/////////////////////////////////////////////////////////////////////

/// REAL-TIME & EMBEDDED SYSTEMS

/// Project 2a: Group 6

///

/// Authors: Rasika Kangutkar, Sree

/// File: main.c

/////////////////////////////////////////////////////////////////////

//Include Headerfiles

#include <hidef.h>

#include <stdio.h>

#include "derivative.h"

#include "types.h" //General DataTypes

#include "serial.h" //SerialPort Control

//#include "recipe.h" //Recipe/Servo Control

//Master Timer Compare Value at 100ms

#define MASTER\_TMR\_VAL ((UINT16) 0xC350) //100ms(50,000 \* 2us)

int move=0;

/////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////

// ISR 9 Routine - 100ms Master Timer handler

/////////////////////////////////////////////////////////////////////

#pragma push

#pragma CODE\_SEG \_\_SHORT\_SEG NON\_BANKED

void interrupt 9 IC1\_isr( void )

{

TC1 += MASTER\_TMR\_VAL;

TFLG1 = TFLG1\_C1F\_MASK;

}

#pragma pop

/////////////////////////////////////////////////////////////////////

// postRoutine() - Check that the timer is incrementing properly

/////////////////////////////////////////////////////////////////////

int postRoutine()

{

UINT16 t0, t1;

UINT8 i;

i = 0x0;

t0 = TCNT;

while( i < 200)

{

i++;

}

t1 = TCNT;

if((t1 - t0) != 0x0)

{

return 1;

}

return 0;

}

/////////////////////////////////////////////////////////////////////

// initGPIO() - Initialize PORTB GPIO to output and clear

/////////////////////////////////////////////////////////////////////

void initGPIO()

{

DDRB = 0xFF;

PORTB = 0x00;

}

/////////////////////////////////////////////////////////////////////

// initData() - perform any necessary data initialization for control

// blocks

/////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////

// initTimers() - Initialize the 100ms master timer

/////////////////////////////////////////////////////////////////////

void initTimers()

{

//Set Prescale Value

//[010] = 1/4

//(2Mhz /4) = 500khz (2us resolution)

TSCR2\_PR0 = 0x0;

TSCR2\_PR1 = 0x1;

TSCR2\_PR2 = 0x0;

//Enable Compare

TIOS\_IOS1 = 0x1;

//Enable compare interrupt for IC

TIE\_C0I = 0x0;

//Master Timer @ 100ms

TC1 = MASTER\_TMR\_VAL;

//Clear interrupts

TFLG1 = TFLG1\_C1F\_MASK;

TIE\_C1I = 0x1;

TSCR1\_TEN = 0x1;

}

/////////////////////////////////////////////////////////////////////

// initPWM() - Initialize and enable channel 0,1 of the pulse width

// modulation

/////////////////////////////////////////////////////////////////////

void initPWM()

{

//Set the PreScalar value to 1/160 (25kHz) (.08ms res)

//ClockSA = BUS / (PCKA \* 2 \* SCLA)

// 2MHz /( 4 \* 2 \* 20) = 25kHz

PWMPRCLK\_PCKA0 = 0x0; //PCK = 1/4

PWMPRCLK\_PCKA1 = 0x1;

PWMPRCLK\_PCKA2 = 0x0;

PWMSCLA = 0xA; //SCLA = 1/20

//Channel 0

PWMCLK\_PCLK0 = 0x1; //Select ClockSA

PWMPOL\_PPOL0 = 0x1; //Polarity (duty is high)

PWMCAE\_CAE0 = 0x1; //Center Alignment

PWMPER0 = 0xFA; //Period = 20ms = (.08 \* 250)

PWMDTY0 = 0x6; //Duty = something out of range

//Channel 1

PWMCLK\_PCLK1 = 0x1; //Select ClockSA

PWMPOL\_PPOL1 = 0x1; //Polarity (duty is high)

PWMCAE\_CAE1 = 0x1; //Center Alignment

PWMPER1 = 0xFA; //Period = 20ms = (.08 \* 250)

PWMDTY1 = 0x6; //Duty = something out of range

//Enable both PWM channels

PWME\_PWME0 = 0x1;

PWME\_PWME1 = 0x1;

}

/////////////////////////////////////////////////////////////////////

// loadRecipes(UINT8 id) - allows user to select a predefined

// recipe

// @parameters

// id - specify recipe for channel 0 or 1

/////////////////////////////////////////////////////////////////////

// resetOverride() - reset the override ctrl structure

/////////////////////////////////////////////////////////////////////

UINT8 servoPos[6] = { 6, //0

10, //1

14, //2

18, //3

22, //4

30}; //5

void driveServo(UINT8 id, UINT8 pos)

{ move=1;

if(id == 0)

{

PWMDTY0 = servoPos[pos];

}

else if(id == 1)

{

PWMDTY1 = servoPos[pos];

}

move=0;

}

void main(void)

{

int i =0;

// initData(); //Initialize Any DataStructs

initSerial(); //Initialize SerialPort for I/O

initTimers(); //Initialize Timer

initPWM(); //Initialize PWM

initGPIO(); //Initialize GPIO

EnableInterrupts; //Enable Interrupts

// if(!postRoutine()) //Exit on POST failure

// return;

(void)printf("...POST...\r\n");

// loadRecipes( 0 ); //Load recipes for execution

// loadRecipes( 1 );

//Initialized to paused State

// RECIPE\_CTRL[0].recState = PAUSE;

// RECIPE\_CTRL[1].recState = PAUSE;

//Start the override prompt

// (void)printf("\r\n>");

i=0;

driveServo(1,5) ;

driveServo(0,5);

}