MUSIC

****

MUSIC: Adam's Sound Generator can be activated by a little program in SmartBasic using POKES and CALLS. Here are some definitions you need:

SOUND: Sound on the ADAM is generated by a SOUND Chip that is located in ADAM'S I/O space. The Sound Chip is a Texas instruments SN76489AN. It has three music channels and a noise channel. In this Memo, we will only deal with the Music Channels.

CALl: Is a Command or Statement in SmartBasic which specifies the address in Memory that contains the start address of a Z-80 Assembly Language program (that we will type into memory later on this page.)

POKE: Is a command in BASIC which stores a number at a specified location in RAM. The expression is POKE location, number.

FREQUENCY: Is the Pitch of the note. The pitch is determined by how fast the speaker is vibrated. If it vibrates 440 times per second, it will produce a pitch equivalent to midddle A.

DURATION: Means how long each notes is played.

Here are the steps to follow:

1. Set up a Workspace to put the Machine Language program.
   (Reserve a space in memory the BASIC will not use.)

```
 5 LOMEM: 29000
```

2. Write a program to POKE a 6 byte Z-80 program into memory. The Program loads microprocessor register A with the byte we will later POKE into location 28006, outputs it to port 255, and returns to BASIC.

```
10000 REM LAUNCH PAD FOR SOUND
10010 DATA 58,102,109,211,255,201
10020 startsound = 28000
10030 For address = startsound TO startsound + 5
10040     READ byte
10050     POKE address, byte
10060     NEXT address
10070     RETURN
```

3. Get the Pitch and Duration. Pitch can be any number between 65 (low note) and 3856. Duration can be 1 (brief) and 64 (long).

```
1000 REM Setup
     1010 INPUT "Pitch:"; pitch
     1020 INPUT "Duration:"; dur
     1030 GOSUB 2000
     1040 GOTO 1000
```

4. Set up some variables:

```
10 GOSUB 10000 : REM Make the Launchpad
20 toot = 144
30 offtoot = 159
40 bytespot = 28006
50 mhertz = 3597000
60 tempo = 20
```

Toot and offtoot turn the note on and off.
Bytespot is the memory location that each variable is POKED into.
Mhertz is the speed of ADAM's clock which will be divided by the Pitch to get the number of times the speaker has to be vibrated.
Tempo is multiplied by duration to give the length of each note.

5. Tell the Sound Chip to Play the Note.

```
2000 REM Play the Note
2010 pitch = mhertz/(32*pitch)
2020 twopitch = pitch/16
2030 onepitch = 128 + (pitch - twopitch*16)
2040 POKE bytespot, onepitch : CALL startsound
2050 POKE bytespot, twopitch : CALL startsound
2060 POKE bytespot, toot : CALL startsound
2070 FOR delay = 1 TO dur * tempo : NEXT delay
2080 POKE bytespot, offtoot : CALL startsound
2090 RETURN
```

It is necessary to break the pitch into two parts to send to the sound chip. The Sound Chip Voice Registers each contain a 10 place counter (register) which must be filled with the correct number to generate all of the characteristics of the sound.
Adam's Z-80 chip only sends messages in 8 bit sizes, so the pitch is broken into 2 parts and sent as two messages.

Now all you need are the notes. Here are a few:

LO C	130.81	C	261.62	HI C	523.24
LO C#	138.59	C#	277.18	HI C#	554.36
LO D	146.83	D	293.66	HI D	587.32
LO D#	155.57	D#	311.13	HI D#	622.26
LO E	164.82	E	329.63	HI E	659.26
LO F	174.62	F	349.23	HI F	698.46
LO F#	185	F#	370	HI F#	740
LO G	196	G	392	HI G	784
LO G#	207.65	G#	415.3	HI G#	830.6
LO A	220	A	440	HI A	880
LO A#	233.08	A#	466.16	HI A#	932.32
LO B	246.94	B	493.88	HI B	987.76

Your program will now run, allowing you to program one note at a time, using Voice 1. If you would like to program chords, see the next page

6. To play a Chord (make 3 voices active) change or add these program lines:

```
1000 FOR voice = 0 TO 64 STEP 32
1005 t = t + 1 : PRINT "Tone:"; t
1015 GOSUB 2000 : NEXT voice

1030 GOSUB 2070

1035 FOR offtoot = 159 TO 223 STEP 32 :
GOSUB 2080 : NEXT offtoot

2030 onepitch = 128 + voice +
(pitch-twopitch*16)
2060 POKE bytespot, toot + voice :
CALL startsound
2065 RETURN

2075 RETURN
```

Each voice is activated by manipulating two bits of the Vocie Register. We do it by adding 32 to the value of the onepitch. Offtoot must also address each voice. This is done in Line 1040.

7. If you really wish to get fancy and add different loudnesses to each note (volume), lets make a volume control that goes from 1 (soft) to 15 (loud.)

```
1012 INPUT "Volume :"; loud : loud = 16 - loud

2060 POKE bytespot, toot + loud + voice : CAll startsound
```

This Technical Bulletin is only meant to get you started. For a more complete explanation of Music on the ADAM, consult one of the many references available. One such reference is:

Adam's Companion Benson and Rochester, (Avon, New York, 1983.)

RANDOMIZE NUMBERS

RANDOMIZE NUMBERS: SmartBasic's RND function returns a random real number less than 1 and greater than or equal to zero. The sequence of random numbers is the same each time Smart Basic is booted, and each time you RUN and program. This arrangement is consistent with ANSI standards, but it differs from AppleSoft tm Basic which produces a new random number each time the program runs.

Here is a little program that will randomize the numbers each time you start the program. The randomization is based on the variable length of time it takes a person to press a number on the front Paddle's keypad that is requested in the first line. ADAM strobes the keypad to see if any number has been pressed. If it was not pressed, ADAM decrements counter rx. Since it is very unlikely that people will press the keypad's number keys in the same length of time, a random seed (negative number) is produced.

Here is a program that can be tucked into the first 3 lines of your program:

```
1 PRINT "Press # on the front
keypad to continue"
2 rx= rx - 1 : IF NOT PDL(11) THEN
2
3 junk = RND(rx)
```

If you want to use this in a little test program that shows a bunch of random numbers between 1 and 6, type in lines 1 through 3, then add:

```
10 FOR Y = 1 TO 5
20     FOR X = 1 TO 10
30         PRINT INT(RND(1)*6+1);"  ";
40 NEXT : PRINT : NEXT
```

You may wish to have a truly randomized number to start your program the first time, but start from the same number each time thereafter. If so, have your program loop back to line 3.

Your Text Window

****

SETTING THE SIZE OF THE TEXT WINDOW: If your television screen seems too small for SmartBasic (because characters disappear off the left or right edge), or if you want to have your text appear in a "Window" on one section of the screen, here is a program that will help.

The program peeks into Basic and finds four important memory addresses. It is written to list these locations on your printer:

```
 10 PRINT "Put paper in the printer." : PRINT
 20 PRINT "Press any key when ready." : GET qS
 30 PR#1
 40 seek = 16000
 50 IF PEEK(seek) = 1 AND PEEK(seek + 3) = 17 AND
    PEEK(seek + 6) = 33 AND PEEK(seek + 9) =-62 AND
    PEEK(seek + 11) = 8 THEN 70
 60 seek = seek + 1 : GOTO 50
 70 PRINT "Lines Address is "; seek + 1
 80 PRINT TAB(5) "Present Value is "; PEEK(seek + 1)
 90 PRINT "Columns Address is "; seek + 2
100 PRINT TAB(5) "Present Value is "; PEEK(seek + 2)
110 PRINT "Top Line Address is "; seek + 4
120 PRINT TAB(5) "Present Value is "; PEEK(seek + 4)
130 PRINT "Left Margin Address is "; seek + 5
140 PRINT TAB(5) "Present Value is "; PEEK(seek + 5)
150 PR#0
```

Explanation: Text displayed on ADAM'S screen is actually a map of a portion of ADAM'S memory. What you see on the screen is simply copied from that section of memory. The boundaries of the memory section are set by ADAM to 31 columns by 24 rows. You change the size of this "window" by changing numbers in the 4 memory locations.

Caution: While it is possible to put any number (0 to 255) in these locations, you can create a situation in which ADAM is not putting characters on the screen, or one in which ADAM has no place in memory to place screen information. If you do this, you will get unsatisfactory results, and likely wind up re-booting BASIC.

When you RUN the program, it will display the current value stored in each location. This value is called the DEFAULT value (the number stored in that location when BASIC is booted.)

See the second page for a description of the four locations.

Your Text Window

Number of lines on your screen. ADAM'S Address: _____
(Apple's PEEK 35) The default value is 23.
To change the number of lines, POKE a number (between 3 and 23) into the "Lines Address" that you obtained when you ran the program.

POKE (address), number : TEXT

After you POKE this location, it is necessary to enter the TEXT command. ADAM only looks at the margin addresses when BASIC is first booted up, and after the TEXT command, so you must enter TEXT after changing the dimensions of your window.

Number of columns. ADAM'S Address: _____
(Apple's PEEK 33) Default is 30. To make the screen one column narrower, change this to a value of 29.
The number of columns is added to the left margin. If your left margin is set at 2, a setting of 28 columns will take the cursor over to the 30th column on the screen.
CAUTION: If you use the CONTROL-N function to insert characters on a line, and the end of the line spills past the right margin of the screen, the text that was pushed off of the screen will be ignored by ADAM.

Top Margin: ADAM'S Address: _____
(Apple's PEEK 34) Default = 0.

Left Margin: ADAM'S Address: _____
(Apple's PEEK 32) Default = 1.

Closing Data Files

****

CLOSING DATA FILES: If you have worked with Sequential or Random Access files, you may have noticed an intermittent problem when ADAM was attempting to read data and close the file. The result was either an END OF DATA ERROR, or garbage appearing in your files.

The problem is caused by a safety feature built into SmartBasic. As with AppleSoft, SmartBasic uses PRINT CHR$ (4) as a signal to ADAM to change the I/O message to the Data Drives. But CHR$ (4) also can be used to print a heart on the screen. To help ADAM distinguish which signal is being used, PRINT CHR$(4) is only interpreted as an I/O signal when the instruction begins in the left margin of the screen, directly after the line numbers, or after a carriage return.

Your program is often written so that the statement immediately preceding the file closing places the cursor somewhere on the screen other than the left margin. (A NEXT, or a MON,C,O,L will do this.) As a result, a heart followed by a "close...." statement appears on the screen, followed by the END OF DATA error message.
This problem can be corrected by adding an HTAB 1 to the beginning of the Close File line.

```
160 HTAB 1 : PRINT D$; "close myfile"
```

The Question Marks: When ADAM reads a data file, it prints the question mark each time it looks for an Input, resulting in as many question marks as you have data items. If you find this aggravating, you can erase the question mark by adding to the INPUT line:

```
140 INPUT add$(x) : PRINT CHR$(8);
```

This causes the cursor to backspace over the question mark and rub it out each time it is printed.

These modifications are written into a sequential file program on Page 2.

An example of a Sequential File with 10 items.

Writing the File out to the Disk. The fix is on line 150.

```
    5 d$  = CHR$(4)
   10 REM **** Read the DATA for filing.
   20 FOR x = 1 TO 10
   30 READ stuff$(x)
   40 NEXT x
  100 REM **** Write a File called Games
  110 PRINT d$; "Open Games"
  120 PRINT d$; "Write Games"
  130 For x = 1 TO 10
  140 PRINT stuff$(x)
  150 NEXT x
  160 HTAB 1 : PRINT d$; "Close Games"
 1000 DATA Buck Rogers, Zaxxon, Donkey Kong, Tarzan,
      Choplifter, Star Trek, AE, Congo Bongo, Telly Turtle,
      Dragon's Lair
```

A Program to read the file. The fixes are on lines 140 and 160.

```
    5 d$ = CHR$(13) + CHR$(4)
  100 REM **** Read the File called Games.
  110 PRINT d$; "Open Games"
  120 PRINT d$; "Read Games"
  130 For x = 1 TO 10
  140 INPUT stuff$(x) : PRINT CHR$(8);
  150 NEXT x
  160 HTAB 1 : PRINT d$; "Close Games"
  170 REM **** Print out your file.
  180 For x = 1 to 10
  190 PRINT stuff$(x)
  200 NEXT x
```

If you wish to print the contents of the file on the printer, turn the printer on after line 170. Otherwise, the printer will print the messages that go out to the data drive, and all of the question marks.

The HTAB fix was contributed by ADAM owner and writer Gary Cornell, Storrs, CT.

ALIGNMENT of COLUMNS (ZONE)

****

Zones: If you wish to have variables printed in columns on the screen or printer, the convention used in BASIC is a comma, printed just after the variable.

Try this program:

```
10 FOR X = 1 to 20
20 PRINT X,
30 NEXT X
```

On the ADAM, this prints on the screen in two zones that are 15 columns apart, with the first zone offset to the second column.
It looks like this:

```
 1              2
 3              4
 5              6
 7              8
 9              10
 11             12
   etc.....
```

But why is that 1 not in the column? Unfortunately, the comma that tells ADAM that the numbers are to be arranged in Zones is not read until after ADAM has received the command to print 1, so it is printed out of place.
To fix the columns so that they are in line, insert this line:

```
5 PRINT TAB (2)
```

This causes ADAM to position the cursor in the second column, and when the order is given to print the loop of numbers, the cursor is in the right place to begin.

ZONES ON THE PRINTER: The printer is not programmed to provide zones. It will dutifully print each number 15 or 16 spaces apart, but this produces an unpleasant arrangement on the page (which is 80 columns instead of the 31 columns provided on the SmartBasic screen.)
Here is one way to get the printer to print your data in five even zones:

```
5 PRINT : PRINT TAB (2)
10 FOR X = 1 TO 20
20 PRINT X,
30 IF INT(X/5) = X/5 THEN  PRINT : PRINT
   TAB(2)
40 NEXT X
```

It is necessary to issue two print statements in line 5. The first statement takes the printer off of the line on which it placed the Basic Bracket]. That bracket throws the printer off by one space.
Line 20 throws in a Carriage Return after the fifth zone.