```c
// Frequency Counter / Timer Example
// Jason Losh

//-----------------------------------------------------------------------------
// Hardware Target
//-----------------------------------------------------------------------------

// Target Platform: EK-TM4C123GXL with LCD/Temperature Sensor
// Target uC:       TM4C123GH6PM
// System Clock:    40 MHz

// Hardware configuration:
// Red LED:
//   PF1 drives an NPN transistor that powers the red LED
// Green LED:
//   PF3 drives an NPN transistor that powers the green LED
// Blue LED:
//   PF2 drives an NPN transistor that powers the blue LED
// Pushbutton:
//   SW1 pulls pin PF4 low (internal pull-up is used)
// Red Backlight LED:
//   PB5 drives an NPN transistor that powers the red LED
// Green Backlight LED:
//   PE5 drives an NPN transistor that powers the green LED
// Blue Backlight LED:
//   PE4 drives an NPN transistor that powers the blue LED
// LM60 Temperature Sensor:
//   AN0/PE3 is driven by the sensor (Vout = 424mV + 6.25mV / degC with +/-2degC uncalibrated
error)
// ST7565R Graphics LCD Display Interface:
//   MOSI (SSI2Tx) on PB7
//   MISO (SSI2Rx) is not used by the LCD display but the pin is used for GPIO for A0
//   SCLK (SSI2Clk) on PB4
//   A0 connected to PB6
//   ~CS connected to PB1
// Frequency counter and timer input:
//   FREQ_IN (WT5CCP0) on PD6

//-----------------------------------------------------------------------------
// Device includes, defines, and assembler directives
//-----------------------------------------------------------------------------

#include <stdint.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include "tm4c123gh6pm.h"
#include "graphics_lcd.h"
#include "wait.h"
#define RED_BL_LED   (*((volatile uint32_t *)(0x42000000 + (0x400053FC-0x40000000)*32 + 5*4)))
#define GREEN_BL_LED (*((volatile uint32_t *)(0x42000000 + (0x400243FC-0x40000000)*32 + 5*4)))
#define BLUE_BL_LED  (*((volatile uint32_t *)(0x42000000 + (0x400243FC-0x40000000)*32 + 4*4)))

#define RED_LED      (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 1*4)))
#define GREEN_LED    (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 3*4)))
#define BLUE_LED     (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 2*4)))
```

```c
#define PUSH_BUTTON  (*((volatile uint32_t *)(0x42000000 + (0x400253FC-0x40000000)*32 + 4*4)))

//-----------------------------------------------------------------------------
// Global variables
//-----------------------------------------------------------------------------

bool timeMode = false;
uint32_t frequency = 0;
uint32_t time = 0;
bool freqUpdate = false;
bool timeUpdate = false;

//-----------------------------------------------------------------------------
// Subroutines
//-----------------------------------------------------------------------------

void setCounterMode()
{
    SYSCTL_RCGCWTIMER_R |= SYSCTL_RCGCWTIMER_R5;     // turn-on timer
    WTIMER5_CTL_R &= ~TIMER_CTL_TAEN;                // turn-off counter before reconfiguring
    WTIMER5_CFG_R = 4;                               // configure as 32-bit counter (A only)
    WTIMER5_TAMR_R = TIMER_TAMR_TAMR_CAP | TIMER_TAMR_TACDIR; // configure for edge count mode, count up
    WTIMER5_CTL_R = 0;                               //
    WTIMER5_IMR_R = 0;                               // turn-off interrupts
    WTIMER5_TAV_R = 0;                               // zero counter for first period
    WTIMER5_CTL_R |= TIMER_CTL_TAEN;                 // turn-on counter
    NVIC_EN3_R &= ~(1 << (INT_WTIMER5A-16-96));      // turn-off interrupt 120 (WTIMER5A)
}

void setTimerMode()
{
    SYSCTL_RCGCWTIMER_R |= SYSCTL_RCGCWTIMER_R5;     // turn-on timer
    WTIMER5_CTL_R &= ~TIMER_CTL_TAEN;                // turn-off counter before reconfiguring
    WTIMER5_CFG_R = 4;                               // configure as 32-bit counter (A only)
    WTIMER5_TAMR_R = TIMER_TAMR_TACMR | TIMER_TAMR_TAMR_CAP | TIMER_TAMR_TACDIR; // configure for edge time mode, count up
    WTIMER5_CTL_R = TIMER_CTL_TAEVENT_POS;           // measure time from positive edge to positive edge
    WTIMER5_IMR_R = TIMER_IMR_CAEIM;                 // turn-on interrupts
    WTIMER5_TAV_R = 0;                               // zero counter for first period
    WTIMER5_CTL_R |= TIMER_CTL_TAEN;                 // turn-on counter
    NVIC_EN3_R |= 1 << (INT_WTIMER5A-16-96);         // turn-on interrupt 120 (WTIMER5A)
}
// Initialize Hardware
void initHw()
{
    // Configure HW to work with 16 MHz XTAL, PLL enabled, system clock of 40 MHz
    SYSCTL_RCC_R = SYSCTL_RCC_XTAL_16MHZ | SYSCTL_RCC_OSCSRC_MAIN | SYSCTL_RCC_USESYSDIV | (4 << SYSCTL_RCC_SYSDIV_S);

    // Set GPIO ports to use APB (not needed since default configuration -- for clarity)
    // Note UART on port A must use APB
    SYSCTL_GPIOHBCTL_R = 0;

    // Enable GPIO port B, D, E, and F peripherals
    SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOB | SYSCTL_RCGC2_GPIOD | SYSCTL_RCGC2_GPIOE | SYSCTL_RCGC2_GPIOF;

    // Configure LED and pushbutton pins
    GPIO_PORTF_DIR_R = 0x0E;  // bits 1-3 are outputs, other pins are inputs
    GPIO_PORTF_DR2R_R = 0x0E; // set drive strength to 2mA (not needed since default configuration -- for clarity)
    GPIO_PORTF_DEN_R = 0x1E;  // enable LEDs and pushbuttons
    GPIO_PORTF_PUR_R = 0x10;  // enable internal pull-up for push button

    // Configure three backlight LEDs
    GPIO_PORTB_DIR_R |= 0x20;  // make bit5 an output
    GPIO_PORTB_DR2R_R |= 0x20; // set drive strength to 2mA
    GPIO_PORTB_DEN_R |= 0x20;  // enable bit5 for digital
    GPIO_PORTE_DIR_R |= 0x30;  // make bits 4 and 5 outputs
    GPIO_PORTE_DR2R_R |= 0x30; // set drive strength to 2mA
    GPIO_PORTE_DEN_R |= 0x30;  // enable bits 4 and 5 for digital

    // Configure A0 and ~CS for graphics LCD
    GPIO_PORTB_DIR_R |= 0x42;  // make bits 1 and 6 outputs
    GPIO_PORTB_DR2R_R |= 0x42; // set drive strength to 2mA
    GPIO_PORTB_DEN_R |= 0x42;  // enable bits 1 and 6 for digital

    // Configure SSI2 pins for SPI configuration
    SYSCTL_RCGCSSI_R |= SYSCTL_RCGCSSI_R2;           // turn-on SSI2 clocking
    GPIO_PORTB_DIR_R |= 0x90;                        // make bits 4 and 7 outputs
    GPIO_PORTB_DR2R_R |= 0x90;                       // set drive strength to 2mA
    GPIO_PORTB_AFSEL_R |= 0x90;                      // select alternative functions for MOSI, SCLK pins
    GPIO_PORTB_PCTL_R = GPIO_PCTL_PB7_SSI2TX | GPIO_PCTL_PB4_SSI2CLK; // map alt fns to SSI2
    GPIO_PORTB_DEN_R |= 0x90;                        // enable digital operation on TX, CLK pins

    // Configure the SSI2 as a SPI master, mode 3, 8bit operation, 1 MHz bit rate
    SSI2_CR1_R &= ~SSI_CR1_SSE;                      // turn off SSI2 to allow re-configuration
    SSI2_CR1_R = 0;                                  // select master mode
    SSI2_CC_R = 0;                                   // select system clock as the clock source
    SSI2_CPSR_R = 40;                                // set bit rate to 1 MHz (if SR=0 in CR0)
    SSI2_CR0_R = SSI_CR0_SPH | SSI_CR0_SPO | SSI_CR0_FRF_MOTO | SSI_CR0_DSS_8; // set SR=0, mode 3 (SPH=1, SPO=1), 8-bit
    SSI2_CR1_R |= SSI_CR1_SSE;                       // turn on SSI2

    // Configure FREQ_IN for frequency counter
    GPIO_PORTD_AFSEL_R |= 0x40;                      // select alternative functions for FREQ_IN pin
    GPIO_PORTD_PCTL_R &= ~GPIO_PCTL_PD6_M;           // map alt fns to FREQ_IN
    GPIO_PORTD_PCTL_R |= GPIO_PCTL_PD6_WT5CCP0;
    GPIO_PORTD_DEN_R |= 0x40;                        // enable bit 6 for digital input

    // Configure Wide Timer 5 as counter
    if (timeMode)
        setTimerMode();
    else
        setCounterMode();

    // Configure Timer 1 as the time base
    SYSCTL_RCGCTIMER_R |= SYSCTL_RCGCTIMER_R1;       // turn-on timer
    TIMER1_CTL_R &= ~TIMER_CTL_TAEN;                 // turn-off timer before reconfiguring
    TIMER1_CFG_R = TIMER_CFG_32_BIT_TIMER;           // configure as 32-bit timer (A+B)
    TIMER1_TAMR_R = TIMER_TAMR_TAMR_PERIOD;          // configure for periodic mode (count
down)
    TIMER1_TAILR_R = 0x2625A00;                      // set load value to 40e6 for 1 Hz
interrupt rate
    TIMER1_IMR_R = TIMER_IMR_TATOIM;                 // turn-on interrupts
    NVIC_EN0_R |= 1 << (INT_TIMER1A-16);             // turn-on interrupt 37 (TIMER1A)
    TIMER1_CTL_R |= TIMER_CTL_TAEN;                  // turn-on timer
}

// Frequency counter service publishing latest frequency measurements every second
void Timer1Isr()
{
    if (!timeMode)
    {
        frequency = WTIMER5_TAV_R;                   // read counter input
        WTIMER5_TAV_R = 0;                           // reset counter for next period
        freqUpdate = true;                           // set update flag
        GREEN_LED ^= 1;                              // status
    }
    TIMER1_ICR_R = TIMER_ICR_TATOCINT;               // clear interrupt flag
}

// Period timer service publishing latest time measurements every positive edge
void WideTimer5Isr()
{
    if (timeMode)
    {
        time = WTIMER5_TAV_R;                        // read counter input
        WTIMER5_TAV_R = 0;                           // zero counter for next edge
        time /= 40;                                  // scale to us units
        timeUpdate = true;                           // set update flag
        GREEN_LED ^= 1;                              // status
    }
    WTIMER5_ICR_R = TIMER_ICR_CAECINT;               // clear interrupt flag
}

//-----------------------------------------------------------------------------
// Main
//-----------------------------------------------------------------------------

int main(void)
{
    // Initialize hardware
    initHw();

    // Turn-on all LEDs to create white backlight
    RED_BL_LED = 1;
    GREEN_BL_LED = 1;
    BLUE_BL_LED = 1;

    // Initialize graphics LCD
    initGraphicsLcd();

    // Draw legend
    setGraphicsLcdTextPosition(0, 0);
    putsGraphicsLcd("Frequency (Hz)");
    setGraphicsLcdTextPosition(0, 2);
    putsGraphicsLcd("Period (us)");

    BLUE_LED = timeMode;

    // Endless loop performing multiple tasks
    // If frequency is updated, then update the display
    char str[10];
    while (1)
    {
        if (freqUpdate)
        {
            freqUpdate = false;
            sprintf(str, "%7lu", frequency);
            setGraphicsLcdTextPosition(0, 1);
            putsGraphicsLcd(str);
        }
        if (timeUpdate)
        {
            timeUpdate = false;
            sprintf(str, "%7lu", time);
            setGraphicsLcdTextPosition(0, 3);
            putsGraphicsLcd(str);
        }
        if (!PUSH_BUTTON)
        {
            timeMode = !timeMode;
            BLUE_LED = timeMode;
            if (timeMode)
                setTimerMode();
            else
                setCounterMode();
            waitMicrosecond(250000);
        }
    }
}


```