CPE403 – Advanced Embedded Systems

# Design Assignment #3

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Xianjie Cao

Email: [Caox2@unlv.nevada.edu](mailto:Caox2@unlv.nevada.edu)

Github Repository link (root): <https://github.com/c1029324620/Flat-White.git>

Youtube Playlist link (root): <https://www.youtube.com/playlist?list=PLY90fbcjLcrnosJGw9U__oC1jxRqwp8w8>

**Follow the submission guideline to be awarded points for this Assignment.**

Submit the following for all Assignments:

1. In the document, for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only.
2. Create a private Github repository with a random name (no CPE/403, Lastname, Firstname). Place all labs under the root folder TIVAC, sub-folder named Assignment1, with one document and one video link file for each lab, place modified c files named as asng\_taskxx.c.
3. If multiple c files or other libraries are used, create a folder asng1\_t01 and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) with startup\_ccs.c and other include files, c) text file with youtube video links (see template).
5. Submit the doc file in canvas before the due date. The root folder of the github assignment directory should have the documentation and the text file with youtube video links.
6. Organize your youtube videos as playlist under the name “cpe403”. The playlist should have the video sequence arranged as submission or due dates.
7. Only submit pdf documents. Do not forget to upload this document in the github repository and in the canvas submission portal.
8. Code for Tasks. for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only. Use separate page for each task.

\* ======== empty\_min.c ========

\*/

/\* XDCtools Header files \*/

**#include** <xdc/std.h>

**#include** <xdc/runtime/System.h>

**#include** <xdc/runtime/Log.h> //needed for any Log\_info() call

/\* BIOS Header files \*/

**#include** <ti/sysbios/BIOS.h>

**#include** <ti/sysbios/knl/Task.h>

/\* TI-RTOS Header files \*/

**#include** <ti/drivers/GPIO.h>

**#include** <ti/drivers/PWM.h>

// #include <ti/drivers/I2C.h>

// #include <ti/drivers/SDSPI.h>

// #include <ti/drivers/SPI.h>

**#include** <ti/drivers/UART.h>

// #include <ti/drivers/Watchdog.h>

// #include <ti/drivers/WiFi.h>

/\* Board Header file \*/

**#include** "Board.h"

//------------------------------------------

// TivaWare Header Files

//------------------------------------------

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <stdlib.h>

**#include** "inc/hw\_types.h"

**#include** "inc/hw\_memmap.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/gpio.h"

**#include** "inc/hw\_ints.h"

**#include** "driverlib/interrupt.h"

**#include** "driverlib/timer.h"

**#include** "driverlib/adc.h"

**#define** TASKSTACKSIZE 512

**#define** TASKSTACKSIZE2 768

Task\_Struct task0Struct, task1Struct, task2Struct;

Char task0Stack[TASKSTACKSIZE], task1Stack[TASKSTACKSIZE], task2Stack[TASKSTACKSIZE2];

Task\_Handle task;

uint16\_t adcValue;

uint16\_t duty = 0;

uint8\_t counter = 0;

uint32\_t ui32ADCValue[1];

UART\_Handle uart;

UART\_Params uartParams;

**char** buffer[20];

UART\_Handle uart;

UART\_Params uartParams;

**const** **char** echoPrompt[] = "\fUART Initialized!!\r\n";

**const** **char** writeDone[] = "ADC Value: ";

/\*

\* ======== heartBeatFxn ========

\* Toggle the Board\_LED0. The Task\_sleep is determined by arg0 which

\* is configured for the heartBeat Task instance.

\*/

Void **heartBeatFxn**(UArg arg0, UArg arg1)

{

**while** (1) {

Task\_sleep((UInt)arg0);

**GPIO\_toggle**(Board\_LED2); //PF1, RED LED

}

}

**void** **pwmFxn**(UArg arg0, UArg arg1)

{

PWM\_Handle pwm1;

PWM\_Params pwmParams;

uint16\_t pwmPeriod = 4096; //period and duty in ms

**PWM\_Params\_init**(&pwmParams);

pwmParams.period = pwmPeriod;

pwm1 = **PWM\_open**(Board\_PWM0, &pwmParams);

**if**(pwm1 == NULL)

{

System\_abort("Board\_PWM0 did not open");

}

//updating the duty cycle.

**while**(1)

{

**PWM\_setDuty**(pwm1, duty);

Task\_sleep((UInt) arg0);

}

}

**void** **buttonFxn**(**unsigned** **int** index)

{

//callback function to the button 0 interrupt

duty = adcValue;

}

Void **echoFxn**(UArg arg0, UArg arg1)

{

/\* Create a UART with data processing off. \*/

**UART\_Params\_init**(&uartParams);

uartParams.writeDataMode = *UART\_DATA\_BINARY*;

uartParams.readDataMode = *UART\_DATA\_BINARY*;

uartParams.readReturnMode = *UART\_RETURN\_FULL*;

uartParams.readEcho = *UART\_ECHO\_OFF*;

uartParams.baudRate = 9600;

uart = **UART\_open**(Board\_UART0, &uartParams);

**if** (uart == NULL) {

System\_abort("Error opening the UART");

}

**UART\_write**(uart, echoPrompt, **sizeof**(echoPrompt));

/\* Loop forever echoing \*/

**while**(1){

sprintf(buffer, "%i\r\n", adcValue);

**UART\_write**(uart, writeDone, **sizeof**(writeDone));

**UART\_write**(uart, buffer, **sizeof**(buffer));

}

}

**void** **hardware\_init**(**void**)

{

uint32\_t ui32Period;

//Set CPU Clock to 40MHz. 400MHz PLL/2 = 200 DIV 5 = 40MHz

//SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_16MHZ|SYSCTL\_OSC\_MAIN);

// Timer 2 setup code

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER2); // enable Timer 2 periph clks

**TimerConfigure**(TIMER2\_BASE, TIMER\_CFG\_PERIODIC); // cfg Timer 2 mode - periodic

ui32Period = (**SysCtlClockGet**() /1000); // period = CPU clk div 1000 (1ms)

**TimerLoadSet**(TIMER2\_BASE, TIMER\_A, ui32Period); // set Timer 2 period

**TimerIntEnable**(TIMER2\_BASE, TIMER\_TIMA\_TIMEOUT); // enables Timer 2 to interrupt CPU

**TimerEnable**(TIMER2\_BASE, TIMER\_A); // enable Timer 2

//set up ADC

//using the A11.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOB);

**GPIOPinTypeADC**(GPIO\_PORTE\_BASE, GPIO\_PIN\_5);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0);

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 32);

**ADCSequenceConfigure**(ADC0\_BASE, 0, ADC\_TRIGGER\_PROCESSOR, 0); // Changed to sequencer #2

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 0, ADC\_CTL\_CH11|ADC\_CTL\_IE|ADC\_CTL\_END);

**ADCIntEnable**(ADC0\_BASE, 0);

**ADCSequenceEnable**(ADC0\_BASE, 0);

}

/\*

\* ======== main ========

\*/

**int** **main**(**void**)

{

Task\_Params taskParams;

Task\_Params pwmTask;

Task\_Params uartTask;

/\* Call board init functions \*/

Board\_initGeneral();

Board\_initGPIO();

Board\_initPWM();

hardware\_init();

// Board\_initI2C();

// Board\_initSDSPI();

// Board\_initSPI();

Board\_initUART();

// Board\_initUSB(Board\_USBDEVICE);

// Board\_initWatchdog();

// Board\_initWiFi();

Task\_Params\_init(&taskParams);

taskParams.arg0 = 500;

taskParams.stackSize = TASKSTACKSIZE;

taskParams.stack = &task0Stack;

Task\_construct(&task0Struct, (Task\_FuncPtr)heartBeatFxn, &taskParams, NULL);

Task\_Params\_init(&pwmTask);

pwmTask.arg0 = 50;

pwmTask.stackSize = TASKSTACKSIZE;

pwmTask.stack = &task1Stack;

Task\_construct(&task1Struct, (Task\_FuncPtr)pwmFxn, &pwmTask, NULL);

task = Task\_handle(&task1Struct);

Task\_Params\_init(&uartTask);

uartTask.stackSize = TASKSTACKSIZE2;

uartTask.stack = &task2Stack;

uartTask.instance->name = "echo";

Task\_construct(&task2Struct, (Task\_FuncPtr)echoFxn, &uartTask, NULL);

/\* Turn on user LED \*/

// GPIO\_write(Board\_LED2, Board\_LED\_ON);

/\* Start BIOS \*/

BIOS\_start();

**return** (0);

}

**void** **timer2\_ISR**(voide)

{

**TimerIntClear**(TIMER2\_BASE, TIMER\_TIMA\_TIMEOUT); // must clear timer flag FROM timer

**ADCIntClear**(ADC0\_BASE, 0);

**ADCProcessorTrigger**(ADC0\_BASE, 0);

counter++;

**if**(counter == 5)

{

//perform ADC read

**ADCSequenceDataGet**(ADC0\_BASE, 0, ui32ADCValue);

adcValue = ui32ADCValue[0];

}

**if**(counter == 10)

{

//print ADC value to the UART

//

}

**if**(counter ==15)

{

//read the value of the button 0. and print the status to the console.

**if**(**GPIO\_read**(Board\_BUTTON0) == 1)

{

System\_printf("Button 0 not pressed\n");

System\_flush();

}

**else**

{

System\_printf("Button 0 is pressed\n");

System\_flush();

}

**GPIO\_setCallback**(Board\_BUTTON0, buttonFxn);

**GPIO\_enableInt**(Board\_BUTTON0);

//reset the counter.

counter = 0;

}

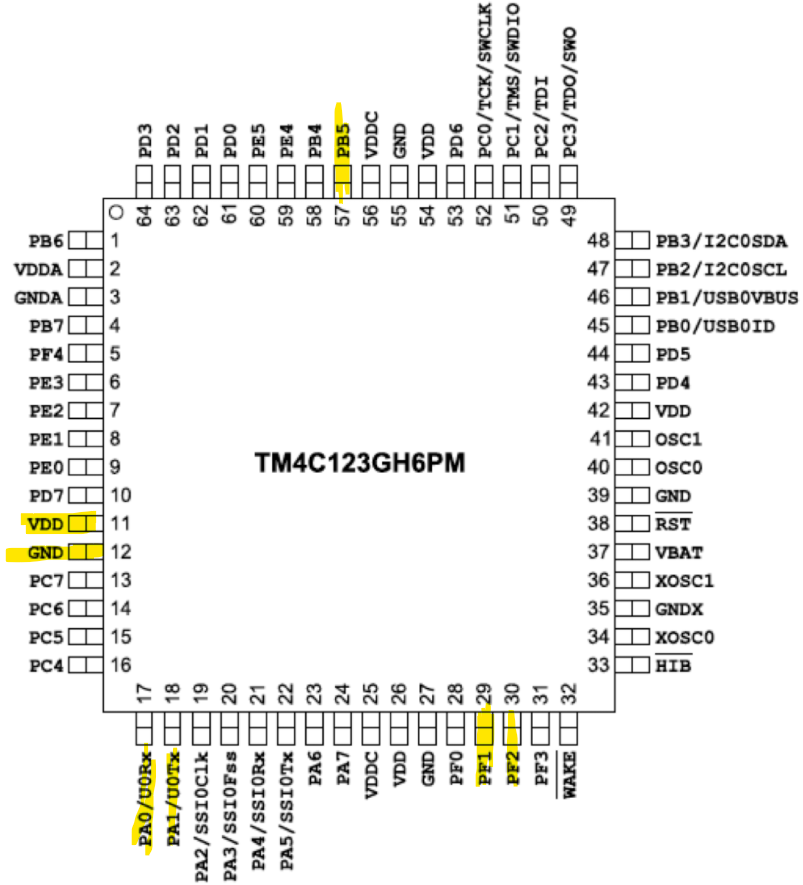
}

1. Block diagram and/or Schematics showing the components, pins used, and interface.

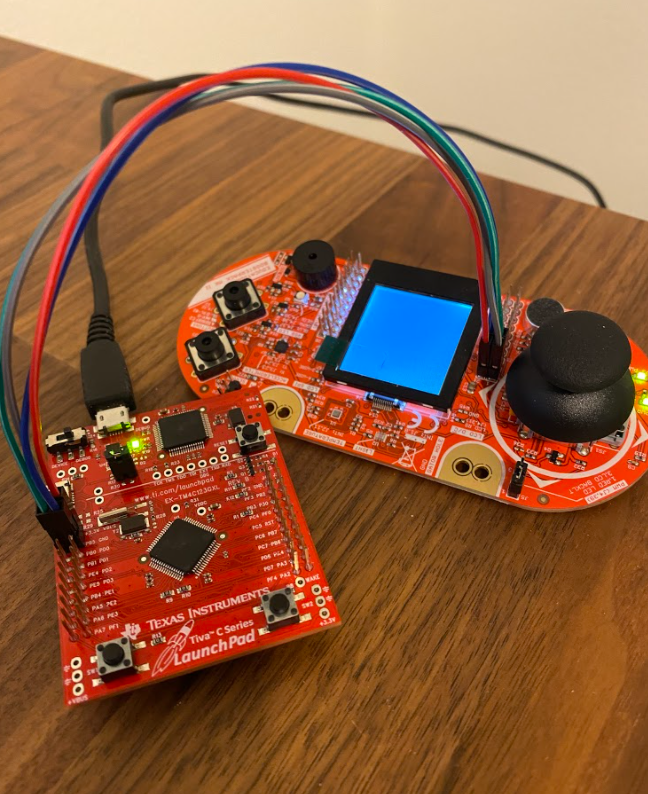
Components used: TM4C123GXL, BoosterPack MKII, jump wires, code composer studio.

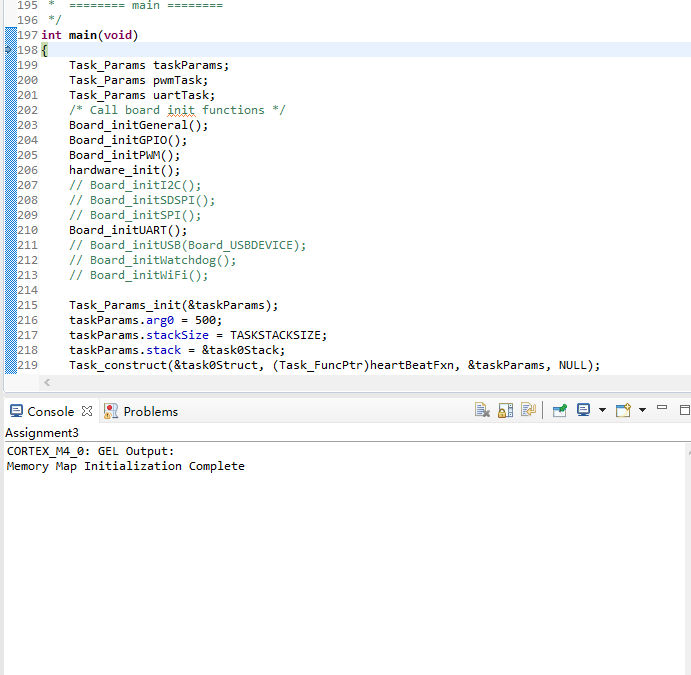
Pins used: 3.3V, 5V, GND, PF1, PF2, PA0, PA1, PB5.

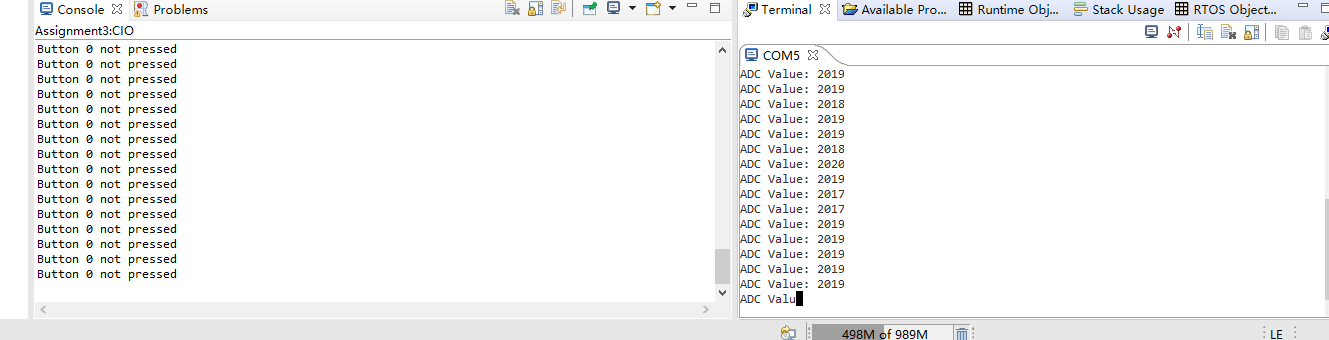
Interface: UART, ADC.

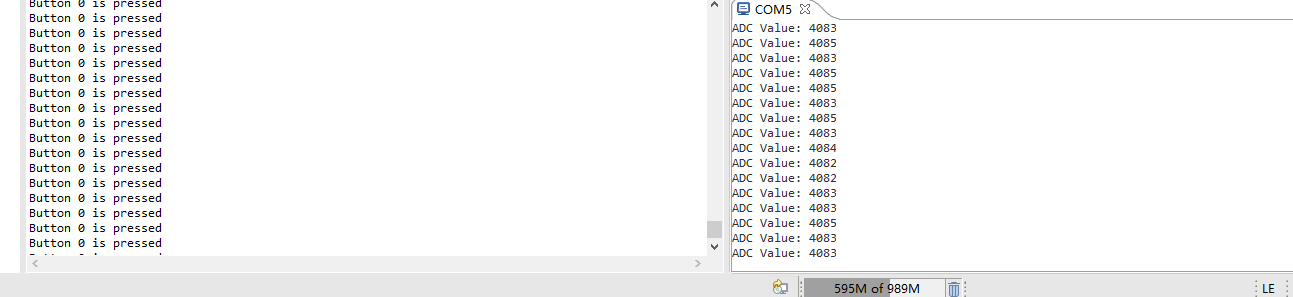


1. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.









1. Declaration

I understand the Student Academic Misconduct Policy - http://studentconduct.unlv.edu/misconduct/policy.html

“This assignment submission is my own, original work”.

Xianjie Cao