),K,1))
 320 SL=SL+1:NEXTK
 330 POKESL,32:RETURN
 500 FORK=1TOZ
 510 D=INT(RND(1)*9+1)
 520 ONDGOTO530,540,550,560,570,580,590,600,610
 530 D=1:GOTO620
 540 D=-1:GOTO620
 550 D=33:GOTO620
 560 D=-33:GOTO620
 570 D=32:GOTO620
 580 D=-32:GOTO620
 590 D=31:GOTO620
 600 D=-31:GOTO620
 610 D=0:POKELT(K),32
 620 IFPEEK(LT(K)+D)<>32THEN510
 625 POKELT(K),32
```

```
 70 POKE54111,32
 75 BL=Z*4
 80 LG=54095:POKELG,29
 90 POKE530,1
 95 P=57088:POKEP,127
 100 IFPEEK(P)<>127THEN140
 110 IFPEEK(LG-1)<>32THEN500
 115 POKELG,32
 120 LG=LG-1:POKELG,29
 130 GOTO500
 140 IFPEEK(P)<>191THEN160
 150 IFPEEK(LG+1)<>32THEN500
 155 POKELG,32:LG=LG+2:GOTO120
 160 IFPEEK(P)<>223THEN500
 165 IFBL=0THEN1100
 167 BL=BL-1:GOSUB300
 170 LB=LG
 175 LB=LB-32
 180 IFPEEK(LB)<>32THEN210
 190 POKELB,149:POKELB+32,32:POKELG,29
 200 GOTO175
 210 FC=PEEK(LB)
 215 POKELB+32,32
 220 FORK=1TO10
 230 POKELB,232:FORG=1TO10:NEXTG:POKELB,233:FORG=1TO10:NEXTG
 240 NEXTK:POKELB,32
 250 IFFC=187THENPOKELB,187
 260 IFFC=4THENS=S+1
 270 IFS=Z*2THEN1000
 280 POKE53382,ASC(MID$(STR$(S),2,1))
 290 GOTO500
 300 SL=53399
 310 FORK=1TOLEN(STR$(BL)):POKESL,ASC(MID$(STR$(BL),K,1))
 320 SL=SL+1:NEXTK
 330 POKESL,32:RETURN
 500 FORK=1TOZ
 510 D=INT(RND(1)*9+1)
 520 ONDGOTO530,540,550,560,570,580,590,600,610
 530 D=1:GOTO620
 540 D=-1:GOTO620
 550 D=33:GOTO620
 560 D=-33:GOTO620
 570 D=32:GOTO620
 580 D=-32:GOTO620
 590 D=31:GOTO620
 600 D=-31:GOTO620
 610 D=0:POKELT(K),32
 620 IFPEEK(LT(K)+D)<>32THEN510
 625 POKELT(K),32
 630 LT(K)=LT(K)+D
 640 POKELT(K),4
 650 NEXTK
 660 GOTO100
 1000 FORK=53248TO54272:POKEK,32:NEXT
 1010 PRINT"YOU WIN!!!":GOTO1110
 1100 FORK=53248TO54271:POKEK,32:NEXT:PRINT"YOU LOSE !!"
 1110 FORK=1TO10:PRINT:NEXT
 1120 INPUT"TRY AGAIN";Y$
 1130 IFLEFT$(Y$,1)="Y"THENRUN
 1140 POKE530,0
OK
```