Project 9:
Real-Time Clock

EE 371

Jake A. Cole

11/5/2013

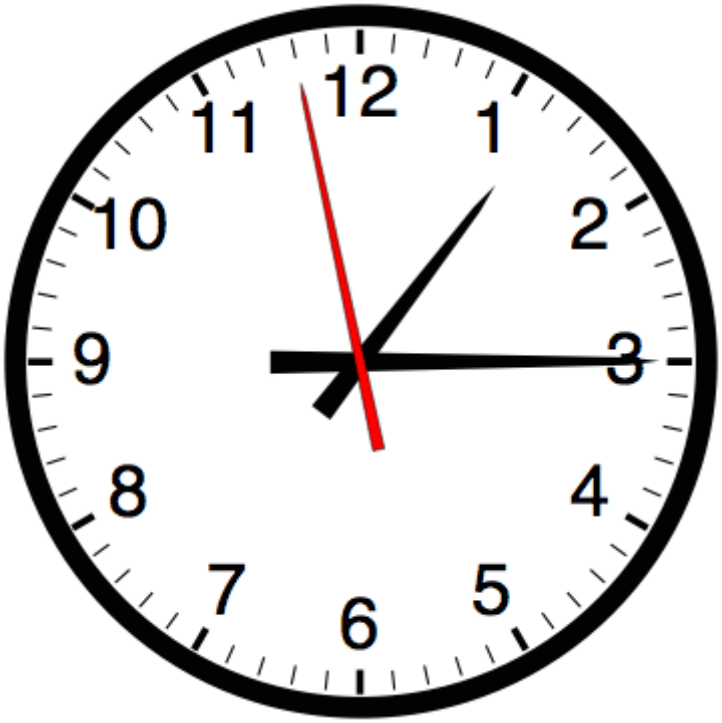


Table of Contents:

Program 9:

```
/*

Jake A. Cole
Project 9
11/5/2013
In the Real Time Clock project Timer1 is used to generate
interrupts every hundredth of a second to create a clock
that is displayed on the LCD screen that is able
to be enabled, disabled, reset, and refreshed
*/
void convert(int n, char s[]);
void interrupt (void);

int counter = 0, dflag = 0; //Display flag is used to determine if the LCD
                    //needs to refresh the current value on the screen
char string[6];

void main () {
  TMR1H = 0xF6;  //Load Timer for a hundredth of a second interrupts
  TMR1L = 0x3B;

  PIR1.TMR1IF = 0;      //Clear Timer overflow flag

  T1CON.TMR1ON = 1;     //Enable Timer
  T1CON.T1CKPS1 = 1;    //Set Prescaler
  T1CON.T1CKPS0 = 1;    //to 1:8
  T1CON.T1OSCEN = 0;    //Turn off oscillator

  PIE1.TMR1IE = 1;      //Enable timer overflow interrupts

  ANSEL  = 0x00;        //Clear analog input register
  ANSELH = 0x00;

  CM1CON0 = 0x00;       //Clear the comparators
  CM2CON0 = 0x00;

  IOCB = 0x00;          //Ensures mismatch interrupts are not enabled
  TRISB = 0x00;         //Sets TRISB as an output
  TRISC = 0xFF;         //Sets TRISC as an input
  TRISD = 0x00;         //Sets TRISD as an output

  INTCON.GIE = 1;       //Global interrupt enabled
  INTCON.PEIE = 1;      //Peripheral interrupt enabled

  Lcd_Config(&PORTB, 4, 5, 6, 3, 2, 1, 0); //Configures the LCD screen to PORTB
  LCD_Cmd(LCD_CURSOR_OFF);
  LCD_Cmd(LCD_CLEAR);
```

```
while (1) {     //infinite loop
   if (PORTC.F2 == 0) //The LCD will refresh the timer if RC2 is released
      if (dflag == 1) Lcd_Out(1, 1, string); //Displays the Clock on the LCD
   if (PORTC.F3 == 1) {
      counter = 0;       //Resets the timer if RC3 is pressed
      PIR1.TMR1IF = 1;   //Ensures the board displays the value 00.00
      T1CON.TMR1ON = 0;          //Disables the timer
   }
   if (PORTC.F4 == 1) T1CON.TMR1ON = 0;  //Disables the timer if RC4 is pressed
   if (PORTC.F5 == 1) T1CON.TMR1ON = 1;  //Enables the timer if RC5 is pressed
  }
}

/*
Interrupt reloads the timer when it overflows and
increments the counter every hundredth of a second
*/
void interrupt (void) {
  if (PIR1.TMR1IF == 0) { //Returns if the timer hasnt overflown
    dflag = 0;  //Flags that the board doesn't need to display the Clock
    return;
  }

  TMR1H = 0xF6;  //Load Timer
  TMR1L = 0x3B;
  PIR1.TMR1IF = 0;      //Clear Timer overflow flag
  convert(counter, string);
  dflag = 1; //Flags that the board needs to display the Clock on the LCD
  counter++;   //Increments the counter
  if (counter > 9999) counter = 0;   //Resets the counter if it hits 10,000

  if ((counter / 100) % 5 == 0) PORTD = 0xFF; //Lights up LEDs RD0 to RD7
  else PORTD = 0x00;  //Turns off LEDs RD0 to RD7
  return;
}

/*
Convert converts an integer to a string so it can be displayed on the LCD
*/
void convert(int n, char s[]) {
  s[5] = '\0';            //Adds null onto the string
  s[4] = n % 10 + '0';     //Hundredth of a second
  n /= 10;
  s[3] = n % 10 + '0';     //Tenth of a second
  n /= 10;
  s[2] = '.';             //Decimal place
  s[1] = n % 10 + '0';     //A second
  n /= 10;
  s[0] = n % 10 + '0';     //Ten seconds
  return;
}
```

Description of the Project:

The Real-Time Clock project utilizes Timer1 on the Pic16F887 board to generate an interrupt every one-hundredth of a second. Then the board utilizes this interrupt to generate a real-time clock and display its time on the LCD display. When the clock reaches 99.99 seconds it resets back to 00.00 seconds and then starts counting up again. The LED's on PORTD are lit for one second every five seconds. The program also has interactivity with four switches on PORTC. When RC2 is pressed it prevents the display from refreshing, and when it is released the display refreshes as it normally would. If RC3 is pressed it disables the timer and resets the counter to value 0. If RC4 is pressed, it disables the timer. If RC5 is pressed, it enables the timer.

Materials used:

Pic16F887 Board, USB Link Cable, Mikro C compiler

Solving the Problem:

To solve this problem I made sure to initialize the Pic16F887 board so that it could utilize Timer1 for the interrupt service routine. I loaded Timer1 with the hex value 0xF63B so that it would interrupt every one-hundredth of a second. I then made sure to clear the timer overflow flag. After this I enabled Timer1, set the pre-scaler to 1:8, and turned off the oscillator. I then enabled Timer1 overflow interrupts, turned off the comparators, and cleared the analog input register. Following that, I enabled global and peripheral interrupts. I also ensured that mismatch interrupts were not enabled. I set the TRISC as an input, and both the TRISB and TRISD as outputs. I also configured the LCD display to make sure that the value could be displayed to the screen.

I used if statements to meet the requirements for the inputs RC2-RC5. I used the LCD out function to display the clock on the display. I configured the interrupt function so that if the timer overflow flag was 0, it would return back to the main function; however, if the timer overflow flag was 1, the program would then go into the interrupt service routine. Timer1 would reset back to the hex value 0xF63B, the timer overflow flag would be reset, and the counter would be incremented. Also the convert function would be called to convert the counter value from an integer to a string. The display flag would be changed to one so that the LCD out function in main would be accessed. If the counter reached a value above 9,999 it would be reset back to 0. Also, if the clock displayed a second value of five, it would turn on the LEDs on PORTD.

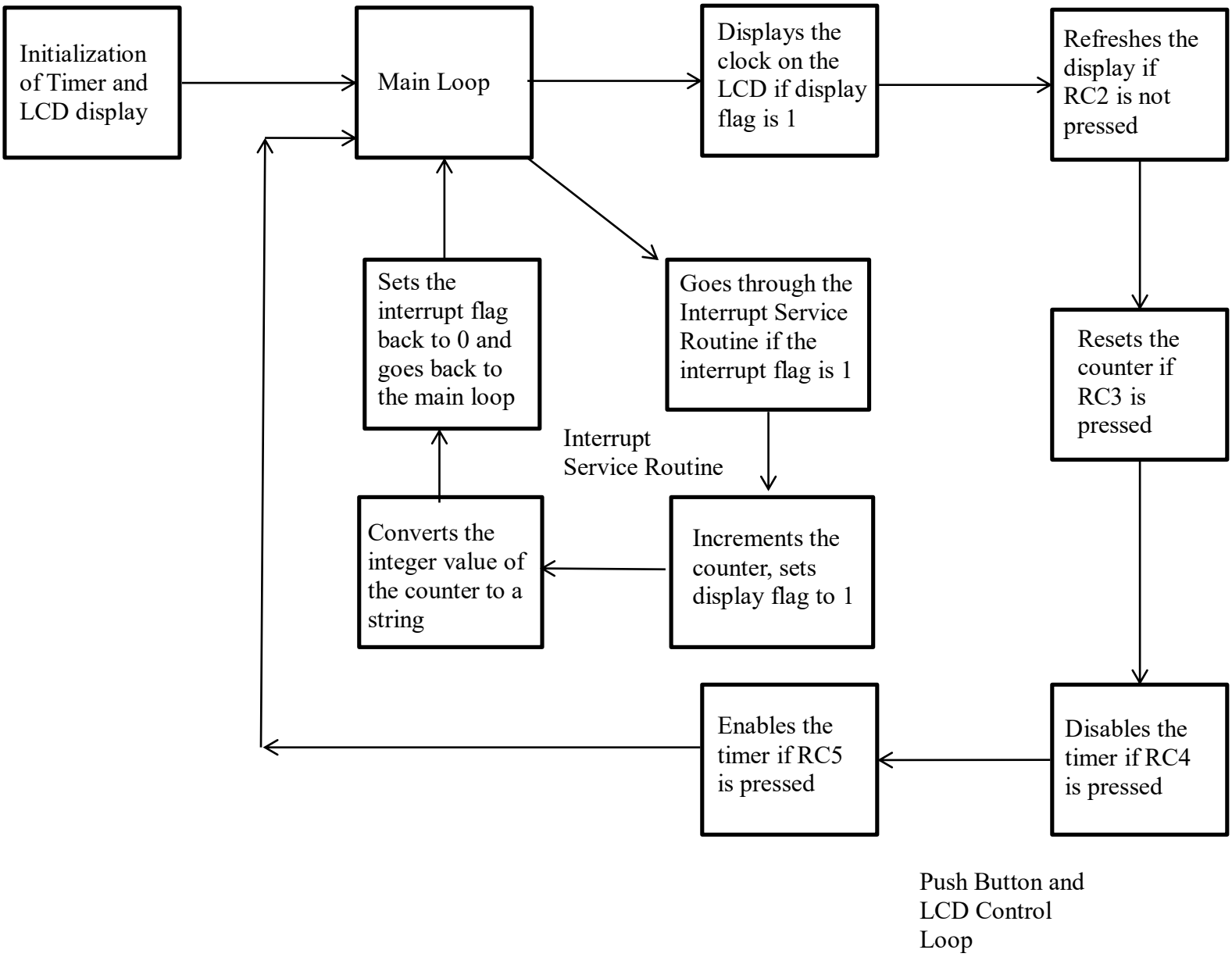
Explanation of the Modules:

The main function is used to initialize both the LCD display and Timer1. The main function also serves as a main loop that controls displaying the clock to the LCD screen as well as how the program interacts with input from PORTC.

The interrupt function is the major part of the interrupt service routine. It returns back to the main function if the interrupt flag is zero, or it executes the interrupt service routine if the flag is one. The interrupt service routine resets the value of Timer1 as well as sets the interrupt flag to zero. It also increments the value of the counter, converts the value of the counter to a string so it can be displayed, and it handles if the LEDs on PORTD should be lit or not.

The convert function converts an integer value to a string. In this particular case it converts the value of the counter to a string to be displayed on the LCD screen.

Flowchart, Figure 1:



Discussion of Results:

The Real-Time Clock works as intended and is functioning in real time. It utilizes the interrupt service routine to increment a counter every one-hundredth of a second to the LCD display. The push button RC5 enables the timer, RC4 disables the timer, RC3 resets the counter as well as disables the timer, and RC2 stops the display from refreshing when it is pressed.

Conclusion:

After completing the Real-Time Clock project I have learned how to utilize Timer1 to create a clock that is incremented every one-hundredth of a second. I have also learned how to implement the interrupt service routine by having Timer1 generate interrupts. I have become more proficient with the LCD display and the push buttons on the Pic16F887 board. I also learned that when a program has to work in real time, the program has to be as efficient as possible to ensure that it can fulfill its purpose. This project has helped to improve my methods of designing and mapping out flow charts. I have also learned how to initialize and manage Timer1 to increment a real time clock.

References:

Karimi, Bijan. "Embedded Systems Course Manual."
www.newhaven.edu. University of New Haven. Web.
4 Nov 2013.

"Pic16F887 Data Sheet."
http://www.microchip.com/. Microchip Technology Inc., n.d. Web.
4 Nov 2013.