```c
/*
 * File:   WiFi_Test_main.c
 * Author: Austin
 *
 * Created on July 11, 2017, 9:42 PM
 */

/* Configuration Bits **********************************************************/
// FBS
#pragma config BWRP = OFF         // Boot Segment Write Protect (Disabled)
#pragma config BSS = OFF          // Boot segment Protect (No boot program
// flash segment)

// FGS
#pragma config GWRP = OFF         // General Segment Write Protect (General
// segment may be written)
#pragma config GSS0 = OFF         // General Segment Code Protect (No
// Protection)

// FOSCSEL
//#pragma config FNOSC = FRCDIV     // Oscillator Select (8MHz FRC oscillator With
// Postscaler (FRCDIV))
//#pragma config FNOSC = PRI//SOSC
#pragma config FNOSC = FRCPLL

#pragma config SOSCSRC = DIG      // SOSC Source Type (Analog Mode for use with
// crystal) (Digital Mode for not crystal)
#pragma config LPRCSEL = HP       // LPRC Oscillator Power and Accuracy (High
// Power, High Accuracy Mode)
#pragma config IESO = ON          // Internal External Switch Over bit (Internal
// External Switchover mode enabled (Two-speed
// Start-up enabled))

// FOSC
//#pragma config POSCMOD = NONE     // Primary Oscillator Configuration bits
#pragma config POSCMOD = HS     // Primary Oscillator Configuration bits
// (Primary oscillator disabled)
#pragma config OSCIOFNC = OFF     // CLKO Enable Configuration bit (CLKO output
// signal is active on the OSCO pin)
#pragma config POSCFREQ = HS      // Primary Oscillator Frequency Range
// Configuration bits (Primary oscillator/
// external clock input frequency greater than
// 8MHz)
#pragma config SOSCSEL = SOSCHP   // SOSC Power Selection Configuration bits
// (Secondary Oscillator configured for
// high-power operation)
#pragma config FCKSM = CSDCMD     // Clock Switching and Monitor Selection (Both
// Clock Switching and Fail-safe Clock Monitor
// are disabled)

// FWDT
#pragma config WDTPS = PS32768    // Watchdog Timer Postscale Select bits
// (1:32768)
#pragma config FWPSA = PR128      // WDT Prescaler bit (WDT prescaler ratio of
// 1:128)
#pragma config FWDTEN = OFF       // Watchdog Timer Enable bits (WDT disabled in
// hardware; SWDTEN bit disabled)
#pragma config WINDIS = OFF       // Windowed Watchdog Timer Disable bit
// (Standard WDT selected(windowed WDT
// disabled))

// FPOR
#pragma config BOREN = BOR0       // Brown-out Reset Enable bits (Brown-out
// Reset disabled in hardware, SBOREN bit
// disabled)
#pragma config LVRCFG = OFF       //  (Low Voltage regulator is not available)
#pragma config PWRTEN = ON        // Power-up Timer Enable bit (PWRT enabled)
#pragma config I2C1SEL = PRI      // Alternate I2C1 Pin Mapping bit (Use Default
// SCL1/SDA1 Pins For I2C1)
```

```
#pragma config BORV = V20          // Brown-out Reset Voltage bits (Brown-out
// Reset set to lowest voltage (2.0V))
#pragma config MCLRE = ON          // MCLR Pin Enable bit (RA5 input pin
//disabled,MCLR pin enabled)

// FICD
#pragma config ICS = PGx3          // ICD Pin Placement Select bits (EMUC/EMUD
// share PGC2/PGD2)

// FDS
#pragma config DSWDTPS = DSWDTPSF // Deep Sleep Watchdog Timer Postscale Select
// bits (1:2,147,483,648 (25.7 Days))
#pragma config DSWDTOSC = LPRC     // DSWDT Reference Clock Select bit (DSWDT
// uses Low Power RC Oscillator (LPRC))
#pragma config DSBOREN = OFF       // Deep Sleep Zero-Power BOR Enable bit (Deep
// Sleep BOR enabled in Deep Sleep)
#pragma config DSWDTEN = ON        // Deep Sleep Watchdog Timer Enable bit (DSWDT
// enabled)


#include <p24FV32KA302.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#include "test_LEDs.h"
#include "pic_frequency.h"
#include "limits.h"
#include <libpic30.h>

#include "WiFi.h"


unsigned long timer_ms = 0;
unsigned long oldTimer = 0;

void init( void );
void initTimer( void );
void initUART1( void );

/*
 *
 */
int main( int argc, char** argv )
{
    ANSBbits.ANSB14 = 0;
    ANSBbits.ANSB15 = 0;

    LED1DIR = 0;
    LED2DIR = 0;

    LED1SET = 0;
    LED2SET = 0;



    __delay_ms( 500 );
    for( int inx = 0; inx < 3; inx++ )
    {
    if( LED1READ == 0 )
    {
        LED1SET = 1;
    }
    else
    {
        LED1SET = 0;
    }
    __delay_ms( 500 );
    }
    init( );

    wifi( true );


    for( int inx = 0; inx < 10; inx++ )
    {
    if( LED1READ == 0 )
    {
        LED1SET = 1;
    }
    else
    {
        LED1SET = 0;
    }
    __delay_ms( 100 );
    }

    LED1SET = 0;
    LED2SET = 0;


    LED1SET = 1;
    __delay_ms( 500 );
    LED2SET = 1;
    __delay_ms( 500 );
    LED1SET = 0;
    __delay_ms( 500 );
    LED2SET = 0;


    __delay_ms( 1000 );

    //    unsigned long oldTimer = 0;

    U2TXREG = '\r';
    __delay_ms( 50 );
    U2TXREG = '\n';
    __delay_ms( 50 );
    U2TXREG = 's';
    __delay_ms( 50 );
    U2TXREG = 't';
    __delay_ms( 50 );
    U2TXREG = 'a';
    __delay_ms( 50 );
    U2TXREG = 'r';
    __delay_ms( 50 );
    U2TXREG = 't';
    __delay_ms( 50 );
    U2TXREG = '\r';
    __delay_ms( 50 );
    U2TXREG = '\n';
    __delay_ms( 50 );


    while( 1 )
    {
#define TIMER_MS_COUNT 1000
    wifi( false );
    if( TMR1 > TIMER_MS_COUNT )
    {
        if( timer_ms == ULONG_MAX )
        {
        timer_ms = 0;
        }
        else
        {
        timer_ms++;
        }
        TMR1 = 0x0000;
    }


    //  if( timer_ms > (oldTimer + 1000) )
    //  {
    //      oldTimer = timer_ms;
    //      if( LED2READ == 0 )
    //      {
    //      LED2SET = 1;
    //      U2TXREG = 't';
    //      }
    //      else
    //      {
    //      LED2SET = 0;
    //      }
    //  }

    }

    return (EXIT_SUCCESS);
}

void init( void )
{
    ANSBbits.ANSB14 = 0;
    ANSBbits.ANSB15 = 0;

    TRISBbits.TRISB14 = 0;
    TRISBbits.TRISB15 = 0;

    initTimer( );

    //set timer and interrupt to run a ms timer
    // no more __delay_ms if at all possible

}

void initTimer( void )
{
    // set timer up here
    T1CONbits.TSIDL = 0b1; //Discontinue module operation when device enters idle mode
    T1CONbits.T1ECS = 0b00; // doesn't matter because we use internal FOSC
    T1CONbits.TGATE = 0b0; // Gated time accumulation is disabled
    T1CONbits.TSYNC = 0b0; // Do not synchronize external clock input (asynchronous)
    T1CONbits.TCS = 0b0; //use internal clock


    T1CONbits.TCKPS = 0b01; // Timer 1 Input Clock Prescale (11-256)(10-64) (01-8) (00-1)
    TMR1 = 0x0000; // start timer at 0

    timer_ms = 0;

    T1CONbits.TON = 0b1; //turn on timer

}
```