APES Homework 5

CODE SNIPPETS:

Problem 2:

```
//*****************************************************************************
// Copyright (c) 2013-2017 Texas Instruments Incorporated.  All rights reserved.
// Software License Agreement
//
// Texas Instruments (TI) is supplying this software for use solely and
// exclusively on TI's microcontroller products. The software is owned by
// TI and/or its suppliers, and is protected under applicable copyright
// laws. You may not combine this software with "viral" open-source
// software in order to form a larger program.
//
// THIS SOFTWARE IS PROVIDED "AS IS" AND WITH ALL FAULTS.
// NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT
// NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. TI SHALL NOT, UNDER ANY
// CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
// DAMAGES, FOR ANY REASON WHATSOEVER.
//
// This is part of revision 2.1.4.178 of the EK-TM4C1294XL Firmware Package.
//
//*****************************************************************************

#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "drivers/pinout.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "utils/uartstdio.h"

uint32_t g_ui32SysClock; //System clock in Hz
static int count;
char buffer[50];

#define BAUD_RATE 115200
#define SysClock 120000000

//*****************************************************************************
//
// The error routine that is called if the driver library encounters an error.
//
//*****************************************************************************
```

```
#ifdef DEBUG
void
__error__(char *pcFilename, uint32_t ui32Line)
{
}
#endif

//*****************************************************************************
//
// Configure the UART and its pins.  This must be called before UARTprintf().
//
//*****************************************************************************
void ConfigureUART(void)
{
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA); //Enable GPIO peripheral

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0); //Enable UART0

    ROM_GPIOPinConfigure(GPIO_PA0_U0RX);             //Configure GPIO pins
    ROM_GPIOPinConfigure(GPIO_PA1_U0TX);
    ROM_GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    UARTStdioConfig(0, BAUD_RATE, g_ui32SysClock);     //Initialize UART
}

int main(void)
{
    g_ui32SysClock = MAP_SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |
                SYSCTL_OSC_MAIN | SYSCTL_USE_PLL |
                SYSCTL_CFG_VCO_480), SysClock);         //Configure System Clock at
120 MHz

    PinoutSet(false, false);

    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_1); //Enable GPIO pins

    ConfigureUART();                                    //Initialize UART

    UARTprintf("Project for: Nikhil Divekar, Date: 04/08/2018 \n");

    while(1)
    {
        LEDWrite(CLP_D1, 1);                            //Turn on LED

        SysCtlDelay(g_ui32SysClock / 2 / 3);            //500ms delay

        count++;
        UARTprintf("Count: %d \n", count);

        LEDWrite(CLP_D1, 0);                            //Turn off LED

        SysCtlDelay(g_ui32SysClock / 2 / 3);            //500ms delay
    }
}
```

Problem 3:

```
//FreeRTOS LED Task

#include <stdint.h>
#include <stdbool.h>
#include "main.h"
#include "drivers/pinout.h"
#include "driverlib/gpio.h"
#include "utils/uartstdio.h"
#include "inc/hw_memmap.h"
#include "driverlib/rom_map.h"
#include "driverlib/sysctl.h"

// TivaWare includes
#include "driverlib/sysctl.h"
#include "driverlib/debug.h"
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"

// FreeRTOS includes
#include "FreeRTOSConfig.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "timers.h"

// Demo Task declarations
void LED1Task(void *pvParameters);
void LED2Task(void *pvParameters);
void TimerCallback1(TimerHandle_t xTimer1);
void TimerCallback2(TimerHandle_t xTimer2);

int a, b;

// Main function
int main(void)
{
    // Initialize system clock to 120 MHz
    uint32_t output_clock_rate_hz;
    output_clock_rate_hz = ROM_SysCtlClockFreqSet(
                               (SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN |
                                SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480),
                               SYSTEM_CLOCK);
    ASSERT(output_clock_rate_hz == SYSTEM_CLOCK);

    // Initialize the GPIO pins for the Launchpad
    PinoutSet(false, false);

    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0);
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_1);
    a = 0x00;
    b = 0x00;

    // Create demo tasks
    xTaskCreate(LED1Task, (const portCHAR *)"LED1",
                configMINIMAL_STACK_SIZE, NULL, 1, NULL);

    xTaskCreate(LED2Task, (const portCHAR *)"LED2",
                configMINIMAL_STACK_SIZE, NULL, 1, NULL);

    vTaskStartScheduler();

    return 0;
}

void TimerCallback1(TimerHandle_t xTimer1)
{
    GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, a);
    a ^= GPIO_PIN_0;
}

void TimerCallback2(TimerHandle_t xTimer2)
{
    GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, b);
    b ^= GPIO_PIN_1;
}

// Flash the LEDs on the launchpad
void LED1Task(void *pvParameters)
{
    TimerHandle_t xTimer1 = NULL;
    xTimer1 = xTimerCreate("MyTimer1", pdMS_TO_TICKS(500), pdTRUE, (void
*)pvTimerGetTimerID(xTimer1), TimerCallback1);
    xTimerStart(xTimer1, 500);
    while(1);
}


// Write text over the Stellaris debug interface UART port
void LED2Task(void *pvParameters)
{
    TimerHandle_t xTimer2 = NULL;
    xTimer2 = xTimerCreate("MyTimer2", pdMS_TO_TICKS(250), pdTRUE, (void
*)pvTimerGetTimerID(xTimer2), TimerCallback2);
    xTimerStart(xTimer2, 250);
    while(1);
}

/*  ASSERT() Error function
 *
 *  failed ASSERTS() from driverlib/debug.h are executed in this function
 */
void __error__(char *pcFilename, uint32_t ui32Line)
{
    // Place a breakpoint here to capture errors until logging routine is finished
    while (1)
    {
    }
}
```

Problem 4:

```
/* FreeRTOS 8.2 Tiva Demo
 *
 * main.c
 *
 * Andy Kobyljanec
 *
 * This is a simple demonstration project of FreeRTOS 8.2 on the Tiva Launchpad
 * EK-TM4C1294XL.  TivaWare driverlib sourcecode is included.
 */

#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include "main.h"
#include "drivers/pinout.h"
#include "driverlib/gpio.h"
#include "utils/uartstdio.h"
#include "inc/hw_memmap.h"
#include "driverlib/rom_map.h"
#include "driverlib/sysctl.h"
#include "utils/uartstdio.h"
#include "driverlib/pin_map.h"
#include <stdio.h>
#include <stdlib.h>
#include "inc/hw_types.h"
#include "driverlib/rom.h"
#include "driverlib/uart.h"
#include "utils/uartstdio.h"

#define LED_TOGGLE 0x00000001
#define LOG_STRING 0x00000002

#define ULONG_MAX 0xFFFFFFFF

// TivaWare includes
#include "driverlib/sysctl.h"
#include "driverlib/debug.h"
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"

// FreeRTOS includes
#include "FreeRTOSConfig.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "timers.h"

#define BAUD_RATE 115200
#define SysClock 120000000

// Demo Task declarations
void LED1Task(void *pvParameters);
void LED2Task(void *pvParameters);
```

```c
void Task3(void *pvParameters);
void TimerCallback1(TimerHandle_t xTimer1);
void TimerCallback2(TimerHandle_t xTimer2);

int a, b, c;

TaskHandle_t Task1Handle;
TaskHandle_t Task2Handle;
TaskHandle_t Task3Handle;
xQueueHandle queue_handle;

uint32_t g_ui32SysClock;

typedef struct notifying_data
{
    char message[50];
    TickType_t current_ticks;
}notifying_data;

void ConfigureUART(void)
{
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);    //Enable GPIO

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);    //Enable UART0

    ROM_GPIOPinConfigure(GPIO_PA0_U0RX);                //Configure UART pins
    ROM_GPIOPinConfigure(GPIO_PA1_U0TX);
    ROM_GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    UARTStdioConfig(0, BAUD_RATE, g_ui32SysClock);         //Initialize UART
}

// Main function
int main(void)
{
        g_ui32SysClock = MAP_SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |
                    SYSCTL_OSC_MAIN | SYSCTL_USE_PLL |
                    SYSCTL_CFG_VCO_480), SysClock);

        PinoutSet(false, false);

        ROM_GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_1);

        ConfigureUART();        //Initialize UART

        UARTprintf("This is Nikhil");

        GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0);
        GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_1);
        a = 0x00;
        b = 0x00;
        c = 0x00;

        // Create demo tasks
        xTaskCreate(LED1Task, (const portCHAR *)"LED1",
                configMINIMAL_STACK_SIZE, NULL, 1, &Task1Handle);

        xTaskCreate(LED2Task, (const portCHAR *)"LED2",
                configMINIMAL_STACK_SIZE, NULL, 1, &Task2Handle);

        xTaskCreate(Task3, (const portCHAR *)"Task_3",
                configMINIMAL_STACK_SIZE, NULL, 1, &Task3Handle);

        vTaskStartScheduler();

        return 0;
}

void TimerCallback1(TimerHandle_t xTimer1)
{
    xTaskNotify(Task3Handle, LED_TOGGLE, eSetBits);
}

void TimerCallback2(TimerHandle_t xTimer2)
{
    notifying_data current_data;
    strcpy(current_data.message, "Notification from Task 2");
    current_data.current_ticks = xTaskGetTickCount();

    queue_handle = xQueueCreate(10, sizeof(notifying_data));

    xQueueSend(queue_handle, &current_data, 500);

    xTaskNotify(Task3Handle, LOG_STRING, eSetBits);
}

// Flash the LEDs on the launchpad
void LED1Task(void *pvParameters)
{
    TimerHandle_t xTimer1 = NULL;
    xTimer1 = xTimerCreate("MyTimer1", pdMS_TO_TICKS(500), pdTRUE, (void
*)pvTimerGetTimerID(xTimer1), TimerCallback1);
    xTimerStart(xTimer1, 500);
    while(1);
}


// Write text over the Stellaris debug interface UART port
void LED2Task(void *pvParameters)
{
    TimerHandle_t xTimer2 = NULL;
    xTimer2 = xTimerCreate("MyTimer2", pdMS_TO_TICKS(250), pdTRUE, (void
*)pvTimerGetTimerID(xTimer2), TimerCallback2);
    xTimerStart(xTimer2, 250);
    while(1);
}

void Task3(void *pvParameters)
{
    BaseType_t returned_notification;
    int returned_val;
    notifying_data received_data;
    while(1)
    {
        returned_notification = xTaskNotifyWait(0, 0xFF, &returned_val,
portMAX_DELAY);
        if(returned_notification == pdTRUE)
        {
            if(returned_val & LED_TOGGLE)
            {
                GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, a);
                a ^= GPIO_PIN_0;
                UARTprintf("Task1 notified \n");
            }
            if(returned_val & LOG_STRING)
            {
                xQueueReceive(queue_handle, &received_data, 500);
                UARTprintf("Message received: %s, Current ticks: %d \n",
received_data.message, received_data.current_ticks);
            }
        }
    }
}

/*  ASSERT() Error function
 *
 *  failed ASSERTS() from driverlib/debug.h are executed in this function
 */
void __error__(char *pcFilename, uint32_t ui32Line)
{
    // Place a breakpoint here to capture errors until logging routine is finished
    while (1)
    {
    }
}
```

Video Links

Problem 2:

https://drive.google.com/open?id=1K3kJM-To4cj4bconSxCVoSeul8xLOKLW

Problem 3:

https://drive.google.com/open?id=1Ms0IPFIc-uWk8tUNbEJ5grQybJXDcriO

Problem 4:

https://drive.google.com/open?id=1BkDisigJXGYYWvxsarpr0S_FwDRy1NmU