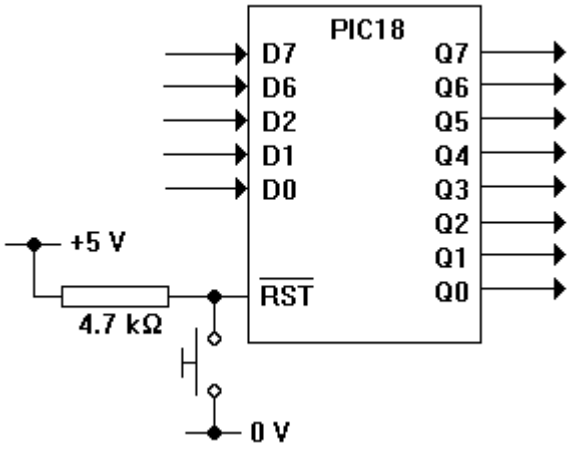
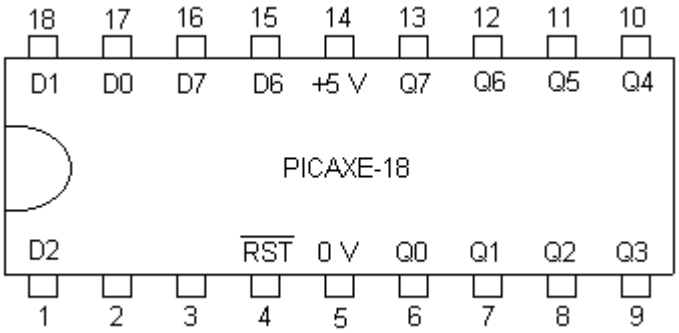
PICAXE introduction

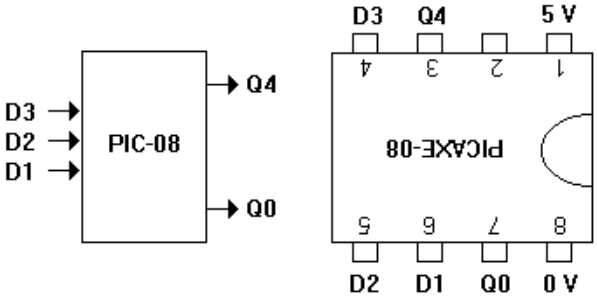
The PICAXE-18 is a cheap programmable logic device with five inputs (D7, D6, D2, D1 and D0) and eight outputs (Q7, Q6, Q5, Q4, Q3, Q2, Q1 and Q0). Programs written in BASIC can be downloaded into the PICAXE-18 via the serial port of a computer. Programs are retained in the PICAXE-18 in the absence of a power supply, and can be over-written as many times as you like. Programs are executed when the $\overline{\text{RST}}$ pin is pulsed low.



Full details of the free programming software and interfacing are to be found at www.picaxe.co.uk.



The PICAXE-08 is a similar programmable logic device with only three inputs (D3, D2, and D1) and two outputs (Q4 and Q0). The BASIC instruction **let dirs = $11** is necessary to assign the input and output pins. Note that there is no $\overline{\text{RST}}$ pin - programs start automatically when power is applied to the i.c. The PICAXE-08 is best inserted upside down on breadboard.



Both the PICAXE-18 and PICAXE-08 can hold programs up to 128 bytes in length, corresponding to about 40 separate commands in BASIC.

The table shows a few of the BASIC commands which can be used to program the PICAXE-18. They use only two eight-bit registers, called the **accumulator** and the **index register**. Use of **only** these 19 commands mimics the machine code instructions used by OCR for their exam questions, providing useful practice in understanding the programming techniques required for machine code.

BASIC command	effect of the command	bytes
let AL = n	place the byte n in the accumulator	3
let AL = AL + n	add the byte n to the byte in the accumulator	4
let AL = AL - n	subtract the byte n from the byte in the accumulator	4
let AL = AL ^ n	exclusive-OR the byte n with the byte in the accumulator	4
let AL = AL & n	AND the byte n with the byte in the accumulator	4
write m,AL	copy byte in accumulator to location whose address is the byte m	5
read m,AL	copy byte from location whose address is the byte m into accumulator	5
let X = AL	copy the byte in the accumulator to the index register	3
let AL = X	copy the byte in the index register to the accumulator	3
let X = X + 1	increment the byte in the index register	3
read X, AL	copy byte at location pointed to by index register into accumulator	4
write X, AL	copy byte in accumulator to location pointed to by index register	4
let pins = AL	copy byte in accumulator to the output port	2
let AL = pins	copy byte at input port to accumulator	3
if AL = 0 then e	jump to program address e if accumulator is zero	4
if AL <> 0 then e	jump to program address e if accumulator is not zero	4
goto e	jump to program address e	2
write m, n	copy byte n into location whose address is the byte m	4
end	stop executing instructions	1

All bytes are written in hexadecimal notation. For example $F2 is the byte 1111 0010.
All programs must start by defining the AL and X registers with the **symbol** command.
Program addresses end with a colon. All comments start with a semi-colon. Here is an example.

```
        symbol AL = b0        ;defines accumulator
        symbol X = b1         ;defines index register

a00:    let AL = pins         ;copy input port to accumulator
a01:    let AL = AL&$C7       ;mask off D5 to D3
a02:    if AL = 0 then a00    ;if accumulator is zero then jump back

a03:    let pins = AL         ;copy accumulator to output port
a04:    goto a00              ;jump back to start
```

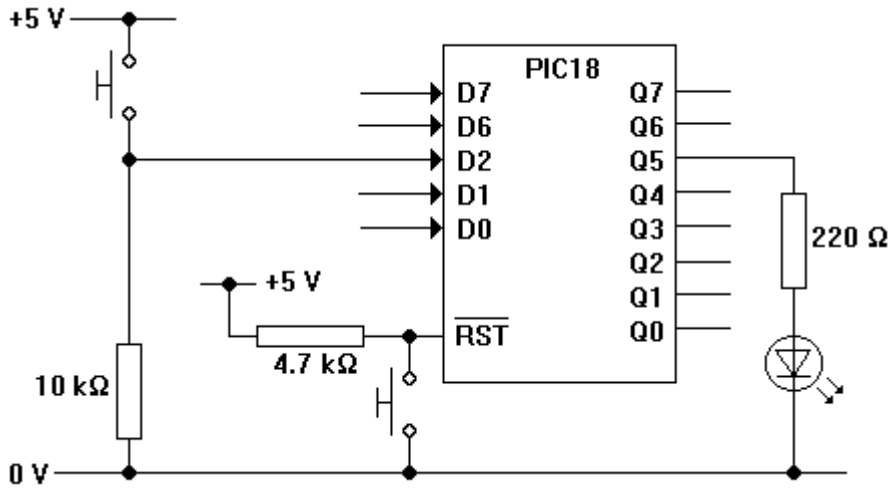
PIC inputs and outputs

You are going to find out how to use the input and output ports of the PICAXE-18. It will help if you have been introduced to the concept of masking and the use of hexadecimal notation.

1 Write the following program into the PIC.

```
        symbol AL = b0
        symbol X = b1

a00:    let AL = $20            ;set accumulator to 0010 0000
a01:    let pins = AL           ;set Q5 to 1, making LED glow

a02:    let AL = pins           ;copy input port to accumulator
a03:    let AL = AL&$04         ;mask off all input pins except D2
a04:    if AL = 0 then a00      ;if D2 = 0 then return to start

a05:    let AL = $00            ;set accumulator to 0000 0000
a06:    goto a01                ;jump to a01
```

2 Set up this circuit on breadboard. Use a red LED. If all is well, the LED should glow only when the pull-up switch is not pressed.



3 Connect the pull-up switch to D7 instead of D2. Connect the LED to Q1 instead of Q5. Adapt the program so that the LED glows only when the pull-up switch is being pressed.

4 Connect a yellow LED and its series resistor to Q6. Adapt the program so that the yellow LED glows whenever the red LED is not glowing.

5 Connect a second pull-up switch and its pull-down resistor to D1. Adapt the program so that the red LED glows when either - or both - of the pull-up switches are pressed. The yellow LED only glows when the red LED is not glowing.

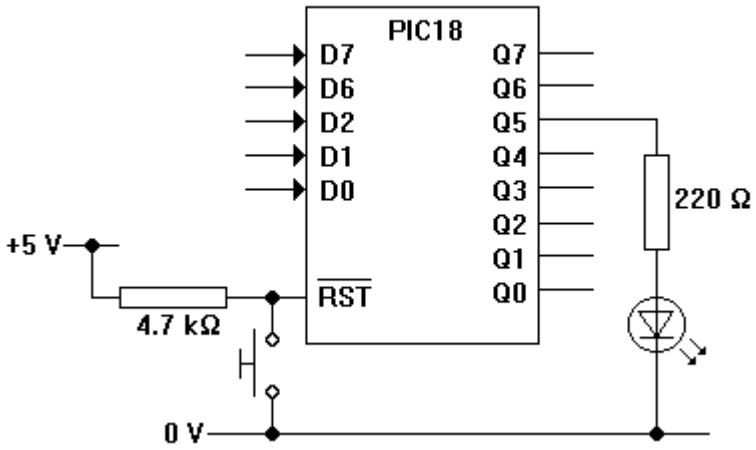
PIC time delays

You are going to find out how to slow down the PICAXE-18. It will help if you have been introduced to the concept of delay loops and the use of the Exclusive-Or function.

1 Write the following program into the PIC.

```
        symbol AL = b0
        symbol X = b1

a00:    let AL = $00            ;set accumulator to 0000 0000
a01:    let X = AL              ;copy accumulator to index register

a02:    let AL = $FF            ;set accumulator to 1111 1111
a03:    let AL = AL - $01       ;subtract one from accumulator
a04:    if AL <> 0 then a03     ;repeat if accumulator not zero

a05:    let AL = X              ;copy index register into accumulator
a06:    let AL = AL^$20         ;toggle bit 5 of accumulator
a07:    let pins = AL           ;copy accumulator to output port

a08:    goto a01                ;jump back to a01
```

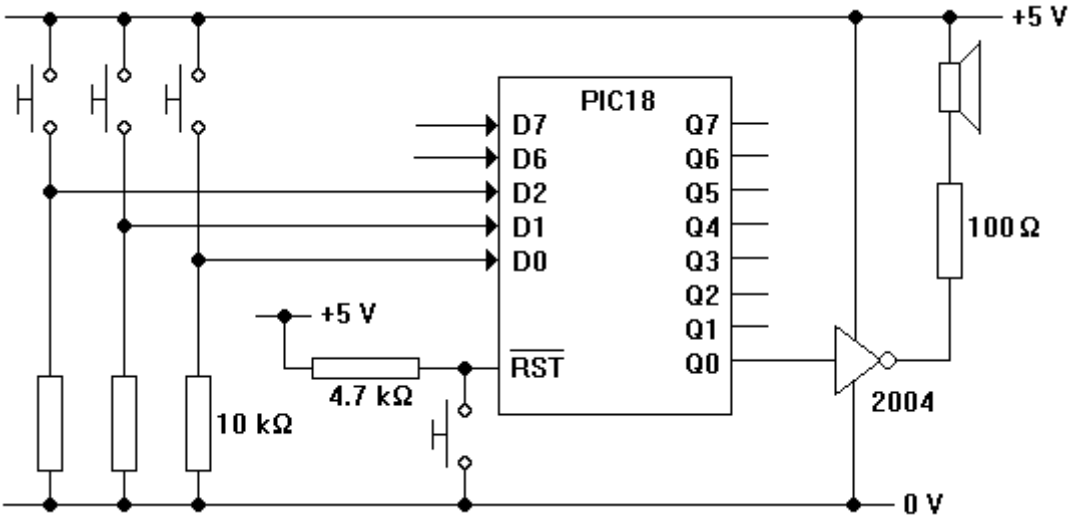
2 Set up this circuit on breadboard. If all is well, the LED should oscillate on and off with a period of about half a second.



3 The program contains a time-delay loop with 255 steps. Adapt the program so that the delay loop has only 8 steps. Use an oscilloscope to measure the frequency of the square wave at Q5. Hence estimate the average time it takes the PIC to execute a single BASIC command.

4 Adapt the program so that the output Q2 feeds out a square wave of frequency 50 Hz.

5 Assemble this circuit on breadboard. Details of the program that will turn this circuit into a musical instrument are given below.



6 Write a program that will make the PIC do the following.

- set Q0 low
- copy the input port to the accumulator
- mask off all bits except 2, 1 and 0
- add one to the accumulator
- subtract one from the accumulator until it reaches zero
- change the state of Q0
- return to the second step

7 If all is well, the pitch of the note produced by the circuit should depend on the binary word D2D1D0 fed in via the three pull-up switches.

8 Very long time delays require the use of a nested delay loop. Study this example.

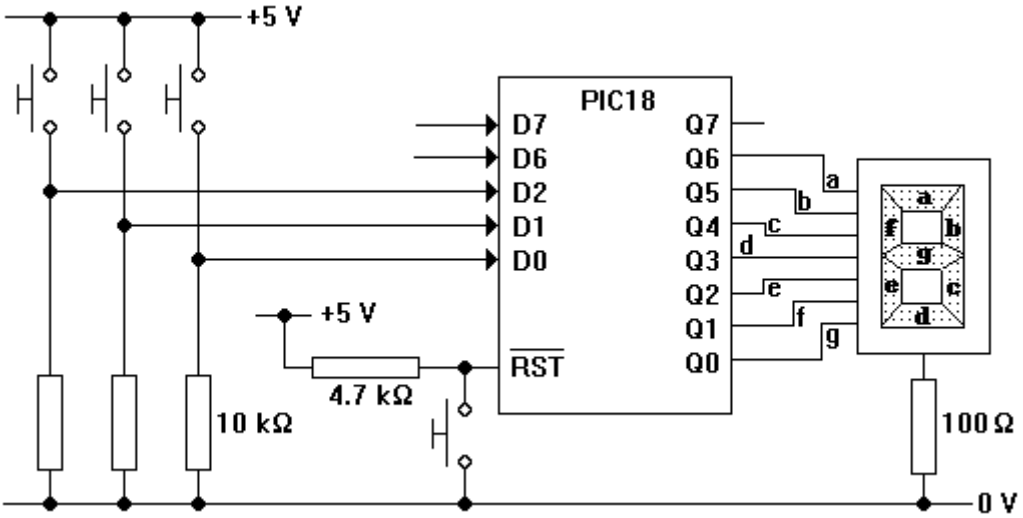
```
        symbol AL = b0
        symbol X = b1
a00:    let AL = $0A            ;set accumulator to 0000 1010
a01:    write $00,AL            ;copy accumulator to location $00
a02:    let AL = $FF            ;short delay loop of length 100 ms
a03:    let AL = AL - $01       ;
a04:    if AL <> 0 then a03     ;
a05:    read $00,AL             ;copy location $00 to accumulator
a06:    let AL = AL - $01       ;subtract one from accumulator
a07:    if AL <> 0 then a01     ;if location $00 not zero, then delay again
a08:    let AL = $01            ;pulse Q0 high
a09:    let pins = AL
a0A:    let AL = $00
a0B:    let pins = AL
a0C:    goto a00                ;return to start
```

9 Write the program into your PIC and replace it in the circuit above. If all is well, the speaker should click at intervals of about one second.

10 Adapt the circuit so that the PIC makes its Q6 output produce a square wave with a frequency of exactly 1 Hz.

PIC look-up tables

1 Assemble this circuit on breadboard.



2 Write this program into the PIC.

```
        symbol AL = b0
        symbol X = b1

a00:    write $00,$7E       ;start of look-up table
a01:    write $01,$30
a02:    write $02,$30
a03:    write $03,$6D
a04:    write $04,$30
a05:    write $05,$6D
a06:    write $06,$6D
a07:    write $07,$79       ;end of look-up table
a08:    let AL = pins       ;copy input port to accumulator
a09:    let AL = AL&$07     ;mask off all bits except 2, 1 and 0 to create an address
a0A:    let X = AL          ;copy look-up table address into index register
a0B:    read X, AL          ;copy byte from look-up table into accumulator
a0C:    let pins = AL       ;copy byte to output port
a0D:    goto a08            ;do it again
```

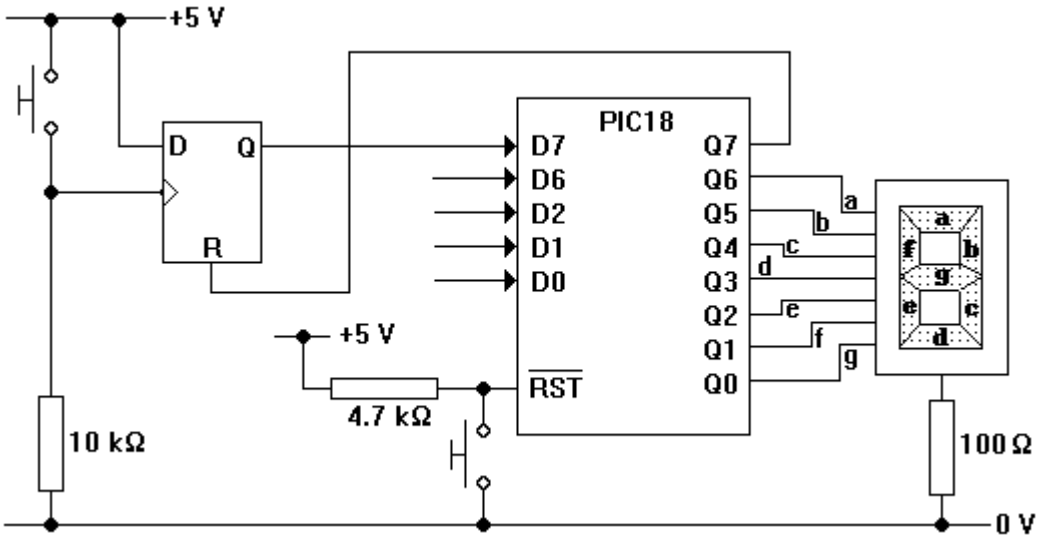
3 Place the PIC in the circuit. If all is well, the number shown on the seven segment LED should equal the number of pull-up switches being pressed.

4 Adapt the program so that the number displayed on the LEDs is as follows.

D2D1D0	display
000	zero
001	one
010	two
011	three
100	four
101	five
110	six
111	seven

PIC handshaking

1 Assemble this circuit on breadboard. Hold the S terminal low!



2 Write this program into the PIC.

```
        symbol AL = b0
        symbol X = b1

a00:    let AL = $80            ;set accumulator to 1000 0000
a01:    let pins = AL           ;push R high and reset flip-flop
a02:    let AL = $00
a03:    let pins = AL           ;pull R low on flip-flop so it can respond to clock pin

a04:    let AL = pins           ;copy input port to accumulator
a05:    let AL = AL & $80       ;mask off all bits except 7
a06:    if AL = 0 then a04      ;repeat until Q = 1

a07:    let AL = $7F            ;set accumulator equal to 0111 1111
a08:    let pins = AL           ;copy accumulator to output port
a09:    let AL = AL - $01       ;subtract one from accumulator
a0A:    if AL <> 0 then a08     ;repeat until accumulator is zero

a0B:    goto a00                ;return to the start
```

3 Place the PIC in the circuit. If all is well, there should be a rapidly changing sequence displayed on the LEDs each time the pull-up switch is pressed.

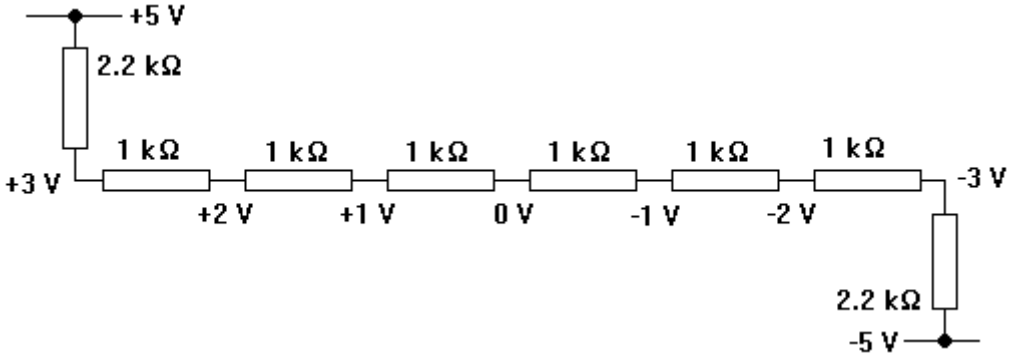
4 Now adapt the program so that the system behaves as follows.

- the flip-flop is reset
- D7 is tested until it goes high
- the LEDs display three, two, one and zero, with a short delay between each change
- the display goes blank and the program returns to the start.

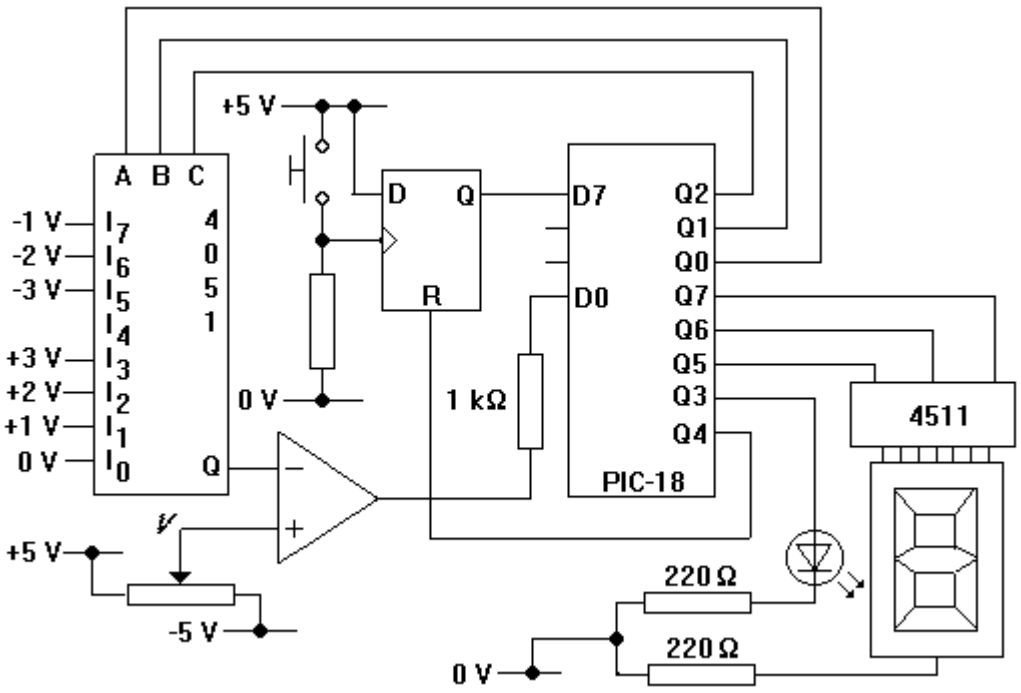
Analgue-to-digital conversion with a PIC

You are going to convert your PC into a simple digital voltmeter.

1 Assemble the resistor ladder shown below. Use a voltmeter to check that the seven reference voltages are approximately as indicated.



2 Now connect the seven reference voltages to the inputs of a 4051 multiplexer, as shown below. Pin 7 of the multiplexer i.c. will need to be connected to -5 V, to allow negative signals through to the output. Assemble the rest of the circuit.



3 Program the PIC so that whenever the switch is pressed, the LEDs display the value of *V*. The program will need to go through the following steps:

- reset the flip-flop and wait until the switch is pressed
- increase the voltage at Q from -3 V to +3 V until the output of the op-amp goes low
- feed appropriate signals to the 4511 decoder and the single LED
- return to the first step

You will need to use look-up tables!

Using a PIC to run OCR programs

You are going to use a program written in BASIC to run a machine code program written using the OCR A2 Electronics instruction set. The BASIC program is called OCRMLTR.BAS, and it runs very slowly in a PICAXE-18A.

1 Use a word processing package, such as EDIT, NOTEPAD or WRITE to open up OCRMLTR.BAS.

2 Follow the instructions in OCRMLTR.BAS to load this OCR machine code program.

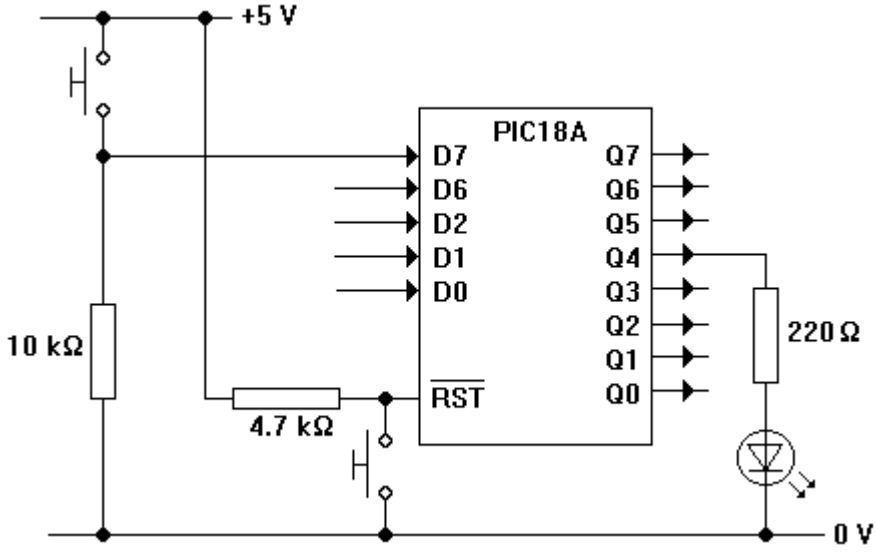
address	code	function
00	3E FF	Load the accumulator with FF
02	32 FF 6F	Copy FF to output port and to index register
05	3A EF E6 80 C2 05	Copy input port to accumulator until D7 is high
0B	7D EE 10	Copy index register to accumulator and toggle bit 4
0E	32 FF 6F	Copy accumulator to output port and index register
11	3A EF E6 80 CA 11	Copy input port to accumulator until D7 is low
17	C3 05	Jump to step three

The program should look as follows when copied into OCRMLTR.BAS:

```
'address 00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F
eeprom ($3E,$FF,$32,$FF,$6F,$3A,$EF,$E6,$80,$C2,$05,$7D,$EE,$10,$32,$FF)

'address 10  11  12  13  14  15  16  17  18  19  1A  1B  1C  1D  1E  1F
eeprom ($6F,$3A,$EF,$E6,$80,$CA,$11,$C3,$05,$00,$00,$00,$00,$00,$00,$00)
```

2 Download OCRMLTR.BAS into a PICAXE-18A integrated circuit.

3 Place the PICAXE-18A into the circuit shown below.



If all is well, the LED should change state every time that you press the switch.