Musical Notes for the Apple

With a 200 note table and a five octive range covering the entire treble, bass and alto clefs, you can do more than just whistle 'Dixie'.

The greatest limitation I found with my APPLE II+(APPLESOFT) is its inability to produce a variety of sounds. To overcome this I wrote a small assembler program (28 bytes long) that is CALLed by a Basic program using at least one POKE command.

Once loaded, the assembler program is capable of producing tones from 1.54hz to well over 15,000hz (cycles per second). Any tone can be held, from several seconds for very high tones, to minutes for lower tones. Four bytes are used in page zero (from $FC to $FF: 252 to 255) to control both the frequency of the tone and its length. The first two bytes set the frequency, and the last two limit the lingth (time) of the tone. Because two bytes are used for each, their combined values range from 1 to 65024 ($01 to $FE00). The use of the values 0($00) and 255 ($FF) are restricted by the relationship between the two programs.

By using page zero, it allows the assembler program to be located anywhere in RAM where it will be safe. For our purposes here the assembler program will be at address $6000, and the variable ASSEM, in the Basic program, will equal 24576. Once the values are set in page zero, the Basic program uses the command

CALL ASSEM

to produce the desired tone. Moving the assembler program only requires that the variable ASSEM be set to the decimal equivalent of the assembler's starting address.

The assembler program can reproduce any musical note, including sharps/flats, up to G# below A of 880hz. Above 880hz the rounding errors for many notes are too great to be of much use. Up to 880hz the largest error is 2.74 cycles, at G and G# just below 880hz. As the frequency decreases, so does the error in cycles.

The useable musical notes are from G# below 880hz, to A at 27.5hz. This gives a 5 octive range covering the entire treble, bass, and alto clefs. The Basic program will allow a 10 octive range of inputs, from 0 to 9, but octives 1 to 5 cover the above clefs, with 1 being the lower octive.

To setup the assembler program enter Monitor

CALL - 151

then the RETURN key. Once you have the asterisk prompt, enter the following after it:

6000:A5 FE 38 EA A6 FC AE 30
CO A6 FC A4 FD CA DO FD
88 DO FA E9 01 DO EB C6
FF DO EB 60 00

then RETURN.

The above should be entered as one continuous string, with each byte seperated by a space. It is shown as it would basically appear in a memory dump. A memory dump is done by entering

6000.601C

By entering

6000L

you will get an assembler listing. The listing is reproduced as a debugging aid.

```
6000- A5 FE     LDA $FE
6002- 38        SEC
6003- EA        NOP
6004- A6 FC     LDX $FC
6006- AE 30 CO  LDX $CO30
6009- A6 FC     LDX $FC
600B- A4 FD     LDY $FD
600D- CA        DEX
600E- DO FD     BNE $600D
6010- 88        DEY
6011- DO FA     BNE $600D
6013- E9 01     SBC #
6015- DO EB     BNE $6002
6017- C6 FF     DEC $FF
6019- DO E8     BNE $6006
601B- 60        RTS
601C- 00        BRK
```

To save this to disk, use the following:

BSAVE ASSEM SOUND, A$6000, L$1C

The length shown ($1C) will not save the last byte at $601C; it is included to show the ending address only.

Still in Monitor, enter

FC:F9 02 07 18

then

6000G

when the RETURN key is pressed it will execute the assembler program using the values at $FC to $FF. The note should be G below middle C, and last 15 seconds. After the assembler program has executed, the value at address $FF should be equal to 0 ($00), the other values should be untouched. If your values differ, check for an error.

For A below middle C:

FC:C2 02 D9 1A

for D above middle C:

FC:50 02 81 23

these values should all give 15 second notes when the assembler program is executed.

After the "Musical Notes" program is entered the keyboard keys "QWERTYU"should be marked to read "ABCDEFG", respectively. The sharp/flat of a note is obtained by pressing the "CNTL" key and the note key together. A rest note is given by the space bar.

As each note is entered, it will be stored in an internal table for replaying, and displayed to the upper 20 text lines. The note table (NT) holds the values for the note frequency and its length in a low order, high order format that are POKEed directly into addresses $FC to $FF. Each note displayed to a text line will have the format "ABCb", where:

A = the octive the note is in (0 to 9).
B = the note value, A through G.
C = the note time;
W : whole
H : half
4 : fourth
8 : eighth
1 : sixteenth
3 : thirty-second
6 : sixty-fourth
b = space, note separator.

A sharp/flat of the note will have the same format, except that it will be in the INVERSE mode. A rest note will be displayed as "bRCb". The "R" meaning a rest note, and the "C" the rest time.

This format allows 10 notes per text line, and by using the first 20 text lines, it permits 200 notes to be displayed. The last 4 text lines are for program information.

When the program is executed, the first input line will be:

ENTER BEATS PER MINUTES (4TH NOTE)

If no value is entered, the beats per minute will be set to 120. Some sheet music will have a note symbol (like a quarter note), followed by an equal sign, then a number in parenthesis near the upper left corner. This number is the beats per minutes. Whatever value is entered at this point will be the base for timing all other notes.

Once the beat is entered, you are ready to start entering notes. To change the octive, press any of the numeric keys (0 to 9). Line 22 (VTAB 22) will show the current octive.

The "BEAT =" shows the base beat of 120 at a quarter note value. To change the beat for any note, or group of notes, the keys "ASDFGHJ" are used.

A = whole note.
S = half note.
D = fourth note.
F = eighth note.
G = sixteenth note.
H = thirty-second note.
J = sixty-fourth note.

By pressing the "A" key, the beat will change from:

BEAT = 120 AT 4TH = 120

to:

BEAT = 120 AT WHOLE = 30

the last number is the beats per minute for a whole note.

Once a note is entered it cannot be deleted, but it can be changed to any value. The "," and "." keys are used to move the cursor over any note you want to change. The "," will move the cursor to the left, lower in the note table, while the "." will move it to the right, but will not allow movement beyond the next enterable note position.

While the left and right arrow keys would have been better, they were not used because the right key has the same value as the "CNTL" and "U" keys, which would give the note G#.

The option "Z : RUN NOTES (0)" shows how many notes are currently being stored, and will run all notes regardless of the cursor position.

In the event either the "X" or "C" is pressed, you will be asked if you really want it before they do their thing. Option "X" will CLEAR everything, and restart the program to

Listing 1

```
5   GOSUB 255: HIMEM: ASSEM
10  VTAB 21: PRINT "ENTER NOTE OR OPTION :                    ":
    PRINT "OCTIVE="OL" BEAT="BM" AT "BV$" = "BM / BV"   "
15  PRINT "Z:RUN NOTES("NO")" TAB( 20)"X:NEW NOTES": PRINT "C:
    END PROGRAM"
20  VTAB VT: HTAB HT: GET IN$:ER = 0:KI =  ASC (IN$): GOSUB 100:
    IF ER = 0 THEN VTAB 23: HTAB 1: GOTO 15
25  IF KI = 65 THEN BV$ = "WHOLE":BV = 4: GOTO 70
26  IF KI = 83 THEN BV$ = "HALF":BV = 2: GOTO 70
27  IF KI = 68 THEN BV$ = "4TH":BV = 1: GOTO 70
28  IF KI = 70 THEN BV$ = "8TH":BV = .5: GOTO 70
29  IF KI = 71 THEN BV$ = "16TH":BV = .25: GOTO 70
30  IF KI = 72 THEN BV$ = "32ND":BV = .125: GOTO 70
31  IF KI = 74 THEN BV$ = "64TH":BV = .0625: GOTO 70
35  IF KI >   = 48 AND KI <   = 57 THEN OL = KI - 48:
    HTAB 1: GOTO 10
40  IF KI = 44 AND NP - 1 >   - 1 THEN NP = NP - 1:
    GOSUB 225: GOTO 20
```

Listing 1

```
 45  IF KI = 46 AND (NP < NO) THEN NP = NP + 1: GOSUB 215: GOTO 20
 50  IF KI = 90 THEN  GOSUB 200
 55  IF KI = 88 OR KI = 67 THEN 180
 60  GOTO 20
 70  SB = (60 / BM) * BV:TM =  INT (((SB * (1E + 6)) / 36372) + .5):
     HTAB 1: GOTO 10
100  SV = OL: IF KI = 32 THEN OL = 0:XF = 0: GOSUB 145:
     IN$ = " R" +  LEFT$ (BV$,1) + " ": PRINT IN$:
     NT(NP,0) = FL * ( - 1):NT(NP,1) = FH:NT(NP,2) = LL:
     NT(NP,3) = LH:OL = SV: GOSUB 215: GOSUB 170: RETURN
105  FOR X = 0 TO 11: IF KI = KV(X,0) THEN NV = KV(X,1):
     XF = X:X = 98
110  NEXT : IF X = 12 THEN ER = 1: RETURN
115  GOSUB 145:NT(NP,0) = FL:NT(NP,1) = FH:NT(NP,2) = LL:
     NT(NP,3) = LH
120  IN$ = "":IN$ =  STR$ (OL) +  MID$ (KL$,NV,1) +
     LEFT$ (BV$,1) + " ": IF KI < 32 THEN  INVERSE
125  PRINT IN$: NORMAL : GOSUB 215
130  POKE 252,FL: POKE 253,FH: POKE 254,LL: POKE 255,LH:
     CALL ASSEM
135  IF (NP + 1 > NO) THEN NO = NO + 1:NP = NP + 1: RETURN
140  NP = NP + 1: RETURN
145  OC = (2 ↑ OL) * (2 ↑ (XF / 12)):CY = SC * OC:TC = TM * OC:
     PS = 1E + 6 / (2 * CY)
150  FH =  INT ((PS + 1254) / 1279):FT = 21 + (5 * FH - 1) +
     (1274 * (FH - 1)):FL = INT (((PS - FT) / 5) + .5)
155  TI = TC:LH =  INT (TI / 255):LL = (((TI / 255) - LH) * 255):
     IF LL = 0 THEN LL = 1
160  LH = LH + 1
165  RETURN
170  IF (NP + 1 > NO) THEN NO = NO + 1:NP = NP + 1: RETURN
175  NP = NP + 1: RETURN
180  IN$ = "END": IF KI = 88 THEN IN$ = "NEW"
185  VTAB 21: HTAB 1: INVERSE : PRINT "ENTER 'Y' FOR "IN$",
     ANY KEY TO IGNORE. ";: GET IN$: NORMAL :
     IF IN$ <  > "Y" THEN  HTAB 1: GOTO 10
190  IF KI = 67 THEN 300
195  CLEAR : HOME : GOSUB 260: GOTO 10
200  IF NO = 0 THEN  RETURN
205  FOR X = 0 TO NO - 1: POKE 252, ABS (NT(X,0)):
     POKE 253,NT(X,1): POKE 254,NT (X,2): POKE 255,NT(X,3):
     IF NT(X,0) < 0 THEN  POKE ASSEM + 8,0
210  CALL ASSEM: POKE ASSEM + 8,192: NEXT : RETURN
215  HT = HT + XI: IF HT = 41 THEN HT = 1:VT = VT + 1:
     IF VT = 21 THEN  GOSUB 235
220  RETURN
225  HT = HT - XI: IF HT < 1 THEN HT = 37:VT = VT - 1:
     IF VT = 0 THEN VT = 1:HT = 1
230  RETURN
235  INVERSE : VTAB 21: PRINT "TABLE FULL !! ANY KEY
     TO CONTINUE. ";: GET IN$: NORMAL
240  NP = NP - 1:VT = VT - 1:HT = 37: IF NO < 200 THEN NO = NO + 1
245  RETURN
255  HOME : PRINT "MUSICAL NOTES FOR THE APPLE": PRINT :
     PRINT "BY PHILLIP BOWERS": PRINT :
     PRINT "ROCHESTER, N.Y.":PRINT
260  BM = 120: INPUT "ENTER BEATS PER MINUTE (4TH NOTE) ";B$:
     SB =  VAL (B$): IF SB > 0 THEN BM = SB
265  BV$ = "4TH":BV = 1:SB = 60 / BM:
     TM =  INT (((SB * (1E + 6)) / 36372) + .5)
270  ASSEM = 24576: DIM KV(11,1): DIM NT(199,3): FOR X = 0 TO 11:
     READ NO,NP:KV(X,0) = NO:KV(X,1) = NP: NEXT :KL$ = "ABCDEFG"
275  VT = 1:HT = 1:XI = 4:NO = 0:NP = 0:ST = 13.75:SC = ST:OL = 1
280  X =  PEEK (ASSEM): IF X <  > 165 THEN IN$ =  CHR$ (4):
     PRINT IN$;"BLOAD ASSEM SOUND"
285  HOME : RETURN
290  DATA  81,1,17,1,87,2,69,3,5,3,82,4
295  DATA  18,4,84,5,89,6,25,6,85,7,21,7
300  VTAB 21: HTAB 1:
     PRINT "PROGRAM END ROUTINE                    ": END
```

the initial beats per minute.

Option "C" does not directly END the program, rather it passes control to line 300. The lines 300 through end have been left open so that you can save the note table, or whatever else you may want to do before ENDing.

Any key other than those mentioned above will be ignored. The program will just continue along its merry way.

The note table is defined in line 270

DIM NT(199,3)

where

NT(NO,0) = low order value for the NT(NO,1) = high order value for the note.

NT(NO,2) = low order value for the note length.

NT(NO,3) = high order value for the note length.

The current number of table entries is equal to NO - 1. The value of ASSEM is also set in line 270.

Even though the number of entries can be made greater than 200, it is not suggested because you will lose the relationship between the screen notes and the notes in the note table once more than 200 notes are entered.

In conclusion, I would like to point out that the note table (NT) uses over 9 times more space than is necessary to store the note values and their lengths. This is because we are using an array defined by a Basic program. Because of this it is not possible to use HGR (page 1). While the basic program uses about 2400 bytes, the note table requires an additional 9600 bytes to save the 800 bytes needed by the assembler program.

While it is possible to POKE these values directly into RAM, it should be noted that it will actually require 1000 bytes to store the data. An additional byte is needed for each note to indicate a rest value. In the basic program a rest note uses the same 4 data bytes as any other note, except that the low order rest value is negative (NT(NP,0))...line 100 in the program.

When the notes are replayed the absolute value (ABS) value is POKEed into address $FC (252), and the assembler program is altered so as not to reference the speaker location when a negative value is encountered. But it is executed in the same manner as any note. So if you decide you need more space, or want to use HGR, then remember to include the additional byte for each note.