Peek And Poke

These two commands are used whenever you want to program something that Basic can't cope with, but every machine uses them differently

POKEing

The POKE statement needs to be used with care as it changes the contents of memory locations and this could affect the running of the computer. No damage can be caused by this, but it could mean the loss of a program. Here are a few 'safe' POKE statements for you to try.

On the Atari 400 or 800, POKEing a 1 into location 751 will turn the screen cursor off; try POKE 751,1.

On the Commodore 64, try POKE 1024,1. 1024 is the address of the first screen location.

On the Sinclair Spectrum, try:

```
100 FOR N = 0 TO 6 STEP 2
110 POKE USR"A" +
    N,BIN01010101
120 POKE USR"A" + N -
    1,BIN10101010
130 NEXT N
140 PRINT "AAAAAAAA"
```

The As in line 140 must be typed in the graphics mode. Running the program will produce a line of miniature checkerboard symbols. However, it should also result in some interesting interference patterns on your TV set

PEEK and POKE are two 'statements' from the BASIC language used in more advanced programming when individual bits and bytes need to be manipulated in memory. The PEEK statement is used to examine (peek at) the contents of a specific address (location) in memory, and POKE is used to store a number (ranging from 0 to 255) in a specific memory location.

PEEK and POKE statements allow the BASIC programmer to gain access to the inner workings of the computer in a way that is not otherwise possible. Normally, the built-in BASIC in your computer takes care of the actual locations where such things as variables and the data defining the characters to be displayed on the screen are stored. Although we do not usually worry about where such things are in the memory, occasionally we need to find out. The PEEK statement allows us to do this.

A short program to examine any memory location can easily be written:

```
10 REM LOOKING AT MEMORY LOCATIONS
20 PRINT "ENTER MEMORY LOCATION IN
   DECIMAL"
30 INPUT M
40 P = PEEK(M)
50 PRINT "CONTENTS OF LOCATION ";M;" ARE ";P
60 GOTO 20
70 END
```

This will print the contents of the specified address expressed as a decimal number. (In fact, of course, the computer stores it in binary.) If you would like to see what the contents are equivalent to in terms of 'printable' characters, BASIC includes a function to convert decimal numbers into their character equivalents. This is the CHR$ function and changing line 50 will print character equivalents of the memory locations instead:

```
50 PRINT "CONTENTS OF LOCATION ";M;" ARE ";
   CHR$(P)
```

To examine the whole of memory, a FOR...NEXT loop can be added by deleting line 30, changing line 20 to FOR X = 0 TO 65535 and replacing line 60 with NEXT X.

To give enough time to see each character as it is printed, you may need to add a delay loop after the PRINT statement and before the NEXT X statement. Note also that the upper limit of the FOR...NEXT loop assumes you have a 64 Kbyte memory. This number can be changed for smaller memories: 16 Kbytes requires 16383 in decimal, 32 Kbytes requires 32767, and 48 Kbytes requires 49151. A full listing of this program is:

```
10 REM PEEKING AND PRINTING ALL MEMORY
   LOCATIONS
20 FOR X=0 TO 65535
30 LET Y=PEEK(X)
40 PRINT "LOCATION ";X;" = ";Y;" = ";
50 PRINT CHR$(Y)
60 FOR D=1 TO 200
70 NEXT D
80 NEXT X
90 END
```

Although the CHR$ function converts decimal numbers into their character equivalents, printable characters are represented by the numbers 32 to 127. Most computers use the numbers between 128 and 255 (the largest number representable in a single byte) for special graphics characters. Many of the numbers between 0 and 31 have special screen control functions. When these are encountered in memory as the program is run, they will be converted by CHR$ into curious screen effects. These may make the screen go blank, for example, or cause the cursor to move to the top left-hand corner of the screen.

The POKE statement is essentially the opposite of PEEK. It allows you to 'write' a byte of data (any number between 0 and 255) into any memory location. POKE must be used with care: if you POKE a number into the wrong part of memory you could 'crash' the computer by corrupting part of an essential program. The only way to recover from this is to reset the computer (switching it off and then on, unless it has a reset button), and this risks destroying one of your programs. Before using POKE, therefore, check the manual to find an area in the memory map designated a 'user area'.

Most home computers make the video memory (the memory used for storing the characters to be displayed on the screen) available to the user. Normally, the computer gets the shape of the characters to be displayed from a special ROM called a character generator, which stores the patterns of dots for each character. But it is usually also possible to use RAM as well. When the pattern codes for characters are stored in RAM, new patterns, specified as decimal numbers, can be POKEd to the appropriate RAM location and used to define completely new displayable characters.