Digital Clock

A Project By

Perry Andrews

Based on the PIC16F84A Micro controller.

Revision C

23rd January 2011

Contents

Contents 2
Introduction 2
Design and Development 3
Construction 7
Conclusion 9
References 9

Figure 1 - Main Flow Diagram 4
Figure 2 – Interrupt Flow Diagram 5
Figure 3 - Configuration Bit (Fuse) Settings 6
Figure 4 - Circuit Diagram 6
Figure 5 - The Completed Project 9

Table 1 - Parts List 7

Introduction

I wanted to make a desk clock for my desk at work that would show both the time and date. I have a preference for LED displays as they are bright and give a large display.

The requirements are:

- Use LED 7 segment displays.
- Display both the time and date at the same time.
- Buttons to set the time and date.
- Take care of leap years.
- Fit in a small red translucent box.

Design and Development

This project creates a clock for a desk which displays the time and date on ten seven segment displays. The project fits in a small red see-through box. The circuit uses eight outputs for the segments and decimal points, and a 4 line to 10 line decoder (4028B) to select the digit via a transistor.

To allow the setting of the time two buttons are provided. The first selects the digit and the second increments the digits. The order is hours, minutes, day, month, year. When the setting mode is active only the digits being set are displayed, all the others are shown as '--'.

Before creating the program I listed the steps the program requires:

1. Initialise the PIC and set all digits to zero.
2. Start the timer interrupts to count the seconds elapsed.
3. When seconds reach 60 increment minutes and reset seconds.
4. When minutes reach 60 increment hours and reset minutes.
5. When hours reach 24 increment days and reset hours.
6. When days reach 28 or more increment months and reset days to 1. This is dependant on the current month.
7. When months reach 13 increment years and reset months to 1.
8. When the first button is pressed stop updating the clock, reset all digits to – apart from the hours. If the button is pressed again set hours to – and display the minutes. Each press moves through the digits. The next press after year displays all digits and restarts the clock.
9. After the first button has been pressed the second button increments the number on the digits. When the digit reaches its intended maximum it returns back to its lowest value. For example, hours rang from 00 to 23. When the digit increments to 24 it returns to 00.

It is possible to set an invalid date e.g. 30th February but this will be corrected when the clock starts the next day. A future revision could fix this issue as it is a matter of setting the years, then month and then day. Knowing the year and month allows you to check the range of days available.

Revision A differs from the first version only in the program and this is reflected in the flow charts below. Revision B is a modification to the circuit diagram only, adding the switches and a decoupling capacitor. In Revision C I corrected errors on the drawing and I changed the program to allow it to be compiled by the mikroC compiler.

Based on these steps I created a flow diagram to help create the program.

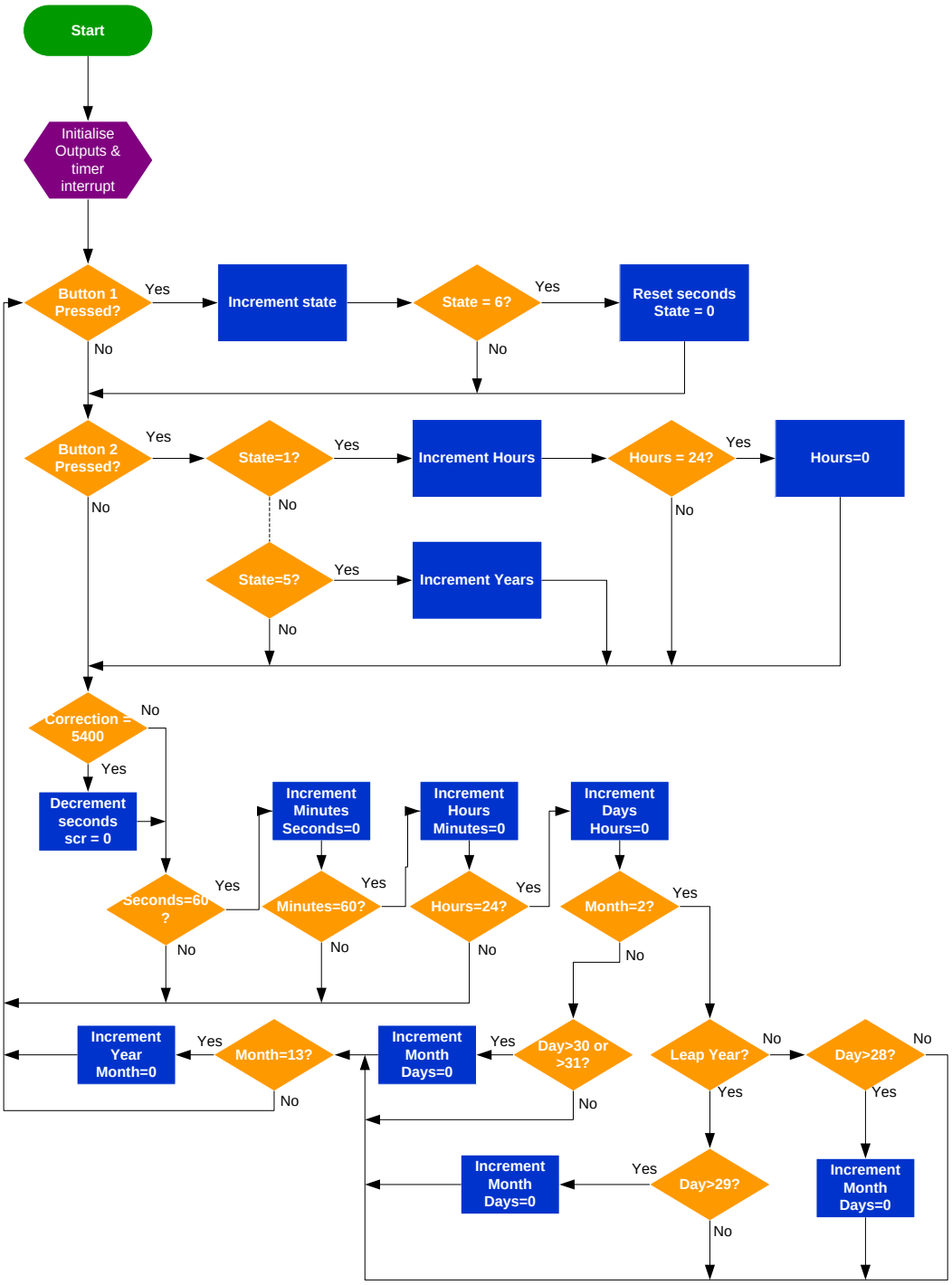


Figure 1 - Main Flow Diagram

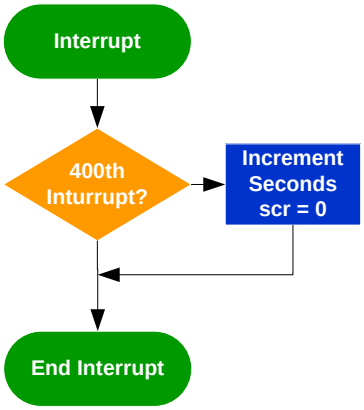


Figure 2 – Interrupt Flow Diagram

The Main Flow Diagram constitutes the main sub program. After setting up the ports and starting the interrupt the main sub program polls the buttons. With Revision A the counters are incremented and the date calculated. The difficult part to this program is how to deal with months having a different number of days, including February which can have 28 or 29 depending on if it is a leap year or not. The interrupt flow diagram now shows only the seconds being updated.

Not shown in the flow charts, code is also required to display the digits and to read the inputs.

I found in use that the clock gained time as the day went on. After a few days the time difference was noticeable. I calculated that every day 16 seconds were being gained. I decided to setup a new counter that was incremented each second and it would be used to lose a second at the correct moment. I needed to loose 1 second every 24 / 16 = 1.5 hours. This was 5400 seconds (1.5x60x60). When the counter reached 5400 seconds the seconds counter was decreased by 1 and the counter was reset. The clock now keeps very good time.

In this project I used MikroC to compile the program. There were two reasons for this. The first was to have a new compiler that supports many different PIC microcontrollers (the C2C version I had only supported 16F84). The second reason was to include the fuse configuration bits in the hex file.

When you use a PIC device for the first time (or in a new project) you need to change the configuration bits to make sure it is suitable for the way you are using it. There are only a few on the 16F84/A so this is very simple. In the MikroC IDE you set the configuration bits in the edit project window. This is shown in figure 3 on the next page.

If your programmer does not see the configuration bit settings in the hex file you need to manually set them using the window on the following page as a guide.

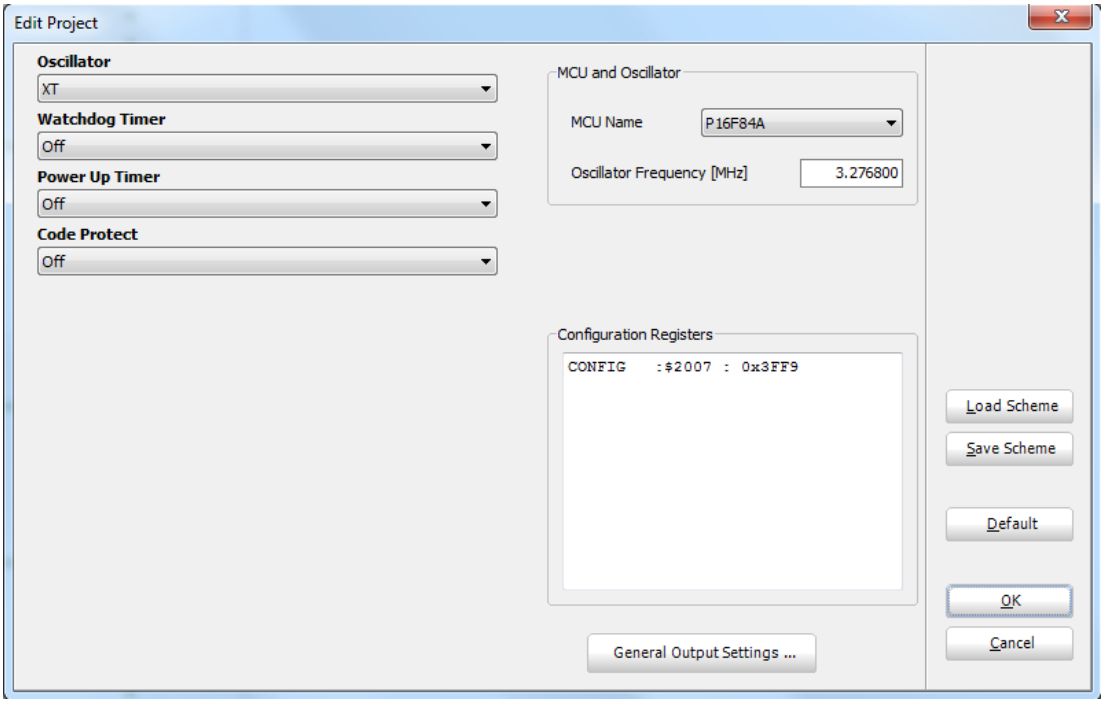


Figure 3 - Configuration Bit (Fuse) Settings

The circuit diagram is more complex because 10 displays are being used. The diagram is shown in figure 4 below:

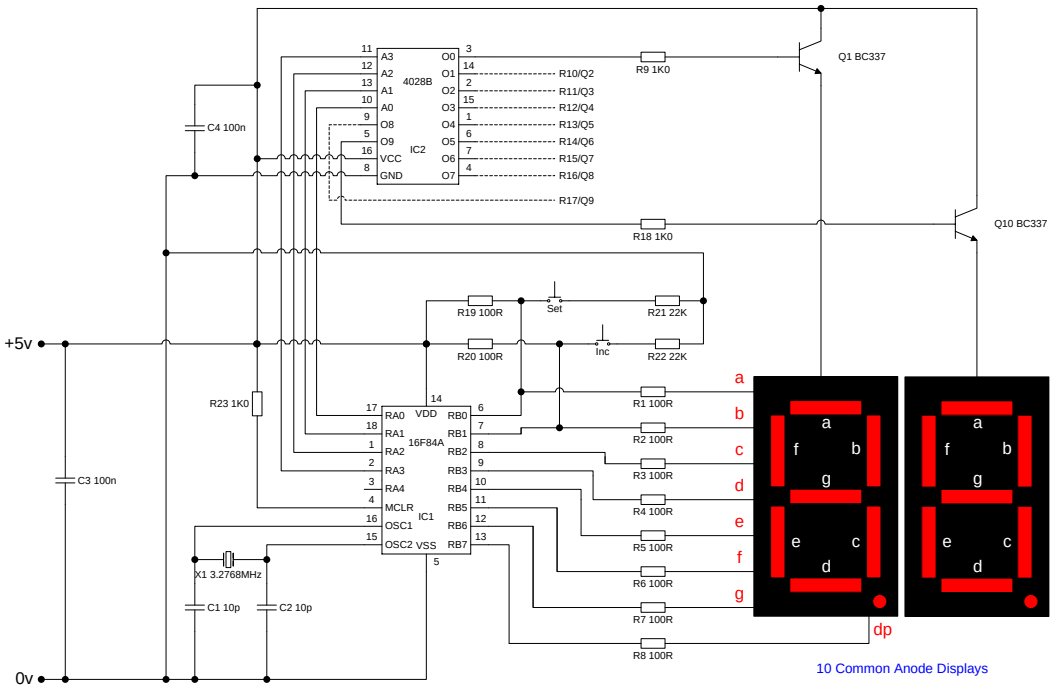


Figure 4 - Circuit Diagram

The circuit diagram is also supplied as a separate PDF file.

Eight outputs are used to turn the 7 segment and decimal points on and off. The displays are common anode so a port set low turns the segment on and a port set high turns it off.

Each of the digits is turned on sequentially. The segments for that digit are only on while that digit is on. This technique is used because 80 outputs would be required to power each individual segment otherwise. We only have five outputs left for the digits but 10 are required. This is where a 4 – 10 line decoder is used to convert 4 outputs into 10. Each of the digits is powered by a transistor connected to each of these outputs.

Construction

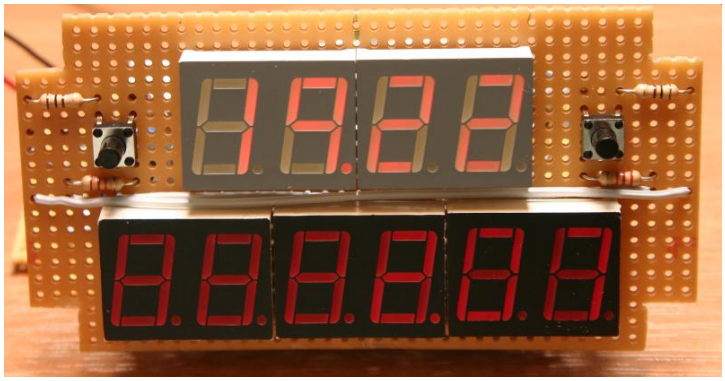
This project was constructed using Vero strip board, as it is quick to work with. A parts list is shown below:

Identifier	Description	Quantity	Maplin Stock Code
	Vero Strip Board	2 pc 100mm x 60mm	JP49D
IC1	PIC16F84A	1	VS87U
	Dual Red LED Displays	5	WL27E
IC2	4028 4-10 Decoder	1	QX17T
Q1-Q10	BC337	10	QB68Y
X1	3.2768MHz Crystal	1	FY86T
C1-C2	10pf Capacitors	2	RA33L
C3-C4	100nf Capacitor	2	RA49D
R9-R18, R23	1K0 Resistor	11	M1K
R1-R8, R19-R20	100R Resistor	10	M100R
R21-R22	22K Resistor	2	M22K
	PCB Switch	2	KR88V
	18Pin DIL Socket	1	HQ76H
	16Pin DIL Socket	1	BL19V
	Infra-Red Box 112x62x27	1	N70AL

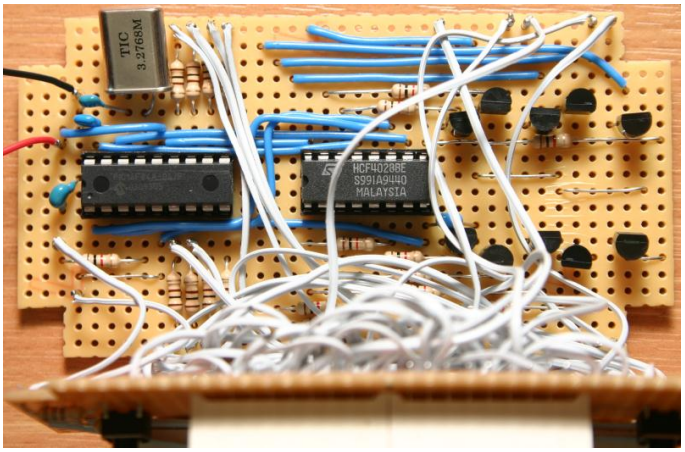
Table 1 - Parts List

I used a crystal as the timing source because of its accuracy. However I did notice that a certain amount of seconds were gained during the day.

This project is constructed on two circuit boards. The top one contains the displays and the two switches. There is not enough room on the boards to link the displays on the top so the displays are linked together on the copper side using stranded insulated wire. This is time consuming and tedious but necessary to fit the project into the box size I wanted.



Once the top board was complete I started on the lower board. This contains all other components including the PIC Micro, 4028 and the transistors. The two sockets were soldered in first followed by the wire links. I then soldered in the transistors before locating the resistors. Usually the resistors are soldered first as they are less prone to damage by excess heat. However, with the transistors in place the resistors were easy to locate. Finally solder in the rest of the components.



The copper strips on the two boards are gapped and then connected together.

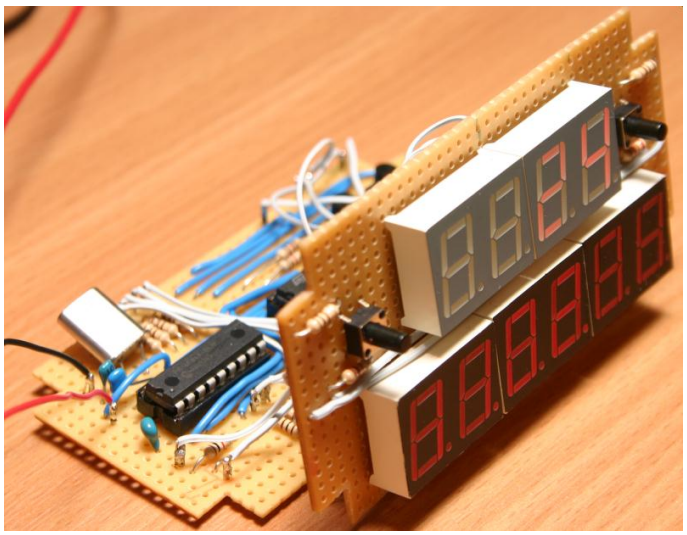


Figure 5 - The Completed Project

Before plugging in the IC's the other components should be tested. This is done using the following procedure.

1. Connect 5v Power supply
2. Measure the voltage and polarity of the pins that supply power to the IC's.
3. Connect the Segment pins to 0v (e.g. link IC1 pins 5 & 6) then connect each digit in turn to 5v (e.g. link IC2 pins 16 & 3 then 16 & 14 etc). The same segment should light on each digit.
4. Repeat step 3 for all other segments.

All that was left to do was to plug in the PIC, the 4028 and connect the 5V supply to test. Set the time to an accurate clock and check periodically to see if it gains or looses time. The PIC can then be reprogrammed to make it more accurate.

Conclusion

There is scope for additional features.

- Change the program so you cannot set an invalid date.
- A count up timer could be incorporated using another button. This could use the date display to count up from 0 in hours, minutes and seconds.
- A world clock can be created.
- Another additional feature could be to automatically change the clock for daylight saving. At the end of March the clocks go forward 1 hour and at the end of October the clocks go back 1 hour.
- Countdown until it's time to go home.
- Countdown timer.
- Hourly 'Chime'.

References

MikroElektronika
http://www.mikroe.com/eng/home/index/
Development tools and compilers etc.

Maplin Electronic Supplies
Supplier of all components including the PIC development board.

Rapid Online
Supplier of components.

http://www.pelnet.co.uk/elect/index.html
Visit my website for more PIC projects.