CPE403 – Advanced Embedded Systems

Design Assignment 4

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name:Jeb Marinas

Email:marinj4@unlv.nevada.edu

Github Repository link (root): https://github.com/jebmarinas/Micro_projects

Youtube Playlist link (root):

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.

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.

```
/*
 * Copyright (c) 2015-2019, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/*
 *  ======== mutex.c ========
 */

 //XDC module Headers /
#include <xdc/std.h>
#include <xdc/runtime/System.h>

// BIOS module Headers /
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Clock.h>
#include <ti/sysbios/knl/Task.h>
#include <ti/sysbios/knl/Semaphore.h>
#include <ti/drivers/Board.h>
#include <stddef.h>

//Driver Header files /
#include <ti/drivers/GPIO.h>
```

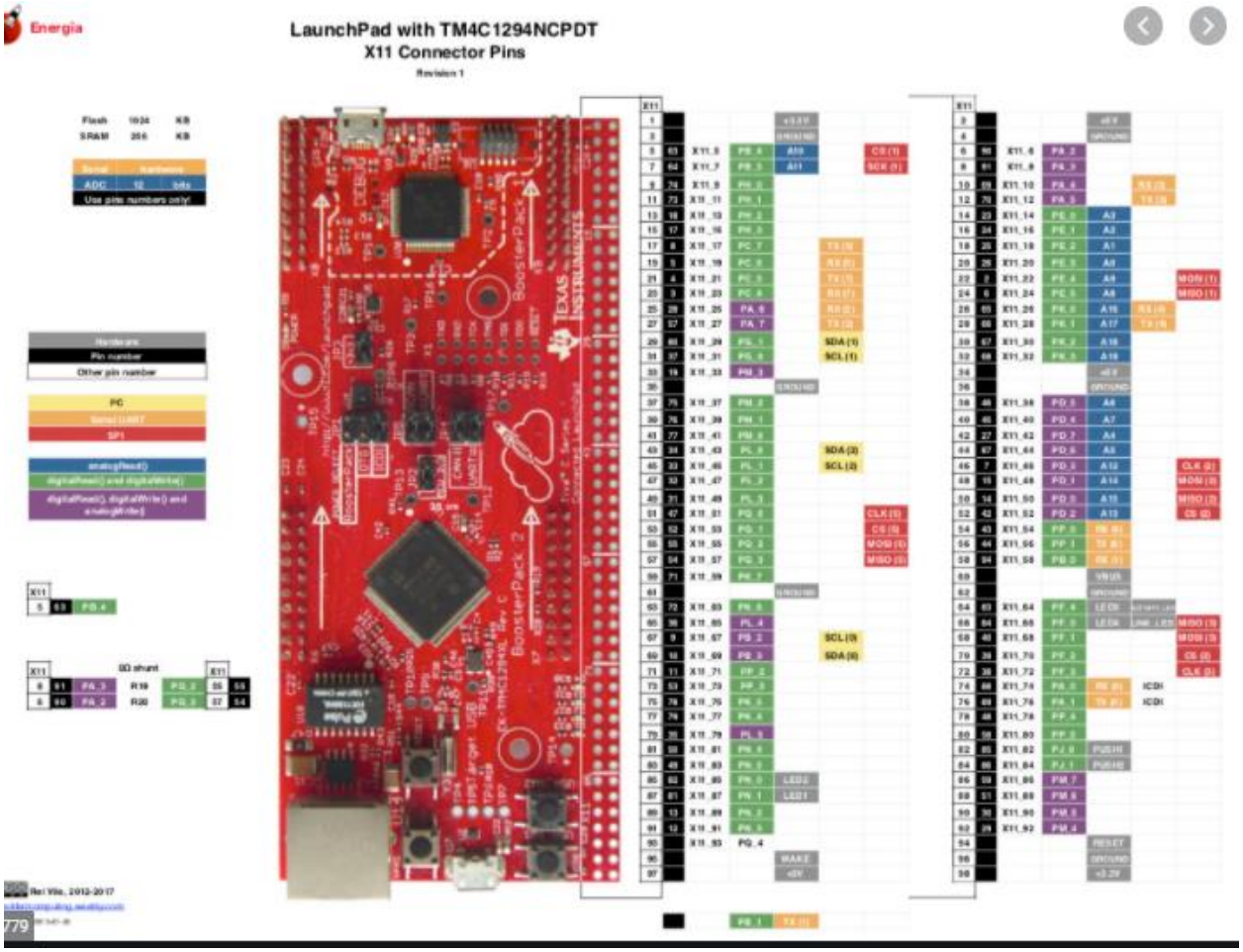
```
52. #include <ti/drivers/Timer.h>
53. #include <ti/drivers/ADC.h>
54. #include <ti/drivers/UART.h>
55. #include <ti/drivers/PWM.h>
56.
57. //Board Header file
58. #include "ti_drivers_config.h"
59. #include <xdc/cfg/global.h>
60.
61.
62. #include <stdio.h>
63. #include <string.h>
64.
65. #define TASKSTACKSIZE   512
66.
67. Void task1Fxn(UArg arg0, UArg arg1);
68. Void task2Fxn(UArg arg0, UArg arg1);
69.
70. Int resource = 0;
71. Int finishCount = 0;
72. UInt32 sleepTickCount;
73. UInt32 global_timer = 0;
74.
75. Task_Struct task1Struct, task2Struct;
76. Char task1Stack[TASKSTACKSIZE], task2Stack[TASKSTACKSIZE];
77. Semaphore_Struct semStruct;
78. Semaphore_Handle semHandle;
79.
80. Void heartBeatFxn(UArg arg0, UArg arg1)
81. {
82.     while (1) {
83.         Task_sleep(500000 / Clock_tickPeriod);
84.         GPIO_toggle(CONFIG_LED_0);
85.     }
86. }
87.
88. /*
89.  *  ======== main ========
90.  */
91. void ConfigurePWM(void) // this will generate pwm signal
92. {
93.
94.
95.
96. }
97.
98. void software_interrupt_timer(void)
99. {
100.
101.          global_timer++;
102.          if(global_timer == 5)
103.          {
104.              Semaphore_post(adc_trigger);
105.          }
106.          else if(global_timer == 10)
```

```
        {
            Semaphore_post(uart_trigger);
        }
        else if(global_timer == 15);
        {
            Semaphore_post(pwm_trigger);
            global_timer = 0;
        }
    }
    void adc_handler(void)
    {
        while(1)
        {
            Semaphore_pend(adc_trigger,BIOS_WAIT_FOREVER);

        }
    }
    void uart_handler(void)
    {
        while(1)
        {
            Semaphore_pend(uart_trigger,BIOS_WAIT_FOREVER);
        }
    }
    void pwm_handler(void)
    {
        while(1)
        {
            Semaphore_pend(pwm_trigger,BIOS_WAIT_FOREVER);
        }
    }
    void ConfigureUART(void){


    }

    void ConfigureADC(void){




    }
    void timerCallback(Timer_Handle myHandle, int_fast16_t status)
    {
        global_timer++;
            if(global_timer == 5)
            {
                Semaphore_post(adc_trigger);
            }
            else if(global_timer == 10)
            {
                Semaphore_post(uart_trigger);
            }
            else if(global_timer == 15);
            {
```

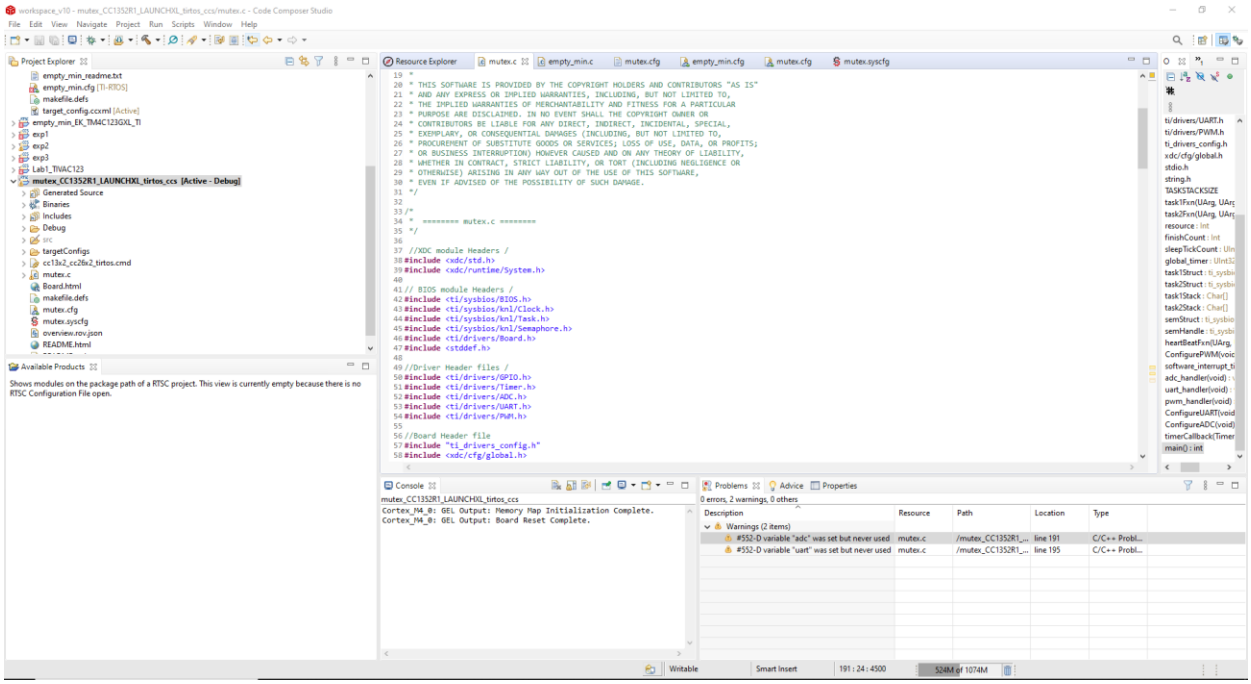
```
                Semaphore_post(pwm_trigger);
                global_timer = 0;
            }
    }
    /*
     *  ======== main ========
     */
    int main()
    {
            Timer_Handle timer0;
            Timer_Params par1;
            /*
             * PWM_init();

        PWM_Params_init(&params);
        params.dutyUnits = PWM_DUTY_US;
        params.dutyValue = 0;
        params.periodUnits = PWM_PERIOD_US;
        params.periodValue = pwmPeriod;
        pwm1 = PWM_open(CONFIG_PWM_0, &params);
             */

            /* Call driver init functions */
            Board_init();
            GPIO_init();
            Timer_init();
            PWM_init();
            ADC_init();
            UART_init();

            ADC_Handle adc;
            ADC_Params par2;
            PWM_Handle pwm;
            PWM_Params par3;
            UART_Handle uart;
            UART_Params par4;


            ADC_Params_init(&par2);
            adc = ADC_open(CONFIG_ADC_0, &par2);
            PWM_Params_init(&par3);
            par3.idleLevel = PWM_IDLE_LOW;
            par3.periodUnits = PWM_PERIOD_US;
            par3.periodValue = 1000;
            par3.dutyUnits = PWM_DUTY_FRACTION;
            par3.dutyValue = 0;
            pwm = PWM_open(CONFIG_LED_1_PWM,&par3);
            PWM_start(pwm);
            UART_Params_init(&par4);
            par4.writeDataMode = UART_DATA_BINARY;
            par4.readDataMode = UART_DATA_BINARY;
            par4.readReturnMode = UART_RETURN_FULL;
            par4.baudRate = 115200;
            uart = UART_open(CONFIG_UART_0, &par4);

```

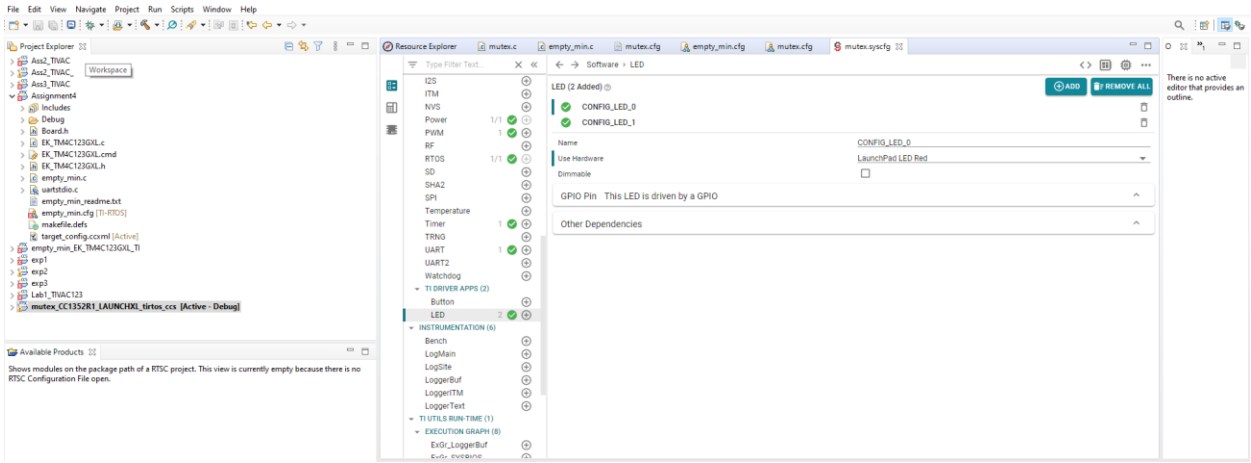
```
217.
218.             /*
219.              * Setting up the timer in continuous callback mode that calls
    the callback
220.              * function every 1,000,000 microseconds, or 1 second.
221.              */
222.             Timer_Params_init(&par1);
223.             par1.period = 1000000;
224.             par1.periodUnits = Timer_PERIOD_US;
225.             par1.timerMode = Timer_CONTINUOUS_CALLBACK;
226.             par1.timerCallback = timerCallback;
227.
228.             timer0 = Timer_open(CONFIG_TIMER_0, &par1);
229.
230.             if (timer0 == NULL) {
231.                 /* Failed to initialized timer */
232.                 while (1) {}
233.             }
234.
235.             if (Timer_start(timer0) == Timer_STATUS_ERROR) {
236.                 /* Failed to start timer */
237.                 while (1) {}
238.             }
239.
240.             BIOS_start();
241.     }
242.
243.     /*
244.      *  ======== task1Fxn ========
245.      */
246.
247.
```

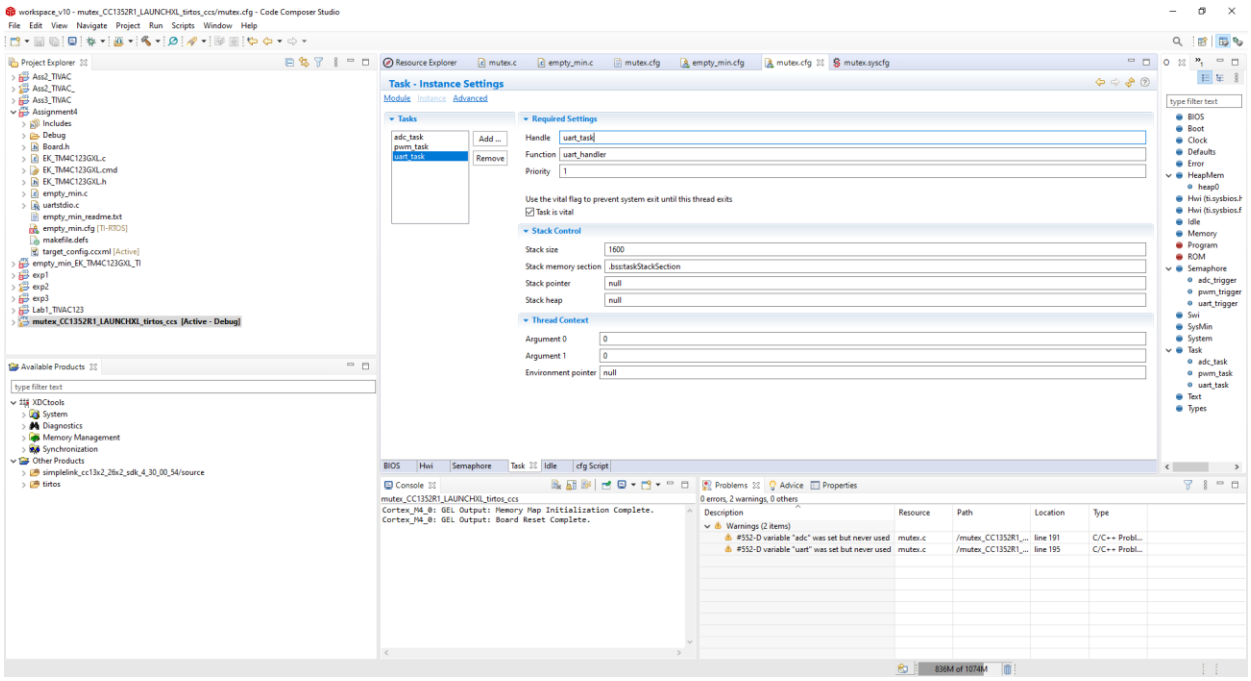
248. Block diagram and/or Schematics showing the components, pins used, and interface.

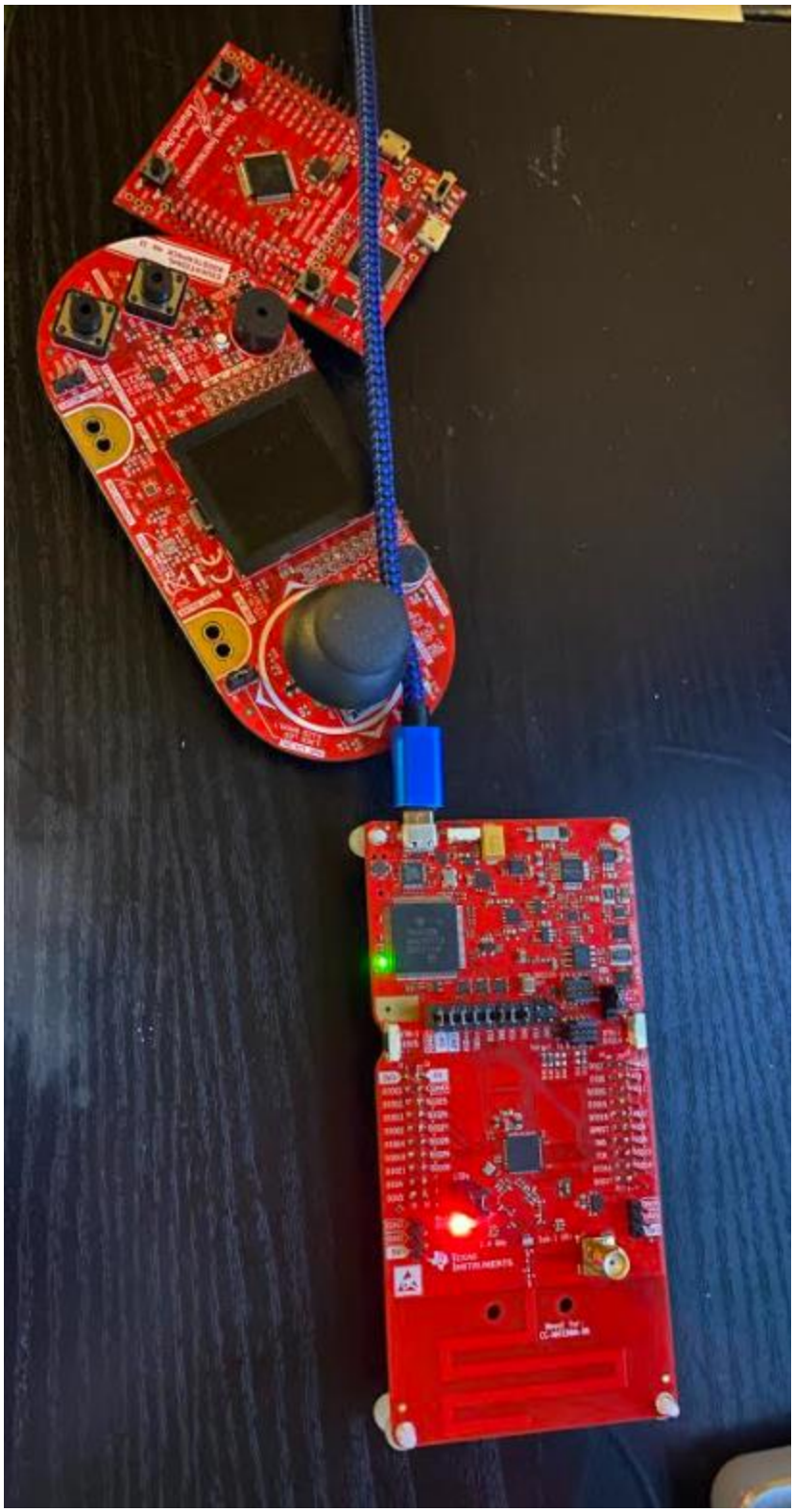


249. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.






workspace_v10 - mutex_CC1352R1_LAUNCHXL_tirtos_ccs/mutex.cfg - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Project Explorer
Ass2_TIVAC
Ass2_TIVAC_
Ass3_TIVAC
Assignment4
Includes
Debug
Board.h
EK_TM4C123GXL.c
EK_TM4C123GXL.cmd
EK_TM4C123GXL.h
empty_min.c
uartstdio.c
empty_min_readme.txt
empty_min.cfg [TI-RTOS]
makefile.defs
target_config.ccxml [Active]
empty_min_EK_TM4C123GXL_TI
exp1
exp2
exp3
Lab1_TIVAC123
mutex_CC1352R1_LAUNCHXL_tirtos_ccs [Active - Debug]
Available Products
type filter text
XDCtools
System
Diagnostics
Memory Management
Synchronization
Other Products
simplelink_cc13x2_26x2_sdk_4_30_00_54/source
tirtos
Resource Explorer
mutex.c
empty_min.c
mutex.cfg
empty_min.cfg
mutex.cfg
mutex.syscfg
Task - Instance Settings
Module Instance Advanced
Tasks
adc_task
pwm_task
uart_task
Add ...
Remove
Required Settings
Handle uart_task
Function uart_handler
Priority 1
Use the vital flag to prevent system exit until this thread exits
Task is vital
Stack Control
Stack size 1600
Stack memory section .bss:taskStackSection
Stack pointer null
Stack heap null
Thread Context
Argument 0 0
Argument 1 0
Environment pointer null
BIOS Hwi Semaphore Task Idle cfg Script
type filter text
BIOS
Boot
Clock
Defaults
Error
HeapMem
heap0
Hwi (ti.sysbios.h
Hwi (ti.sysbios.f
Idle
Memory
Program
ROM
Semaphore
adc_trigger
pwm_trigger
uart_trigger
Swi
SysMin
System
Task
adc_task
pwm_task
uart_task
Text
Types
Console
mutex_CC1352R1_LAUNCHXL_tirtos_ccs
Cortex_M4_0: GEL Output: Memory Map Initialization Complete.
Cortex_M4_0: GEL Output: Board Reset Complete.
Problems
Advice
Properties
0 errors, 2 warnings, 0 others
Description
Resource
Path
Location
Type
Warnings (2 items)
#552-D variable "adc" was set but never used
mutex.c
/mutex_CC1352R1_...
line 191
C/C++ Probl...
#552-D variable "uart" was set but never used
mutex.c
/mutex_CC1352R1_...
line 195
C/C++ Probl...
836M of 1074M



Declaration

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

"This assignment submission is my own, original work".
Name of the Student
Jeb Marinas