

Display Real Time Clock (RTC) On LCD



Version 1.1

Jan 2008

Cytron Technologies Sdn. Bhd.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Cytron Technologies Incorporated with respect to the accuracy or use of such information or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Cytron Technologies's products as critical components in life support systems is not authorized except with express written approval by Cytron Technologies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

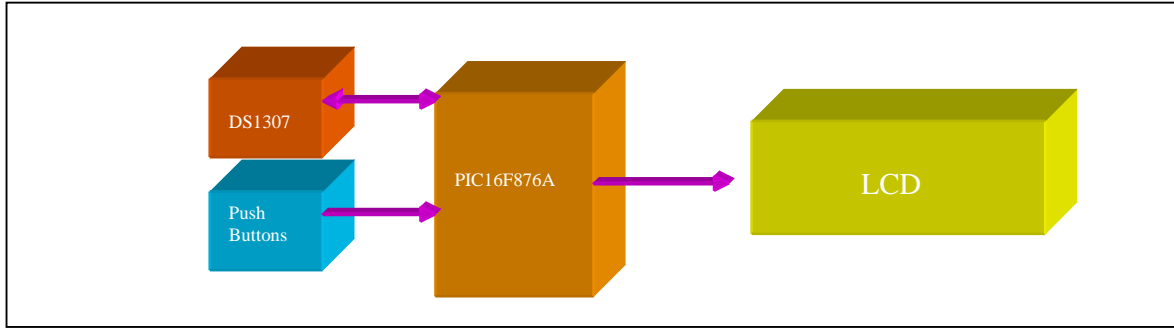
OVERVIEW

This document describes the development of Cytron Technologies DIY (Do It Yourself) Project No.12 (PR12). This project will use PIC16F876A to read or write DS1307 RTC chip and display the time/calendar using LCD screen. The value of time and calendar can be adjusted by push button. Circuit schematic and 2 options of sample PIC source code will be provided. First sample, the time and calendar can only be changed in the program. Second sample, the time and calendar can be changed using push button.

FEATURES

- Real-Time Clock (RTC) Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the week, and Year with Leap-Year compensation valid up to 2100.
- Consist of backup battery which allows the Oscillator keep running even the main power is OFF.
- I²C Serial Interface.
- LCD screen is used to display the time and date.
- Time and calendar can be set by using push button.
- In circuit serial programming (ICSP) used
- Extra switch for further development. Example:-
 - Time alarm by adding a buzzer
 - Timer – activate relay when preset time reached.
 - Record data with time and date known.

SYSTEM OVERVIEW



GENERAL DESCRIPTION

This project uses DS1307 real time clock chip to develop a clock/calendar system. DS1307 will provide the information of seconds, minutes, hours, day, date, month and year. This information will be read by the PIC and displayed on LCD.

PIC16F876A

This powerful (200 nanosecond instruction execution) yet easy-to-program (only 35 single word instructions) CMOS FLASH-based 8-bit microcontroller packs Microchip's powerful PIC® architecture into an 28-pin package and is upwards compatible with the PIC16C5X, PIC12CXXX and PIC16C7X devices. Feature of the device:

- 256 bytes of EEPROM data memory
- Self programming
- ICD (In Circuit Debugging function)
- 2 Comparators
- 5 channels of 10-bit Analog-to-Digital (A/D) converter
- 2 capture/compare/PWM functions
- the synchronous serial port can be configured as either 3-wire Serial Peripheral Interface (SPI™) or the 2-wire Inter-Integrated Circuit (I²C™) bus
- Universal Asynchronous Receiver Transmitter (UART)

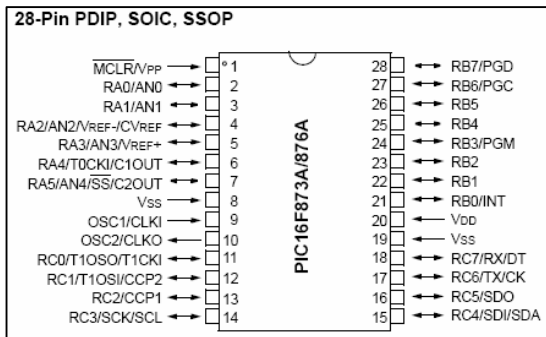


Figure 1

Figure 1 shows the pin diagram of the PIC16F876A. For more detail, please download the datasheet from microchip web site at: <http://www.microchip.com>

DS1307

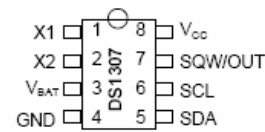


Figure 2

The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM chip. Address and data are transferred serially through an I²C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply. For further information about DS1307, please refer to DS1307 datasheet from MAXIM which can be downloaded from the link below:

http://www.datasheetcatalog.com/datasheets_pdf/D/S/1/3/DS1307.shtml

HARDWARE

This project will require following hardware:

- a. 1 x PIC16F876A
- b. 1 x PR12 Printed Circuit Board (PCB)
- c. 1 x DS1307 RTC chip
- d. 1 x 2X16 LCD
- e. Related electronic components

Please refer to the schematic diagram of PR12. The schematic is provided free therefore Cytron Technologies will not be responsible for any further modification or improvement.

Interface PIC16F876A with DS1307 chip

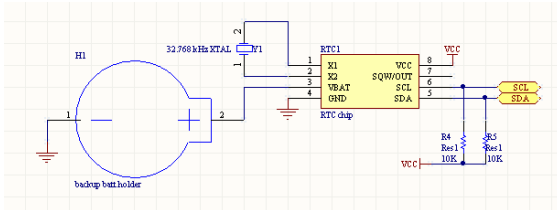


Figure 3

DS1307 chip consists of 8 pins. The first 2 pins (X1 & X2) are connected to a 32.768KHz crystal. The 3rd pin (VBAT) is the backup supply input which connected to a 3V battery. The SDA pin is the data input/output pin for the I²C serial interface while the SCL pin is the clock input for the I²C interface and is used to synchronize data movement on the serial interface. The 2 pins must be connected to the I²C pins at microcontroller (RC3 and RC4) and pulled high. The SQW/OUT pin is the Square Wave/Output Driver. When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square-wave frequencies (1Hz, 4KHz, 8kHz, 32kHz). In this project, the pin is not used.

Interface PIC16F876A with LCD (2X16 character)

To use the LCD, user has to solder 16 pin header pin to the LCD. LCD used in this project is JHD162A, for other type of LCD, please refer to its data sheet.

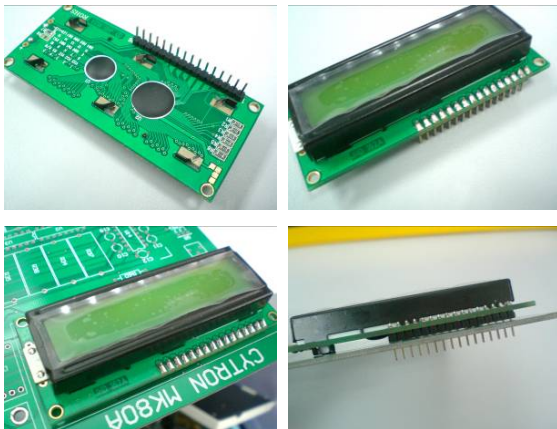


Figure 4

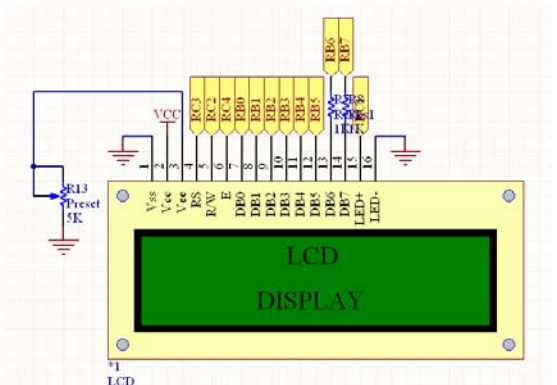


Figure 5

Figure 4 is a 2X16 character LCD. LCD connection pins and function of each pin are shown in table below:

Pin	Name	Pin function	Connection
1	VSS	Ground	GND
2	VCC	Positive supply for LCD	5V
3	VEE	Brightness adjust	Connected to a preset to adjust brightness
4	RS	Select register, select instruction or data register	RA0
5	R/W	Select read or write	GND
6	E	Start data read or write	RA1
7	DB0	Data bus pin	RB0
8	DB1	Data bus pin	RB1
9	DB2	Data bus pin	RB2
10	DB3	Data bus pin	RB3
11	DB4	Data bus pin	RB4
12	DB5	Data bus pin	RB5
13	DB6	Data bus pin	RB6
14	DB7	Data bus pin	RB7
15	LED+	Backlight positive input	VCC
16	LED-	Backlight negative input	GND

Power supply for the circuit

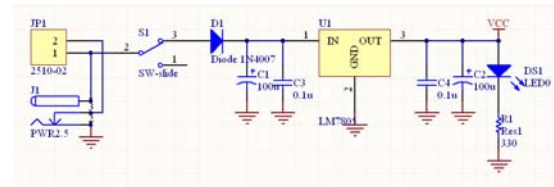


Figure 6

User can choose either to use the AC to DC adaptor or 9V-12V battery to power up the circuit. Higher input voltage will produce more heat at LM7805 voltage regulator. Typical voltage is 12V. Anyhow, LM7805 will still generate some heat at 12V. There are two type of power connector for the circuit, DC plug (J1) and 2510-02 (JP1). Normally AC to DC adaptor can be plugged to J1 type connector. Shown in Figure 6, the D1 is use to protect the circuit from wrong polarity supply. C1 and C3 is use to stabilize the voltage at the input side of the LM7805 voltage regulator, while the C2 and C4 is use to stabilize the voltage at the output side of the LM7805 voltage supply. DS1 is green LED to indicate the power status of the circuit. R1 is resistor to protect DS1 from over current that will burn the DS1.

Push Button as input for PIC microcontroller

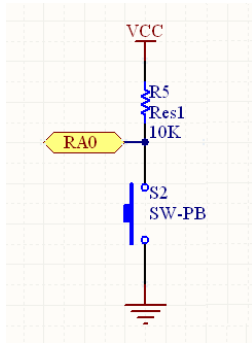


Figure 7

One I/O pin is needed for one push button as input of PIC microcontroller. The connection of the push button to the I/O pin is shown in Figure 7. The I/O pin should be pull up to 5V using a resistor (with value range 1K-10K) and this configuration will result an active-low input. When the button is being pressed, reading of I/O pin will be in logic 0, while when the button is not pressed, reading of that I/O pin will be logic 1.

LED as output for PIC microcontroller

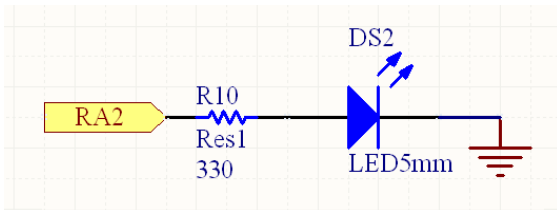


Figure 8

One I/O pin is needed for one LED as output of PIC microcontroller. The connection for a LED to I/O pin is shown in the schematic above. The function of R10 is to protect the LED from over current that will burn the LED. When the output is in logic 1, the LED will ON, while when the output is in logic 0, the LED will OFF.

ICSP for programming PIC microcontroller

ICSP stands for In Circuit Serial Programming and describes the serial programming interface for PIC microcontroller. ICSP gives you a convenient way of programming PIC Microcontroller without removing the chip from the development or production board. All you need is a programmer that provides the ICSP connector.

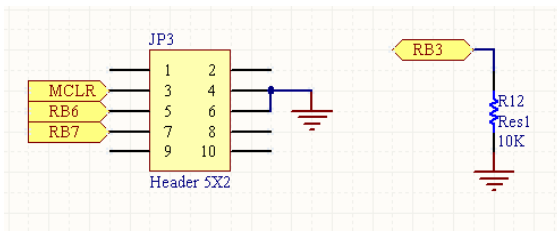


Figure 9

MCLR, RB6 and RB7 need to be connected to the ICSP socket to program the PIC microcontroller. At the same time, RB3 need to be pull down to 0V to disable low voltage programming, because the programmer is using high voltage programming.

PCB circuit board



Figure 10

Component:

1. Extra button (for further development).
2. Increase value button.
3. Mode select button.
4. Variable resistor (adjust LCD contrast).
5. Reset button.
6. Slide switch (Power ON/OFF).
7. Power connector (12V).
8. DC plug socket (To 12V ac to dc adaptor).
9. Box header (To ICSP programmer).

Please refer to Appendix A for the PCB layout of PR12. The PCB layout is provided free therefore Cytron Technologies will not be responsible for any further modification or improvement.

SOFTWARE

Flowchart:

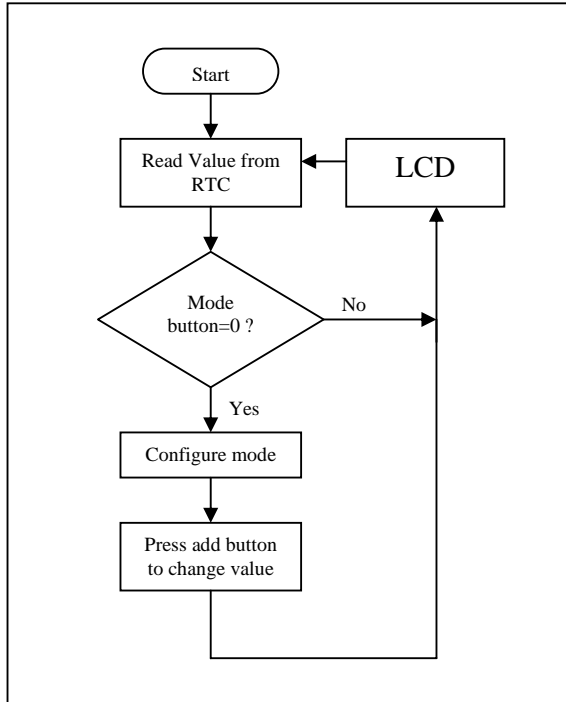


Figure 11

I²C data bus

The DS1307 supports the I²C protocol. The chip has two control lines, SCL and SDA (clock and data) which are connected to the PIC PortC pins RC3 and RC4, which also have the same functional names of SCL and SDA. RTC operates in I²C mode, a technique that allows 2 line data transfer between a host device and a slave. In this project, the host is the PIC and the slave is the RTC. Since I²C devices only respond to commands that are addressed specifically to them, several I²C devices can share the same two lines. Because the RTC SCL and SDA pins can be left floating when PIC's controlling pins are put into high impedance read mode, these lines are biased high by a resistor, typically of 10K ohm, although the value is not critical. Any time the RTC is to be read from or written to, its address command is sent serially to the chip via the SDA line, each data bit being clocked into the chip using the SCL line. Only if that address is valid to the RTC will it accept further commands, either telling it to receive new data being written to it, or to output its current clock and calendar data. The control wave forms are shown in Figure 12.

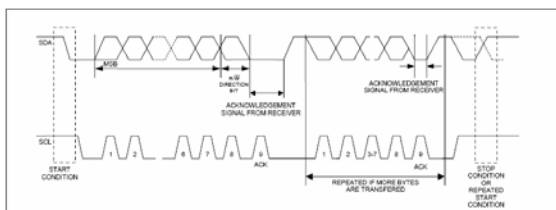


Figure 12

Example of simple RTC control code

```

//=====write to RTC=====
TRISC3=0;           //set SCL as output
TRISC4=0;           //set SDA as output
tclk=1;             //set tclk pin
sdata=1;           //set sdata pin
setstop();
setstart();
rtclkout(0b11010000); //set RTC for write mode
rtclkout(0b00000000); //set RTC for address 0
  
```

Listing 1

The initial stage of the routine for writing data to the RTC is shown in Listing 1. In it the equated names for the SDA and SCL line are SDATA and TCLK. At first, configure both SDA and SCL pin into output by writing to TRISC3 and TRISC4. Both the port pin are set to high. Call 'setstop' (Listing 2) first to ensure RTC is in stop mode. Calling 'setstart' (Listing 2) will inform RTC to expect an ID address. The only ID address it will respond to at this time is '0b11010000'. In the ID code, bit 0 determines whether the code sets the RTC into write mode (0) or read mode (1). The ID code is sent via routine 'rtclkout' in Listing 3.

```

void setstop(void)
{
  sdata=0;           //take data low while clk is low
  tclk=1;            //take clk high while data is low
  sdata=1;           //take data high while clk is high
}

void setstart(void) //TCLK & SDATA assumed to be high
{
  sdata=0;           //take data low while clk is high
  tclk=0;            //take clk low while data is low
}
  
```

Listing 2

```

void rtclkout(unsigned char data)
{
  unsigned char loop;
  unsigned char result;

  for(loop=8;loop>0;loop--1)
  {
    result = ((data>>(loop-1)) & 0x01); //MSB first
    sdata=0; //clear SDATA

    if(result!=0) //result not equal to zero?
    {
      sdata=1; //set SDATA
      tclk=1; //clk up
      tclk=0; //clk down
    }
    else
    {
      tclk=1; //clk up
      tclk=0; //clk down
    }
  }

  waitackrtc();
}
  
```

Listing 3

The command in Listing 3 shows that the bit are sent serially in order of most significant bit (MSB) first, least significant bit (LSB) last. After all eight bits have been sent, 'waitackrtc' function has been called in which acknowledgment from the RTC is waited to confirm data receipt.

```

void waitackrtc(void)
{
  TRISC4=1; //set SDATA as input
  tclk=1; //clk up
  while(sdata)continue; //continuous loop if sdata is set
  tclk=0; //clk down
  TRISC4=0; //set SDATA as output
}
  
```

Listing 4

It is assumed in Listing 4 that acknowledgement will be received and the routine waits indefinitely until it is done. Next the address at which the first data byte to be stored is sent, zero in this case (0b00000000). Now the actual time and calendar data can be sent to the RTC. Listing 5 shows part of routines. Here the register values have all been previously preset.

```
rtcclkout(0b00110011); //send init second 33
rtcclkout(0b00110011); //send init minute 33
rtcclkout(0b00100001); //send init hour 21
```

Listing 5

Note that after each write operation, the RTC address counter is automatically increased, ready for next data byte to be sent. The formatting for the data is shown in Table 1. Note in particular the CH bit (bit7) in address line 00H. This bit controls the RTC's oscillator. When the bit is set to 1, the oscillator is disabled. The oscillator is only enabled when the bit is set to 0. When the RTC is powered up for the first time, the bit7 of address 00H must be cleared to 0 in order to operate. Any program sending the second data to RTC must also clear this bit. Having set the current time and calendar data into the RTC, time can be read as real time data whenever user want by calling 'readrtc' function (Listing 6).

```
// Read RTC function
//=====
void readrtc(void)
{
    TRISC3=0;
    setstop();
    setstart();
    rtcclkout(0b11010000); //set to write mode
    rtcclkout(0b00000000);

    setstop();
    setstart();
    rtcclkout(0b11010001); //set to read mode

    TRISC4=1; //set SDA as input

    rtcclkread(); //read 1 byte sec
    sendackread(); //send ACK
    store=store<0b01111111;
    clksec=store;

    rtcclkread(); //read 1 byte min
    sendackread(); //send ACK
    store=store<0b01111111;
    clkmin=store;
}
```

Listing 6

It should be noted that when setting for read mode, the RTC must first be set into write mode, by the ID byte bit 0 being cleared to 0. The start address is sent and the RTC is put into stop mode. Now the start command is sent, followed by the ID code again but this time with bit 0 set to 1, to put it into read mode. It was here that the RTC datasheet was found to be most unclear. Now, after the read mode is set, four commands for reading the 'CLKSEC' data are shown. The same commands are repeated for all values required but with the 'AND' value being changed to suit the specific value being read and with difference destination register. (Refer Table 1)

```
void sendackread(void)
{
    TRISC4=0; //set SDATA as output
    sdata=0; //clear SDATA
    tclk=1; //clk up
    tclk=0; //clk down
    TRISC4=1; //set SDATA as input
}

void sendnotackread(void)
{
    TRISC4=0; //set SDATA as output
    sdata=1; //set SDATA
    tclk=1; //clk up
    tclk=0; //clk down
    TRISC4=1; //set SDATA as input
}
```

Listing 7

The reading is not the last data value from RTC, 'sendackread' function is being called to send acknowledgement to RTC. If the data is last, the 'sendnotackread' is called to stop a read. The difference between 'sendackread' and 'sendnotackread' routine is only the bit of 'sdata'.

```
void rtcclkread(void)
{
    unsigned char loop=0;
    unsigned char orf=0b10000000;
    store=0;

    for(loop=8;loop>0;loop-=1)
    {
        tclk=1; //clk up
        if(sdata) //is SDATA set
        {
            store=store|orf; //or function
            tclk=0; //clk down
        }
        else
        {
            store=store|0b00000000; //or function
            tclk=0; //clk down
        }

        orf=orf>>1; //shift right orf by 1 bit
    }
}
```

Listing 8

At each read step 'rtcclkread' function is called in which the eight data bit of the required value are read back serially, again in order of MSB to LSB, and built up into a byte within store register. Above are the basic commands to control a RTC chip, for further information to use these commands and display the read value on LCD, please refer to the sample program of this project.

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE	
00H	CH	10 Seconds			Seconds		Seconds		Seconds	00-59	
01H	0	10 Minutes			Minutes		Minutes		Minutes	00-59	
02H	0	12	10	10	Hours		Hours		Hours	1-12 +AM/PM	
		24	Hour								Hour
03H	0	0	0	0	DAY		Day		Day	01-07	
04H	0	0	10 Date		Date		Date		Date	01-31	
05H	0	0	0	10	Month		Month		Month	01-12	
06H	10 Year			Year		Year		Year		Year	00-99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—	
08H-3FH									RAM	00H-FFH	

Table1

The source code is provided free and Cytron Technologies will not be responsible for any further modification or improvement.

GETTING START

User can obtain the hardware set for this project (PR12) either by online purchasing (www.cytron.com.my) or purchase it in Cytron Technologies Shop.

1. Once user has the hardware set, soldering process can be started. Please solder the electronic components one by one according the symbols or overlays on the Printed Circuit Board (PCB). Ensure the component value and polarity is correctly soldered. Please refer to PCB Layout in Appendix A.

Caution: Make sure all the connectors (2510) are soldered in proper side. Those electronic components have polarity such as capacitor, diode, PIC, LM7805 and LED should be soldered in right polarity or it may cause the circuit board fail to work.

Warning: Before the battery (Power) is plugged in, make sure the polarity is correct to prevent the explosion. Wrong polarity of capacitor also may cause explosion.

2. Please download the necessary files and document from Cytron Technologies website. These included documentation, sample source code, schematic, component list and software.
3. The next step is to install MPLAB IDE and PICC Lite into a computer. The MPLAB IDE and PICC Lite can be downloaded from www.cytron.com.my.
4. After the installation complete, open the project file provided using MPLAB IDE. Please refer to PR1 and PR5 for the method to use MPLAB and PICC Lite.
5. Plug in power supply for the circuit. User can choose to use battery or AD to DC adaptor.
6. Build the project and load the hex file into the PIC microcontroller using the USB In Circuit Programmer (UIC00A). The programmer is not included in the hardware set but it can be found at Cytron website. (User's manual is provided at website).
7. PIC is now completely programmed.
8. Please takes note that when the initial power on, all the information registers of DS1307 are not defined and the oscillator is not running. Therefore, it is important to enable the oscillator (CH bit = 0). User has to press the select mode button and the program will go to write mode.
9. At the write mode, all the initial value of time and calendar can be set and the oscillator of RTC will be enabled.
10. Press the select mode button again until the program return to the read mode. Now, the time will be running.

REMEMBER! When the RTC is totally out of power (3V backup battery and the main source is unplug), all

the registers will be reset, so configuration as previous need to be redo again to run the RTC. If only one of the 2 power source is unplug, RTC will still functioning using the power from the remaining source.

AC to DC adaptor:

User can decide either uses a 12V battery or an AC to DC adaptor as the power source to the circuit. The picture and the way to use the adaptor are shown in Figure 11 and 12.



Figure 11 (not included in DIY project set)

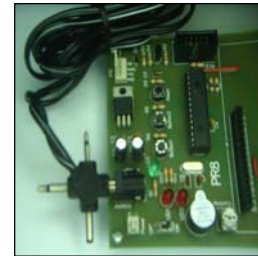


Figure 12

9V battery connector:

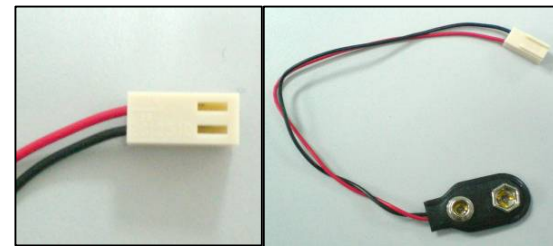


Figure 13

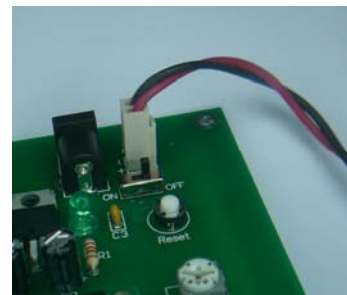


Figure 14

Figure 13 shows the standard of connecting the 2510 socket header to the battery. The red cable is the

positive terminal meanwhile the black cable is negative terminal. Also, be careful on the position of the 2510 socket on the board (refer the PCD layout at Appendix A).

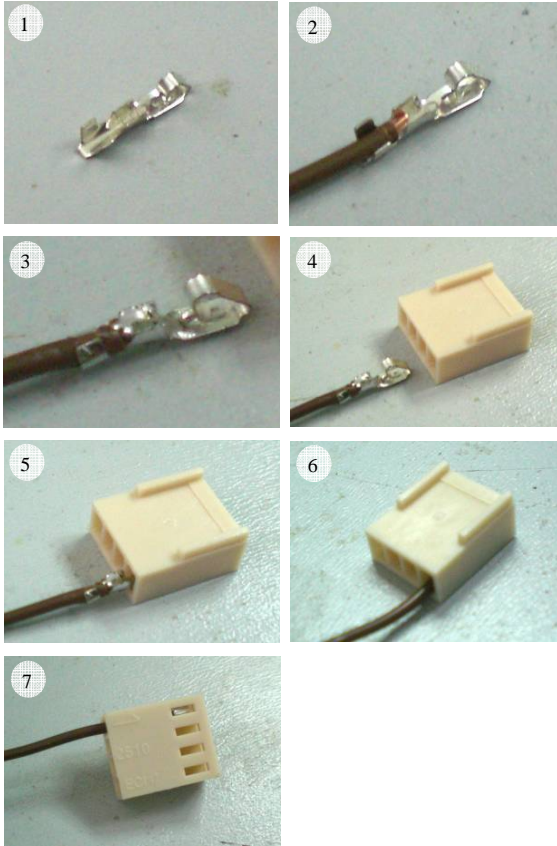
How to connect the wire to 2510 connector:

Figure 15

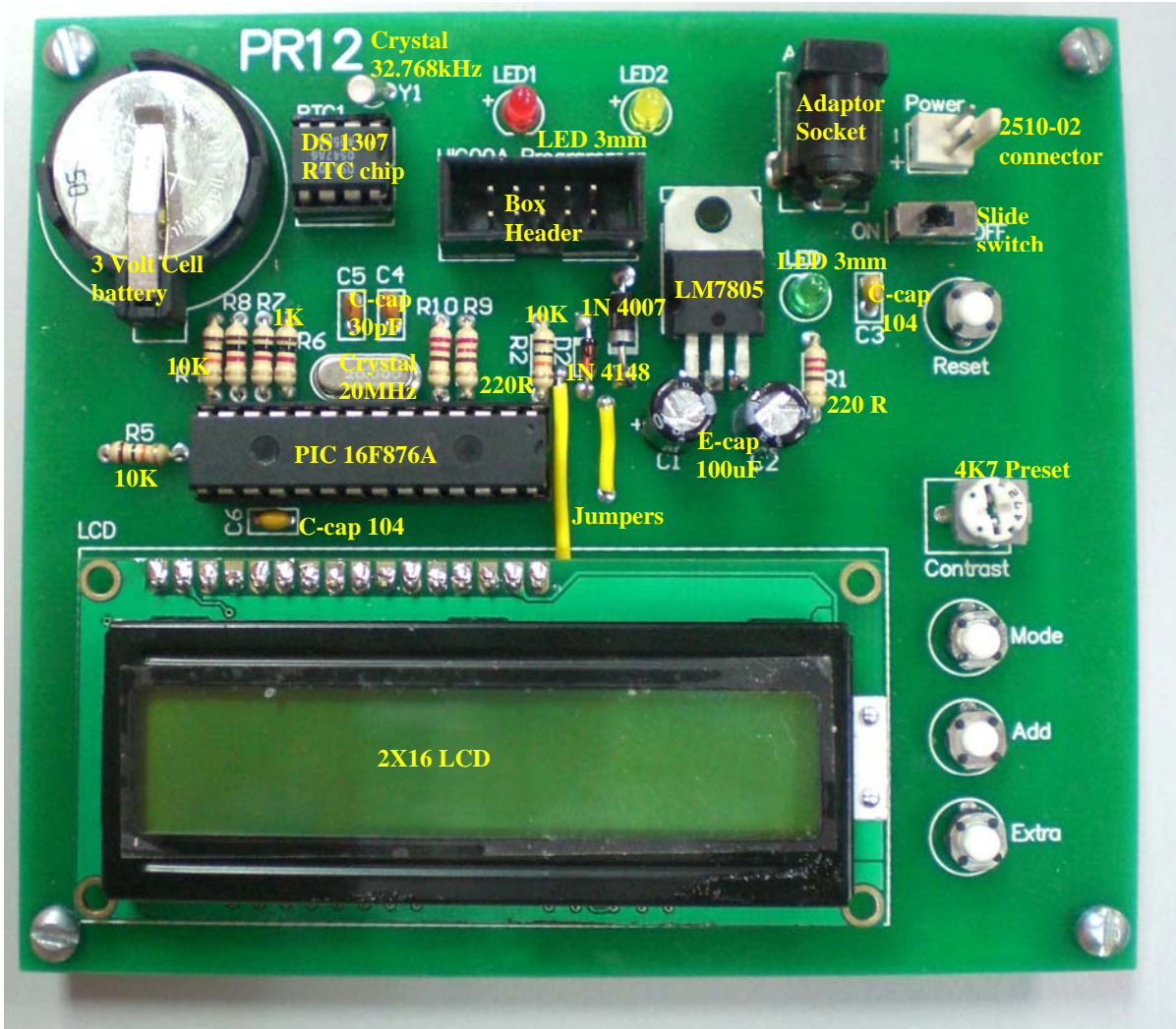
Figure 15 shows the method of connecting the cable to 2510 header.

WARRANTY

No warranty will be provided as this is DIY project. Please check the polarity of each electronic component before soldering it to board.

Appendix A

PCB Layout:



Prepared by
Cytron Technologies Sdn. Bhd.
19, Jalan Kebudayaan 1A,
Taman Universiti,
81300 Skudai,
Johor, Malaysia.

Tel: +607-521 3178
Fax: +607-521 1861

URL: www.cytron.com.my
Email: support@cytron.com.my
sales@cytron.com.my