**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"

#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)

{

}

}