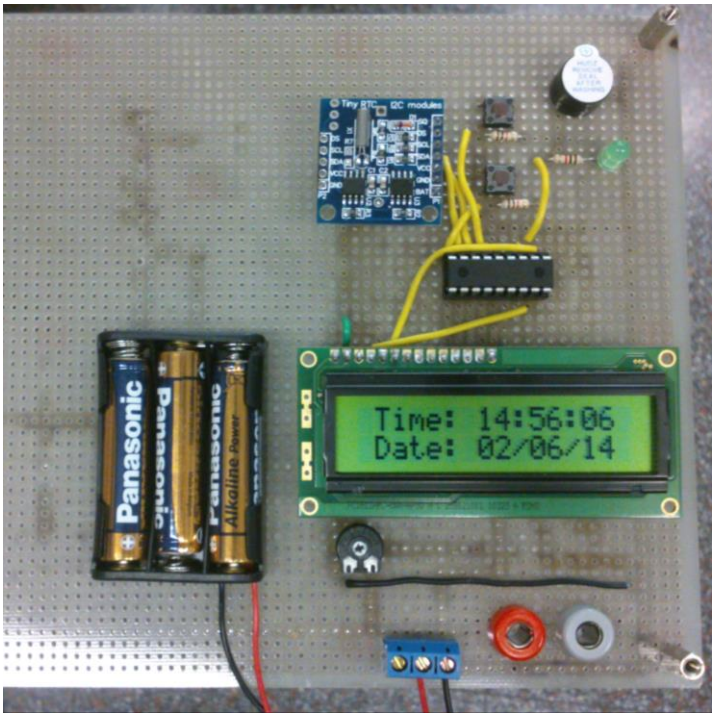
Teamwork proposal



Design of a digital desktop clock

Melvin - Carles Tong González

Jordi Grabulosa Solés

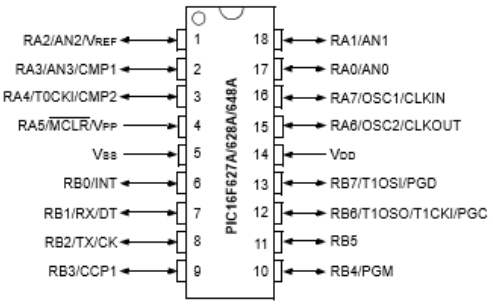
Technical design

Our digital clock will be able to display the current time, date and an operative alarm. For the minimal flash memory that has our PIC, we didn't display also the day of the week.

So, for do all this it was necessary to use a LCD display, a PIC microcontroller, a Real Time Clock module, few buttons and batteries. For the alarm we use a buzzer and a LED.

PIC 16F628A

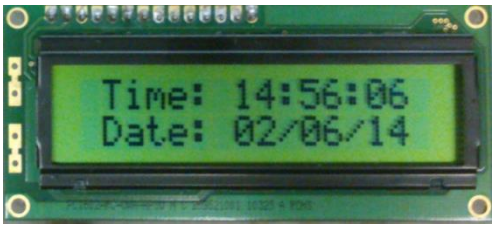
We choose this PIC for several reasons:



- The operating range of Vcc is 3.3 to 5.5 V. So is convenient to use it with batteries.
- They have the just memory flash to do a digital clock with it.
- They have a serial pins, also permit the communication i2c with the RTC module.
- They have 2 ports of pins, PORTA and PORTB, the minimum pins to do a digital clock. More pins will cause more power consumption although the pins are not used.
- Incorporates an internal oscillator, so an external quartz crystal oscillator is not needed.
- They are a lot of tutorials and explanations about this PIC to learn. It's the more used if you are a beginner in the programming of PIC microcontrollers.
- I have been familiarized with this PIC before, in a work for high school programed in assembler.

Ref: http://ww1.microchip.com/downloads/en/DeviceDoc/40044E.pdf

LCD Display



We use an LCD of 16x2 lines. One data line for the time and the other one for the date.

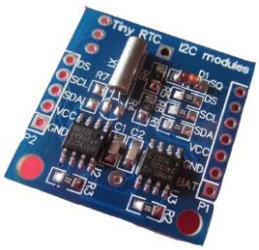
Pin No	Name	Function	Description
1	Vss	Power	GND
2	Vdd	Power	+ 5 V
3	Vee	Contrast Adj.	(-2) 0 - 5 V
4	RS	Command	Register Select
5	R/W	Command	Read / Write
6	E	Command	Enable (Strobe)
7	D0	I/O	Data LSB
8	D1	I/O	Data
9	D2	I/O	Data
10	D3	I/O	Data
11	D4	I/O	Data
12	D5	I/O	Data
13	D6	I/O	Data
14	D7	I/O	Data MSB

Also we choose one LCD without background backlight, because with it involves more power consumption.

Buttons

We use 2 buttons. The minimal quantities of buttons for run our digital clock. One is used to select the mode or the number. And the other one is to change the mode or the number value.

Real Time Clock module



We use a RTC module because is all compact and our design consist in a digital clock not in a Real Time Clock. This module uses a microchip ds1307 with an oscillator and other components that communicate the real time to our digital clock using a communication i2c. We choose this model because is the more used and the cheaper. Also the operating range is between 4.5 – 5.5V, and provides time and date.

Ref: http://datasheets.maximintegrated.com/en/ds/DS1307.pdf

Buzzer and LED

We use a buzzer of 5 V that are commonly used in arduino. Operating range 3-7 V.

Batteries

Technical Detail:

Individual Battery Details (AAA/LR03 Alkaline Battery)

Battery Technology:	Alkaline (Single Use)
Capacity Range:	1000.0mAh - 1200.0mAh
Current:	5.0000A
Diameter:	10.5mm
Height:	44.5mm
Voltage:	1.50V
Weight:	11.5g

We use 3 Individual Batteries to power our PIC with 4.5 V.

With the digital clock operative, we check the power consumption of our circuit.

- At normal operation: 1.82 mA
- When alarm is beeping: 14 mA

So our digital clock has life of:

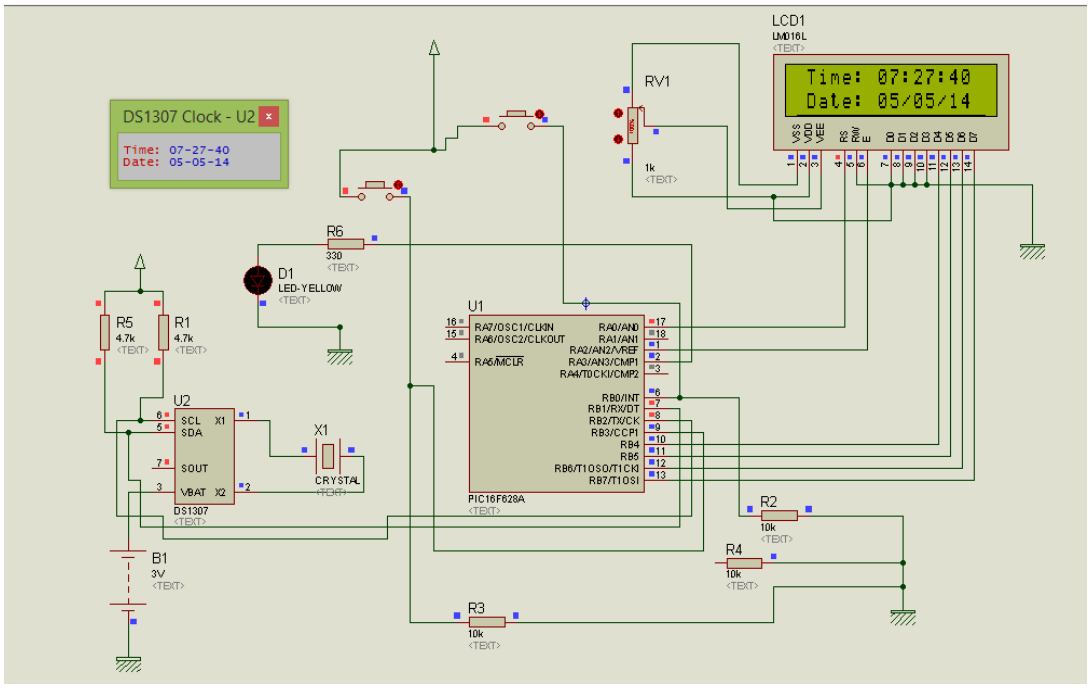
L = 1200*3*1.82 = 6552 h = 9.1 mth

So the clock has an autonomy of approximately 9 months if we consider to use the alarm. In the worst case (1000mAh) we have 7.5 months.

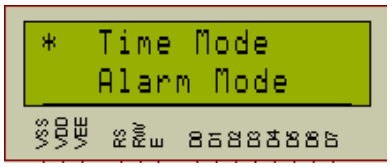
Although if we want more time we can also put 3 AA batteries.

Simulation and working

We use ISIS proteus for check if the C code was correct until to burn it over the PIC in our real circuit.

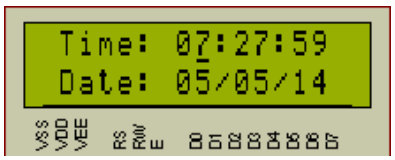


Menu



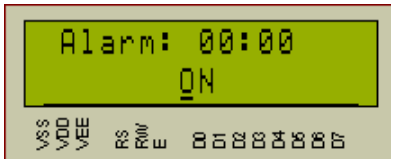
For minimize the number of buttons and the content of code we use an internal menu to select time or alarm configuration.

Time Mode



We use a blink cursor to know what number is selected for change it.

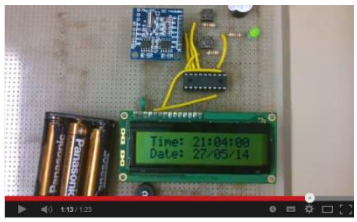
Alarm Mode



We use a blink cursor too and a value that indicates if the alarm is activated or not. If the time is the same as the alarm time, the buzzer and the led will blink.

Then we proceeded to do the physical circuit.

https://www.youtube.com/watch?v=sIwQA44Ctxc



C code

```
#include <main.h>

#include <flex_lcd.c>
#include <ds1307.c>

//Buttons
#bit RB0=0x06.0
#bit RB3=0x06.3

//Time and date
int h = 0;
int min = 0;
int seg = 0;
int day = 0;
int mth = 0;
int year = 0;
int dow = 0;

//Alarm time
int ah = 0;
int amin = 0;

//Alarm ON-OFF
int alarm = 0;
//Counter of alarm beeping
int al = 0;
//Mode menu
int mode = 0;

//For cursor blink
int pass = 0;
//What value of time or date is selected to change.
int B0 = 0;
```

```
//Button RB0 interrupt
#INT_EXT
void interrupcio(void){
    //Wait to the rebounds of button and rising edge.
    while(RB0){
      delay_ms(10);
    }
    //Stop alarm
    if(al < 10){
        al = 10;
    //Next number of time or data
    }else{
        B0++;
        pass = 1;
    }
}

void main(){

   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   enable_interrupts(GLOBAL);
   ENABLE_INTERRUPTS(INT_EXT);

   lcd_init();
   ds1307_init();
   ds1307_set_date_time(5,05,14,3,07,27,05);

    //Program running.
   while(true){
      //If button RB0 is pressed we enter in the menu
      if(B0 != 0){
         lcd_putc("\f*  Time Mode\n   Alarm Mode");
         //Menu to select time or alarm (var. mode)
         while(B0 == 1){
            if(RB3 == 1){
               while(RB3){
                  delay_ms(10);
               }
               mode = !mode;
               if(mode == 1){
                  lcd_putc("\f   Time Mode\n*  Alarm Mode");
               }else{
                  lcd_putc("\f*  Time Mode\n   Alarm Mode");
               }
            }
         }
         //time mode
         if(mode == 0){
            while(B0 > 1){
               //If RB0 is pressed, shift cursor.
               if(pass == 1){
                  pass = 0;
                  printf(lcd_putc,"\f Time: %02D:%02D:%02D\n Date: %02D/%02D/%02D ",h,min,seg,day,mth,year);
                  if(B0 == 2){
                     lcd_gotoxy(15,1);
                     lcd_setcursor_vb(true, true);
                  }else if(B0 == 3){
                     lcd_gotoxy(12,1);
                     lcd_setcursor_vb(true, true);
                  }else if(B0 == 4){
                     lcd_gotoxy(9,1);
                     lcd_setcursor_vb(true, true);
```

```
            }else if(B0 == 5){
                lcd_gotoxy(9,0);
                lcd_setcursor_vb(true, true);
            }else if(B0 == 6){
                lcd_gotoxy(12,0);
                lcd_setcursor_vb(true, true);
            }else if(B0 == 7){
                lcd_gotoxy(15,0);
                lcd_setcursor_vb(true, true);
            }
        }
        //Cursor in seconds position
        if(B0 == 2){
            if(RB3 == 1){
                while(RB3){
                    delay_ms(10);
                }
                seg = 0;
                pass = 1;
            }
        //Cursor in minutes position
        }else if(B0 == 3){
            if(RB3 == 1){
                while(RB3){
                    delay_ms(10);
                }
                min = (min+1)%60;
                pass = 1;
            }
        //Cursor in hours position
        }else if(B0 == 4){
            if(RB3 == 1){
                while(RB3){
                    delay_ms(10);
                }
                h = (h+1)%24;
                pass = 1;
            }
        //Cursor in days position
        }else if(B0 == 5){
            if(RB3 == 1){
                while(RB3){
                    delay_ms(10);
                }
                day = (day+1)%31;
                pass = 1;
            }
        //Cursor in months position
        }else if(B0 == 6){
            if(RB3 == 1){
                while(RB3){
                    delay_ms(10);
                }
                mth = (mth+1)%12;
                pass = 1;
            }
        //Cursor in years position
        }else if(B0 == 7){
            if(RB3 == 1){
                while(RB3){
                    delay_ms(10);
                }
                year = (year+1)%20;
                pass = 1;
```

```
            }
         //End of time mode. Set new time to RTC.
         }else{
            B0 = 0;
            ds1307_set_date_time(day,mth,year,dow,h,min,seg);
            lcd_setcursor_vb(false, false);
         }
      }
   //Alarm mode
   }else{
      while(B0 > 1){
         //If RB0 is pressed, shift cursor.
         if(pass == 1){
            pass = 0;
            if(alarm){
               printf(lcd_putc,"\f Alarm: %02D:%02D\n       ON",ah,amin);
            }else{
               printf(lcd_putc,"\f Alarm: %02D:%02D\n       OFF",ah,amin);
            }
            if(B0 == 2){
               lcd_gotoxy(13,1);
               lcd_setcursor_vb(true, true);
            }else if(B0 == 3){
               lcd_gotoxy(10,1);
               lcd_setcursor_vb(true, true);
            }else if(B0 == 4){
               lcd_gotoxy(8,0);
               lcd_setcursor_vb(true, false);
            }
         }
         //Cursor in minutes position
         if(B0 == 2){
            if(RB3 == 1){
               while(RB3){
                  delay_ms(10);
               }
               amin = (amin+1)%60;
               pass = 1;
            }
         //Cursor in hours position
         }else if(B0 == 3){
            if(RB3 == 1){
               while(RB3){
                  delay_ms(10);
               }
               ah = (ah+1)%24;
               pass = 1;
            }
         //Cursor in ON-OFF position
         }else if(B0 == 4){
            if(RB3 == 1){
               while(RB3){
                  delay_ms(10);
               }
               alarm = !alarm;
               pass = 1;
            }
         //End of alarm mode.
         }else{
            B0 = 0;
            lcd_setcursor_vb(false, false);
         }
      }
```

```
            }
            mode = 0;

        //If button RB0 was not pressed we enter in the normal mode of clock.
        }else{

            //Refresh time and data
            ds1307_get_time(h,min,seg);
            ds1307_get_date(day,mth,year,dow);
            printf(lcd_putc,"\f Time: %02D:%02D:%02D\n Date: %02D/%02D/%02D ",h,min,seg,day,mth,year);
            delay_ms(1000);

            //Alarm
            if(h == ah && min == amin && seg == 0 && alarm){
               al = 0;
               while(al < 10){
                  OUTPUT_HIGH(PIN_A3);
                  delay_ms(500);
                  OUTPUT_LOW(PIN_A3);
                  delay_ms(500);
                  al++;
               }
            }

        }
    }
}
```

Price of the system

Quantity	Component	Price (€)
1	PIC 16F628A	3,10 €
1	Socle 18 pins	0,11 €
4	Support legs	0,96 €
1	Perfboard	11,12 €
1	Screw Terminal Block Connector	0,46 €
1	LED green 5mm	0,08 €
3	Ressistor	0,04 €
1	LCD 2x16	11,00 €
1	headers male and female	0,48 €
2	Button 6x6 5 mm	0,18 €
1	Potentiometer 22 KΩ	0,14 €
1	RTC Module	3,90 €
1	Battery holder	2,50 €
3	Batteries	2,00 €
2	Sokets for bannana	0,47 €
	Total	**36,54 €**

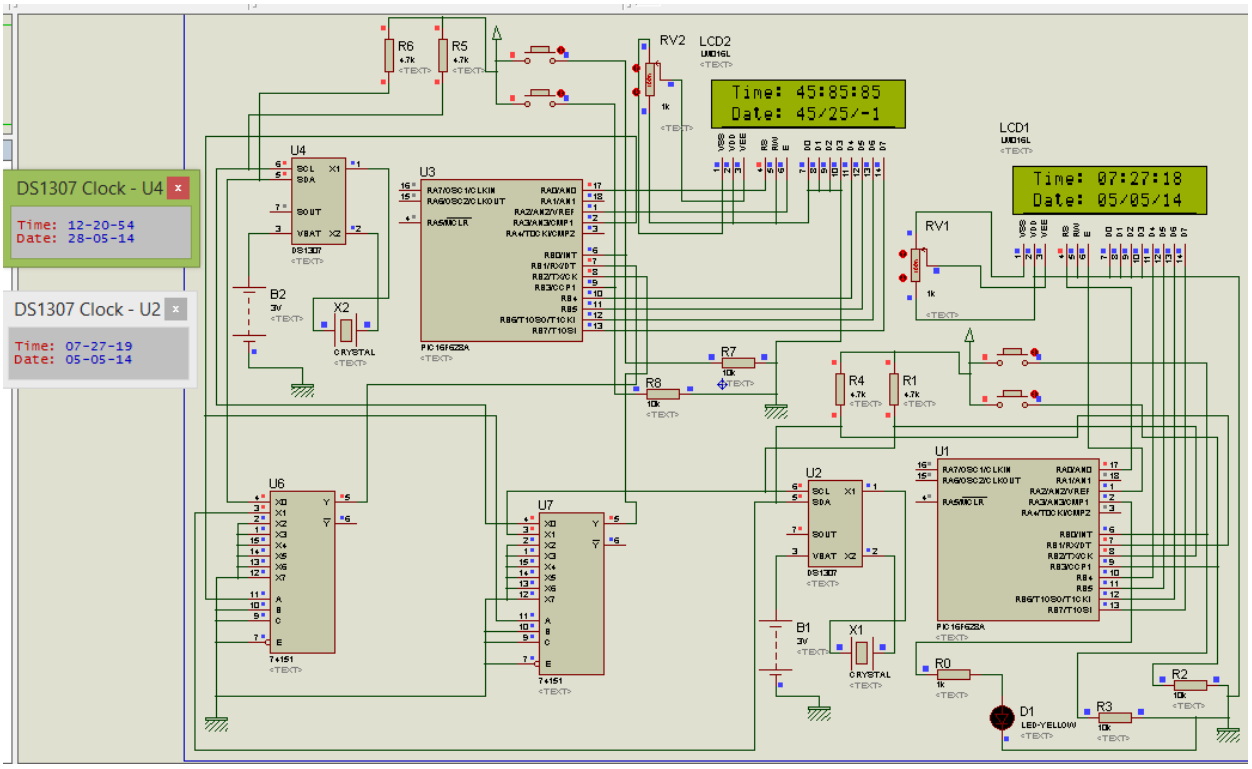
Synchronization Capability

We design two models.

The first one has zero delays between the master and the slave because the clock is get directly from the RTC module of the master clock. So is not need to pass through the master clock. So, in our design we change the “alarm mode” in the slave clock to “upgrade mode”.

This mode allows upgrading the time and data from the clock master every time as you wish. Also the clock is automatically synchronized at 12 o’clock pm.

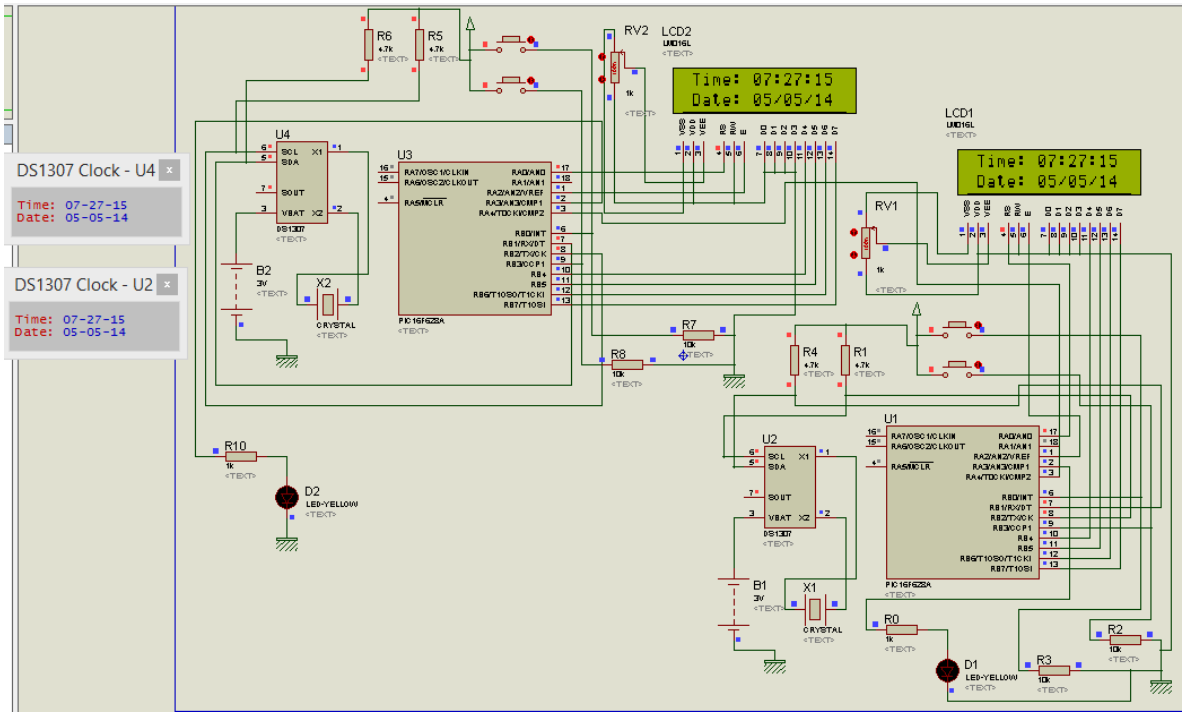
So, we use a multiplexor to take the time from the slave RTC or master RTC. And the alarm pin we change to the select for the multiplexor.



But with this solution we have problems of the communication i2c. Because the multiplexor introduces delays, so the communication of the time and date with i2c fails.

So we choose the second design. One where the master clock communicates directly to the slave clock, avoiding the communications problems with i2c. But with this solution, only the master clock can send data. So instead of upgrade the time every time the slave clock want, the slave clock only will be synchronized at 12:00 pm by the master.

So when the master clock reaches 12:00 pm, they will send a signal to the slave clock by the RA4 pin, telling to change the time at 12:00 pm. This design includes some delays that the master clock and the PIC could do.



For calculate the deviation between clocks after 24 hours, we calculate the deviation of our clock in one day in reference with one atomic clock from internet.

The deviation in one day was approximately of 2.5 seconds.

So the deviation between clocks after 24 hours could be of ±2.5 seconds.