CPE403 – Advanced Embedded Systems

# Design Assignment 4

Name: Elmer Mejia

Email: mejiae4@unlv.nevada.edu

Github Repository link (root): [assignments](https://github.com/cpemejia/Launchpad_assignments.git)

Youtube Playlist link (root): [Tiva\_C](https://www.youtube.com/playlist?list=PL6PbL2NpuYCIJs2P3WZJMcGl-yEjS-LJJ)

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

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

A picture containing text, electronics

Description automatically generated



1. 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.

/\* TI-RTOS Header files \*/

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

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

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

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

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

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

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

**#include** <ti/display/Display.h>

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

/\*

\* ======== uartecho.c ========

\*/

**#include** <stdint.h>

**#include** <stddef.h>

/\* Driver configuration \*/

**#include** "ti\_drivers\_config.h"

/\* ADC sample count \*/

**#define** ADC\_SAMPLE\_COUNT (10)

/\* ADC conversion result variables \*/

uint16\_t adcValue0;

uint32\_t adcValue0MicroVolt;

uint16\_t adcValue1[ADC\_SAMPLE\_COUNT];

uint32\_t adcValue1MicroVolt[ADC\_SAMPLE\_COUNT];

**static** Display\_Handle display;

**void** **myDelay**(**int** count);

/\* Could be anything, like computing primes \*/

**#define** FakeBlockingSlowWork() myDelay(12000000)

**#define** FakeBlockingFastWork() myDelay(2000000)

Task\_Struct workTask;

Task\_Struct ADCTask;

/\* Make sure we have nice 8-byte alignment on the stack to avoid wasting memory \*/

**#pragma** DATA\_ALIGN(workTaskStack, 8)

**#define** STACKSIZE 1024

**static** uint8\_t workTaskStack[STACKSIZE];

**static** uint8\_t ADCTaskStack[STACKSIZE];

**void** **ADCt**(**void**)

{

ADC\_Handle adc;

ADC\_Params params;

int\_fast16\_t res;

**ADC\_Params\_init**(&params);

adc = **ADC\_open**(CONFIG\_ADC\_0, &params);

**if** (adc == NULL) {

Display\_printf(display, 0, 0, "Error initializing CONFIG\_ADC\_0\n");

**while** (1);

}

/\* Blocking mode conversion \*/

res = **ADC\_convert**(adc, &adcValue0);

**if** (res == ADC\_STATUS\_SUCCESS) {

adcValue0MicroVolt = ADC\_convertRawToMicroVolts(adc, adcValue0);

Display\_printf(display, 1, 0, "CONFIG\_ADC\_0 raw result: %d\n", adcValue0);

Display\_printf(display, 1, 0, "CONFIG\_ADC\_0 convert result: %d uV\n",

adcValue0MicroVolt);

}

**else** {

Display\_printf(display, 1, 0, "CONFIG\_ADC\_0 convert failed\n");

}

**ADC\_close**(adc);

}

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

{

**while** (1) {

ADCt();

}

}

**void** **doWork**(**void**)

{

**GPIO\_write**(Board\_GPIO\_LED0, CONFIG\_LED\_OFF);

FakeBlockingSlowWork(); /\* Pretend to do something useful but time-consuming \*/

**GPIO\_write**(Board\_GPIO\_LED0, CONFIG\_LED\_ON);

}

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

{

**while** (1) {

/\* Do work \*/

doWork();

/\* Wait a while, because doWork should be a periodic thing, not continuous.\*/

//myDelay(24000000);

Task\_sleep(500 \* (1000 / Clock\_tickPeriod));

}

}

/\*

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

\*

\*/

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

{

Board\_init();

**GPIO\_init**();

**PWM\_init**();

Display\_init();

// Display\_Handle hSerial = Display\_open(Display\_Type\_UART, NULL);

display = Display\_open(Display\_Type\_UART, NULL);

Display\_printf(display, 1, 0, "Starting the adc channel example\n");

/\* Set up the led task \*/

Task\_Params workTaskParams;

Task\_Params\_init(&workTaskParams);

workTaskParams.stackSize = STACKSIZE;

workTaskParams.priority = 2;

workTaskParams.stack = &ADCTaskStack;

Task\_construct(&ADCTask, ADCTaskFunc, &workTaskParams, NULL);

workTaskParams.priority = 1;

workTaskParams.stack = &workTaskStack;

Task\_construct(&workTask, workTaskFunc, &workTaskParams, NULL);

/\* Start kernel. \*/

BIOS\_start();

**return** (0);

}

/\*

\* ======== myDelay ========

\* Assembly function to delay. Decrements the count until it is zero

\* The exact duration depends on the processor speed.

\*/

**\_\_asm**(" .sect \".text:myDelay\"\n"

" .clink\n"

" .thumbfunc myDelay\n"

" .thumb\n"

" .global myDelay\n"

"myDelay:\n"

" subs r0, #1\n"

" bne.n myDelay\n"

" bx lr\n");

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

Graphical user interface, text, application

Description automatically generated

1. Declaration

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

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

-Elmer Mejia