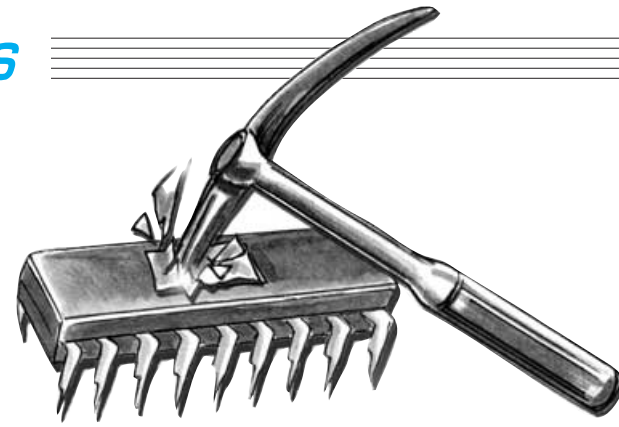
Constructional Projects

PICAXE PROJECTS



MAX HORSEY

Part 1 – Egg Timer, Dice Machine, Quiz Game Monitor

Using the PICAXE system, you do not need specialised equipment or knowledge to program the PIC microcontrollers used in these designs.

THE flexibility of PIC microcontrollers is considerable, and this series of articles is based around a variant of one of them, used with a general-purpose circuit and printed circuit board, allowing nine projects to be realised. The only major difference between them being the program code and casing layouts.

All the projects are based on PICAXE-18 microcontrollers. These are modified versions of Microchip's PIC16F627, a fairly recent addition to the PIC family. They have been modified by Revolution Education to allow them to accept program code written in a form of BASIC. These devices do not need special programming hardware and are simply programmed by means of a serial link to your PC.

IN A FLASH

The basic PIC16F627 and its PICAXE-18 derivative are flash reprogrammable, include an internal oscillator, and analogue as well as digital inputs. The designs presented here can be used with either device, although the standard PIC16F627 needs to be programmed using a conventional PIC programmer.

Details of obtaining the software and the PICAXE system are given later, as are details of obtaining preprogrammed PICs direct from the author, should you not have a computer but still want to build the designs.

The projects to be described are:

Part 1. Digital:
Egg Timer
Dice Machine
Quiz Game Monitor (4 inputs)

Part 2: Analogue:
Temperature Sensor
Voltage Sensor
VU Display

Part 3: Chaser:
Chaser (low voltage)
Interface Circuits
Mains Interface

WHAT IS PICAXE?

The PICAXE system allows you to program a PICAXE-18 device directly in your circuit by means of a 3-wire serial link from your PC-compatible computer. This is achieved by means of a 3-pin connector and 3-wire download cable. The cable is terminated with a 9-pin connector at the PC end, and a 3-pin socket at the circuit board end.

The steps required to construct a PICAXE circuit are as follows:

1. Build the circuit
2. Write the program (already done for you in this series)
3. Connect the circuit to the serial port of your PC
4. Connect the 5V power supply to the circuit
5. Download your program

The PC may now be disconnected from your circuit since the program is safely installed into the PIC. You can modify and download your program as many times as you like (although the PIC has a maximum limit of about 1000 reprogram cycles).

ADVANTAGES OF PICAXE

- Employs easy-to-understand BASIC
- No special programmer required
- Inexpensive
- Offers easy experimentation
- Three of the pins can be used as analogue inputs
- All software is available free of charge

DISADVANTAGES OF PICAXE

- Memory quite limited (128 bytes for the PICAXE-18)
- Limited to a fixed set of five input pins and eight output pins
- BASIC is an inefficient programming method to use with microcontrollers
- Functions such as interrupts are not available

Anyone who is familiar with all the benefits of a normal PIC, and is used to writing programs in assembly code, will see that the disadvantages are considerable. It is quite difficult (impossible?) to use the PICAXE to multiplex an array of 7-segment l.e.d.s., and the limited memory is a considerable problem – although this does encourage the use of more intelligently-written programs.

There is no suggestion that a PICAXE device can replace a conventional PIC in complex systems, but it does provide a very easy introduction to anyone not familiar with assembly code who wishes to join the PIC "club". If you have never programmed a PIC device, you will find that the PICAXE system offers enormous advantages over designing your circuit in a conventional way, i.e. using many logic gates etc.

Having mastered the essentials of the PICAXE system you may then want to progress to assembly code programming. This is quite a leap, but much help is available through, for example, the excellent *PICtutor* (now renamed as *Assembly for PICmicro V2*) by John Becker, available on CD-ROM as detailed elsewhere in this issue.

MASTER CIRCUIT

The general purpose circuit diagram for all the designs in this series of articles is shown in Fig.1.

The PICAXE-18/PIC16F627 microcontroller is shown as IC1. The power supply connections are via pins 5 and 14. Since the chip has an internal oscillator, all the remaining pins are normally available as inputs or outputs. However, to ensure compatibility with the PICAXE system Port A pins RA0, RA1, RA2, RA6 and RA7 are set as inputs, and all Port B pins (RB0 to RB7) are set as outputs.

Pins RA3 and RA4 are configured for serial programming using the PICAXE system. Pin RA5 is not used as a data

input/output pin, but is used in its other role as the $\overline{\text{MCLR}}$ (reset) pin.

The 3-pin connector TB1 and resistors R1 and R2 are required for serial programming if the PICAXE version of the PIC is required. If in-circuit programming is not required then these three components can be omitted, though their inclusion will not otherwise affect the working of the circuit.

Resistor R3 maintains the $\overline{\text{MCLR}}$ pin at logic 1. If reset is required then pin 4 must be briefly connected to logic 0 (i.e. 0V). If reset is required only infrequently then a pair of terminal pins in the TP1 and TP2 positions will suffice, briefly shorting them together when reset is needed.

If reset is required for a particular project, (e.g. to reset the Egg Timer part-way during the timing period) then a pushbutton switch can be connected to TP1 and TP2. Reset also occurs each time you switch off and on.

Note that PICAXE may very occasionally lock up during programming unless a reset is performed.

The first three simple PICAXE projects.

INPUTS

The five digital inputs via Port A are shown connected to pushbutton switches, S1 to S5. In practice these can be any type of switch or a digital logic signal (0V/+5V). The programs assume that if the switch is not pressed then a logic 0 is present, since resistors R4 and R13 to R16 normally bias the inputs to 0V. The pins are held at logic 1 when the switches are pressed.

Note that the switches are labelled in numerical order from left to right, although their numbers as defined in the program are as follows:

S1: Input 2 (RA2)
S2: Input 1 (RA1)
S3: Input 0 (RA0)
S4: Input 7 (RA7)
S5: Input 6 (RA6)

OUTPUTS

The eight digital outputs (from Port B) are coupled to light emitting diodes (l.e.d.s), D1 to D8, via ballast resistors R5 to R12. Each output can supply about 25mA, which can light a standard l.e.d. quite brightly. Note that the maximum total current that the PIC can source or sink via its ports is 200mA.

The three projects discussed here in Part 1 assume that a beeper WD1 is connected to output RB7 in place of l.e.d. D8, with resistor R12 reduced to 12Ω.

The circuit is intended to be powered by a voltage of between 4·5V and 6V, by means of three 1·5V cells, or four 1.2V rechargeable cells. If a mains derived source is employed, then 5V is the ideal supply. The maximum safe voltage that the PIC can accept is 6·5V. Capacitor C1 decouples the supply.

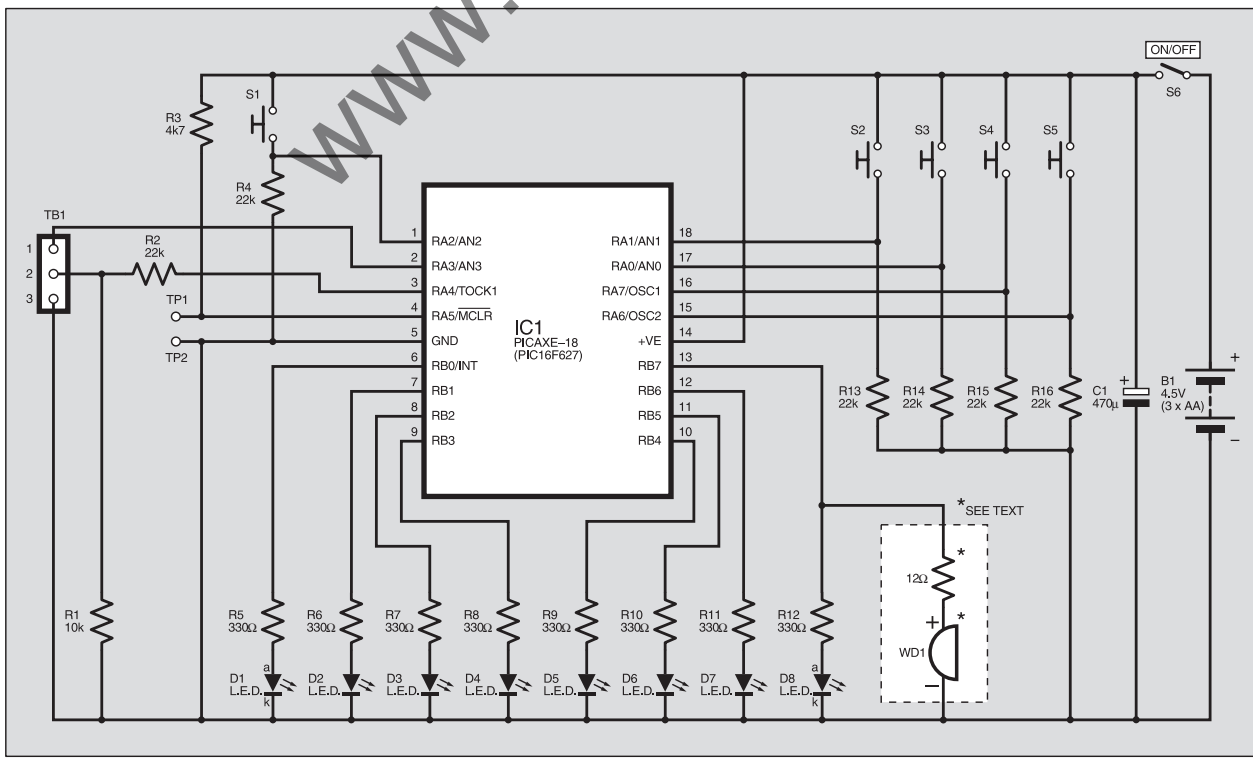


Fig.1. General circuit diagram for all the designs in this PICAXE series of projects.

EGG TIMER

The Egg Timer's design brief was to make a timer which is quick and easy to set (unlike some!), and accurate without the need for calibration. The use of a PIC ensures accuracy and a single pushbutton switch, S2, sets the time required.

When S2 is held pressed, the first seven l.e.d.s, D1 to D7, light one by one, with a brief delay between them responding, each indicating the countdown time required:

D1 1 minute D5 3·5 minutes
D2 2 minutes D6 4 minutes
D3 2·5 minutes D7 4·5 minutes
D4 3 minutes

When the required time is reached, release the switch and the countdown begins. The remaining time is displayed by the appropriate l.e.d. At the end of the timed period the beeper (WD1) sounds for five seconds. The beeper is connected in place of l.e.d. D8 and resistor R12 becomes 12Ω instead of the 330Ω needed for an l.e.d.

A pushbutton reset switch can be added, wired between TP1 and TP2, in case you wish to interrupt the timing cycle, although, as just said, momentarily turning off the power will also cause a reset. This switch was not included with the prototype designs.

Note that the program required for this timer is longer than needed for the other projects, and the full version will not fit into the PICAXE-18. Hence the BASIC program for serial PICAXE in-circuit programming is a cut-down version with just four l.e.d.s. The times chosen are 2·5, 3, 3·5, and 4 minutes. It is very easy to change the program to modify these times. The HEX code file for conventional programming provides the full range of times as described.

DICE MACHINE

With the Dice Machine, again only switch S2 and the first seven l.e.d.s are used, arranged in a pattern as used in dice. The beeper, WD1, is also required. Pressing S2 causes a random number to be displayed.

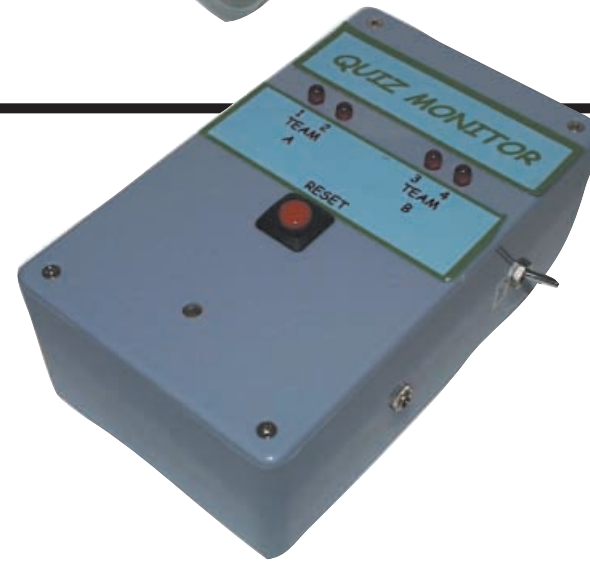
In practice it may be more fun to use a tilt switch instead of S2 so that tilting the circuit will "roll" the dice. Note that the switch must make contact for about a second, hence a vibration switch would not be appropriate.

An in-built delay prevents players from attempting to cheat by knocking the count on by one by tapping the switch quickly. The program counts through the six numbers at high speed, making the l.e.d.s flicker, and stopping when the switch is released. A random number is therefore obtained. The beeper sounds a number of times equal to the number displayed on the l.e.d.s. This provides fun for all, and could also be used by blind players.

QUIZ GAME MONITOR

The Quiz Game Monitor uses switches S1, S2, S3 and S5 for the contestants, and l.e.d.s D4 to D7 indicate who pressed first, mapped as S1/D6, S2/D5, S3/D4, S5/D7. Once more the beeper (WD1) is included in place of l.e.d. D8, operating whenever a contestant's switch is pressed.

Switch S4 is for the Quizmaster to reset the contestant l.e.d.s. Within the software, it simultaneously increments a counter. Although not implemented in the prototype, the value of this counter can be monitored by adding l.e.d.s. D1 to D3. The counter is in binary form and represents the numbers from 0 to 7 and can be used as a Question counter.



CONSTRUCTION

The same printed circuit board (p.c.b.) is used for all the projects in this series. Its full-size, copper foil tracking details are shown in Fig.2, together with all the component positions. The board is available from the *EPE PCB Service*, code 373.

Note that some designs do not need the full set of components, as will be seen from the later wiring diagrams. However, there is no reason why all components should not be included if you wish to experiment with different programs while using the same board.

The component positioning and interwiring details for this month's three circuits are shown in Fig.3 to Fig.5. If you prefer to build the board so that it is specific to the circuit function described, insert only those components that are needed (see Components list and relevant figures), and ignore the p.c.b. holes that are not used. Do make sure that you put the components into the correct holes!

Begin construction by fitting the 18-pin socket for IC1 (but don't insert the PIC itself at this time), followed by the resistors. As mentioned earlier, resistors R1 and R2, and connector TB1, are only required if you wish to use the PICAXE version of the chip and program it via a serial lead, otherwise they can be omitted.

Connector TB1 must be inserted the correct way round, with the plastic tongue nearer the line of l.e.d.s (see Fig.2). Capacitor C1 must also be fitted the correct way round. Attach wires for the l.e.d.s, switches and bleeper as required.

Note that the l.e.d.s have a common cathode (k) and so only one wire is required for all the cathodes as shown in their component layout diagrams. Terminal pins TP1 and TP2 are optional, as discussed earlier.

When assembly has been completed and *thoroughly checked*, insert the PIC the correct way round. If a PICAXE-18 is used, programming should be carried out via the 3-pin serial connector, described shortly. If a normal PIC16F627 is used, then it should have already been programmed using a normal PIC programmer.

COMPONENTS

See SHOP TALK page

Resistors

R1	10k
R2, R4, R13 to R16	22k (6 off)
R3	4k7
R5 to R11	330Ω (7 off)
R12	12Ω or 330Ω (see text)

Capacitor

C1	470μ, radial elect. 16V

Semiconductors

D1 to D8	red l.e.d. and mounting clips (8 off)
IC1	PICAXE-18 microcontroller (see text)

Miscellaneous

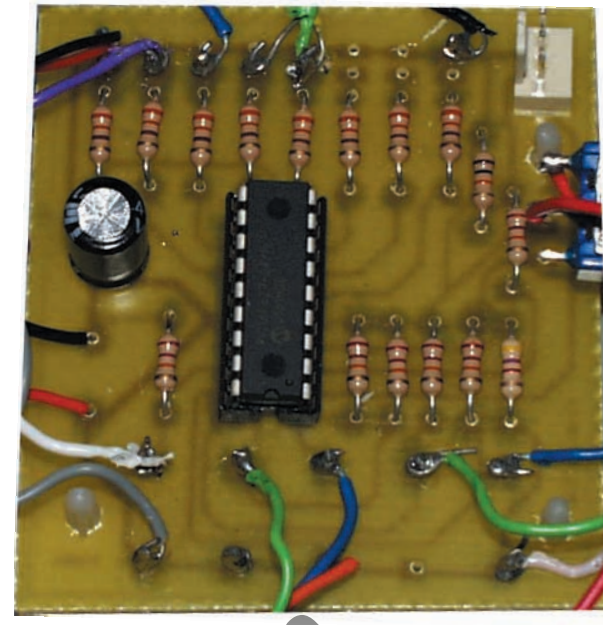
B1	4·5V battery (3 x AA) and clip (see text)
S1 to S5	min. s.p. push-to-make switch (5 off)
S6	min. s.p.s.t. toggle switch
TB1	3-pin serial connector (shrouded 3-pin header) (see text)
TP1, TP2	(see text)
WD1	active buzzer, 5V

Printed circuit board, available from the *EPE PCB Service*, code 373 (1 for each design – see text); 18-pin socket (1 for each p.c.b.); plastic case, size 140mm x 80mm x 30mm approximately (1 per p.c.b.); p.c.b. supports (4 off per p.c.b.); 1mm terminal pins; connecting wires; solder, etc.

Variants

R12 is 12Ω for designs in Part 1 but is 330Ω for some later designs in the series
D8 is not used in Part 1, but is in later parts
R5 to R7 are not used in Quiz Game Monitor (but see text)
D1 to D3 not used in the Quiz Game Monitor (but see text)
S1, S3 to S5 not used in Egg Timer and Dice Machine

Approx. Cost Guidance Only **£18** ***excl. case & batts.***



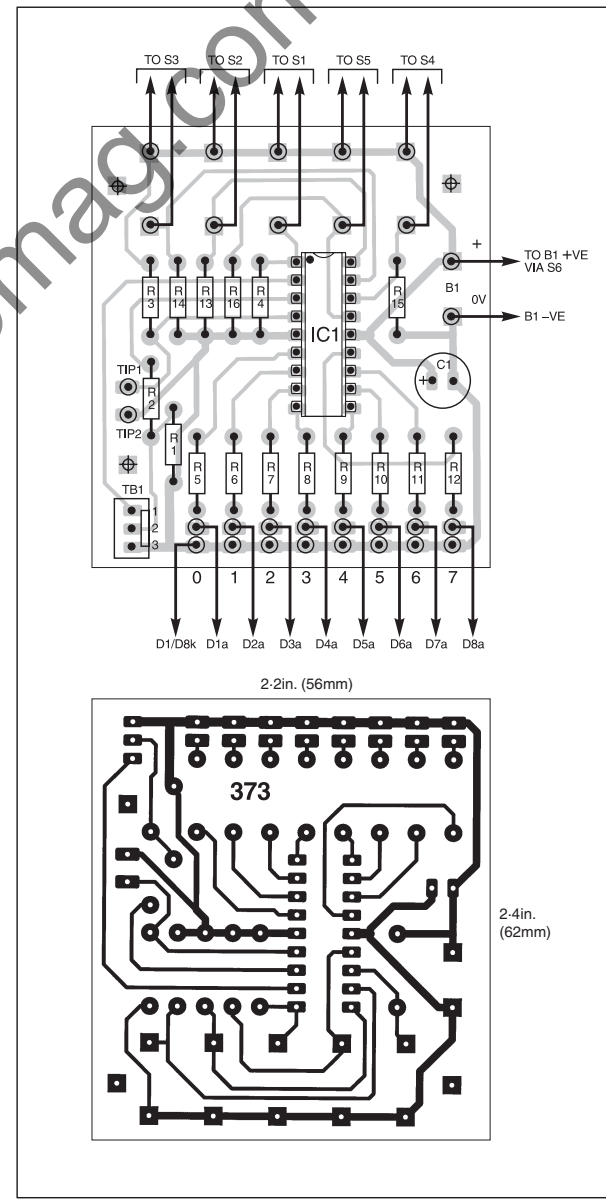
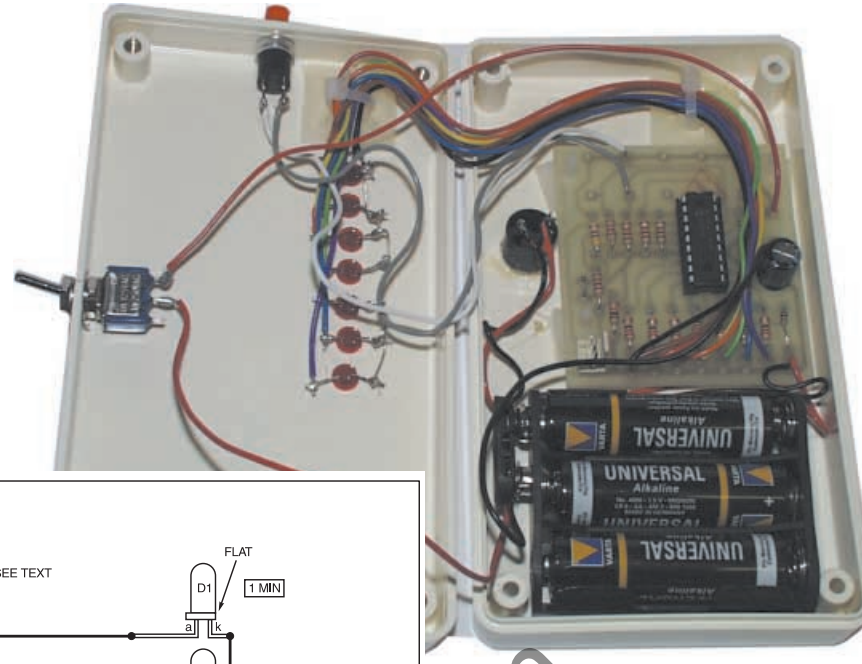


Fig.2. Multiboard topside component layout, full-size foil master and general wiring details.

All three projects described here were housed in plastic cases, measuring approximately 140mm × 80mm × 30mm.

EGG TIMER: This is intended to stand upright on a work surface, and so the batteries should be fitted near the base to aid stability. The component layout and off-board wiring details are shown in Fig.3.

DICE MACHINE: It is intended that this should sit flat on a surface so that the l.e.d.s can be observed from all angles. The component layout and off-board wiring are shown in Fig.4. As mentioned earlier, a tilt switch can be substituted for switch S2 if preferred. Remember that the switch must be closed for at least a second to activate the circuit.



Completed Egg Timer prototype showing general layout and wiring inside the case.

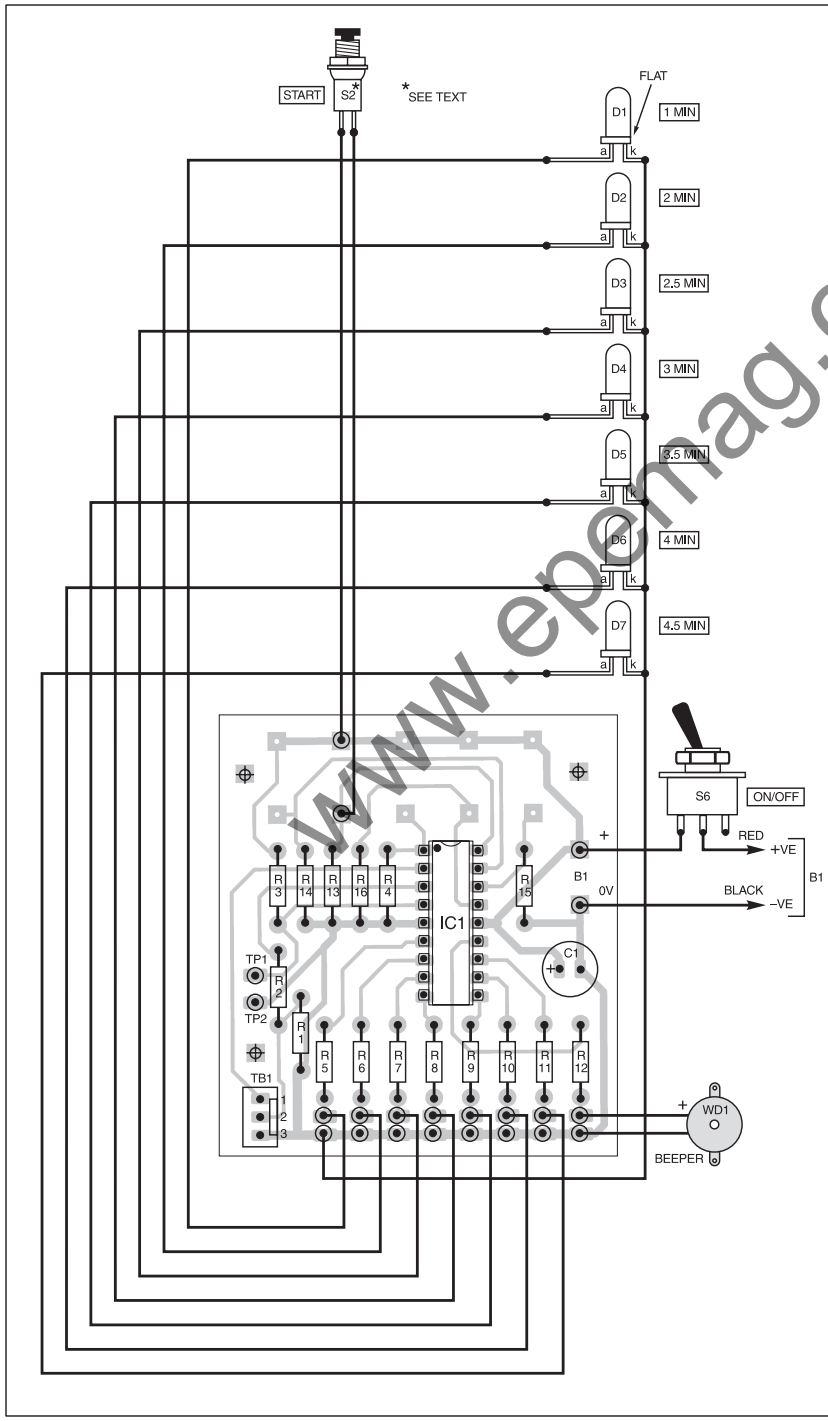


Fig.3. Egg Timer interwiring to off-board components.

QUIZ GAME MONITOR: The component layout and off-board wiring details for the Quiz Game Monitor are shown in Fig.5. Additional small cases are needed for this design, to house the contestant pushbutton switches, one for each contestant, although they could be mounted in pairs in a team contest. They can be directly wired to the main board, or could be connected via jack plugs and sockets if preferred, drilling case holes accordingly.

Note that although the pushbutton switches for the Quiz Game Monitor can all be connected individually to their respective pads on the p.c.b., in a paired situation one wire can be saved by sharing the positive lead as shown, since one side of each switch is connected to positive. This may be useful if the switches are at some distance from the master circuit and allows a 3-core cable to be used. It does not need to be screened.

If the three Question count l.e.d.s (D1 to D3) are required as described earlier, three more holes will need to be drilled in the Quiz Master's case than are shown in the photograph.

PROGRAMMING AND TESTING

PICAXE-18 chips are intended to be programmed in-circuit, and this allows program changes to be made and tested very quickly. Read the instructions provided with the PICAXE system to understand what you need to do to program the code from your computer into the PICAXE-18. The files you need for the PICAXE system are all suffixed with .BAS.

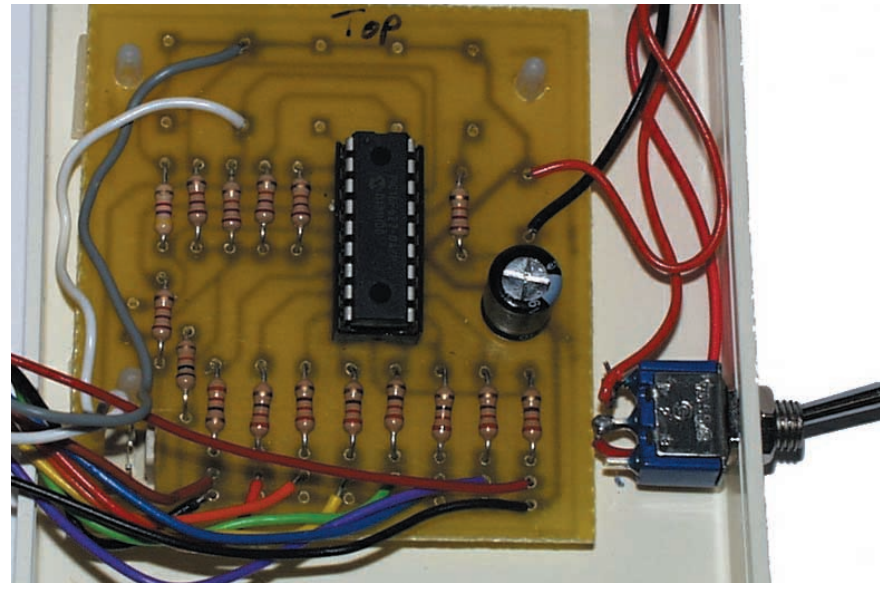
If the computer cannot "find" the PICAXE-18, check the serial connection and see which port is in use. The port setting can be changed within the software at start-up. If the 3-pin connector is the correct way round, resistors R1 and R2 are the correct values and connected correctly, and if the PICAXE-18 is powered correctly

from a supply of about 5V then programming should be successful.

Note that the type of serial cable required is that available at low cost from Revolution Education (whose contact details are given later).

Sometimes it may be necessary to reset the PICAXE-18 just before you send the program. Hold the PIC reset, send the program and release the reset control after about a second. Having released the reset control, the PICAXE then accepts the program.

Still no luck? Ensure that the PIC is a PICAXE chip. PICAXE-18 is a customised PIC16F627 and so the label on the chip will read PIC16F627. It is very easy therefore to get it mixed up with a "normal" PIC16F627. A "normal" PIC16F627 will not work as a PICAXE-18, nor will a PICAXE-18 chip which has been programmed by a standard PIC programmer since the PICAXE code will have been erased in the process.



Component layout on the Dice Machine circuit board.

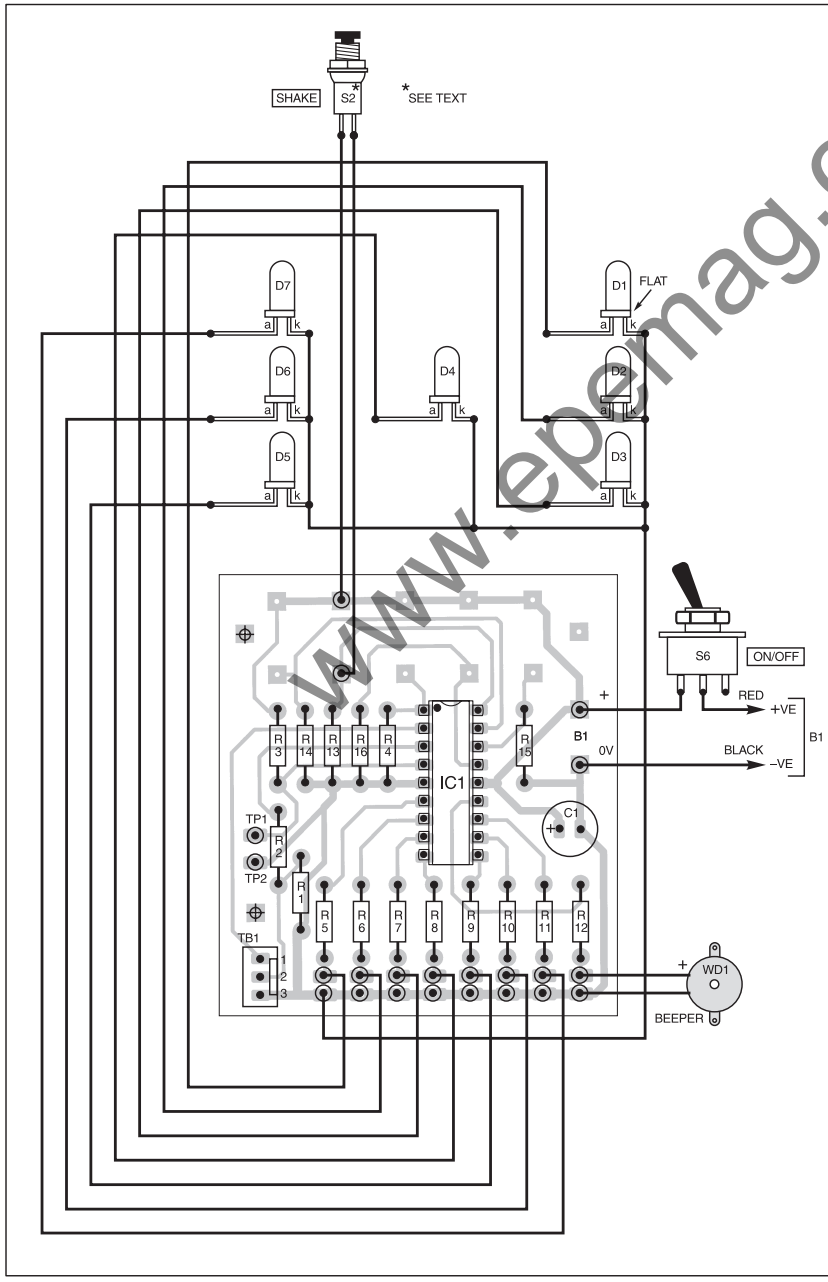


Fig.4. Dice Machine interwiring to off-board components.

Completed Dice Machine showing the front panel layout of the l.e.d.s.

Assuming that the chip has been correctly programmed, faultfinding can be achieved with a voltmeter whose common lead is connected to 0V in the circuit. Now use the positive voltmeter lead to probe around the circuit. Check the voltage at the PIC's power supply pins, then check the voltage at each output.

A +5V reading at any output pin should light the appropriate l.e.d. if it has been connected the correct way round, and assuming that the correct value resistors are fitted. A reading near to 0V should be obtained on pin 18, changing to about +5V (depending on your power supply voltage) when switch S2 is pressed.

Switch anti-bounce protection has been incorporated into each program. Because it causes a delay, it may be necessary to hold

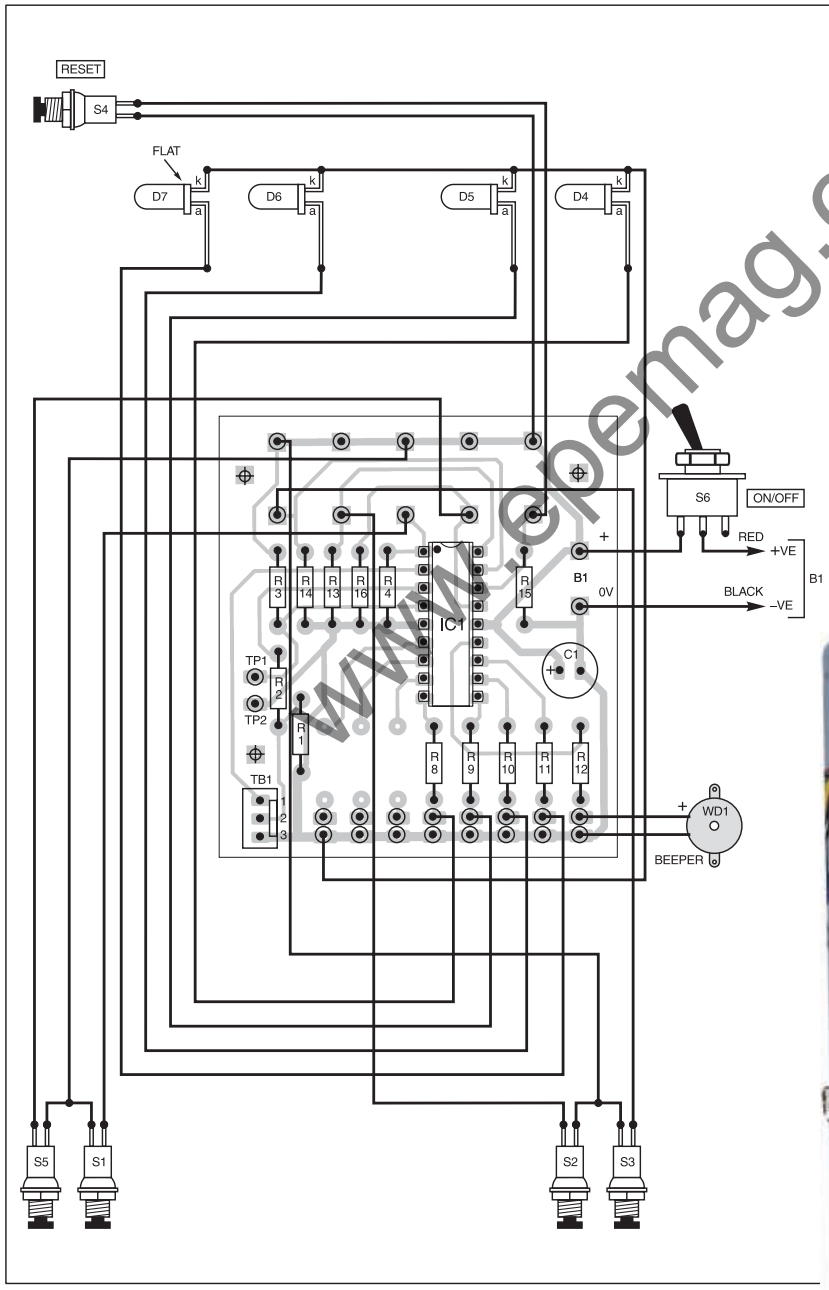


Fig.5. Quiz Monitor interwiring from the p.c.b. to off-board components. Note that "contestant" switches S1,S5/S2,S3 are housed in separate boxes – see photo.

the pushswitch pressed for a second or so. If you think that the PIC has "crashed" try resetting it as discussed earlier.

Note that if you are wishing to program a standard PIC16F627 via a normal PIC programmer, use the HEX file provided. The Watchdog Timer setting must be On.

CREATING A BASIC PROGRAM

We will describe the Egg Timer program (the cut-down version for PICAXE-18) as an example of how to program a PICAXE-18 device. Refer to Listing 1. All the BASIC programs will open in the Windows Notepad text editor and can be modified there.

Note that anything in a program listing which follows an apostrophe is ignored by the system and so is useful for making comments.

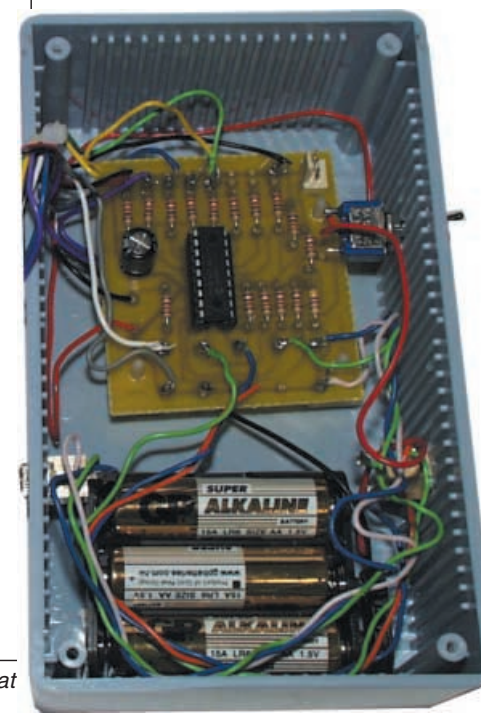
The first nine lines are statements to remind the (human) programmer how the circuit is configured. These lines are ignored by the PIC (or more correctly, the BASIC converter and compiler which generate the code required by the PICAXE-18 microcontroller).

The program begins with the routine starting at the **Make:** label. This examines the logic level at PIC pin 1 (**if pin 1 = 1**). If it is logic 1 (positive – caused by pressing switch S2) then the **Break** routine is entered. Otherwise the program loops back to the **Make:** label. The command **Sleep** is included to reduce power consumption (you will need to read the PICAXE documentation to understand how and why this happens).

NOTATIONS

You will have spotted that switch S2 is connected to pin 18, not pin 1. The command **pin** as used in the program refers to

General component layout in the "master" case.



LISTING 1. Egg Timer program – EGG8.BAS

```
'egg timer by MPH "egg8"  simple version for PICAXE-18), with sleep added
'outputs 0 to 3 = l.e.d.s
'output:
'       0 = 2·5 min.
'       1 = 3 min.
'       2 = 3·5 min.
'       3 = 4 min.
'output 7 = buzzer
'input 1= push switch, high = 1

make:
        if pin1 = 1 then break      'check for switch to be pressed
        sleep 1                     'reduces power consumption
        goto make
break:  high 0
        pause 500
        if pin1 = 0 then twoh       'has switch been released?
        high 1
        pause 500
        if pin1 = 0 then three      'has switch been released?
        high 2
        pause 500
        if pin1 = 0 then threeh     'has switch been released?
        high 3
        pause 500
        goto four
twoh:   let b2 = 25                 'set time factor (2·5 secs)
        goto time
three:  let b2 = 30
        goto time
threeh: let b2 = 35
        goto time
four:   let b2 = 40
time:
        let b3 = b2+b2
        let b3 = b3+b2              'b3 =  b2 X 3
        for b0 = 1 to b3
        let b4 = b3 – b0            'b4 counts down as time elapses
        if b4<75 then twohl
        if b4<90 then threel
        if b4<105 then threehl
        goto fourl
twohl:  let pins = %00000001        'displays one l.e.d.
        goto hold
threel: let pins = %00000011        'displays two l.e.d.s
        goto hold
threehl: let pins = %00000111
        goto hold
fourl:  let pins = %00001111
hold:   pause 2000
        next
        let pins = 0
        high 7                      'sounds beeper
        pause 5000                  'for 5 seconds
        low 7
        goto make
```

the *input number*, and not the i.c. pin number. Fig.1 shows that pin 18 is connected to input 1 (RA1/AN1), so the command **pin 1** in the program refers to pin 18 in the schematic.

Now assume that S2 has been pressed and we have jumped to the **break:** label. The command **high 0** causes output 0 (RB0/INT), to switch high, to turn on the first l.e.d., D1. The system now pauses for 500 milliseconds (**pause 500**). If you release S2 during this time, the next command **if pin 1 = 0 then twoh** causes the system to jump to label **twoh:**. This sets a variable, **b2**, to 25, which will later translate into 2·5 minutes)

There is a limited range of variables in this system. You cannot, for example, state **let x = 25**. A full explanation is available in the help menu provided by the Revolution Education software.

A jump is now made to the label **time:**. The variable **b2** is multiplied by three, by repeated adding. The result is called **b3**. The system now enters a For-Next loop, using another variable, **b0**, (**for b0 = 1 to b3**).

Each time the command **next** is encountered, **b0** advances (increments) by the value of 1, until it reaches the value of **b3**. Variable **b4** counts down and determines the number of l.e.d.s which should be lit. So if **b4** is less than 75 (**if b4<75 then twohl**) the system jumps to label **twohl:**.

This causes Port B to output the binary number %00000001 via the command **let pins = %00000001**. The percentage sign allows the number to be written in binary. The effect is to light the first l.e.d. You could omit the percentage sign and simply use the decimal number 1, but when more l.e.d.s have to be lit, binary notation provides a more visual representation.

After lighting the appropriate l.e.d., a jump is made to **hold:**. This causes a pause for two seconds.

The mathematics of the timing may now be clarified; the original value for **b2** was 25, this was multiplied by three (making 75). The For-Next loop therefore counts from 1 to 75, each time pausing for two seconds. This provides a total delay of 150 seconds, equal to 2·5 minutes.

You may wonder why **b2** is multiplied by three in the program, rather than just stating **let b2 = 75** in the first place. The method was chosen to aid clarity in setting the times. If you look back at the line **twoh: let b2 = 25**, this sets a time of 2·5 minutes. The next setting is 30 (i.e. three minutes), etc.

You can select any time you like by choosing an appropriate number at this stage. Note, however, that no variable can exceed a value of 255, so if you require much longer times, then increase the value in the line **hold: pause 2000**.

When the For-Next loop has finished looping 75 times (i.e. 2.5 minutes) the command **let pins = 0** ensures that no l.e.d.s are lit, and the sequence **high 7, pause 5000, low 7** causes the beeper to sound for five seconds. Note that the command **high 7** has the same effect as **let pins = %10000000**, but is easier if only a single output is being switched. The program now goes back to the **make:** label.

Earlier in this description, during the **break:** sequence, it was assumed that the switch was released during the first 0·5 seconds, hence setting the timer to 2·5 minutes and lighting the l.e.d. connected to output 0. If the switch is held down longer, then the next l.e.d., D2 (output 1) will also light and we jump to a different point, the program thus setting **b2** to a higher number, 30 to achieve three minutes, 35 for three and half minutes or 40 for four minutes.

You can set any time required by changing this number, providing that when multiplied by three the result does not exceed 255.

MORE ON PICAXE

Fuller details on PICAXE programming can be found within the software issued by Revolution Education. Further examples on circuit design and program examples relating to PICAXE devices can be found on the CD-ROM *Modular Circuit Design* available from *EPE*, see CD-ROMs page in the current issue.

RESOURCES

Preprogrammed HEX versions of the PICs for these designs can be obtained from: M.P. Horsey, Electronics Dept., Radley College, Abingdon, Oxon. OX14 2HR. The price is £5 per PIC, including postage. Specify the project for which the PIC is required. Enclose a cheque payable to Radley College.

Software for these three designs (except the PICAXE programming software) is available on 3.5in disk (*EPE* Disk 5), for which a nominal handling charge applies, from the Editorial office. It is also available for free download from the *EPE* ftp site.

PICAXE programming software can be obtained from: Tech-Supplies, Dept. *EPE*, 4 Old Dairy Business Centre, Melcombe Road, Bath, BA2 3LR.

The telephone number of Revolution Education is: 01225 340563, and their web site is at: **www.rev-ed.co.uk**.

Next Month: Temperature Sensor ● Voltage Sensor ● VU Display.