Github: https://github.com/Ber-geb/solid-octo-tribble.git

CPE 403 ADV EMB SYS DES F 2019

TITLE: TIRTOS TIVAC Assignment

Youtube Link: https://youtu.be/9pXOSxHbD2Y

GOAL:

- Create ADC task to run every 10th instance of HWI
- Create UART diplay task to run every 20th instance of HWI
- Create Switch/Read Task to run every 30th instance of HWI
- Repeat the process above every 30 ms

DELIVERABLES:

The project will show an LED that is affected by the PWM signal that takes the ADC value generated every 10th instance of the HWI. Also, a terminal will be shown that is connected to the same port as the TIVAC TM4C123GH6PM MCU to show the UART signals being transmitted/received. Every time SW0/SW1 is pressed, the duty cycle will change. Since the period of the PWM is very small, a logic analyzer will be shown which will convey when the switch is pressed to affect the PWM signal of the LED.

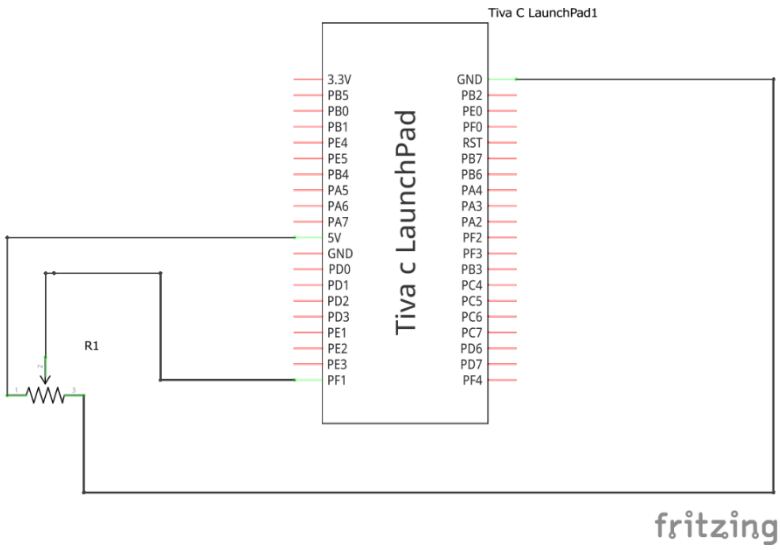
COMPONENTS:

TIVAC TM4C123GH6PM MCU

- Logic Analyzer
- Jumper Wires
- Potentiometer

SCHEMATICS:

Github: https://github.com/Ber-geb/solid-octo-tribble.git



Above is a picture of the execution graph. Due to the sampling of the execution graph being little more than a millisecond, the graph was only able to capture one instance of a task (which is the switch read task denoted by SRfun). The execution graph was able to show the distance between two hardware interrupts which was about 1ms.

Github: https://github.com/Ber-geb/solid-octo-tribble.git

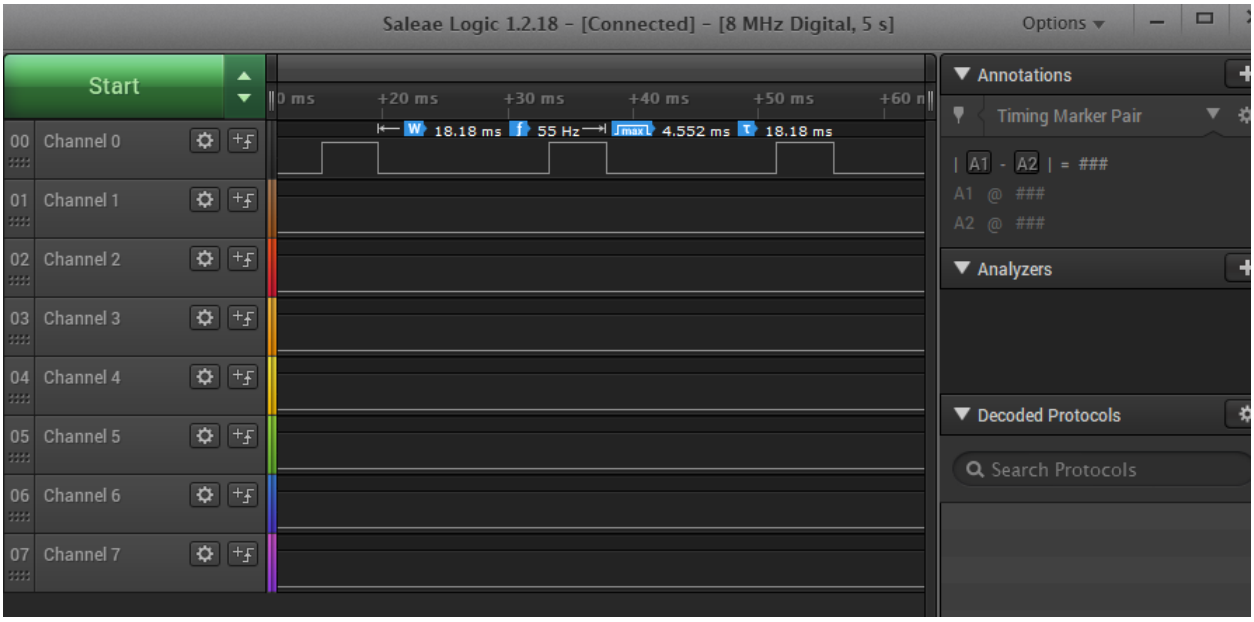
IIMPLEMENTATION:

UART and GPIO will be initialized as well as the ADC. This is the major initializations made for the assignment. The code below will show these initializations.

Picture of the terminal displaying ADC value:

Picture of the Logic Analyzer displaying the LED PWM:

Github: https://github.com/Ber-geb/solid-octo-tribble.git



Github: https://github.com/Ber-geb/solid-octo-tribble.git

CODE:

Github: https://github.com/Ber-geb/solid-octo-tribble.git

```
/*
 * Copyright (c) 2015, 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.
// */
//
///*
// *  ======== empty.c ========
// */
//
////---------------------------------------
//// Prototypes
////---------------------------------------
//void hardware_init(void);
//
//
//#include <stdbool.h>
//
////---------------------------------------
//// BIOS header files
////---------------------------------------
//#include <xdc/std.h>                        //mandatory - have to include first, for BIOS types
//#include <ti/sysbios/BIOS.h>                //mandatory - if you call APIs like BIOS_start()
//#include <xdc/runtime/Log.h>                //needed for any Log_info() call
//#include <xdc/cfg/global.h>                 //header file for statically defined objects/handles
//
//
///* XDCtools Header files */
//#include <xdc/runtime/System.h>
//
```

Github: https://github.com/Ber-geb/solid-octo-tribble.git

```
///* BIOS Header files */
//#include <ti/sysbios/knl/Task.h>
//
///* TI-RTOS Header files */
//#include <ti/drivers/GPIO.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"
//
//#include "driverlib/adc.h"
//#include "inc/hw_memmap.h"
//#include "driverlib/sysctl.h"
//#include "driverlib/timer.h"
//#include "driverlib/interrupt.h"
//
//
//#define TASKSTACKSIZE   512
//
//Task_Struct task0Struct;
//Char task0Stack[TASKSTACKSIZE];
//
///*
// *  ======== 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_LED0);
//    }
//}
//
///*
// *  ======== main ========
// */

//----------------------------------------
// BIOS header files
//----------------------------------------
#include <xdc/std.h>                        //mandatory - have to include first, for BIOS types
#include <ti/sysbios/BIOS.h>                //mandatory - if you call APIs like BIOS_start()
#include <xdc/runtime/Log.h>                //needed for any Log_info() call
#include <xdc/cfg/global.h>                 //header file for statically defined objects/handles


//----------------------------------------
// TivaWare Header Files
//----------------------------------------
```

Github: https://github.com/Ber-geb/solid-octo-tribble.git

```
113  #include <stdint.h>
114  #include <stdbool.h>
115
116  #include "inc/hw_types.h"
117  #include "inc/hw_memmap.h"
118  #include "driverlib/sysctl.h"
119  #include "driverlib/gpio.h"
120  #include "inc/hw_ints.h"
121  #include "driverlib/interrupt.h"
122  #include "driverlib/timer.h"
123  #include "driverlib/adc.h"
124  #include "utils/uartstdio.h"
125  #include "driverlib/uart.h"
126  #include "driverlib/pin_map.h"
127  #include "driverlib/pwm.h"
128
129  //---------------------------------------
130  // Prototypes
131  //---------------------------------------
132  void hardware_init(void);
133  void ledToggle(void);
134  void TIMER_ISR(void);
135  void TIMER2INT(void);
136  void ADCfun(void);
137  void SRfun(void);
138  void UARTfun(void);
139  void reverse(char[], int);
140  char* itoa(int,char*,int);
141  void InitConsole(void);
142
143  #define PWM_FREQUENCY 55
144
145  volatile int16_t i16ToggleCount;
146  uint32_t ui32ADC0Value[4];
147  uint32_t ADCAvg;
148  volatile bool buttonPressed;
149  volatile uint32_t ui32Load;
150  volatile uint32_t ui32PWMClock;
151  //volatile uint32_t ui8Adjust = 83;
152
153  int main(void)
154  {
155      //    Task_Params taskParams;
156      //
157      //    /* Call board init functions */
158      //    Board_initGeneral();
159      //    //Board_initGPIO();
160      //    // Board_initI2C();
161      //    // Board_initSDSPI();
162      //    // Board_initSPI();
163      //    //Board_initUART();
164      //    // Board_initUSB(Board_USBDEVICE);
165      //    // Board_initWatchdog();
166      //    // Board_initWiFi();
167      //
168      //    /* Construct heartBeat Task  thread */
169      //    Task_Params_init(&taskParams);
```

Github: https://github.com/Ber-geb/solid-octo-tribble.git

```
170     //    taskParams.arg0 = 1000;
171     //    taskParams.stackSize = TASKSTACKSIZE;
172     //    taskParams.stack = &task0Stack;
173     //    Task_construct(&task0Struct, (Task_FuncPtr)heartBeatFxn, &taskParams, NULL);
174     //
175     //    /* Turn on user LED */
176     //    GPIO_write(Board_LED0, Board_LED_ON);
177     buttonPressed = false;
178     hardware_init();
179     //
180     //    System_printf("Starting the example\nSystem provider is set to SysMin. "
181     //            "Halt the target to view any SysMin contents in ROV.\n");
182     //    /* SysMin will only print to the console when you call flush or exit */
183     //    System_flush();
184
185     /* Start BIOS */
186     BIOS_start();
187
188 }
189
190 //-------------------------------------------------------------------------
191 // hardware_init()
192 //
193 // inits GPIO pins for toggling the LED
194 //-------------------------------------------------------------------------
195 void hardware_init(void)
196 {
197     uint32_t ui32Period;
198
199     i16ToggleCount = 0;
200
201     // Board_initUART();
202
203     //Set CPU Clock to 40MHz. 400MHz PLL/2 = 200 DIV 5 = 40MHz
204     SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
205     SysCtlPWMClockSet(SYSCTL_PWMDIV_64);
206
207     SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
208     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
209
210     GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0); //PD0 PWM pin
211     GPIOPinConfigure(GPIO_PD0_M1PWM0);
212
213     // ADD Tiva-C GPIO setup - enables port, sets pins 1-3 (RGB) pins for output
214     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
215     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
216
217     GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
218     GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0);
219     GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_2);
220
221     GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0 , GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
222
223     ui32PWMClock = SysCtlClockGet() / 64;
224     ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
225
226     PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN);
```

Github: https://github.com/Ber-geb/solid-octo-tribble.git

```
227         PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load);
228
229         //PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, (ui32ADC0Value[0]/200 * ui32Load)/1000);
230         PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
231         PWMGenEnable(PWM1_BASE, PWM_GEN_0);
232
233         // Turn on the LED
234         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 4);
235
236         //initialize ADC
237         SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
238         ADCHardwareOversampleConfigure(ADC0_BASE, 64);
239         ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
240
241         ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_CH1);
242         ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_CH1);
243         ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_CH1);
244         ADCSequenceStepConfigure(ADC0_BASE, 1, 3, ADC_CTL_CH1 | ADC_CTL_IE | ADC_CTL_END);
245
246         ADCSequenceEnable(ADC0_BASE, 1);
247
248         // Timer 2 setup code
249         SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER2);           // enable Timer 2 periph clks
250         TimerConfigure(TIMER2_BASE, TIMER_CFG_PERIODIC);        // cfg Timer 2 mode - periodic
251
252         ui32Period = (SysCtlClockGet() / 1000);                   // period = CPU clk div 1000 (1ms)
253         TimerLoadSet(TIMER2_BASE, TIMER_A, ui32Period);         // set Timer 2 period
254
255         TimerIntEnable(TIMER2_BASE, TIMER_TIMA_TIMEOUT);        // enables Timer 2 to interrupt CPU
256
257         TimerEnable(TIMER2_BASE, TIMER_A);                      // enable Timer 2
258
259         InitConsole();
260         UARTprintf("WORKING!");
261     }
262
263
264
265 void ADCfun(void){
266     while(1){
267
268         ADCIntClear(ADC0_BASE, 1);
269         ADCProcessorTrigger(ADC0_BASE, 1);
270
271         while (!ADCIntStatus(ADC0_BASE, 1, false)){}
272
273         ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
274         ADCAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
275         //ui32ADC0Value holds the ADC value...choose what to do with it...
276         Semaphore_pend (ADCSem, BIOS_WAIT_FOREVER);
277         //Semaphore_reset(ADCSem, 0);
278         //Semaphore_pend (UARTSem, BIOS_WAIT_FOREVER);
279         //Semaphore_post(UARTSem);
280     }
281 }
282
283 void SRfun(void){
```

Github: https://github.com/Ber-geb/solid-octo-tribble.git

```
284
285     while(1){
286         if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0)==0x00)
287         {
288             buttonPressed = true;
289             PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ADCAvg);
290         }
291         //Semaphore_reset(SRSem, 0);
292         //Semaphore_pend (SRSem, BIOS_WAIT_FOREVER);
293         Semaphore_pend (SRSem, BIOS_WAIT_FOREVER);
294
295     }
296 }
297
298 void UARTfun(void){
299 // display for UART
300     while(1){
301         UARTprintf("ADC Value[0]: %d\n", ADCAvg);
302         Semaphore_pend (UARTSem, BIOS_WAIT_FOREVER);
303         //Semaphore_reset(UARTSem, 0);
304         //Semaphore_pend (SRSem, BIOS_WAIT_FOREVER);
305     }
306 }
307
308 void TIMER2INT(void){
309     TimerIntClear(TIMER2_BASE, TIMER_TIMA_TIMEOUT);         // must clear timer flag FROM timer
310     il6ToggleCount = il6ToggleCount + 1; //increment every time HWI occurs
311     //    System_printf("Timer 2 interrupt occurred\n");
312     //    System_flush();
313
314     if (buttonPressed){
315         if(GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0))
316         {
317             GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 4);
318         }
319         else
320         {
321             GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
322         }
323     }
324
325     if (il6ToggleCount == 10){
326         //count = Semaphore_getCount(ADCSem);
327         Semaphore_post (ADCSem);
328     }
329
330     else if (il6ToggleCount == 20){
331         Semaphore_post (UARTSem);
332     }
333
334     else if (il6ToggleCount == 30){
335         Semaphore_post (SRSem);
336         il6ToggleCount = 0;
337     }
338
339     //Semaphore_post(ADCSem);
340 }
```

Github: https://github.com/Ber-geb/solid-octo-tribble.git

```
341
342 void TIMER_ISR(void){
343     TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);         // must clear timer flag FROM timer
344     if (!buttonPressed){
345         if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
346         {
347             GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
348         }
349         else
350         {
351             GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
352         }
353     }
354
355 }
356
357 void reverse(char str[], int len){
358     int start, end;
359     char temp;
360     for (start=0, end=len-1; start < end; start++, end--){
361         temp = *(str+start);
362         *(str+start) = *(str+end);
363         *(str+end) = temp;
364     }
365 }
366
367 char* itoa( int num, char* str, int base){
368     int i = 0;
369     bool isNegative = false;
370
371     if (num==0){
372         str[i] = '0';
373         str[i+1] = '\0';
374         return str;
375     }
376
377     if (num < 0 && base == 10) {
378         isNegative = true;
379         num = -num;
380     }
381
382     while (num!=0){
383         int rem = num % base;
384         str[i++] = (rem > 9) ? (rem - 10) + 'A' : rem + '0';
385         num = num/base;
386     }
387
388     if (isNegative){
389         str[i++] = '-';
390     }
391
392     str[i] = '\0';
393     reverse(str,i);
394     return str;
395 }
396
397 void InitConsole(void){
```

Github: https://github.com/Ber-geb/solid-octo-tribble.git

```
398     //Enable GPIO port A which is used for UART0 pins
399     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
400     //Configure the pin muxing for UART 0 functions on port A0 and A1
401     //This step is not necessary if your part does not support pin muxing
402     //TODO: change this to select the port/pin you are using.
403     GPIOPinConfigure(GPIO_PA0_U0RX);
404     GPIOPinConfigure(GPIO_PA1_U0TX);
405     //ENable UART0 so that we can configure the clock.
406     SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
407     //Use the internal 16MHz oscillator as the UART clock source.
408     UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);
409     //Select the alternate (UART) function for these pins.
410     GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
411     //Initialize the UART for console I/O.
412     UARTStdioConfig(0,115200,16000000);
413 }
414
```

Github: https://github.com/Ber-geb/solid-octo-tribble.git