Datalogger

Design Brief

Design a datalogging system that can monitor light and temperature values every 10 minutes for a day.

Circuit Explanation

The two sensors, an LDR to detect light and a 22K thermistor to detect temperature, are connected to the analogue input pins of the PICAXE microcontroller. To record values every 10 minutes (6 per hour) for a day for two sensors requires 2 x 6 x 24 = 288 bytes, which is more than is available inside the microcontroller. Therefore an external memory chip, the 93LC66A EEPROM (with 512 bytes of memory) is used.

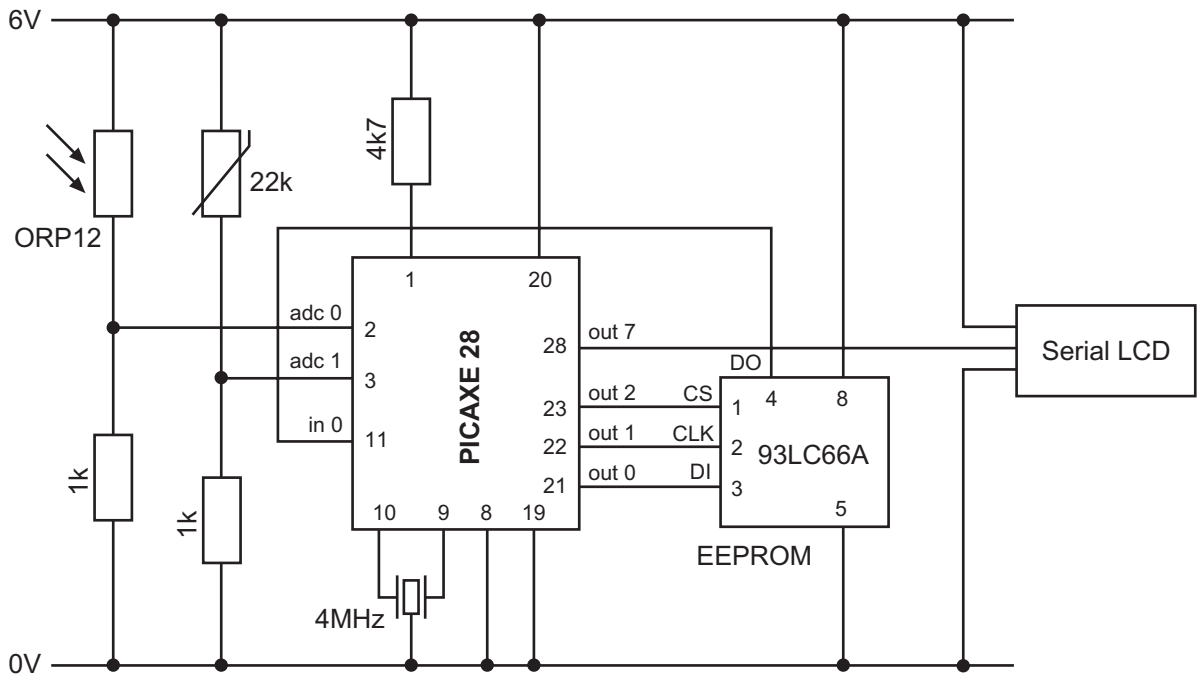
A serial LCD module is used to display the last reading whilst the experiment in underway.

Note that the existing circuit does not have any method of extracting the data from the memory chip after the experiment is over. A simple way to achieve this would be to connect a push switch to input 6, so that every time the switch is pushed the next reading is displayed on the serial LCD.

Program Explanation

The program reads the values from the sensors, and then saves the values into the external EEPROM before updating the serial LCD. The code to drive the EEPROM is fairly complex, but as it is saved as standard sub-procedures it could easily be cut and pasted between programs.

The second program is a simpler program that shows how to read back the data and display it on the serial LCD. The program presumes an extra switch has been connected to input 6. Note the use of the 'pause' command after the switch push has been detected to 'de-bounce' the switch (i.e. prevent multiple pushes being detected by the microcontroller as the switch contact close).



 © copyright 2001 Revolution Education Ltd. Email: info@rev-ed.co.uk Web: www.rev-ed.co.uk

Program Listing

```
‘ Datalogger Experiment
‘ For PICAXE-28


‘ setup symbols for 93LC66 EEPROM
‘ note you must not change the allocated variables
‘ as this will stop the sub-procedures working correctly
symbol    EE_D_I    =    pin0 ‘EEPROM data pin (input 0)
symbol    EE_D_O    =    0    ‘EEPROM data pin (output 0)
symbol    EE_CLK    =    1    ‘EEPROM clock pin (output 1)
symbol    EE_CS=    2         ‘EEPROM chip select pin (output 2)
symbol    data =    b4        ‘data byte
symbol    i    =    b5        ‘scratchpad counter
symbol    ShifReg =  w3        ‘scratchpad shift register
symbol  address =    b8        ‘EEPROM address
symbol    page =    b9        ‘EEPROM page
symbol    clocks    =    b11  ‘scratchpad clock counter


‘First blank the LCD screen
init:
     serout 7,N2400,(254,1)    ‘Clear lcd
     pause 30                  ‘Short pause.


‘ Now take 6x24 = 144 readings every ten minutes
‘ Light is saved on page 1 of the EEPROM memory
‘ Temp is saved on page 1 of the EEPROM memory

main:
     for address = 0 to 143

          readadc 0,data       ‘ read light from adc0
          let page = 0
          gosub eewrite

          serout 7, N2400,(254,128,”Light = “,#data,”  “)

          readadc 1,data       ‘ read temp from adc1
          let page = 1
          gosub eewrite

          serout 7, N2400,(254,192,”Temp  = “,#data,”  “)

          for b0 = 1 to 10 ‘ 10 x 60 second delay
             pause 60000
          next b0

     next address

     end

‘ *** All the code below is standard subs
‘ *** to read/write to EEPROM

‘ This sub-procedure writes a byte to the EEPROM.
‘ ‘Data’ is written to ‘address’ on ‘page’
```

 © copyright 2001

```
eewrite:
     gosub eenabl              'Enable.
     let ShifReg = $A00        'Get the write opcode.
     let ShifReg = ShifReg | w4'OR in the address bits.
     let clocks = 12           'Send 12 bits to EEPROM.
     high EE_CS                'Select EEPROM.
     gosub eeout               'Send the opcode/address.
     let ShifReg = data * 16   'Move bit 7 to bit 11.
     let clocks = 8            'Eight data bits.
     gosub eeout               'Send the data.
     low EE_CS                 'Deselect EEPROM.
     gosub edisbl              'Write Protect.
     return

' This sub-procedure reads a byte from the EEPROM.
' 'Data' is read from 'address' on 'page'

eeread:
     let ShifReg = $C00        'Get the read opcode.
     let ShifReg = ShifReg | w4'OR in the address bits.
     let clocks = 12           'Send 12 bits to EEPROM.
     high EE_CS                'Chip select on.
     gosub eeout               'Send the opcode/address.
     gosub eein                'Receive the byte.
     low EE_CS                 'Deselect the EEPROM.
     return

' Internal EEPROM sub-procedures. Required by eeread and eewrite

eenabl:   let ShifReg = $980   'Get the write-enable opcode.
     high EE_CS                'Chip select on.
     let clocks = 12           'Send 12 bits to EEPROM.
     gosub eeout               'Send the opcode.
     low EE_CS                 'Deselect the EEPROM.
     return

edisbl:   let ShifReg = $800   'Get the write-disable opcode.
     high EE_CS                'Chip select on.
     let clocks = 12           'Send 12 bits to EEPROM.
     gosub eeout               'Send the opcode.
     low EE_CS                 'Deselect the EEPROM.
     return

eein:let data = 0                   'Clear data byte.
     for i = 1 to 8                 'Prepare to get 8 bits.
          let data = data * 2       'Shift EEdata to the left.
          high EE_CLK               'Data valid on rising edge.
          let data = data + EE_D_I  'Move data to lsb of variable.
          low EE_CLK                'End of clock pulse.
     next i                         'Get another bit.
     return

eeout:    for i = 1 to clocks            'Number of bits to shift out.
          let EE_D_I = ShifReg / $800    'Get bit 12 of ShifReg.
        pulsout EE_CLK,10                'Output a brief clock pulse.
        let ShifReg = ShifReg * 2        'Shift register to the left.
     next i                              'Send another bit.
     Return
```

```
‘ Datalogger Read Back Program
‘ For Experiment Done with tut_datalog1.bas
‘ For PICAXE-28


‘ setup symbols for 93LC66 EEPROM
‘ note you must not change the allocated variables
‘ as this will stop the sub-procedures working correctly
symbol    EE_D_I    =    pin0 ‘EEPROM data pin (input 0)
symbol    EE_D_O    =    0    ‘EEPROM data pin (output 0)
symbol    EE_CLK    =    1    ‘EEPROM clock pin (output 1)
symbol    EE_CS=    2    ‘EEPROM chip select pin (output 2)
symbol    data =    b4   ‘data byte
symbol    i    =    b5   ‘scratchpad counter
symbol    ShifReg =  w3   ‘scratchpad shift register
symbol  address =    b8   ‘EEPROM address
symbol    page =    b9   ‘EEPROM page
symbol    clocks    =    b11  ‘scratchpad clock counter


‘First blank the LCD screen
init:
     serout 7,N2400,(254,1)    ‘Clear lcd
     pause 30                  ‘Short pause.

     let address = 0

‘Now show position and light on line1
‘and temp on line2 of the serial lCD
update:
     serout 7, N2400,(254,128,#address, “  “)
     serout 7, N2400,(254,132,”Light=”,#data,”  “)
     serout 7, N2400,(254,196,”Temp =”,#data,”  “)

‘now de-bounce switch and increment address
     pause 500
     let address = address + 1

‘ Now update the LCD display with readings every time switch is pushed
loop:
     if pin6 = 1 then update
     goto loop


‘ *** All the code below is standard subs
‘ *** to read/write to EEPROM

‘ This sub-procedure writes a byte to the EEPROM.
‘ ‘Data’ is written to ‘address’ on ‘page’

eewrite:
     gosub eenabl              ‘Enable.
     let ShifReg = $A00        ‘Get the write opcode.
     let ShifReg = ShifReg | w4‘OR in the address bits.
     let clocks = 12           ‘Send 12 bits to EEPROM.
     high EE_CS                ‘Select EEPROM.
     gosub eeout               ‘Send the opcode/address.
     let ShifReg = data * 16   ‘Move bit 7 to bit 11.
     let clocks = 8            ‘Eight data bits.
     gosub eeout               ‘Send the data.
     low EE_CS                 ‘Deselect EEPROM.
     gosub edisbl              ‘Write Protect.
     return
```

```
‘ This sub-procedure reads a byte from the EEPROM.
‘ ‘Data’ is read from ‘address’ on ‘page’

eeread:
     let ShifReg = $C00         ‘Get the read opcode.
     let ShifReg = ShifReg | w4 ‘OR in the address bits.
     let clocks = 12            ‘Send 12 bits to EEPROM.
     high EE_CS                 ‘Chip select on.
     gosub eeout                ‘Send the opcode/address.
     gosub eein                 ‘Receive the byte.
     low EE_CS                  ‘Deselect the EEPROM.
     return

‘ Internal EEPROM sub-procedures. Required by eeread and eewrite

eenabl:    let ShifReg = $980        ‘Get the write-enable opcode.
     high EE_CS                      ‘Chip select on.
     let clocks = 12                 ‘Send 12 bits to EEPROM.
     gosub eeout                     ‘Send the opcode.
     low EE_CS                       ‘Deselect the EEPROM.
     return

edisbl:    let ShifReg = $800        ‘Get the write-disable opcode.
     high EE_CS                      ‘Chip select on.
     let clocks = 12                 ‘Send 12 bits to EEPROM.
     gosub eeout                     ‘Send the opcode.
     low EE_CS                       ‘Deselect the EEPROM.
     return

eein: let data = 0                   ‘Clear data byte.
     for i = 1 to 8                  ‘Prepare to get 8 bits.
          let data = data * 2        ‘Shift EEdata to the left.
          high EE_CLK                ‘Data valid on rising edge.
          let data = data + EE_D_I   ‘Move data to lsb of variable.
          low EE_CLK                 ‘End of clock pulse.
     next i                          ‘Get another bit.
     return

eeout:     for i = 1 to clocks             ‘Number of bits to shift out.
           let EE_D_I = ShifReg / $800     ‘Get bit 12 of ShifReg.
        pulsout EE_CLK,10                  ‘Output a brief clock pulse.
        let ShifReg = ShifReg * 2          ‘Shift register to the left.
     next i                                ‘Send another bit.
     return
```