CPE403 – Advanced Embedded Systems

# Design Assignment 3

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Joe Uesato

Email: uesatj1@unlv.nevada.edu

Github Repository link (root): https://github.com/joeuesato/lab\_submissions

Youtube Playlist: https://www.youtube.com/playlist?list=PLSBOvuRedzOf8JAhpVx0VsSteisJQUKv3

**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. /\* XDC module Headers \*/
10. **#include** <xdc/std.h>
11. **#include** <xdc/runtime/System.h>
12. **#include** <xdc/runtime/Log.h> //needed for any Log\_info() call
13. **#include** <xdc/cfg/global.h> //header file for statically defined objects/handles
14. **#include** <xdc/runtime/Diags.h>
15. /\* BIOS module Headers \*/
16. **#include** <ti/sysbios/BIOS.h>
17. **#include** <ti/sysbios/knl/Clock.h>
18. **#include** <ti/sysbios/knl/Task.h>
19. **#include** <ti/sysbios/knl/Semaphore.h>
20. /\* Board Header file \*/
21. **#include** "Board.h"
22. /\* Include header files for adc and GPIO functions \*/
23. **#include** <stdint.h>
24. **#include** <stdbool.h>
25. **#include** "inc/hw\_memmap.h"
26. **#include** "inc/hw\_types.h"
27. **#include** "inc/tm4c123gh6pm.h"
28. **#include** "driverlib/gpio.h"
29. **#include** "driverlib/pin\_map.h"
30. **#include** "driverlib/sysctl.h"
31. **#include** "driverlib/uart.h"
32. **#include** "driverlib/interrupt.h"
33. **#include** "driverlib/timer.h"
34. **#include** "driverlib/debug.h"
35. **#include** "driverlib/adc.h"
36. **#include** "driverlib/rom.h"
37. **#include** "utils/uartstdio.h"
38. **#include** <time.h>
39. **#include** <inc/hw\_gpio.h>
40. **volatile** int16\_t i16ToggleCount1 = 0;
41. **volatile** int16\_t i16ToggleCount2 = 0;
42. **void** **timer2Init**();
43. **void** **ADCInit**();
44. **void** **UARTInit**();
45. //---------------------------------------------------------------------------
46. **void** **delay\_simple**(**void**)
47. {
48. **SysCtlDelay**(6700000); // creates ~500ms delay - TivaWare fxn
49. }
50. **void** **taskFxn1**(**void**)
51. {
52. **while**(1)
53. {
54. Semaphore\_pend(task1sem, BIOS\_WAIT\_FOREVER);
55. // LED values - 2=RED, 4=BLUE, 8=GREEN
56. **if**(**GPIOPinRead**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2))
57. {
58. **GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0);
59. }
60. **else**
61. {
62. **GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4);
63. }
64. // delay\_simple(); // create a delay of ~1/2sec
65. i16ToggleCount1 += 1; // keep track of #toggles
66. Log\_info1("LED2 TOGGLED [%u] times", i16ToggleCount1); // send #toggles to Log Display
67. //System\_printf("Count: %d\n", i16ToggleCount);
68. //System\_flush();
69. }
70. }
71. **void** **taskFxn2**(**void**)
72. {
73. **while**(1)
74. {
75. Semaphore\_pend(task2sem, BIOS\_WAIT\_FOREVER);
76. // LED values - 2=RED, 4=BLUE, 8=GREEN
77. **if**(**GPIOPinRead**(GPIO\_PORTF\_BASE, GPIO\_PIN\_3))
78. {
79. **GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0);
80. }
81. **else**
82. {
83. **GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, 8);
84. }
85. // delay\_simple(); // create a delay of ~1/2sec
86. i16ToggleCount2 += 1; // keep track of #toggles
87. Log\_info1("LED3 TOGGLED [%u] times", i16ToggleCount2); // send #toggles to Log Display
88. //System\_printf("Count: %d\n", i16ToggleCount);
89. //System\_flush();
90. }
91. }
92. **void** **taskFxn3**(**void**)
93. {
94. **while**(1)
95. {
96. Semaphore\_pend(task3sem, BIOS\_WAIT\_FOREVER);
97. }
98. }
99. **void** **taskFxn4**(**void**)
100. {
101. **while**(1)
102. {
103. Semaphore\_pend(task4sem, BIOS\_WAIT\_FOREVER);
104. }
105. }
106. **volatile** uint32\_t tickCount=0;
107. /\*
108. \* ======== main ========
109. \*/
110. **int** **main**()
111. {
112. /\* Set up the System Clock \*/
113. **SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);
114. /\* Enable all the peripherals \*/
115. **SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF);
116. /\* Unlock pin PF0 \*/
117. HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_LOCK)= GPIO\_LOCK\_KEY;
118. HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_CR) |= 0x01;
119. HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_LOCK)= 0;
120. /\* Configure Enable pin as output \*/
121. **GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);
122. **GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2);
123. **GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_3);
124. timer2Init();
125. UARTInit();
126. ADCInit();
127. // Initial message to terminal display
128. **UARTprintf**("Program Running...\n");
129. BIOS\_start(); /\* Does not return \*/
130. }
131. //---------------------------------------------------------------------------
132. // Timer ISR to be called by BIOS Hwi
133. //
134. // Posts Semaphore for releasing tasks
135. //---------------------------------------------------------------------------
136. **void** **Timer\_ISR**(**void**)
137. {
138. **TimerIntClear**(TIMER2\_BASE, TIMER\_TIMA\_TIMEOUT); // must clear timer flag FROM timer
139. tickCount++; //tickCount is incremented every 2 ms.
140. **if**(tickCount == 300)
141. {
142. Semaphore\_post(task1sem);
143. }
144. **else** **if**(tickCount == 600)
145. {
146. Semaphore\_post(task2sem);
147. }
148. **else** **if**(tickCount == 900)
149. {
150. Semaphore\_post(task3sem);
151. tickCount = 0;
152. }
153. **else** **if**(tickCount == 1200)
154. {
155. Semaphore\_post(task4sem);
156. }
157. }
158. //---------------------------------------------------------------------------
159. // Initialization Functions
160. //---------------------------------------------------------------------------
161. //Timer 2 setup
162. **void** **timer2Init**()
163. {
164. uint32\_t ui32Period;
165. **SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER2); // enable Timer 2 periph clks
166. **TimerConfigure**(TIMER2\_BASE, TIMER\_CFG\_PERIODIC); // cfg Timer 2 mode - periodic
167. ui32Period = (**SysCtlClockGet**()/500) / 2;
168. **TimerLoadSet**(TIMER2\_BASE, TIMER\_A, ui32Period-1); // set Timer 2 period
169. **TimerIntEnable**(TIMER2\_BASE, TIMER\_TIMA\_TIMEOUT); // enables Timer 2 to interrupt CPU
170. **TimerEnable**(TIMER2\_BASE, TIMER\_A); // enable Timer 2
171. }
172. // ADC0 Setup
173. **void** **ADCInit**()
174. {
175. **SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOD);
176. // Configure ADC
177. **SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0); // Enable ADC0
178. **ADCHardwareOversampleConfigure**(ADC0\_BASE, 32);
179. // ADCSequenceConfigure(ADCModuleAddress, SequencerNumber, TriggerSource, Priority)
180. **ADCSequenceConfigure**(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0); // Changed to sequencer #2
181. // Configure each sample - Sequencer 2 takes four samples
182. // ADCSequenceStepConfigure(ADCModuleAddress, SequencerNumber, StepNumber, Channel)
183. // ADC\_CTL\_TS: Internal temp. sensor channel, use ADC\_CTL\_CH0 - ADC\_CTL\_CH11 for inputs
184. // ADC\_CTL\_IE: Generate an interrupt when sequencer is done
185. // ADC\_CTL\_END: Defines the step as the last step of the sequence
186. **ADCSequenceStepConfigure**(ADC0\_BASE, 2, 0, ADC\_CTL\_CH4);
187. **ADCSequenceStepConfigure**(ADC0\_BASE, 2, 1, ADC\_CTL\_CH4);
188. **ADCSequenceStepConfigure**(ADC0\_BASE, 2, 2, ADC\_CTL\_CH4);
189. **ADCSequenceStepConfigure**(ADC0\_BASE, 2, 3, ADC\_CTL\_CH4|ADC\_CTL\_IE|ADC\_CTL\_END);
190. **ADCSequenceEnable**(ADC0\_BASE, 2);
191. }
192. // UART Setup
193. **void** **UARTInit**()
194. {
195. **SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0);
196. // Configure pins for UART
197. **GPIOPinConfigure**(GPIO\_PA0\_U0RX);
198. **GPIOPinConfigure**(GPIO\_PA1\_U0TX);
199. **GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);
200. **UARTClockSourceSet**(UART0\_BASE, UART\_CLOCK\_PIOSC);
201. **UARTStdioConfig**(0, 115200, 16000000);
202. }
203. Block diagram and/or Schematics showing the components, pins used, and interface.

TivaC with MKII plugged in, no other connections

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

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

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

Joe Uesato