APES - Homework 5
By Anay Gondhalekar
Date – 2018/04/08

Problem 1: Repository for HW5 - https://github.com/AnayGondhalekar/ECEN5013_HW5

Problem 2: DriverLib
Link to Video - https://drive.google.com/open?id=1oQWx7EY0uupTxBk6fFo528XdyFsq59cj

Code:

```
#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"

#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "drivers/pinout.h"

uint32_t g_ui32SysClock;

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

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

void ConfigureUART(void)
{

  // Enable the GPIO Peripheral used by the UART.
```

```
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);


    // Enable UART0

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);

    // Configure GPIO Pins for UART mode.

    ROM_GPIOPinConfigure(GPIO_PA0_U0RX);
    ROM_GPIOPinConfigure(GPIO_PA1_U0TX);
    ROM_GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);


    // Initialize the UART for console I/O.

    UARTStdioConfig(0, 115200, g_ui32SysClock);
}

int main(void)
{
    int count = 0;

    // Run clockat 120 MHz.

    g_ui32SysClock = MAP_SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |
                SYSCTL_OSC_MAIN | SYSCTL_USE_PLL |
                SYSCTL_CFG_VCO_480), 120000000);


    // Configure the device pins.

    PinoutSet(false, false);


    // Enable the GPIO pins for the LED D1 (PN1).

    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_1);


    // Initialize the UART.

    ConfigureUART();
    UARTprintf("Project for: Anay Gondhalekar 4/7/2018\n");

    while(1)
    {

        // Turn on D1.

        LEDWrite(CLP_D1, 1);

        SysCtlDelay(g_ui32SysClock / 2 / 3);

        // Turn off D1.

        LEDWrite(CLP_D1, 0);

        SysCtlDelay(g_ui32SysClock / 2 / 3);
        count++;
        UARTprintf("Count is %d \n",count);
    }
}
```

Problem 3: FreeRTOS
Link to Video – https://drive.google.com/open?id=14dl3f3rN_U3TPoVEUqbpoTbbmu9yAHzr

Code:

```
#include <stdint.h>
#include <stdbool.h>
#include "main.h"
#include "drivers/pinout.h"
#include "utils/uartstdio.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"


// Declarations
void LEDTask1(void *pvParameters);
void LEDTask2(void *pvParameters);

int flag1,flag2;
TimerHandle_t xTimer1,xTimer2;

void vTimerCallback1( TimerHandle_t xTimer1 )
{
  if (flag1 == 0)
  {
    LEDWrite(0x01, 0x01);
    flag1 = 1;
  }
  else if(flag1 == 1)
  {
    LEDWrite(0x01, 0x00);
    flag1 = 0;
  }
}

void vTimerCallback2( TimerHandle_t xTimer2 )
{
  if (flag2 == 0)
  {
    LEDWrite(0x02, 0x02);
    flag2 = 1;
  }
  else if(flag2 == 1)
  {
    LEDWrite(0x02, 0x00);
    flag2 = 0;
  }
}
// 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);


  PinoutSet(false, false);

  // Create tasks
   xTaskCreate(LEDTask1, (const portCHAR *)"LED1",
            configMINIMAL_STACK_SIZE, NULL, 1, NULL);

   xTaskCreate(LEDTask2, (const portCHAR *)"LED2",
            configMINIMAL_STACK_SIZE, NULL, 1, NULL);

  vTaskStartScheduler();
  return 0;
}


// Flash the LEDs on the launchpad
void LEDTask1(void *pvParameters)
{

    // Turn on LED 1
    xTimer1 = xTimerCreate("timer1",pdMS_TO_TICKS( 500 ),pdTRUE,( void * ) 0,vTimerCallback1);
    xTimerStart( xTimer1, 0 );
    for(;;) ;

}


void LEDTask2(void *pvParameters)
{

    // Turn on LED 2
    xTimer2 = xTimerCreate("timer2",pdMS_TO_TICKS( 250 ),pdTRUE,( void * ) 0,vTimerCallback2);
    xTimerStart( xTimer2, 0 );
    for(;;) ;

}

/*  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: Event Driven UI
Link to Video – https://drive.google.com/open?id=191a7_AJ2lmOiqiDt8cLr1YaZ3HxO3-qF

Code:

```
//General Includes
#include <stdint.h>
#include <stdbool.h>
#include "main.h"
#include "drivers/pinout.h"
#include "utils/uartstdio.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"


// TivaWare includes
#include "driverlib/sysctl.h"
#include "driverlib/debug.h"
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"
#include "driverlib/gpio.h"
#include "drivers/pinout.h"
#include "driverlib/pin_map.h"
#include "driverlib/uart.h"


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

```c
#define TOGGLE_LED  0x01
#define LOG_STRING  0X02

// Demo Task declarations
void LEDTask1(void *pvParameters);
void LEDTask2(void *pvParameters);
void Task3(void *pvParameters);
int flag1,flag2;
TimerHandle_t xTimer1,xTimer2;
TaskHandle_t xTaskHandle3;
uint32_t output_clock_rate_hz;
QueueHandle_t xQueue;
struct Message        //structure to send
 {
    TickType_t number;
    char logstring[20];
 } xMessage;

void vTimerCallback1( TimerHandle_t xTimer1 )
{
    //xTaskNotify( xTaskHandle3, TOGGLE_LED , eSetBits );
    xTaskNotify( xTaskHandle3, TOGGLE_LED , eSetBits );
}

void vTimerCallback2( TimerHandle_t xTimer2 )
{

    struct Message pxMessage;
    xQueue = xQueueCreate( 20, sizeof( struct Message ) );
    TickType_t Tick_Count;
    Tick_Count = xTaskGetTickCount();
    pxMessage.number = Tick_Count;

    strcpy( pxMessage.logstring, "Anay here");

    xQueueSend( xQueue, &pxMessage, ( TickType_t ) 0 );
    xTaskNotify( xTaskHandle3,LOG_STRING , eSetBits );

}

void ConfigureUART(void)
{
    //
    // Enable the GPIO Peripheral used by the UART.
    //
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    //
    // Enable UART0
    //
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);

    //
    // Configure GPIO Pins for UART mode.
    //
    ROM_GPIOPinConfigure(GPIO_PA0_U0RX);
    ROM_GPIOPinConfigure(GPIO_PA1_U0TX);
    ROM_GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    //
    // Initialize the UART for console I/O.
    //
    UARTStdioConfig(0, 115200, output_clock_rate_hz);
}

// Main function
int main(void)
{
    // Initialize system clock to 120 MHz
    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);
    ConfigureUART();

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

    // Create demo tasks
     xTaskCreate(LEDTask1, (const portCHAR *)"LED1",
             configMINIMAL_STACK_SIZE, NULL, 1, NULL);

     xTaskCreate(LEDTask2, (const portCHAR *)"LED2",
             configMINIMAL_STACK_SIZE, NULL, 1, NULL);
    xTaskCreate(Task3,(const portCHAR *)"Task3",
             configMINIMAL_STACK_SIZE, NULL, 1,&xTaskHandle3 );
```

```
   vTaskStartScheduler();
   return 0;
}


// Flash the LEDs on the launchpad
void LEDTask1(void *pvParameters)
{

        // Turn on LED 1
        xTimer1 = xTimerCreate("timer1",pdMS_TO_TICKS( 500 ),pdTRUE,( void * ) 0,vTimerCallback1);
        xTimerStart( xTimer1, 0 );
        for(;;) ;

}


void LEDTask2(void *pvParameters)
{

        // Turn on LED 2
        xTimer2 = xTimerCreate("timer2",pdMS_TO_TICKS( 250 ),pdTRUE,( void * ) 0,vTimerCallback2);
        xTimerStart( xTimer2, 0 );
        for(;;) ;

}

void Task3(void *pvParameters)
{
   uint32_t ulNotifiedValue;
   while(1)
   {
   xTaskNotifyWait( 0x00, ULONG_MAX, &ulNotifiedValue, portMAX_DELAY );

                if( ( ulNotifiedValue & 0x01 ) != 0 )         /check if led_toggle or log command
                {
                   if (flag1 == 0)
                      {
                         LEDWrite(0x01, 0x01);
                         flag1 = 1;
                      }
                      else if(flag1 == 1)
                      {
                         LEDWrite(0x01, 0x00);
```

```
                        flag1 = 0;
                    }
              }

              if( ( ulNotifiedValue & 0x02 ) != 0 )
              {
                 struct Message pxRxedMessage;
                 TickType_t count;
                 //char mystring[20];
                 xQueueReceive( xQueue, &pxRxedMessage , ( TickType_t ) 500);
                 count = pxRxedMessage.number ;
                 //mystring = pxRxedMessage.logstring;
                 UARTprintf("Message is %s and tick count is %d \n",pxRxedMessage.logstring,count);  //print
received message to uart
              }
   }
}

/*  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)
   {
   }
}
```