

## Programming in QBASIC

There are hundreds of different computer languages; BASIC is a simple to use computer language first developed in 1964 (FORTRAN was developed in 1954, C in 1971 and Java in 1995). QBASIC is a version of BASIC that was supplied with free with the first PCs and continued to come with the PC until Windows 2000 and Windows XP.

### Getting started

To use QBASIC on the machines in college you need a floppy disc containing the language.

- Put the floppy disc in the computer
- Restart the computer.

After a little while you should get a screen with a message “Welcome to MS-DOS Qbasic”

- Press <Esc>

### Program 1

You are now ready to write a program.

- Type

```
PRINT "Hello"
```

- Click on **Run** then **Start**

You should see a screen with **Hello** written on it under a lot of other things. At the bottom it says “Press any key to continue”.

- Press a key
- Click on **Run** then **Start**

You should see **Hello** written under the previous **Hello**.

- Press a key
- Now press <F5>

There should now be three lines with **Hello** written under each other. <F5> does the same thing as **Run -> Start**

### Program 2

- Change your program to

```
CLS  
PRINT "Hello"  
PRINT "world"
```

- Press <F5>

CLS stands for clear screen.

### **Program 3**

- Change the program to

```
CLS
PRINT "Hello",
PRINT "world"
```

- Run the program (press <F5>)

What difference does the comma (,) make?

### **Program 4**

- Now change the program to

```
CLS
PRINT "Hello";
PRINT "world"
```

- Run the program (press <F5>)

What does the semicolon (;) do?

### **Program 5**

- Try this program

```
100 CLS
110 PRINT "Hello"
120 GOTO 110
```

- Run the program
- The program will run until you stop it. Hold down <Ctrl> and press <Break> to stop

What does GOTO do?

The numbers 100, 110 and 120 are called line numbers. The only line here that needs a number is 110. Why does line 110 need a number?

### **Program 6**

- Change the program to

```
100 CLS
110 PRINT "Hello";
120 GOTO 110
```

Can you guess what will happen before you run it?

### **Program 7**

- Try this program

```
100 CLS
110 A = 3
120 B = 5
130 C = A + B
```

```
140 PRINT A
150 PRINT B
160 PRINT C
```

A and B are called variables. You can give A and B any value

### **Program 8**

- Change the program to subtract 14 from 30.

### **Program 9**

You can improve the format. Use the example below to make the output of your program easier to read.

```
100 CLS
110 X1 = 3
120 X2 = 5
130 Z = X1 * X2
140 PRINT X1; " times "; X2; " equals "; Z
```

### **Program 10**

You can change a program to allow the user to put their own numbers in.

- Type in this program

```
100 CLS
110 INPUT number1
120 INPUT number2
130 result = number1 / number2
130 PRINT number1; " divided by "; number2; " equals "; result
```

- Run the program

A “?” appears on the screen

- Type in a number and press <return>

Now another “?” appears

- Type in another number and press <return>

### **Program 11**

You can make the program easier to use by adding some words

```
100 CLS
110 INPUT "Enter a number"; first
120 INPUT "Enter another number"; second
130 sum1 = first + second
140 PRINT first; " plus "; second; " equals "; sum1
```

Notice that the variable names can have lots of letters and digits but it must start with a letter.

### **Program 12**

- Write a program to work out the value of resistor to use with an LED when the user puts in the supply voltage.

### **Program 13**

This program uses another type of variable

```
100 CLS
110 INPUT "Enter your name"; name$
120 PRINT "Hello "; name$
```

Any variable that ends in \$ is called a "string". String variables store letters instead of numbers so are not used in calculations.

Change the program to say goodbye

### **Program 14**

Try this program

```
100 CLS
110 INPUT "Enter a number"; first
120 INPUT "Enter another number"; second
130 PRINT first; " plus "; second; " equals "; first + second
140 INPUT "Do you want to do another calculation"; answer$
150 IF answer$="y" THEN GOTO 100
```

What does line 150 do?

### **Program 15**

- Write a program to calculate the frequency of a 555 astable.

### **Program 16**

This program counts

```
100 FOR X=1 TO 15
110 PRINT X
120 NEXT X
```

Can you explain how this program works

### **Program 17**

- Change the program and explain its output

```
100 FOR index=1 to 12
110 fives=index*5
120 PRINT "1 x "; index; " = "; fives,
```

```
130 NEXT index
```

### **Program 18**

- Write a program to output a times table

### **Program 19**

To output the 5 times table you could write

```
100 FOR fives=5 to 60 step 5
110 PRINT fives,
120 NEXT index
```

- Change this program to show the sevens times table.

### **Program 20**

When you use the INPUT command the computer waits for the user to press <return> before continuing. INKEY\$ has the value of whatever key is being pressed at the time or nothing if no key is being pressed.

- Try this program, press keys whilst it is running.

```
DO
KEY$=INKEY$
PRINT KEY$;
LOOP
```

Notice that it is not necessary to use line numbers in your program; you only need line numbers if you need to refer to a particular line.

### **Program 21**

If you just want the user to enter <y> or <n> and not wait for return you can use INKEY\$. If you want to wait for the user to press <y> or <n> you need to keep checking INKEY\$, this is called polling.

```
100 CLS
110 INPUT "Enter a number"; first
120 INPUT "Enter another number"; second
130 PRINT first; " plus "; second; " equals "; first + second
140 PRINT "Do you want to do another calculation (y/n)?"
150 DO
160 A$ = INKEY$
170 LOOP UNTIL A$<>""
180 IF A$="y" THEN GOTO 100
```

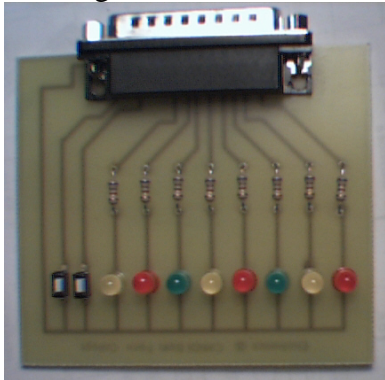
- What does the line 170 LOOP UNTIL A\$<>"" do?

## Further programming in QBASIC

On these sheets you will learn how to use inputs and outputs and use subroutines.

### Program 22

- Plug the LED module into LPT1



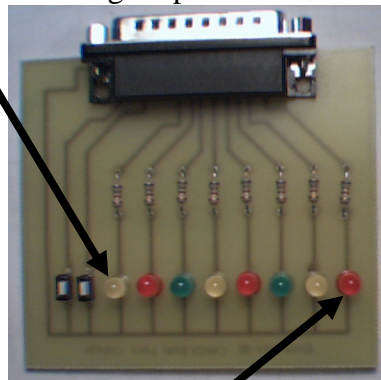
- In the immediate window at the bottom of the QBASIC screen type  
`OUT &h378,&h04<return>`

What happens?

Why is it called the immediate window?

### Program 23

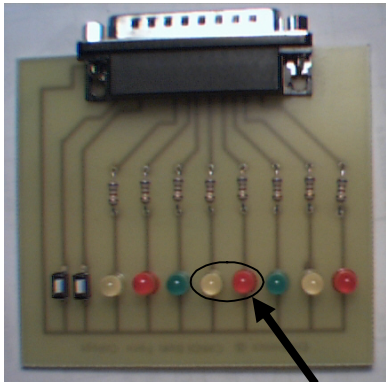
- Make the LED on the left hand side light up



### Program 24

- Make the LED on the right hand side light.

## Program 25



- Turn on the two LEDs in the middle

## Program 26

- Type this program in the program window and run it

```
CLS
PRINT "Press any key to stop"
DO WHILE INKEY$=""
    OUT &h378,&h80
    OUT &h378,&h00
LOOP
END
```

What did you expect to happen?

## Program 27

Program 26 ran too fast for you to see the LED flashing

Change the program to include some time delays

```
CLS
PRINT "Press any key to stop"
DO WHILE INKEY$=""
    OUT &h378,&h80
    FOR J=1 TO 50000
    NEXT J
    OUT &h378,&h00
    FOR J=1 TO 50000
    NEXT J
LOOP
END
```

The FOR – NEXT loop wastes some time so now you can see the LED flash.

### Program 28

- Change the program to make the LED flash quickly

### Program 29

- Write a program to make the LEDs produce a set of traffic lights (red, red and amber, green, amber, red, red and amber, ...)

### Program 30

To save typing and computer memory and to make the program easier to read and more reliable we use *subroutines*. A subroutine is a set of instructions that can be used from anywhere in a program. A subroutine is called with the instruction GOSUB address. At the end of the subroutine the instruction RETURN makes the program continue where it left off.

- Type in this program that uses a subroutine for the time delay

```
100 CLS
110 PRINT "Press any key to stop"
120 DO WHILE INKEY$=""
130 OUT &h378,&h18
140 GOSUB 1000
150 OUT &h378,&h24
160 GOSUB 1000
170 OUT &h378,&h42
180 GOSUB 1000
190 OUT &h378,&h81
200 GOSUB 1000
210 LOOP
220 END
1000 FOR J=1 TO 50000
1010 NEXT J
1020 RETURN
```

### Program 31

- Change the program to make the lights change more quickly

### Program 32

Instead of writing out separate OUT instructions for each output the different patterns can be stored in an *array*. When using an array it must first be *dimensioned* with a DIM statement that gives the number of elements in the *array*.

- Try this program

```
100 CLS
110 PRINT "Press any key to stop"
120 DIM L(4)
130 L(1)=1
140 L(2)=3
150 L(3)=4
160 L(4)=2
170 DO WHILE INKEY$=""
180   FOR X=1 TO 4
190     OUT &h378,L(X)
200     GOSUB 1000
210   NEXT X
220 LOOP
230 END
1000 FOR J=1 TO 100000
1010 NEXT J
1020 RETURN
```

### Program 33

- Change the program to make the lights display the sequence

1	○	○	○	○	○	○	○	●
2	○	○	○	○	○	○	●	○
3	○	○	○	○	○	○	●	○
4	○	○	○	○	○	○	○	○
5	○	○	○	○	○	○	○	○
6	○	○	○	○	○	○	○	○
7	○	○	○	○	○	○	○	○
8	○	○	○	○	○	○	○	○

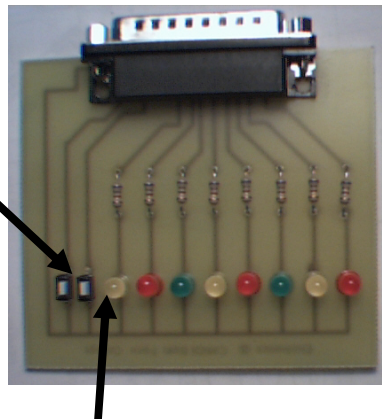
### Program 34

The input port is at address 379H

- Try this program to look at the effect of pressing the buttons on the LED board

```
CLS
DO WHILE INKEY$=""
  CLS
  A=INP(&h379)
  PRINT A
  FOR X=1 TO 10000
  NEXT X
LOOP
END
```

The switch on the left is connected to pin 13, what happens when you press the left button.



The switch on the right is connected to pin 12, what happens when you press the right button.

### **Program 35**

To read a particular button read the input, mask for the required bit and wait for the button to be pressed.

- Change program 32 to respond to the left button

```
100 CLS
110 PRINT "Press any key to stop"
120 DIM L(4)
130 L(1)=1
140 L(2)=3
150 L(3)=4
160 L(4)=2
170 DO
180   FOR X=1 TO 4
190     OUT &h378,L(X)
200     GOSUB 1000
210   NEXT X
220   DO
230     A=INP(&h379)
240     A=A AND &h10
250   LOOP UNTIL A=0
260 LOOP
1000 FOR J=1 TO 100000
1010 NEXT J
1020 RETURN
1030 END
```

### **Program 36**

- Write a program to simulate a pelican crossing