very fast (a note has a short duration). In addition, note lengths can be more flexibly defined by the addition of dots such as L1...or L5. where each dot increases the note length by half its normal value. Therefore L1... = $1+\frac{1}{2}+\frac{1}{2}+\frac{1}{2}=2\frac{1}{2}$ notes and L5. = $\frac{1}{5}+\frac{1}{10}=\frac{3}{10}$ note.

There is no absolute way in which the relationship between note and tempo can be represented. The values required can vary for each tune and are best selected by trial and error. This may be a little time consuming but it makes the command very flexible.

The parameter O specifies the octave in which the next note is to be played. O1 starts with C at 131Hz and O5 ends with B at 2093Hz. Middle C begins O2 which is the default octave. Within an octave, notes can be specified in two ways. In the first case a number can be used that corresponds to a musical note as follows:

1	2	3	4	5	6
C	C#	D	D#	E	F
7	8	9	10	11	12
F#	G	G#	A	A#	B

This makes it possible to specify a note as a variable within a selected octave. Alternatively, the required note letter can be used directly to make the statement easier to understand in a listing.

The above explanations are best illustrated with an example. The following command plays F (6) in the default octave O2, for half a note length (L2) at default volume V15. It then pauses for a quarter note length (L1) at volume V20. Tempo is set at T3:

```
PLAY "T3;L2;6;P4;O3;V20;L1;A#"
      < F >  <   A#   >
           pause
```

In addition, the T, O, V, and L parameters can be varied by preset amounts from within the command by the addition of a suffix:

Suffix	Effect
+	Adds one to current value
-	Subtracts one from current value
>	Multiplies current value by two
<	Divides current value by two

The format is: T+, T-, T > or T< for each parameter.

The most useful Dragon facility is the ability to PLAY tunes using substrings. These are first defined, and then PLAYed in any order or repeated:

```
10 A$="F;A#;G"
20 B$="C;D#;F;P4;XA$;"
30 PLAY B$
```

This defines A$ and then includes it in B$ as substring XA$. The resulting tune is C—D#—F—P4—F—A#—G. This technique can be continued as necessary where sequences of notes are repeated a number of times within a piece of music. In all cases the semi-colon following a substring must be included, as in XA$, above.

Higher numbers repeat these eight functions but with extra effects, such as dotted lines instead of solid lines. Values of k between 80 and 87 fulfil a particularly useful function. PLOT80,x,y joins the point (x,y) to the two previously plotted points to form a triangle. The triangle is then filled in with the current foreground colour. This provides the only simple means of PAINTing graphic shapes.

VDU x is equivalent to the more usual BASIC command PRINT CHR$(x). We saw in the introduction to graphics on the BBC Micro that VDU can be followed by a series of numbers. VDU v,w,x,y,z is equivalent to:

```
PRINT CHR$(v);CHR$(w);CHR$(x);CHR$(y);
  CHR$(z)
```

The VDU commands allow the user access to the part of the BBC's operating system that controls graphics and screen display. Although VDU commands may be used within BASIC programs they actually work independently of the language employed. Thus the same VDU commands could be used for a graphics display in PASCAL or any other language offered for the BBC. Each of the BASIC graphics facilities so far discussed can also be implemented by the appropriate VDU command.

Defining characters is very easy on the BBC Micro. VDU 23 controls this function. In the section on user-defined graphics (see page 247) we learned that normal ASCII codes are constructed from a block of eight by eight pixels. The pixels that are visible can be represented by a 1 in binary and those not visible by a 0. Each row of eight bits can then be converted to its decimal equivalent, giving a total of eight decimal numbers to define a character. VDU 23 allows the user to redefine the character with an ASCII code between 224 and 255. For example:

```
10 REM DEFINE A CHARACTER
20 MODE 2
30 VDU 23,240,16,56,124,146,16,16,16,0
40 PRINT CHR$(240)
50 END
```

This short piece of program redefines the character with ASCII code 240 to create an arrow shape. The last eight numbers define this new shape, and line 40 PRINTs the character on the screen.

VDU 24 and VDU 28 respectively control the creation of graphics and text 'windows' on the screen. Using these functions, graphics and text output to the screen can be limited to definable areas. This can be particularly useful when designing interactive programs where a split screen is desirable. All that is required to define a graphics window is to specify the co-ordinates of the bottom left- and top right-hand corners.

MODE 1

This short program listing draws a colourful spiral flower on the screen using MODE 1 resolution. Note the use of filled in triangles to produce the flower petals.

```
10 REM FLOWER
20 CLS
30 MODE 1
40 FOR D=1 TO 3
50 A=600 : B=500
60 MOVEA,B
70 FOR C=1 TO 550 STEP3
80 GCOL0,RND(3)
90 S=(C/(RND(5)+10))
100 X=S*5*SIN(C/16)+A
110 Y=S*5*COS(C/16)+B
120 PLOT85,X,Y
130 NEXT C
140 NEXT D
150 END
```

The spiral pattern is produced by the combination of sine and cosine in lines 100 and 110. Normally this relationship between the x and y co-ordinates produces a circle but the FOR...NEXT loop gradually increases the radius C producing the spiral effect. The co-ordinates of the centre of the spiral, A and B, may be altered to re-position the flower