Assignment 2 - Timers

EGC 443 Embedded Systems

George Dagis (CE), NAME OMITTED (CE), NAME OMITTED (CE)

March 13th, 2018

Instructor: Michael Otis

Fade LED In and Out

```
/* p11_3.c: Use PWM to control LED intensity */

/* This program is based on p10_2. In the infinite loop, the value of CMPA register is
decremented by 100 every 20 ms. The decreasing CMPA value causes the duty cycle to
lengthen. */

#include "TM4C123GH6PM.h"

int main(void)

{

    void delayMs(int n);

    int x = 0;


    /* Enable Peripheral Clocks */

    SYSCTL->RCGCPWM |= 2;       /* enable clock to PWM1 */

    SYSCTL->RCGCGPIO |= 0x20;   /* enable clock to PORTF */

    SYSCTL->RCC &= ~0x00100000; /* no pre-divide for PWM clock */


    /* Enable port PF3 for PWM1 M1PWM7 */

    GPIOF->AFSEL = 8;           /* PF3 uses alternate function */

    GPIOF->PCTL &= ~0x0000F000; /* make PF3 PWM output pin */

    GPIOF->PCTL |= 0x00005000;

    GPIOF->DEN |= 8;            /* pin digital */


    PWM1->_3_CTL = 0;           /* stop counter */

    PWM1->_3_GENB = 0x0000008C; /* M1PWM7 output set when reload, */

                                /* clear when match PWMCMPA */

    PWM1->_3_LOAD = 16000;      /* set load value for 1kHz (16MHz/16000) */
    PWM1->_3_CMPA = 15999;      /* set duty cycle to min */
    PWM1->_3_CTL = 1;           /* start timer */
    PWM1->ENABLE = 0x80;        /* start PWM1 ch7 */


    for(;;)
    {
        do
        {
                x = x + 100;
                PWM1->_3_CMPA = x;
                 delayMs(20);
        }
                while (x < 15900);
                        do
                        {
                                x = x - 100;
                                PWM1->_3_CMPA = x;
                                 delayMs(20);
                        }
                while (x > 0);
                        //PWM1->_3_CMPA = x;
                         //delayMs(20);
          }
}
```

```
/* delay n milliseconds (16 MHz CPU clock) */

void delayMs(int n)

{

   int i, j;

   for(i = 0 ; i < n; i++)

      for(j = 0; j < 3180; j++)

         {}  /* do nothing for 1 ms */

}


/* This function is called by the startup assembly code to perform system specific initialization
tasks. */

void SystemInit(void)

{

   /* Grant coprocessor access */

   /* This is required since TM4C123G has a floating point coprocessor */

   SCB->CPACR |= 0x00f00000;

}
```

Calculate Period of Pulse

```
/* p5_8.c */

/* square wave signal should be fed to PB6 pin. Make sure it is 3.3 to 5V peak-to-peak. Initialize
Timer0A in edge-time mode to capture rising edges. The input pin of Timer0A is PB6. See Table
5-2.*/

#include <stdint.h>

#include "tm4c123gh6pm.h"

#include <stdio.h>

#include <stdlib.h>
```

```
void delayMs(int n);

void UART0Tx(char c);

char UART0Rx();




int main(void)

{

	char c;		//UART0 character to be received by the board from the terminal

				//(prevents code from looping until user inputs a character to

				//indicate they wish to rerun the code)


	char data;		//variable to compare Port B input to a desired result

				//in order to determine when to begin/end measurement

	int lastEdge, thisEdge, period, most, middle, least;

	char x[80];


	int i; //for loop counter


	int div;;



// UART0 INITIALIZATION
```

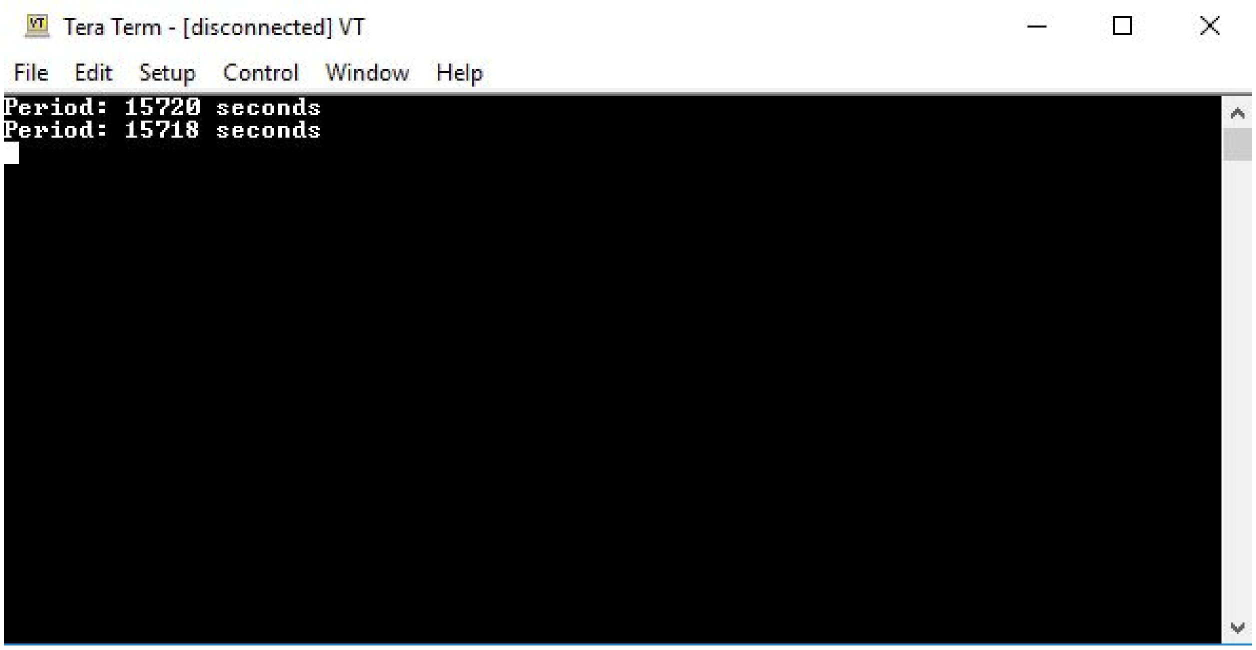
```
  SYSCTL->RCGCUART |= 1;  /* clock to UART0 */
  SYSCTL->RCGCGPIO |= 1;  /* enable clock to PORTA */

  UART0->CTL = 0;         /* disable UART0 */
  UART0->IBRD = 104;      /* 16MHz/16=1MHz, 1MHz/104=9600 baud rate */
  UART0->FBRD = 11;       /* fraction part, see Example 4-4 */
  UART0->CC = 0;          /* use system clock */
  UART0->LCRH = 0x60;     /* 8-bit, no parity, 1-stop bit, no FIFO */
  UART0->CTL = 0x301;     /* enable UART0, TXE, RXE */

   /* UART0 TX0 and RX0 use PA0 and PA1. Set them up. */
  GPIOA->DEN = 0x03;      /* Make PA0 and PA1 as digital */
  GPIOA->AFSEL = 0x03;    /* Use PA0,PA1 alternate function */
  GPIOA->PCTL = 0x11;     /* configure PA0 and PA1 for UART */

  delayMs(1);             /* wait for output line to stabilize */

SYSCTL->RCGCTIMER |= 1;                 //enable clock to Timer Block 0
SYSCTL->RCGCGPIO |= 2;          //enable clock to Port Block

 GPIOB->DIR &= ~0x40;        /* make PB6 an input pin */
 GPIOB->DEN |= 0x40;         /* make PB6 as digital pin */
 GPIOB->AFSEL |= 0x40;       /* use PB6 alternate function */
 GPIOB->PCTL &= ~0x0F000000;  /* configure PB6 for T0CCP0 */

        GPIOB->PCTL |= 0x07000000;
        TIMER0->CTL &= ~1;                    //disable timer0A during setup

        TIMER0->CFG = 4;              //16 bit timer mode

        TIMER0->TAMR = 0x17;          //up-count, edge-time, capture mode

        TIMER0->CTL &= ~0x0C;         //capture the rising edge

        TIMER0->CTL |= 1;             //enable timer0A


        delayMs(1);         /* wait for output line to stabilize */




for (;;)

        {

                c = UART0Rx();


                TIMER0->ICR = 4;   //Clear the timer0A capture flag

                while((TIMER0->RIS & 4) == 0); //Loop until first rising edge is captured

                lastEdge = TIMER0->TAR;           //Save the timestamp


                TIMER0->ICR = 4;

                while ((TIMER0->RIS & 4) == 0);

                thisEdge = TIMER0->TAR;


                if (lastEdge > thisEdge)

                        {

                                period = thisEdge + (0xFFFFFF - thisEdge);

                        }
                        else {period = thisEdge - lastEdge;}


                        period =  thisEdge - lastEdge;


    sprintf(x, "Period: %d seconds", period);
                        for (i = 0; i < 30; i++)
                        {
                                UART0Tx(x[i]);
                        }
                        UART0Tx('\n');
                        UART0Tx('\r');
            }


            return 0;
}
// UART0 Tx
// Make sure Tx buffer is not occupied prior to transmission
void UART0Tx(char c)
{
    while((UART0->FR & 0x20) != 0);
    UART0->DR = c;
}
// UART0 Receive
// Make sure Rx buffer is not empty before receiving data
char UART0Rx()
```

```
{
    char c;

    while((UART0->FR & 0x10) != 0);
    c = UART0->DR;
    return c;
}
void delayMs(int n)
{
    int i, j;

    for(i = 0 ; i < n; i++)
        for(j = 0; j < 3180; j++)
            {}  /* do nothing for 1 ms */
}

void SystemInit(void)
{
    SCB->CPACR |= 0x00f00000;
}
```

Summary

For part 1 of this experiment, the group had come up with an algorithm in order to fade an LED in and out by using variable loops with integer parameters and a delay function to reach those parameters in a reasonable amount of time (enough time to see the LED fade). Initially for-loops were going to be used but after some thinking, it was thought that do-while loops would be more appropriate. At the bottom of our main code we added a small SystemInit subroutine. This gives the coprocessor access to the control register. The delay function was also added at the bottom. There was no external system besides the board, as we used the on-board LED to test our code.

For part 2, the group had to measure the period and pulse width of an incoming signal by using the Timer sections that is built in the board. Once the signals are captured, the group had to output the results into a terminal using UART. In order to measure the period, pulse and captured the signals, the group use UART0 to measure the pulse width and sprintf function to capture the signal. In order to capture two period signal and return the difference, Timer0A_periodCapture function to capture the first rising edge and the second rising edge and return the difference between them.


Tera Term - [disconnected] VT
File Edit Setup Control Window Help

```
Period: 15720 seconds
Period: 15718 seconds
```