AXE114S BINARY CLOCK

Features:

The PICAXE binary clock kit 'tells the time' by lighting up blue LEDs in a binary pattern. This is a useful tool for teaching students binary code or simply just confusing/ amazing those who do not understand the binary system!

- Binary blue LED displays for both date and time.
- Optional switches to implement programmable alarm features
- Optional LDR to implement light sensitive features.
- Optional transparent blue plastic case available (part AXE114C, not included).

Also required:

- 9V DC power supply (e.g. PWR009A - UK only), with a 2.1mm tip positive connector. The clock is not designed to be powered by batteries, and so is supplied with a 1.5m extension lead fitted with 2.1mm connectors.
- AXE026 or AXE027 PICAXE download cable

Optional additions:

- Transparent blue plastic case (part AXE114C)
- Piezo sounder (part SPE002)

Before you start!

The Binary Clock project has been designed to be easily customised by the end user. Several inputs (e.g. three push switches and an LDR light sensor) are provided to allow the user to add customised additional features, e.g. programming an alarm via the switches. The user can therefore decide the purpose of these switches and modify the example program as appropriate.

The PCB has also been designed to optionally fit in the AXE114C plastic case. If this plastic case is used it is necessary to remove the 4 corners of the PCB by carefully cutting along the predrilled route. It may also be necessary to mount the LEDs with 'long legs' and/or mount the push switches on the rear of the PCB. It is not always be necessary to drill holes in the case for the LEDs, as the case is transparent. These various casing/mounting design decisions are left to the end user and must be made before assembly.

Power Supply

The clock is designed to run from a regulated 9V DC (tip positive) power supply (e.g. part PWR009A)

Self-Assembly Kit - Overview:

The Binary Clock is a high quality plated through PCB and is therefore relatively straight forward to assemble. However a number of the electronic components are polarised, so please ensure these components are fitted the correct way around before soldering (see table below).

Tools required (not supplied):

- Soldering iron and solder
- Side Cutters
- Small pair of pliers

Soldering experience is assumed.

Contents:

• PCB	1	Binary Clock PCB		
• IC1	1	18 pin IC socket		
• IC2	1	8 pin IC socket		
• R1	1	22k resistor	(red red orange gold)	
• R2-3	2	1k resistor	(brown black red gold)	
• R4	1	470 resistor	(yellow violet brown gold)	
• A1-4	4	470 resistor array	(6X-1-471LF)	*** text faces bottom of PCB
• A5	1	10k resistor array	(6X-2-103LF)	*** text faces bottom of PCB
• A6	1	4k7 resistor array	(6X-1-472LF)	*** text faces bottom of PCB
• C1	1	100nF (104) polyester capacitor		
• C2	1	100uF electrolytic capacitor		*** + marked on PCB
• LED1-20	20	blue LED		*** + marked on PCB
• R5	1	miniature LDR		
• S1-3	3	miniature push switch (can be mounted either side of PCB)		
• X1	1	4MHz 3 pin resonator		
• Q1-2	2	BC548B transistor		*** flat marked on PCB

Items mounted on bottom of PCB:

• X2	1	miniature watch crystal	
• CT1	1	3.5mm stereo socket	
• REG1	1	7805 regulator	
• BAT1	1	CR2032 cell holder	
• POWER GND	1	power cable	*** white wire to V+, black to

Items inserted into sockets:

• IC1	1	PICAXE-28X microcontroller	*** pin 1 faces left
• IC2	1	DS1307 RTC	*** pin 1 faces left
• BAT1	1	CR2032 cell	

(*** denotes components which must be soldered the correct way around. See notes above).

(Piezo sounder PZ (SPE002 - mounted on rear) is an optional upgrade not included in pack)

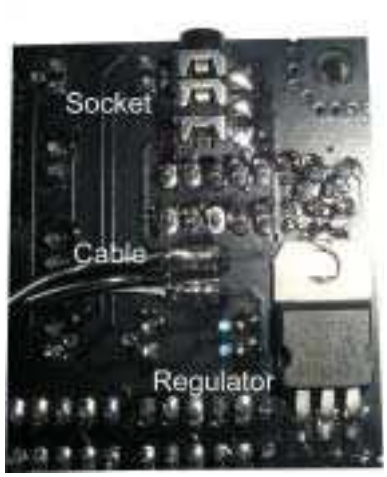
Assembly - Reverse Side:

To reduce PCB size, several components are mounted on the rear of the PCB. These components must be mounted first, before the main assembly.

1. Solder the Watch Crystal in position as shown in photo 1. The body of the crystal can be soldered to the PCB to provide a strong mount. Take care not to overheat the crystal. Carefully trim any excess wire off the two legs after soldering.
2. Solder the coin cell clip in position. Note that the clip will become hot whilst soldering, it is recommended to hold it with miniature pliers!
3. Solder the 3.5mm stereo socket in position as shown in photo 2.
4. Bend and cut the legs of the 7805 voltage regulator as shown in photo 2. Solder in position.
5. Note that it is recommended that the main power cable in photo 2 is connected after the main side assembly.

Assembly - main side

1. Solder the resistors in position.
2. Solder the IC sockets in position.
3. Solder the 6 resistor arrays in position. Note the different values for different positions. The printed text side of the arrays should always face the bottom of the PCB.
4. Solder the resonator in position. Experienced PICAXE users may optionally choose to 'overclock' the chip using a 8 or 16MHz resonator instead (not supplied).
5. Solder the transistors in position, correctly aligning the flat edge.
6. Solder the miniature push switches in position. The switches may be soldered on either side of the board as desired.
7. Solder the LEDs in position, ensuring that the 'positive' side of the LEDs is correctly aligned (note that the top row is different to the other rows). If using the optional plastic case you may choose to leave 'longer legs' on the LEDs, so that they are raised off the PCB.
8. If desired a piezo sounder (not supplied) can be connected to the PZ connection points.
9. Power connection is made to direct solder pads on the board. This is to simplify use of the AXE114C case - if this is used a simple hole can be drilled in the case to pass the power cable through. Therefore cut the 'plug' off the extension cable (leaving the other (socket) end to 'mate' with the power supply connector). Bare the ends of the two cores and solder in position on the rear of the PCB as shown in photo 2 (white marked wire to +9V, black wire to 0V).
10. Insert the ICs into their sockets, ensuring correct orientation.
11. Insert the lithium coin cell into the socket, +ve side up, -ve side touching the PCB.



Operation

The time function is provided by the DS1307 real time clock chip. This requires programming with the correct date and time upon first use. The lithium coin cell maintains the time when the main power supply is removed (the coin cell does not power the LEDs). The DS1307 communicates with the PICAXE-28X chip via the I2C bus.

The LEDs are controlled via the PICAXE-28X chip. The LEDs are multiplexed to both a date and a time LED, as there are more LEDs than available outputs. Transistors Q1 and Q2 control which set of LEDs is displayed at any one time.

The kit will also function with 28X1 and 28X2 PICAXE chips.

Programming the DS1307 RTC

The DS1307 requires programming with the current time upon first use. This is carried out via a Wizard built into the Programming Editor software. Connect power to the Binary Clock and connect the AXE026 or AXE027 download cable. From the Programming Editor software select the PICAXE>Wizards>DS1307 RTC wizard and set the current time. Upon OK the current time will be downloaded and programmed into the RTC. If programming is successful the seconds indicator on the PCB will start flashing every second. This pulsing 'seconds' LED is controlled directly via the DS1307 RTC output pin, it is not under PICAXE program control.

This process is only required once as the lithium coin cell maintains the time when the main power supply is removed.

Programming the PICAXE chip

The PICAXE-28X chip requires programming with a suitable clock program before it will light the LEDs correctly. A default clock program is provided in Appendix A of the full datasheet (download from www.rev-ed.co.uk/docs/axe114.pdf) . This program is also available in the \samples folder of the Programming Editor software. The program is designed as a basic starting point for users to modify and create their own more complex clock programs. This program explains how the date and time are retrieved from the DS1307 (via readi2c commands) and converted to the appropriate LED patterns.

The switches (inputs 0-3), LDR (analogue 0) and piezo (output 6) are not used in the sample program - these are left to the end user to incorporate into their own programs.

Note that the da/time and hours/min LEDs are multiplexed - ie each output is connected to both a time LED and a date LED. Therefore you can not display both time and date at exactly the same time. The transistorw Q1 and Q2 enable/disable each set of LEDs.

To read the time add up the binary weighting of each LED that is lit. For instance if the 'hours' LEDs marked '8' and '2' are lit, and the minutes '16' and '1' LEDs are lit, the time is 10:17

Appendix A - Sample program

```
'***** AXE114 Binary Clock Program *****
#picaxe 28x
'***** Input/Output Pins *****
'- Output LEDs
'output7          min32       -
'output6          min16       day16
'output5          min8        day8
'output4          min4        day4
'output3          min2        day2
'output2          min1        day1
'output0          hours1      month1
'outputc7         hours2      month2
'outputc6         hours4      month4
'outputc5         hours8      month8
' - Other outputs
'output1          (optional piezo sounder PZ)
'outputc2         transistor 2
'outputc1         transistor 1
'- Inputs
'input4           DS1307 SDA
'input3           DS1307 SCL
'input0           switch A (S1)
'analogue 0       LDR
'porta pin1       switch < (S3)
'porta pin2       switch > (S2)
'analogue 3       not used

'***** Variables *****
symbol mins = b0              ' store for minutes value
symbol hours = b1             ' store for hours value
symbol days = b2              ' store for days value
symbol months = b3            ' store for months value
symbol light_value  = b4      ' store for LDR value
symbol temp = b5              ' temporary
symbol temp_port = b6         ' temporary port value

'***** Initialisation *****
init:
      ' setup portC
      ' as some inputs are changed to be outputs
      let dirsc = %11100110
      ' setup i2c for DS1307 clock  operation
      i2cslave %11010000, i2cslow, i2cbyte
```

```
'***** Main Loop *****
main:
      ' read current time from DS1307
      readi2c 0,(temp,mins,hours,temp,days,months)
      'change clock values from BCD to binary
      gosub bcd_bin
      'optionally display on computer via debug
      'debug
      'display the hours and minutes
      gosub display_hours_mins
      'display the days and months
      gosub display_days_months
      goto main

'***** Sub Procedure - convert BCD to binary *****
bcd_bin:
      ' convert the DS1307 BCD values to binary
      temp = mins AND %00001111
      mins = mins AND %11110000 * 10 / 16 + temp
      temp = hours AND %00001111
      hours = hours AND %11110000 * 10 / 16 + temp
      temp = days AND %00001111
      days = days AND %11110000 * 10 / 16 + temp
      temp = months AND %00001111
      months = months AND %11110000 * 10 / 16 + temp
      ' convert 24 hour clock values to 12 hour clock
      if hours < 13 then do_return
      hours = hours - 12
do_return:
      return

'***** Sub Procedure - display the hours/mins *****
display_hours_mins:
      ' display hours and minutes
      let temp_port = mins * 4 AND %11111100
      let temp = hours AND %00000001
      if temp = 0 then hm1
      let temp_port = temp_port OR %00000001
hm1:
      let pins = temp_port
      let pinsc = 0
      let temp= hours AND %00000010
      if temp = 0 then hm2
      high portc 7
hm2:
      let temp= hours AND %00000100
      if temp = 0 then hm3
      high portc 6
hm3:
      let temp= hours AND %00001000
      if temp = 0 then hm4
      high portc 5
```

```
hm4:
      ' now actually light LEDs for a while
      ' by enabling the correct transistor
      high portc 2
      pause 2000
      low portc 2
      return

'***** Sub Procedure - display the days/months *****
display_days_months:
      ' display days and months
      let temp_port = days * 4 AND %01111100
      let temp = months AND 1
      if temp = 0 then dm1
      let temp_port = temp_port OR %00000001
dm1:
      let pins = temp_port
      let pinsc = 0
      let temp = months AND %00000010
      if temp = 0 then dm2
      high portc 7
dm2:
      let temp= months AND %00000100
      if temp = 0 then dm3
      high portc 6
dm3:
      let temp= months AND %00001000
      if temp = 0 then dm4
      high portc 5
dm4:
      ' now actually light LEDs for a while
      ' by enabling the correct transistor
      high portc 1
      pause 2000
      low portc 1
      return


'***** Extra Information *****
' Examples on how to use the other inputs/outputs
' to react to LDR light sensor
'     readadc 0,light_value
'     if light_value > 80 then...
' to make a sound on piezo
'     sound 1,(50,50)
' to read the three switches
'     if pin0 = 1 then...
'     if porta pin0 = 1 then...
'     if porta pin1 = 1 then...
```

Appendix B - Circuit Diagram

to be added