22 Making sounds

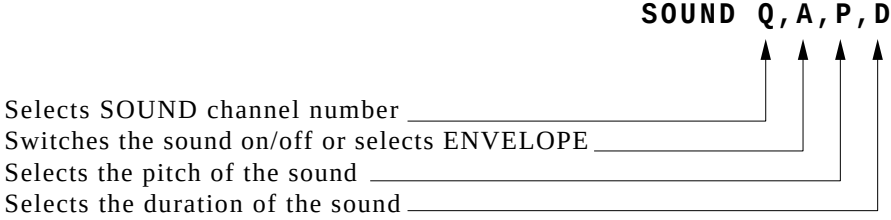
Introduction

Inside the Electron is a sound synthesiser which you can program to generate virtually any sound you like. The synthesiser is controlled by two BASIC commands: **SOUND** and **ENVELOPE**, and each command requires you to type in a series of numbers (or parameters) after it. These parameters determine the type of sound you will hear from the Electron's internal loudspeaker; ie the type of sound, pitch, duration, and so on.

There is a vast range of parameter values which gives you an almost unlimited range of possibilities when writing programs for the synthesiser. However, you don't need to be a wizard in order to use the sound system on a relatively simple level, and this chapter will serve to get you started. Once you've had some practice, read the last section in this chapter which goes into the SOUND commands in more detail.

The SOUND command

The **SOUND** command must be followed by four parameters which we will call **Q**, **A**, **P** and **D**. So the **SOUND** command takes the form



Press ESCAPE and type the following:

```
SOUND 1,-15,0,100 RETURN
```

This will produce a single tone of fairly long duration. Type this fine in again and increase the P parameter (the third number). You'll notice that the pitch increases.

The Q parameter
There are four values of Q, and each one selects the SOUND channel number.

Q=0 Channel 0, selects noise
Q=1 Channel 1, selects tone
Q=2 Channel 2, selects tone
Q=3 Channel 3, selects tone

You may wonder why there are three **SOUND** channels which all select tone as opposed to noise. The only reason for this is to make the sound system as compatible with the BBC Microcomputer as possible. From now on, we will ignore channels 2 and 3.

The A parameter
This parameter does three things, depending on the value you give it:

When A is a negative number, the amplitude is at maximum, ie 'on'. When A is zero, the amplitude is at minimum, ie 'silence'. When A is between 1 and 16 inclusive, an **ENVELOPE** command of the same number is selected. Forget the **ENVELOPE** command for the moment, you will meet it later in this chapter.

Note that for the sake of compatibility with the BBC Microcomputer, use A=–15 for 'sound on', and A=8 for 'silence'.

The P parameter
The value of this parameter controls the pitch of the generated sound. The range of values are from 8 to 255, and each consecutive value will give a quarter-semitone pitch change. The lowest note (P=0) is the B one octave and a semitone below middle C, and the highest accurate note is the B one octave above middle C (P=100). Although values above P=100 are not accurate to the even-tempered scale, they can still be put to good use for making sound effects. The table below is a quick reference guide to help you find the pitches you want.

	Octave Number						
Note	**1**	**2**	**3**	**4**	**5**	**6**	
B	0	48	96	144	192	240	
C#	4	*52	100	148	196	244	*middle C
D	12	60	108	156	204	252	
D#	16	64	112	160	208		
E	20	68	116	164	212		
F	24	72	120	168	216		
F#	28	76	124	172	220		
G	32	80	128	176	224		
G#	36	84	132	180	228		
A	40	88	136	184	232		
A#	44	92	140	188	236		

When the noise channel (channel 9) is selected, the P parameter has the following effect:

P=0 Tone – high pitch
P=1 Tone – intermediate pitch
P=2 Tone – low pitch
P=3 Tone – intermediate pitch
P=4 Noise – short period
P=5 Noise – intermediate period
P=6 Noise – long period
P=7 Noise – intermediate period

The D parameter
The value of this parameter sets the duration of the tone in steps of 50mS. So if D=100, then the duration will be (100 × 50)/100, or five seconds. The maximum value of D is 254 (12.75 seconds).

If D=–l, then the tone will carry on until it is turned off.

Using the SOUND command in a program

Before using the SOUND command in a program, try following through the example below, in order to produce a single sound.

First of all, decide which channel to use: channel 0 will give noise, so use one of the three tone producing channels, say channel 1. So the first number is 1 – this is the value of Q.

Now choose the value for the second number – the A parameter. At the moment we aren't using any **ENVELOPE**s, and we want to hear the sound, so the next number is –15 – this is the value of A.

The third number – the P parameter – determines the pitch. If you look at the pitch table, the C below middle C is the value 4. So P=4.

How long shall the sound last? The fourth number – the P parameter – gives the duration in 50mS (five hundredths of a second). So for a duration of five seconds, P=100.

Now type the following **SOUND** command with the number above:

```
SOUND 1,-15,4,100 RETURN
```

This produces a tone which lasts for five seconds, and is the C below middle C.

Remembering that the pitch value is in quarter semitone steps, then to produce the sound corresponding to C (one semitone higher than the previous one), we must increase the pitch value by four.

```
SOUND 1,-15,8,100 RETURN
```

To play the next note up the scale, increase the P parameter by four again, and so on.

Instead of using numbers in the **SOUND** command, you can incorporate the **SOUND** command into a program, and use real or integer variables (see chapter 11). For example, let's produce a chromatic scale over an octave (equivalent to playing every note in sequence on a piano keyboard over an octave).

Instead of having to type in 13 **SOUND** commands to get the 13 different notes in the scale, we can use a variable for the P parameter, and change the variable value 13 times by using a **FOR . . . NEXT** loop. Try the program below which demonstrates this.

```
5 REM chromatic scale starting at C below m
iddle C up to middle C
10 FOR X%=4 TO 52 STEP 4
20 SOUND 1,-15,X%,5
30 NEXT
```

Lines 10 and 30 set up a loop with variable **X%**, whose values are passed to the pitch parameter in line 20. Line 20 contains the **SOUND** command: channel 1 is selected, the sound is 'on' (A=–15), the pitch is determined by the value of **X%**, and each sound plays for 5 × 50mS, or 0.25 seconds.

As you can hear, each new note only starts after the one before has finished. If you want to put a pause between each note, insert another **SOUND** command between lines 20 and 30, but turn it 'off' by giving the A parameter value 0. The length of the pause will of course be decided by the value you give the last parameter. Try this line:

```
25 SOUND 1,0,X%,10
```

(Obviously, the pitch value here can be anything you like). Now list the program.

```
5 REM chromatic scale starting at C below m
iddle C up to middle C
10 FOR X%=4 TO 52 STEP 4
20 SOUND 1,-15,X%,5
25 SOUND 1,0,X%,10
30 NEXT
```

When you **RUN** this program, you will hear a pause between each note. Try changing the values of **X%** and the duration of the tones and pauses to get different effects. If you want to make a crude sequencer, put the complete program into a **REPEAT...UNTIL FALSE** loop.

ENVELOPE

Each **SOUND** command, as we have seen, allows you to generate a single tone, whose pitch and duration is defined by you within the **SOUND** command. The **ENVELOPE** command allows you to change the single tone into something far more complex. Try typing the following:

```
10 SOUND 1,2,100,100
```

If you run this, you will hear a continuous tone which lasts for five seconds. Now add this fine:

```
200 ENVELOPE 2,1,4,-4,4,10,20,10,0,0,0,0,0,0
```

When you run this program, you'll hear a sound rather like a police siren. Ignore all the numbers after the **ENVELOPE** command, except for the very first one, which defines the **ENVELOPE** number, in this case 2. The second parameter in the **SOUND** command selects **ENVELOPE** number 2; (**ENVELOPE** 2, . . .); the third parameter selects the starting pitch and the fourth parameter selects a duration of five seconds.

Note that once an **ENVELOPE** statement has been executed, it will stay in the computer until it is either re-defined, or until the computer is switched off. So even if you delete line 20 in the program above, when you run the program **ENVELOPE** number 2 will still be selected and used.

The ENVELOPE command

The **ENVELOPE** command is followed by 14 parameters, each separated by a comma. To make life easier, only the first eight actually do anything on the Electron, but the rest must be included for the sake of compatibility with the BBC Microcomputer. These eight parameters are shown below, and are followed by six zeros to make up the I 4 parameters necessary.

```
ENVELOPE n,s,Pi1,Pi2,Pi3,Pr1,Pr2,Pr3,0,0,0,
0,0,0
```

If you want to run Electron programs containing **ENVELOPE** statements on the BBC Microcomputer, then you should get into the habit of always using the following numbers for the last six parameters:

```
ENVELOPE x,x,x,x,x,x,x,x,126,8,8,-126,126,1
26
```

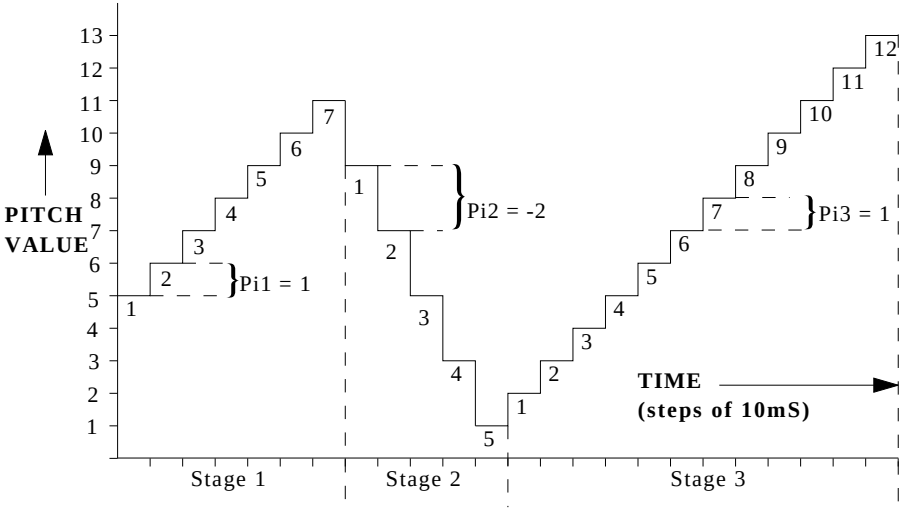
This will ensure that the resulting sound will be the same on both computers. For those of you who are not concerned with compatibility, use any numbers you like. In the rest of this section, 'zeros' have been used to save initially confusing you with too many numbers!

The duration of the **ENVELOPE** effect varies, depending on the values in the **ENVELOPE** command. However, as long as the duration is less than that of its associated **SOUND** command, it will repeat itselfuntil the **SOUND** command finishes. For example, if you set the duration of a **SOUND** command to five seconds and select an **ENVELOPE** which makes a 'WOW' sound lasting for one second, then the result will be 'WOWOWOWOWOW'. With the **ENVELOPE** command, you can:

– Alter the SOUND pitch parameter by a specified increment
– Specify how many increments you want
– Set the length of time each increment 'plays'

The length of each increment can only be specified once in the **ENVELOPE** command, but the increment value itself, and the number of increments can both be specified three times!

Have a look at the graph below, which represents pitch plotted against time for an **ENVELOPE** command. As you can see, the resulting envelope contains three stages. In the first stage, the pitch increases by an increment of 1, then by –2 in the second stage, then 1 in the third stage. In the first stage there are seven pitch increments, in the second stage five, and in the third stage 12. (Notice that the duration of every pitch increment is the same).



The first eight parameters of the **ENVELOPE** command are as follows.

First parameter n
This is the **ENVELOPE** number, and is selected by the second parameter of the **SOUND** command. Values are from 1 to 16.

Second parameter s
This gives the duration of every pitch change in the **ENVELOPE** command, (ie all three stages), and each value corresponds to 18mS. Values are from 0 to 255. Values 1 to 127 give durations of 1 × 10mS to 127 × 10mS, and

when the **ENVELOPE** is finished, it repeats itself until the duration of the associated **SOUND** command has run out Values 128 to 255 are equivalent to 1 to 127, except that at the end of the **ENVELOPE**, the final pitch is held until the associated **SOUND** command has run out.

Third parameter Pi1
This gives the value of every pitch increment duringthe first stage. Values are from –128 to 127.

Fourth parameter Pi2
This gives the value of every pitch increment during the second stage. Values are from –128 to 127.

Fifth parameter Pi3
This gives the value of every pitch increment during the third stage. Values are from –128 to 127.

Sixth parameter Prl
This gives the number of pitch increments in the first stage. Values are from 1 to 255.

Seventh parameter Pr2
This gives the number of pitch increments in the second stage. Values are from 1 to 255.

Eighth parameter Pr3
This gives the number of pitch increments in the third stage. Values are from 1 to 255.

Parameters 9 to 14
These parameters must be put into the **ENVELOPE** command, but their values will have no effect on the effect produced by the Electron's **ENVELOPE** command. In order to keep the command compatible with the BBC Microcomputer, these values should be 126,0,0,–126,126,126.

Constructing an ENVELOPE

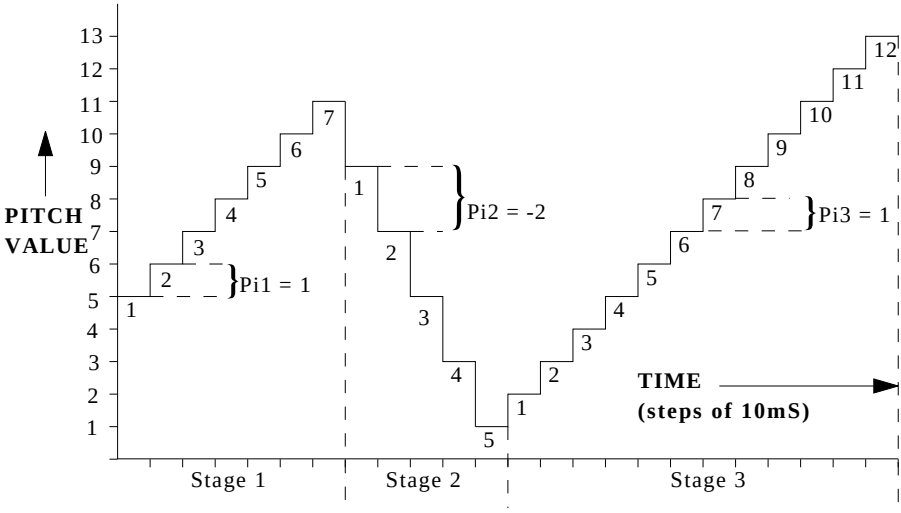
Type in the following program:

```
10 SOUND 1,2,4,50
20 ENVELOPE 2,1,1,-2,1,7,5,12,0,0,0,0,0,0
```

If you run this, you will hear a whining sound. The sound is being transmitted on channel 1, **ENVELOPE** 2 is selected, the pitch value is 4, and the sound lasts for 50 × 50mS or 2.5 seconds.

The graph below shows the effect of the **ENVELOPE** on the sound. This graph is exactly the same as the one at the beginning of this section on the **ENVELOPE** command, and was drawn using the parameter values in the **ENVELOPE** statement above. The parameters are as follows:

n ENVELOPE number 2
s Pitch change duration is 1 × 10mS
Pi1 Stage 1 pitch changes INCREASE in increments of 1, (quarter semitones)
Pi2 Stage 2 pitch changes DECREASE in increments of 2, (half semitones)
Pi3 Stage 3 pitch changes INCREASE in increments of 1, (quarter semitones)
Pr1 Number of pitch changes in stage 1 is 7
Pr2 Number of pitch changes in stage 2 is 5
Pr3 Number of pitch changes in stage 3 is 12



The total time taken for this **ENVELOPE** command is 10ms × total number of pitch changes, or 24 × 0.01 seconds, = 0.24 seconds. So the **ENVELOPE** is repeated over and over again, until the **SOUND** command finishes.

Additionl SOUND features

The first parameter of the **SOUND** command can be given extra values which provide you with some more facilities. To use these values, you must enter the first parameter as a four digit hexadecimal number (which must be preceded by a & sign, to tell the computer that a hexadecimal number follows). The four digits in this number can be represented by HOFC, and here is a description of each digit.

H HOLD — H = 0 for compatibility with the BBC Microcomputer.

O Not used, but should be entered with a value of 0 for compatibility.

F FLUSH — F = 0 sound is queued. This means that the sound won't start playing until the previous one has finished. This is only time if the previous sound is on the same channel as this one. If the channel numbers are different, then the latest **SOUND** instruction to be executed by the computer will immediately start playing in place of the previous one.

F = 1 sound is not queued. This means that as soon as the sound is executed by the computer, any sound previously playing will immediately stop and be replaced by this sound.

C CHANNEL — C = 0 noise channel
C = 1, 2 or 3 tone channels

Example SOUND and ENVELOPE programs

The program below tums the Electron into a keyboard instrument by using the 12345 and QWERTY keys.

```
10 REM Keyboard - Caps Lock on!
20 *FX12,4
30 *FX11,4
40 S$= " Q2W3ER5T6Y7UI9OOP"
50 REPEAT
60 SOUND &11,-15,INSTR(S$,GET$)*4,1
70 UNTIL FALSE
```

This next program takes a string of note names, and plays the notes. The range of notes you can play is from the C below rifiddle C, to the B above middle C. To produce the full range in sequence enter the following: **cdegabCDEFGAB (RETURN)**. This will give you two octaves in the key of C major.

```
10 REM Tune player
20 S$ = " c d ef g a bC D EF G A B"
30 REPEAT
40 INPUT "=>" T$
50 FOR N% = 1 TO LEN T$
60 P% = INSTR(S$,MID$(T$,N%,1))
70 SOUND 1,-15,P%*4,4
80 NEXT
90 UNTIL FALSE
```

Laser Zap! This program uses an ENVELOPE with a downwards pitch sweep. To fire the laser, press any key.

```
10 REM Zap!
20 ENVELOPE 1,129,-15,-8,-3,10,10,10,126,0,
0,-126,126,126
30 REPEAT
40 SOUND &11,1,255,5
50 UNTIL GET = FALSE
```

This last program uses a repeated ENVELOPE with an upwards pitch sweep to produce a spaceship take-off sound.

```
10 REM Liftoff
20 ENVELOPE 1,1,6,6,6,2,2,1,126,0,0,-
126,126,126
30 FOR S% = 0 TO 220
40 SOUND 1,1,S%,1
50 NEXT
```