CPE403 – Advanced Embedded Systems

# Design Assignment 4

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name:Jeb Marinas

Email:marinj4@unlv.nevada.edu

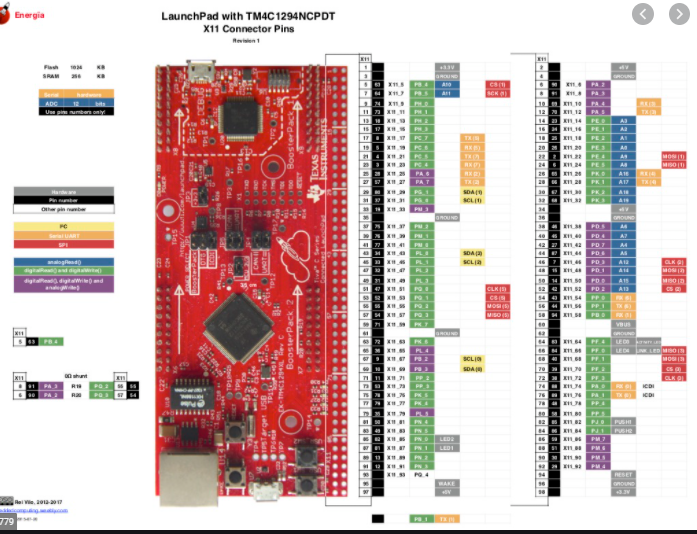
Github Repository link (root): https://github.com/jebmarinas/Micro\_projects

Youtube Playlist link (root):

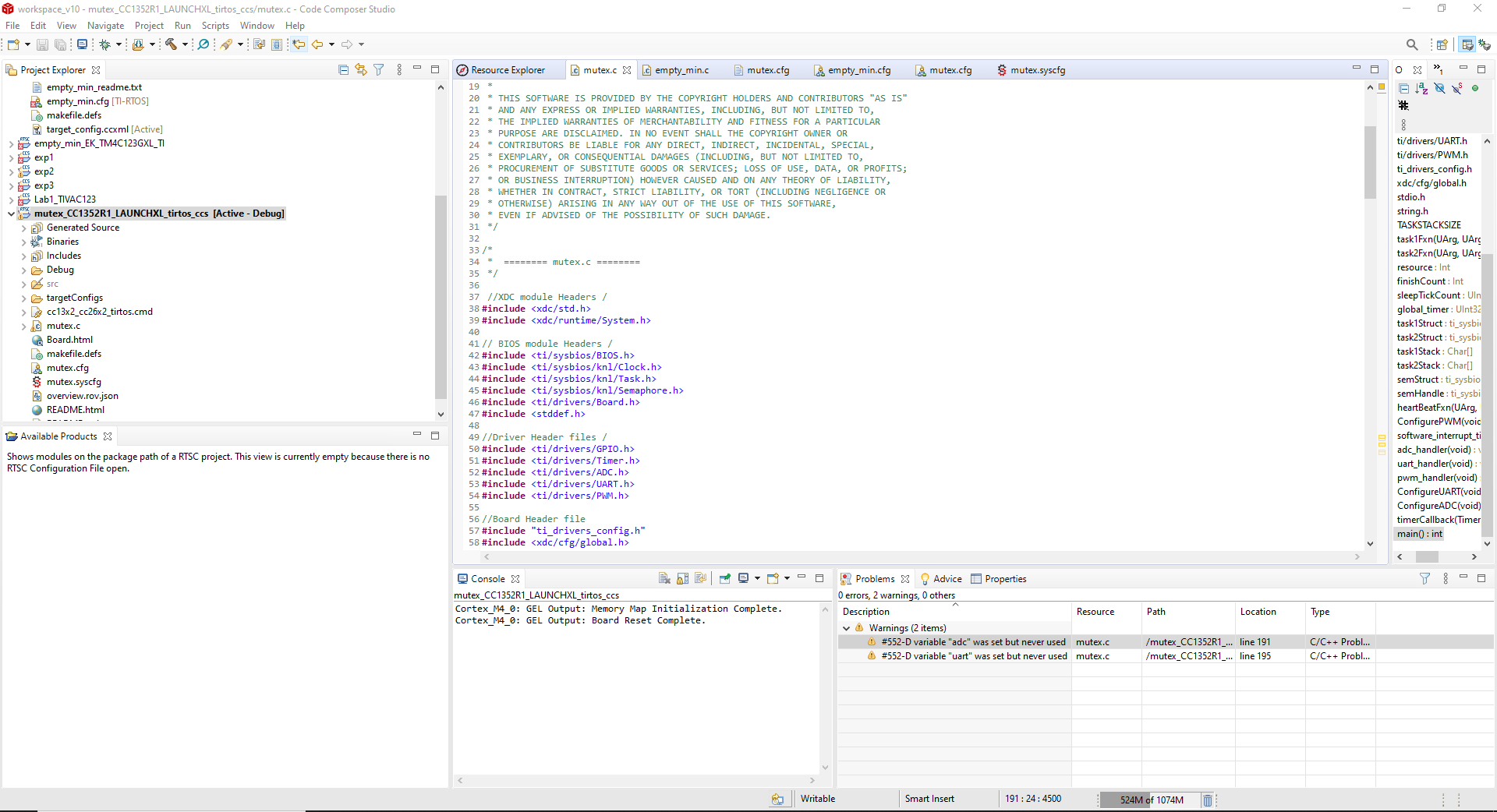
**Follow the submission guideline to be awarded points for this Assignment.**

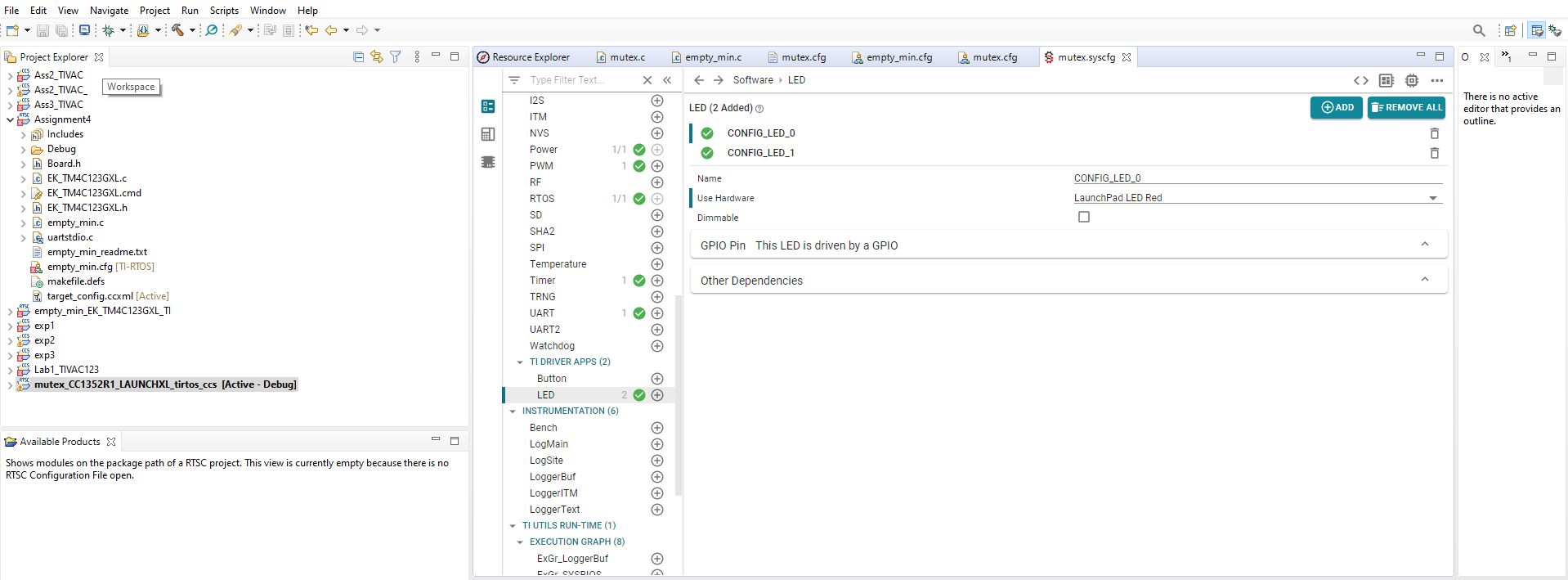
Submit the following for all Assignments:

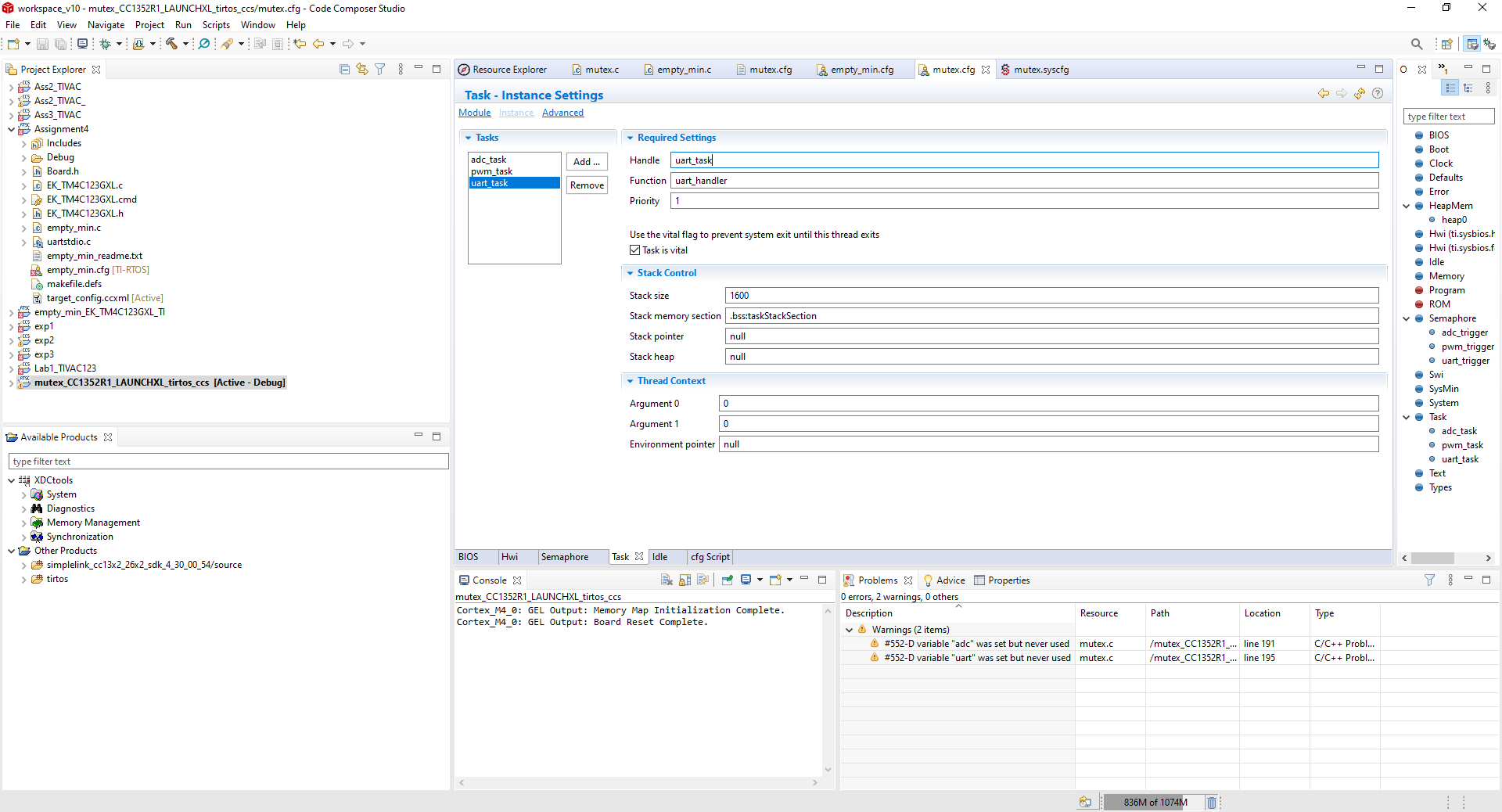
1. In the document, for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only.
2. Create a private Github repository with a random 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.
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.
9. /\*
10. \* Copyright (c) 2015-2019, Texas Instruments Incorporated
11. \* All rights reserved.
12. \*
13. \* Redistribution and use in source and binary forms, with or without
14. \* modification, are permitted provided that the following conditions
15. \* are met:
16. \*
17. \* \* Redistributions of source code must retain the above copyright
18. \* notice, this list of conditions and the following disclaimer.
19. \*
20. \* \* Redistributions in binary form must reproduce the above copyright
21. \* notice, this list of conditions and the following disclaimer in the
22. \* documentation and/or other materials provided with the distribution.
23. \*
24. \* \* Neither the name of Texas Instruments Incorporated nor the names of
25. \* its contributors may be used to endorse or promote products derived
26. \* from this software without specific prior written permission.
27. \*
28. \* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
29. \* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
30. \* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
31. \* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
32. \* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
33. \* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
34. \* PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
35. \* OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
36. \* WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
37. \* OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
38. \* EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
39. \*/
40. /\*
41. \* ======== mutex.c ========
42. \*/
43. //XDC module Headers /
44. **#include** <xdc/std.h>
45. **#include** <xdc/runtime/System.h>
46. // BIOS module Headers /
47. **#include** <ti/sysbios/BIOS.h>
48. **#include** <ti/sysbios/knl/Clock.h>
49. **#include** <ti/sysbios/knl/Task.h>
50. **#include** <ti/sysbios/knl/Semaphore.h>
51. **#include** <ti/drivers/Board.h>
52. **#include** <stddef.h>
53. //Driver Header files /
54. **#include** <ti/drivers/GPIO.h>
55. **#include** <ti/drivers/Timer.h>
56. **#include** <ti/drivers/ADC.h>
57. **#include** <ti/drivers/UART.h>
58. **#include** <ti/drivers/PWM.h>
59. //Board Header file
60. **#include** "ti\_drivers\_config.h"
61. **#include** <xdc/cfg/global.h>
62. **#include** <stdio.h>
63. **#include** <string.h>
64. **#define** TASKSTACKSIZE 512
65. Void **task1Fxn**(UArg arg0, UArg arg1);
66. Void **task2Fxn**(UArg arg0, UArg arg1);
67. Int resource = 0;
68. Int finishCount = 0;
69. UInt32 sleepTickCount;
70. UInt32 global\_timer = 0;
71. Task\_Struct task1Struct, task2Struct;
72. Char task1Stack[TASKSTACKSIZE], task2Stack[TASKSTACKSIZE];
73. Semaphore\_Struct semStruct;
74. Semaphore\_Handle semHandle;
75. Void **heartBeatFxn**(UArg arg0, UArg arg1)
76. {
77. **while** (1) {
78. Task\_sleep(500000 / Clock\_tickPeriod);
79. **GPIO\_toggle**(CONFIG\_LED\_0);
80. }
81. }
82. /\*
83. \* ======== main ========
84. \*/
85. **void** **ConfigurePWM**(**void**) // this will generate pwm signal
86. {
87. }
88. **void** **software\_interrupt\_timer**(**void**)
89. {
90. global\_timer++;
91. **if**(global\_timer == 5)
92. {
93. Semaphore\_post(adc\_trigger);
94. }
95. **else** **if**(global\_timer == 10)
96. {
97. Semaphore\_post(uart\_trigger);
98. }
99. **else** **if**(global\_timer == 15);
100. {
101. Semaphore\_post(pwm\_trigger);
102. global\_timer = 0;
103. }
104. }
105. **void** **adc\_handler**(**void**)
106. {
107. **while**(1)
108. {
109. Semaphore\_pend(adc\_trigger,BIOS\_WAIT\_FOREVER);
110. }
111. }
112. **void** **uart\_handler**(**void**)
113. {
114. **while**(1)
115. {
116. Semaphore\_pend(uart\_trigger,BIOS\_WAIT\_FOREVER);
117. }
118. }
119. **void** **pwm\_handler**(**void**)
120. {
121. **while**(1)
122. {
123. Semaphore\_pend(pwm\_trigger,BIOS\_WAIT\_FOREVER);
124. }
125. }
126. **void** **ConfigureUART**(**void**){
127. }
128. **void** **ConfigureADC**(**void**){
129. }
130. **void** **timerCallback**(Timer\_Handle myHandle, int\_fast16\_t status)
131. {
132. global\_timer++;
133. **if**(global\_timer == 5)
134. {
135. Semaphore\_post(adc\_trigger);
136. }
137. **else** **if**(global\_timer == 10)
138. {
139. Semaphore\_post(uart\_trigger);
140. }
141. **else** **if**(global\_timer == 15);
142. {
143. Semaphore\_post(pwm\_trigger);
144. global\_timer = 0;
145. }
146. }
147. /\*
148. \* ======== main ========
149. \*/
150. **int** **main**()
151. {
152. Timer\_Handle timer0;
153. Timer\_Params par1;
154. /\*
155. \* PWM\_init();
156. PWM\_Params\_init(&params);
157. params.dutyUnits = PWM\_DUTY\_US;
158. params.dutyValue = 0;
159. params.periodUnits = PWM\_PERIOD\_US;
160. params.periodValue = pwmPeriod;
161. pwm1 = PWM\_open(CONFIG\_PWM\_0, &params);
162. \*/
163. /\* Call driver init functions \*/
164. Board\_init();
165. **GPIO\_init**();
166. **Timer\_init**();
167. **PWM\_init**();
168. **ADC\_init**();
169. **UART\_init**();
170. ADC\_Handle adc;
171. ADC\_Params par2;
172. PWM\_Handle pwm;
173. PWM\_Params par3;
174. UART\_Handle uart;
175. UART\_Params par4;
176. **ADC\_Params\_init**(&par2);
177. adc = **ADC\_open**(CONFIG\_ADC\_0, &par2);
178. **PWM\_Params\_init**(&par3);
179. par3.idleLevel = *PWM\_IDLE\_LOW*;
180. par3.periodUnits = *PWM\_PERIOD\_US*;
181. par3.periodValue = 1000;
182. par3.dutyUnits = *PWM\_DUTY\_FRACTION*;
183. par3.dutyValue = 0;
184. pwm = **PWM\_open**(CONFIG\_LED\_1\_PWM,&par3);
185. **PWM\_start**(pwm);
186. **UART\_Params\_init**(&par4);
187. par4.writeDataMode = *UART\_DATA\_BINARY*;
188. par4.readDataMode = *UART\_DATA\_BINARY*;
189. par4.readReturnMode = *UART\_RETURN\_FULL*;
190. par4.baudRate = 115200;
191. uart = **UART\_open**(CONFIG\_UART\_0, &par4);
192. /\*
193. \* Setting up the timer in continuous callback mode that calls the callback
194. \* function every 1,000,000 microseconds, or 1 second.
195. \*/
196. **Timer\_Params\_init**(&par1);
197. par1.period = 1000000;
198. par1.periodUnits = *Timer\_PERIOD\_US*;
199. par1.timerMode = *Timer\_CONTINUOUS\_CALLBACK*;
200. par1.timerCallback = timerCallback;
201. timer0 = **Timer\_open**(CONFIG\_TIMER\_0, &par1);
202. **if** (timer0 == NULL) {
203. /\* Failed to initialized timer \*/
204. **while** (1) {}
205. }
206. **if** (**Timer\_start**(timer0) == Timer\_STATUS\_ERROR) {
207. /\* Failed to start timer \*/
208. **while** (1) {}
209. }
210. BIOS\_start();
211. }
212. /\*
213. \* ======== task1Fxn ========
214. \*/
215. Block diagram and/or Schematics showing the components, pins used, and interface.

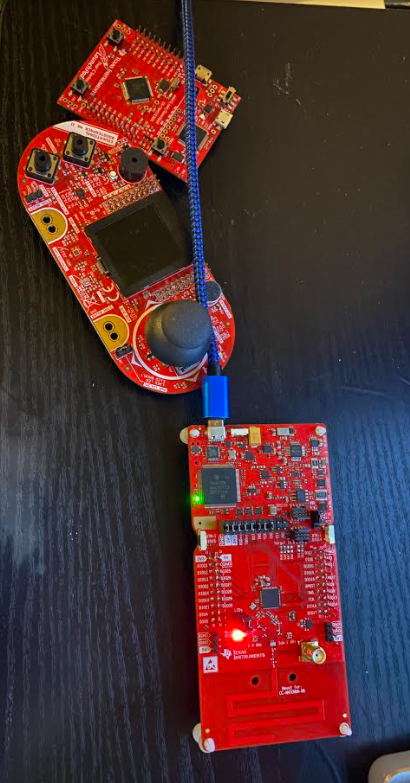


1. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.









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