# SINE WRITING

**In the last section of Workshop we built a digital-to-analogue converter to expand our user port system. We can now begin to design software to produce sound signals from this device. In this instalment we look at the production of different waveforms and discover how to determine the duration of a note.**

## The Project

You can monitor output either via headphones or through a stereo, using the pair of output sockets nearest the potentiometer on the D/A box. You will also need to make some simple connecting leads. For headphones: use the stereo jack plug fitted, obtain a suitable in-line socket and solder the two positive tags in this socket to a lead connected to the red socket on the D/A box. For a stereo: consult your amplifier manual to locate the audio-IN and earth connections and then make the lead. In-line stereo jack sockets are obtainable from many electrical shops. Now follow these steps: connect the buffer box and the D/A converter and plug the buffer box into the user port; plug in the headphones or stereo amp leads to the D/A sockets; turn the potentiometer on the D/A converter fully to the left and then provide transformer power to the box.

IAN McKINNELL

Once these steps have been followed, we can test the system using a short BASIC program. In essence, sound is generated electrically by providing an oscillating voltage to a speaker. We can generate a crude oscillating voltage output from our D/A converter by changing the contents of the user port data register from 0 to 255 and back in rapid succession. Type in the following program and RUN it. Turn the D/A potentiometer clockwise until sound can be heard.

```
10 REM **** CBM BASIC SOUND GENERATOR ****
20 DDR=56579:DATREG=56577
30 POKE DDR,255
35 N=1
40 POKE DATREG,0:FOR I=1TO N:NEXT:POKE DATREG,255:GOTO40

10 REM **** BBC BASIC SOUND GENERATOR ****
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=255
35 N=1
40 ?DATREG=0:FOR I=1TO N:NEXT:?DATREG=255:GOTO40
```

```
      LDX #0          2
LOOP1
      LDA TABLE,X     4
      STA PORT        4
      INX             2
      CMP #STEPS      2
      BNE LOOP1       3
      (2 if loop fails)
```

Notice that the BASIC program has a repeating structure, all crunched down onto a single line to produce maximum speed. There is a delay loop inserted between the data register being set to 255 and it being set to 0. The value N in line 35 sets the length of this delay. Try altering the value of N and re-running the program. You will notice that the pitch of the tone heard goes down as the value of N increases.

The highest pitch obtainable from this BASIC program will occur when the delay loop is removed altogether. Even a loop executed once has an audible effect on the pitch of the note heard.

If you have experimented with different values of N in the BASIC program given, you will have noticed that changing the value of N by 1 has a significant effect on the pitch of the note. BASIC is just not fast enough to allow us to control the rate of oscillation accurately. Instead we must use machine code.

In the next instalment of Workshop we shall look at the difficult problem of controlling pitch and volume from machine code. Here we concentrate on devising a program to produce different waveforms. The waveform produced by the BASIC program used earlier was a square wave. It is, however, possible to produce other waveforms, which alter the 'quality' of the tone heard. We can digitally synthesise sine and saw-tooth waves by taking a number of samples of the waveform and putting them in a look-up table. The machine code program required to place these samples one after another in the data register is in essence very simple. Our illustration shows these three waveforms, together with accompanying look-up tables for sine and saw-tooth waves. If the waveform cycle is divided into steps, and these steps sampled, then the program loop that sends these samples out through the user port is shown left.

In producing sound, timing is crucial. Next to each instruction is the number of machine cycles required to execute that instruction. From this formula we can calculate the total number of machine cycles it takes to produce one complete waveform cycle: number of machine cycles = $2+(4+4+2+2+3)\times$ steps $-1 = 1 + 15 \times$ steps.

If the wave is split into 80 samples then the number of machine cycles required to produce one waveform cycle is 1,201.

As each machine cycle in 6502 code takes about a millionth of a second, the total number of waveform cycles that can be produced in one second (i.e. the frequency of the note) is given by this calculation: frequency = 1,000,000/1201 = 832 Hz. As middle C is 512 Hz, the note produced