Quang Nguyen

CPE 409 Lab

# Goals

To learn how to use interrupts in C programming language

# Equipment used

## Hardware

* Microchip Explorer 16 board
* PIC kit 3

## Software

* MPLAB X IDE 2.00

# Design Specifications

* Must act as a real time clock
  + Must write an ISR for timer 1
  + Timer 1 must be connected to the external 32.678 kHz watch crystal
  + Must store the value of seconds, minutes and hours.
  + Must display the minutes on the onboard LEDs
    - The output for the minutes will blink at the same frequency as seconds.
    - Displayed values must be in BCD
    - Seconds and minutes must roll over after 59
    - Hours must roll over after 12
* Optional:
  + A press of switch S3 will shows the seconds
  + A press of switch s6 will shows the hours

# Design

* Refer to figure 1 and 2 for flow diagram of the design.

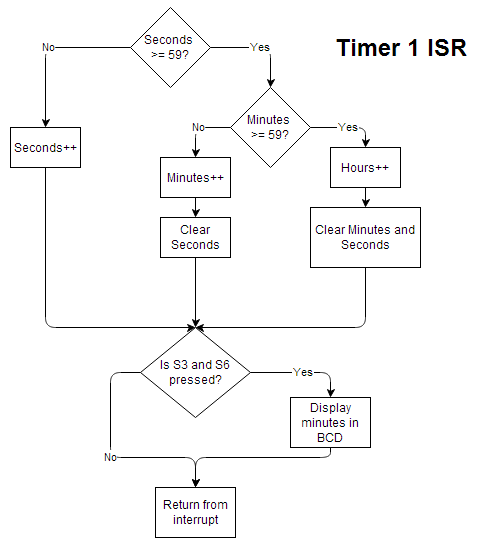


Figure 1: Flow diagram of Timer 1 ISR

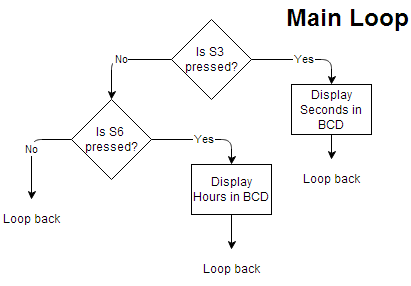


Figure 2: Flow diagram of the main loop

* Refer to the end of the report for the code of the design

# Verification

* The program was allowed to run for 5 minutes.
  + During this time, the minutes display was observed to blink with a frequency of 1 Hz.
  + Also during this time, the BCD value of minutes was observed to working as expected.
  + Using a phone’s stopwatch function, a minute of the RTC was measure to be approximately equal to one minute of the stopwatch
* S3 was then pressed
  + It was observed that the value of seconds does change everyone 1 second.
  + It was observed that the BCD value of seconds was represented correctly
  + The display of seconds observed to roll over after 59 seconds.
* Minutes was allow to reach 59
  + It was observed that the display of minutes does indeed roll over after 59 minutes
* S6 was then pressed after 4 hours has elapsed
  + It was observed that the BCD value of hours does indeed represent 5.

# Conclusions and Limitations

* No limitations were found
* The program works perfectly as expected.

# Programming Code

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Project: Lab 4

\*

\* Name: Quang Nguyen

\*

\* Date: 2/7/2014

\*

\* Function: Use the board as a Real Time Clock (RTC).

\* The program will keep track of seconds, minutes and hours.

\* The external 32.768KHz watch crystal will be use for timer 1.

\* The lower 8 bits of PORT A will be used as indicators.

\*

\* Operation:

\* [Normal]

\* The 8 LEDs of PORTA will tell the minutes in BCD. The lower

\* nibble will indicate the 1's place and the upper nibble will

\* indicate the 10's place.

\* The 8 LEDs of PORTA will turn on and off at a 1 second interval

\* to indicate that 1 second has past

\*

\* [Extra Credit]

\* Press S3 (RD6) to show Seconds

\* Press S6 (RD7) to show Hours.

\*

\*

\* Pins used:

\* [Normal]

\* The lower 8 bits of PORTA

\* [Extra Credit]

\* RD6 and RD7

\*TIMER 1

\* To get an interrupt every 1 seconds

\* External clock = 32.768 KHz

\* 1/32.768 KHz = 30.517 us

\* with prescaler at 8

\* 30.517 us \* 8 = 0.244136 ms

\* 1/0.244136 = 4096 clock pulses

\* Peripherals used:

\*

\*

\*

\* Comments:

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Configuration Bits

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <xc.h>

// FOSCSEL

#pragma config FNOSC = PRIPLL // Oscillator Mode (Primary Oscillator (XT, HS, EC) w/ PLL)

#pragma config IESO = ON // Two-speed Oscillator Start-Up Enable (Start up with FRC, then switch)

// FOSC

#pragma config POSCMD = XT // Primary Oscillator Source (XT Oscillator Mode)

#pragma config OSCIOFNC = OFF // OSC2 Pin Function (OSC2 pin has clock out function)

#pragma config FCKSM = CSDCMD // Clock Switching and Monitor (Both Clock Switching and Fail-Safe Clock Monitor are disabled)

// FWDT

#pragma config FWDTEN = OFF // Watchdog Timer Enable (Watchdog timer enabled/disabled by user software)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Library includes

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <p33FJ256GP710A.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Constant Declarations

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define TIME\_SECONDS 59 // These contants dictate how many counts till roll over

#define TIME\_MINUTES 59 // Changing these number can help debugging go much quicker

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Global Variable Declarations

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char volatile Hours = 12, Minutes = 58, Seconds = 30;

unsigned char volatile PortAToggleFlag = 1;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Function Prototype

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void initialize();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Function name : BinaryToBCD

\*

\* Returns : Binary coded decimal value of the input (2 digits)

\*

\* Parameters : Value between (0-99) to be encoded into BCD

\*

\* Purpose : Convert a character into a BCD encoded character.

\* The input must be in the range 0 to 99.

\* The result is byte where the high and low nibbles

\* contain the tens and ones of the input.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char BinaryToBcd(unsigned char input);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Main Function

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

// setting up everything

initialize();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Main Loop \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while (1) {

if (\_RD6 == 0) // If S3 button is pressed

{ // Need to display Seconds

PORTA = BinaryToBcd(Seconds); // PORTA = BCD value of Seconds

PortAToggleFlag = 1; // Toggle Flag to ON

}

else if (\_RD7 == 0) // If S6 button is pressed

{ // Need to display HOurs

PORTA = BinaryToBcd(Hours); // PORTA = BCD value of Hours

PortAToggleFlag = 1; // Toggle Flag to ON

}

} // End of the infinite While loop

return 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Initialize Function

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void initialize() {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Setting up for Clock (PLL, M, N1, N2)

\* for 32 MHz and Fcy = 16 MHz

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Fosc = Fin(M/(N1\*N2)) = 8 MHz (32/(2\*4)) = 32 MHz

PLLFBD = 30; // M = 32

// N1 default is 2

// N2 default is 4

// Fcy = Fosc/2 by default

// Setting up PORT A

AD1PCFGH = 0xFF; // Turn off ADC for Module 1

TRISA = 0x00; // Port A to output

PORTA = 0x00; // Keep PORTA off to begin with

// Setting up RD6 and RD7

\_TRISD6 = 1;

\_TRISD7 = 1; // Set RD6 & 7 to inputs

// Setting up Timer 1 module

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* External clock = 32.768 KHz

\* 1/32.768 KHz = 30.517 us

\* with prescaler at 8

\* 30.517 us \* 8 = 0.244136 ms

\* 1/0.244136 = 4096 clock pulses

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\_\_builtin\_write\_OSCCONL(2); // Macro to use secondary oscillator

T1CONbits.TCS = 1; // External clock from T1CK

T1CONbits.TSYNC = 0; // Do not synchronize external clock

T1CONbits.TCKPS = 1; // Set prescaler to 8

// Set initial value for PR1

PR1 = 4095; // T1 interrupt at 4096 clock pulse

// 4096 -1 = 4095

TMR1 = 0; // Clear TMR1

T1CONbits.TON = 1; // Turn on Timer 1

\_T1IE = 1; // Turn on TMR1 interrupt

// The default priority is 4 so lets not worry about it for now

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Interrupt Service Routine

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void \_\_attribute\_\_ ((interrupt, no\_auto\_psv)) \_T1Interrupt (void)

{ //

// Turn off flag //

IFS0bits.T1IF = 0; //

TMR1 = 0; // Clear TMR1

// Check if Seconds is equal to 59 //

if (Seconds >= TIME\_SECONDS) //

{ //

// if so, check if minutes is equal to 59 also

if (Minutes >= TIME\_MINUTES) //

{ //

// if so, increment Hours, Minutes = 0, and Seconds = 0

++Hours; //

Minutes = 0; //

Seconds = 0; //

} //

// if not, increment Minutes, and Seconds = 0

else //

{ //

++Minutes; //

Seconds = 0; //

} //

} //

// If not, Increment seconds //

else //

++Seconds; //

//

if ((\_RD6 == 1) && (\_RD7 == 1)) // Both buttons are not pressed

{ // Display Minutes with 1 s interval

// Decide whether to turn on PORTA or not based on PortAToggleFlag

// 1 = PORTA is on right now //

// 0 = PORTA is off right now //

if (PortAToggleFlag == 1) //PORTA is currently ON, lets turn it OFF

{ //

PORTA = 0; // Turn OFF PORTa

PortAToggleFlag = 0; // Toggle the flag to OFF

} //

else // PORTA is currently OFF, lets set it to Minutes

{ //

PORTA = BinaryToBcd(Minutes); // Set PORTA to the BCD value of minutes

PortAToggleFlag = 1; // Toggle the flag to ON

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Other Functions

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Function name : BinaryToBCD

\*

\* Returns : Binary coded decimal value of the input (2 digits)

\*

\* Parameters : Value between (0-99) to be encoded into BCD

\*

\* Purpose : Convert a character into a BCD encoded character.

\* The input must be in the range 0 to 99.

\* The result is byte where the high and low nibbles

\* contain the tens and ones of the input.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char BinaryToBcd(unsigned char input)

{

char high = 0;

while (input >= 10) // Count tens

{

high++;

input -= 10;

}

return (high << 4) | input; // Add ones and return answer

}