ESE-3025 Embedded Real Time Operating Systems

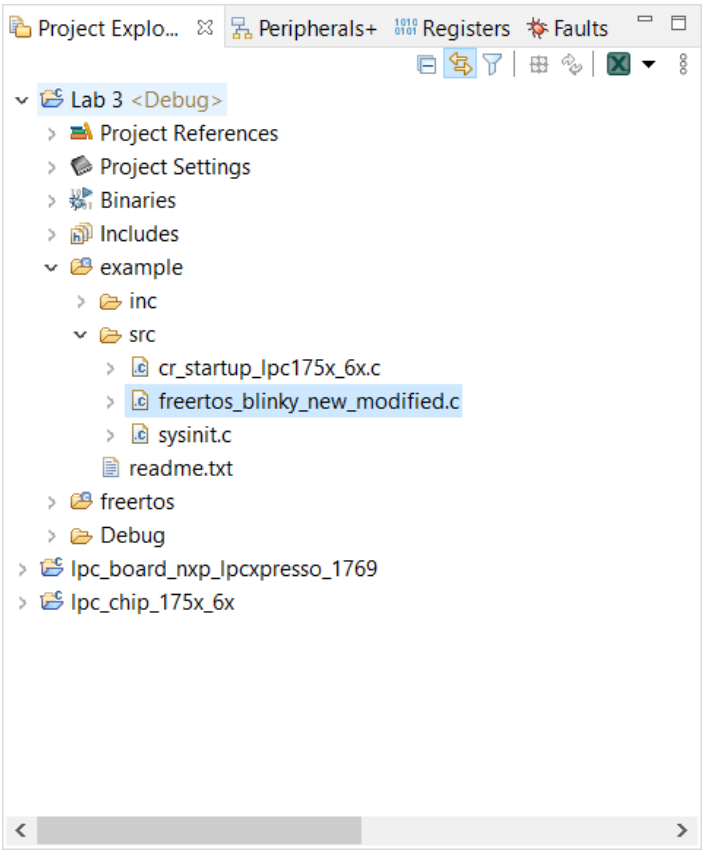
LAB 3

GROUP No. 2

Group Members	Student ID
Christy Rachel Philip	C0765535
Fahad Rahman	C0769871
James M Chacko	C0777192
Premil Presannan	C0777191
Vy Nguyen	C0776242

MODIFYING ON-BOARD LED IN FREE RTOS

Step 1: Create a new project folder.



Step 2: Add retarget.h to the new project folder. You can clone this library header file from https://github.com/mikeshams/ESE3025

Retarget.h file:

```
/*
 * @brief      IO redirection support
 *
 * This file adds re-direction support to the library for various
 * projects. It can be configured in one of 3 ways - no redirection,
 * redirection via a UART, or redirection via semihosting. If DEBUG
 * is not defined, all printf statements will do nothing with the
 * output being throw away. If DEBUG is defined, then the choice of
 * output is selected by the DEBUG_SEMIHOSTING define. If the
 * DEBUG_SEMIHOSTING is not defined, then output is redirected via
 * the UART. If DEBUG_SEMIHOSTING is defined, then output will be
 * attempted to be redirected via semihosting. If the UART method
 * is used, then the Board_UARTPutChar and Board_UARTGetChar
 * functions must be defined to be used by this driver and the UART
 * must already be initialized to the correct settings.
 *
 * @note
 * Copyright(C) NXP Semiconductors, 2012
```

```
 * All rights reserved.
 *
 * @par
 * Software that is described herein is for illustrative purposes only
 * which provides customers with programming information regarding the
 * LPC products.  This software is supplied "AS IS" without any warranties of
 * any kind, and NXP Semiconductors and its licensor disclaim any and
 * all warranties, express or implied, including all implied warranties of
 * merchantability, fitness for a particular purpose and non-infringement of
 * intellectual property rights.  NXP Semiconductors assumes no responsibility
 * or liability for the use of the software, conveys no license or rights under any
 * patent, copyright, mask work right, or any other intellectual property rights in
 * or to any products. NXP Semiconductors reserves the right to make changes
 * in the software without notification. NXP Semiconductors also makes no
 * representation or warranty that such application will be suitable for the
 * specified use without further testing or modification.
 *
 * @par
 * Permission to use, copy, modify, and distribute this software and its
 * documentation is hereby granted, under NXP Semiconductors' and its
 * licensor's relevant copyrights in the software, without fee, provided that it
 * is used in conjunction with NXP Semiconductors microcontrollers.  This
 * copyright, permission, and disclaimer notice must appear in all copies of
 * this code.
 */

#include "board.h"

/* Keil (Realview) support */
#if defined(__CC_ARM)

#include <stdio.h>
#include <rt_misc.h>

#if defined(DEBUG_ENABLE)
#if defined(DEBUG_SEMIHOSTING)
#define ITM_Port8(n)    (*((volatile unsigned char *) (0xE0000000 + 4 * n)))
#define ITM_Port16(n)   (*((volatile unsigned short *) (0xE0000000 + 4 * n)))
#define ITM_Port32(n)   (*((volatile unsigned long *) (0xE0000000 + 4 * n)))

#define DEMCR           (*((volatile unsigned long *) (0xE000EDFC)))
#define TRCENA          0x01000000

/* Write to SWO */
void _ttywrch(int ch)
{
       if (DEMCR & TRCENA) {
              while (ITM_Port32(0) == 0) {}
              ITM_Port8(0) = ch;
       }
}
```

```
#else
static INLINE void BoardOutChar(char ch)
{
	Board_UARTPutChar(ch);
}

#endif /* defined(DEBUG_SEMIHOSTING) */
#endif /* defined(DEBUG_ENABLE) */

struct __FILE {
	int handle;
};

FILE __stdout;
FILE __stdin;
FILE __stderr;

void *_sys_open(const char *name, int openmode)
{
	return 0;
}

int fputc(int c, FILE *f)
{
#if defined(DEBUG_ENABLE)
#if defined(DEBUG_SEMIHOSTING)
	_ttywrch(c);
#else
	BoardOutChar((char) c);
#endif
#endif
	return 0;
}

int fgetc(FILE *f)
{
#if defined(DEBUG_ENABLE) && !defined(DEBUG_SEMIHOSTING)
	return Board_UARTGetChar();
#else
	return 0;
#endif
}

int ferror(FILE *f)
{
	return EOF;
}

void _sys_exit(int return_code)
{
label:
	__WFI();
```

```
        goto label;    /* endless loop */
}

#endif /* defined (__CC_ARM) */

/* IAR support */
#if defined(__ICCARM__)
/*******************
 *
 * Copyright 1998-2003 IAR Systems.  All rights reserved.
 *
 * $Revision: 30870 $
 *
 * This is a template implementation of the "__write" function used by
 * the standard library.  Replace it with a system-specific
 * implementation.
 *
 * The "__write" function should output "size" number of bytes from
 * "buffer" in some application-specific way.  It should return the
 * number of characters written, or _LLIO_ERROR on failure.
 *
 * If "buffer" is zero then __write should perform flushing of
 * internal buffers, if any.  In this case "handle" can be -1 to
 * indicate that all handles should be flushed.
 *
 * The template implementation below assumes that the application
 * provides the function "MyLowLevelPutchar".  It should return the
 * character written, or -1 on failure.
 *
 ********************/

#include <yfuns.h>

#if defined(DEBUG_ENABLE) && !defined(DEBUG_SEMIHOSTING)

_STD_BEGIN

#pragma module_name = "?__write"

/*
   If the __write implementation uses internal buffering, uncomment
   the following line to ensure that we are called with "buffer" as 0
   (i.e. flush) when the application terminates. */
size_t __write(int handle, const unsigned char *buffer, size_t size)
{
#if defined(DEBUG_ENABLE)
        size_t nChars = 0;

        if (buffer == 0) {
                /*
                   This means that we should flush internal buffers.  Since we
                   don't we just return.  (Remember, "handle" == -1 means that all
```

```
                  handles should be flushed.)
                */
               return 0;
        }

        /* This template only writes to "standard out" and "standard err",
           for all other file handles it returns failure. */
        if (( handle != _LLIO_STDOUT) && ( handle != _LLIO_STDERR) ) {
               return _LLIO_ERROR;
        }

        for ( /* Empty */; size != 0; --size) {
               Board_UARTPutChar(*buffer++);
               ++nChars;
        }

        return nChars;
#else
        return size;
#endif /* defined(DEBUG_ENABLE) */
}

_STD_END
#endif

#endif /* defined (__ICCARM__) */

#if defined( __GNUC__ )
/* Include stdio.h to pull in __REDLIB_INTERFACE_VERSION__ */
#include <stdio.h>

#if (__REDLIB_INTERFACE_VERSION__ >= 20000)
/* We are using new Redlib_v2 semihosting interface */
        #define WRITEFUNC __sys_write
        #define READFUNC __sys_readc
#else
/* We are using original Redlib semihosting interface */
        #define WRITEFUNC __write
        #define READFUNC __readc
#endif

#if defined(DEBUG_ENABLE)
#if defined(DEBUG_SEMIHOSTING)
/* Do nothing, semihosting is enabled by default in LPCXpresso */
#endif /* defined(DEBUG_SEMIHOSTING) */
#endif /* defined(DEBUG_ENABLE) */

#if !defined(DEBUG_SEMIHOSTING)
int WRITEFUNC(int iFileHandle, char *pcBuffer, int iLength)
{
#if defined(DEBUG_ENABLE)
        unsigned int i;
```

```
	for (i = 0; i < iLength; i++) {
		Board_UARTPutChar(pcBuffer[i]);
	}
#endif

	return iLength;
}

/* Called by bottom level of scanf routine within RedLib C library to read
   a character. With the default semihosting stub, this would read the character
   from the debugger console window (which acts as stdin). But this version reads
   the character from the LPC1768/RDB1768 UART. */
int READFUNC(void)
{
#if defined(DEBUG_ENABLE)
	char c = Board_UARTGetChar();
	return (int) c;

#else
	return (int) -1;
#endif
}

#endif /* !defined(DEBUG_SEMIHOSTING) */
#endif /* defined ( __GNUC__ ) */
```

Step 3: Go to the lpc_board_nxp_lpcxpresso_1769 project folder available on the project explorer window and cut the original board.c in a safe place on your machine and replace it with the modified_board.c that you can clone it from the same GitHub link. (Do you need to rename this file as board.c?)

Answer:

New board.c file: (was modified_board.c). Yes we have to rename it, in order for the program select the correct file.

```
/*
* @brief NXP LPC1769 LPCXpresso board file
*
* @note
* Copyright(C) NXP Semiconductors, 2012
* All rights reserve
* @par
* Software that is described herein is for illustrative purposes only
* which provides customers with programming information regarding the
```

```
* LPC products. This software is supplied "AS IS" without any warranties
of
* any kind, and NXP Semiconductors and its licensor disclaim any and
* all warranties, express or implied, including all implied warranties of
* merchantability, fitness for a particular purpose and non-infringement
of
* intellectual property rights. NXP Semiconductors assumes no
responsibility
* or liability for the use of the software, conveys no license or rights
under any
* patent, copyright, mask work right, or any other intellectual property
rights in
* or to any products. NXP Semiconductors reserves the right to make
changes
* in the software without notification. NXP Semiconductors also makes no
* representation or warranty that such application will be suitable for
the
* specified use without further testing or modification.
*
* @par
* Permission to use, copy, modify, and distribute this software and its
* documentation is hereby granted, under NXP Semiconductors' and its
* licensor's relevant copyrights in the software, without fee, provided
that it
* is used in conjunction with NXP Semiconductors microcontrollers. This
* copyright, permission, and disclaimer notice must appear in all copies
of
* this code.
*/
#include "board.h"
#include "string.h"
#include "retarget.h"
/*****************************************************************************
***
* Private types/enumerations/variables
*****************************************************************************
*/
#define BUTTONS_BUTTON1_GPIO_PORT_NUM 2
#define BUTTONS_BUTTON1_GPIO_BIT_NUM 10
#define JOYSTICK_UP_GPIO_PORT_NUM 2
#define JOYSTICK_UP_GPIO_BIT_NUM 3
#define JOYSTICK_DOWN_GPIO_PORT_NUM 0
#define JOYSTICK_DOWN_GPIO_BIT_NUM 15
#define JOYSTICK_LEFT_GPIO_PORT_NUM 2
#define JOYSTICK_LEFT_GPIO_BIT_NUM 4
#define JOYSTICK_RIGHT_GPIO_PORT_NUM 0
#define JOYSTICK_RIGHT_GPIO_BIT_NUM 16
#define JOYSTICK_PRESS_GPIO_PORT_NUM 0
#define JOYSTICK_PRESS_GPIO_BIT_NUM 17
/* RED LED */
#define LED0_GPIO_PORT_NUM 0
#define LED0_GPIO_BIT_NUM 22
```

```
/* GREEN LED - (Added 11/7/2019) */
#define LED1_GPIO_PORT_NUM 3
#define LED1_GPIO_BIT_NUM 25
/* BLUE LED - (Added 11/7/2019) */
#define LED2_GPIO_PORT_NUM 3
#define LED2_GPIO_BIT_NUM 26
/*****************************************************************************
***
* Public types/enumerations/variables
*****************************************************************************
*/
/* System oscillator rate and RTC oscillator rate */
const uint32_t OscRateIn = 12000000;
const uint32_t RTCOscRateIn = 32768;
/*****************************************************************************
***
* Private functions
*****************************************************************************
*/
/* Initializes board LED(s) */
static void Board_LED_Init(void)
{
/* Pin PIO0_22 is configured as GPIO pin during SystemInit */
/* Set the PIO_22 as output */
Chip_GPIO_WriteDirBit(LPC_GPIO, LED0_GPIO_PORT_NUM,
LED0_GPIO_BIT_NUM, true);
/* Setting Pins for Blue and Green LED */
Chip_GPIO_WriteDirBit(LPC_GPIO, LED1_GPIO_PORT_NUM,
LED1_GPIO_BIT_NUM, true);
Chip_GPIO_WriteDirBit(LPC_GPIO, LED2_GPIO_PORT_NUM,
LED2_GPIO_BIT_NUM, true);
}
/*****************************************************************************
***
* Public functions
*****************************************************************************
*/
/* Initialize UART pins */
void Board_UART_Init(LPC_USART_T *pUART)
{
/* Pin Muxing has already been done during SystemInit */
}
/* Initialize debug output via UART for board */
void Board_Debug_Init(void)
{
#if defined(DEBUG_ENABLE)
Board_UART_Init(DEBUG_UART);
Chip_UART_Init(DEBUG_UART);
Chip_UART_SetBaud(DEBUG_UART, 115200);
Chip_UART_ConfigData(DEBUG_UART, UART_LCR_WLEN8 | UART_LCR_SBS_1BIT |
UART_LCR_PARITY_DIS);
/* Enable UART Transmit */
```

```
Chip_UART_TXEnable(DEBUG_UART);
#endif
}
/* Sends a character on the UART */
void Board_UARTPutChar(char ch)
{
#if defined(DEBUG_ENABLE)
while ((Chip_UART_ReadLineStatus(DEBUG_UART) & UART_LSR_THRE) == 0)
{}
Chip_UART_SendByte(DEBUG_UART, (uint8_t) ch);
#endif
}
/* Gets a character from the UART, returns EOF if no character is ready */
int Board_UARTGetChar(void)
{
#if defined(DEBUG_ENABLE)
if (Chip_UART_ReadLineStatus(DEBUG_UART) & UART_LSR_RDR) {
return (int) Chip_UART_ReadByte(DEBUG_UART);
}
#endif
return EOF;
}
/* Outputs a string on the debug UART */
void Board_UARTPutSTR(char *str)
{
#if defined(DEBUG_ENABLE)
while (*str != '\0') {
Board_UARTPutChar(*str++);
}
#endif
}
/* Sets the state of a board LED to on or off */
void Board_LED_Set(uint8_t LEDNumber, bool On)
{
bool LEDon;
if (On==false)
{
LEDon=true;
}
else
{
LEDon=false;
}
/* There is only one LED -- Fixing for three LEDs*/
if (LEDNumber == 0)
{
Chip_GPIO_WritePortBit(LPC_GPIO, LED0_GPIO_PORT_NUM,
LED0_GPIO_BIT_NUM, LEDon);
}
else if (LEDNumber == 1)
{
Chip_GPIO_WritePortBit(LPC_GPIO, LED1_GPIO_PORT_NUM,
```

```
LED1_GPIO_BIT_NUM, LEDon);
}
else if (LEDNumber == 2)
{
Chip_GPIO_WritePortBit(LPC_GPIO, LED2_GPIO_PORT_NUM,
LED2_GPIO_BIT_NUM, LEDon);
}
}
/* Returns the current state of a board LED */
bool Board_LED_Test(uint8_t LEDNumber)
{
bool state = false;
if (LEDNumber == 0)
{
state = Chip_GPIO_ReadPortBit(LPC_GPIO, LED0_GPIO_PORT_NUM,
LED0_GPIO_BIT_NUM);
}
else if (LEDNumber == 1)
{
state = Chip_GPIO_ReadPortBit(LPC_GPIO, LED1_GPIO_PORT_NUM,
LED1_GPIO_BIT_NUM);
}
else if (LEDNumber == 2)
{
state = Chip_GPIO_ReadPortBit(LPC_GPIO, LED2_GPIO_PORT_NUM,
LED2_GPIO_BIT_NUM);
}
return !state; // Returns the opposite state
}
void Board_LED_Toggle(uint8_t LEDNumber)
{
if (LEDNumber == 0)
{
Board_LED_Set(LEDNumber, !Board_LED_Test(LEDNumber));
}
else if (LEDNumber == 1)
{
Board_LED_Set(LEDNumber, !Board_LED_Test(LEDNumber));
}
else if (LEDNumber == 2)
{
Board_LED_Set(LEDNumber, !Board_LED_Test(LEDNumber));
}
}
/* Set up and initialize all required blocks and functions related to the
board hardware */
void Board_Init(void)
{
/* Sets up DEBUG UART */
DEBUGINIT();
/* Initializes GPIO */
Chip_GPIO_Init(LPC_GPIO);
```

```
Chip_IOCON_Init(LPC_IOCON);
/* Initialize LEDs */
Board_LED_Init();
}
/* Returns the MAC address assigned to this board */
void Board_ENET_GetMacADDR(uint8_t *mcaddr)
{
const uint8_t boardmac[] = {0x00, 0x60, 0x37, 0x12, 0x34, 0x56};
memcpy(mcaddr, boardmac, 6);
}
/* Initialize pin muxing for SSP interface */
void Board_SSP_Init(LPC_SSP_T *pSSP)
{
if (pSSP == LPC_SSP1) {
/* Set up clock and muxing for SSP1 interface */
/*
* Initialize SSP0 pins connect
* P0.7: SCK
* P0.6: SSEL
* P0.8: MISO
* P0.9: MOSI
*/
Chip_IOCON_PinMux(LPC_IOCON, 0, 7, IOCON_MODE_INACT,
IOCON_FUNC2);
Chip_IOCON_PinMux(LPC_IOCON, 0, 6, IOCON_MODE_INACT,
IOCON_FUNC2);
Chip_IOCON_PinMux(LPC_IOCON, 0, 8, IOCON_MODE_INACT,
IOCON_FUNC2);
Chip_IOCON_PinMux(LPC_IOCON, 0, 9, IOCON_MODE_INACT,
IOCON_FUNC2);
}
else {
/* Set up clock and muxing for SSP0 interface */
/*
* Initialize SSP0 pins connect
* P0.15: SCK
* P0.16: SSEL
* P0.17: MISO
* P0.18: MOSI
*/
Chip_IOCON_PinMux(LPC_IOCON, 0, 15, IOCON_MODE_INACT,
IOCON_FUNC2);
Chip_IOCON_PinMux(LPC_IOCON, 0, 16, IOCON_MODE_INACT,
IOCON_FUNC2);
Chip_IOCON_PinMux(LPC_IOCON, 0, 17, IOCON_MODE_INACT,
IOCON_FUNC2);
Chip_IOCON_PinMux(LPC_IOCON, 0, 18, IOCON_MODE_INACT,
IOCON_FUNC2);
}
}
/* Initialize pin muxing for SPI interface */
void Board_SPI_Init(bool isMaster)
```

```
{
/* Set up clock and muxing for SSP0 interface */
/*
* Initialize SSP0 pins connect
* P0.15: SCK
* P0.16: SSEL
* P0.17: MISO
* P0.18: MOSI
*/
Chip_IOCON_PinMux(LPC_IOCON, 0, 15, IOCON_MODE_PULLDOWN,
IOCON_FUNC3);
if (isMaster) {
Chip_IOCON_PinMux(LPC_IOCON, 0, 16, IOCON_MODE_PULLUP,
IOCON_FUNC0);
Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 16, true);
Board_SPI_DeassertSSEL();
}
else {
Chip_IOCON_PinMux(LPC_IOCON, 0, 16, IOCON_MODE_PULLUP,
IOCON_FUNC3);
}
Chip_IOCON_PinMux(LPC_IOCON, 0, 17, IOCON_MODE_INACT, IOCON_FUNC3);
Chip_IOCON_PinMux(LPC_IOCON, 0, 18, IOCON_MODE_INACT, IOCON_FUNC3);
}
/* Assert SSEL pin */
void Board_SPI_AssertSSEL(void)
{
Chip_GPIO_WritePortBit(LPC_GPIO, 0, 16, false);
}
/* De-Assert SSEL pin */
void Board_SPI_DeassertSSEL(void)
{
Chip_GPIO_WritePortBit(LPC_GPIO, 0, 16, true);
}
void Board_Audio_Init(LPC_I2S_T *pI2S, int micIn)
{
I2S_AUDIO_FORMAT_T I2S_Config;
/* Chip_Clock_EnablePeripheralClock(SYSCTL_CLOCK_I2S); */
I2S_Config.SampleRate = 48000;
I2S_Config.ChannelNumber = 2; /* 1 is mono, 2 is stereo */
I2S_Config.WordWidth = 16; /* 8, 16 or 32 bits */
Chip_I2S_Init(pI2S);
Chip_I2S_TxConfig(pI2S, &I2S_Config);
}
/* Sets up board specific I2C interface */
void Board_I2C_Init(I2C_ID_T id)
{
switch (id) {
case I2C0:
Chip_IOCON_PinMux(LPC_IOCON, 0, 27, IOCON_MODE_INACT,
IOCON_FUNC1);
Chip_IOCON_PinMux(LPC_IOCON, 0, 28, IOCON_MODE_INACT,
```

```
IOCON_FUNC1);
Chip_IOCON_SetI2CPad(LPC_IOCON, I2CPADCFG_STD_MODE);
break;
case I2C1:
Chip_IOCON_PinMux(LPC_IOCON, 0, 19, IOCON_MODE_INACT,
IOCON_FUNC2);
Chip_IOCON_PinMux(LPC_IOCON, 0, 20, IOCON_MODE_INACT,
IOCON_FUNC2);
Chip_IOCON_EnableOD(LPC_IOCON, 0, 19);
Chip_IOCON_EnableOD(LPC_IOCON, 0, 20);
break;
case I2C2:
Chip_IOCON_PinMux(LPC_IOCON, 0, 10, IOCON_MODE_INACT,
IOCON_FUNC2);
Chip_IOCON_PinMux(LPC_IOCON, 0, 11, IOCON_MODE_INACT,
IOCON_FUNC2);
Chip_IOCON_EnableOD(LPC_IOCON, 0, 10);
Chip_IOCON_EnableOD(LPC_IOCON, 0, 11);
break;
}
}
void Board_Buttons_Init(void)
{
Chip_GPIO_WriteDirBit(LPC_GPIO, BUTTONS_BUTTON1_GPIO_PORT_NUM,
BUTTONS_BUTTON1_GPIO_BIT_NUM, false);
}
uint32_t Buttons_GetStatus(void)
{
uint8_t ret = NO_BUTTON_PRESSED;
if (Chip_GPIO_ReadPortBit(LPC_GPIO, BUTTONS_BUTTON1_GPIO_PORT_NUM,
BUTTONS_BUTTON1_GPIO_BIT_NUM) == 0x00) {
ret |= BUTTONS_BUTTON1;
}
return ret;
}
/* Baseboard joystick buttons */
#define NUM_BUTTONS 5
static const uint8_t portButton[NUM_BUTTONS] = {
JOYSTICK_UP_GPIO_PORT_NUM,
JOYSTICK_DOWN_GPIO_PORT_NUM,
JOYSTICK_LEFT_GPIO_PORT_NUM,
JOYSTICK_RIGHT_GPIO_PORT_NUM,
JOYSTICK_PRESS_GPIO_PORT_NUM
};
static const uint8_t pinButton[NUM_BUTTONS] = {
JOYSTICK_UP_GPIO_BIT_NUM,
JOYSTICK_DOWN_GPIO_BIT_NUM,
JOYSTICK_LEFT_GPIO_BIT_NUM,
JOYSTICK_RIGHT_GPIO_BIT_NUM,
JOYSTICK_PRESS_GPIO_BIT_NUM
};
static const uint8_t stateButton[NUM_BUTTONS] = {
```

```
JOY_UP,
JOY_DOWN,
JOY_LEFT,
JOY_RIGHT,
JOY_PRESS
};
/* Initialize Joystick */
void Board_Joystick_Init(void)
{
int ix;
/* IOCON states already selected in SystemInit(), GPIO setup only.
Pullups
are external, so IOCON with no states */
for (ix = 0; ix < NUM_BUTTONS; ix++) {
Chip_GPIO_SetPinDIRInput(LPC_GPIO, portButton[ix],
pinButton[ix]);
}
}
/* Get Joystick status */
uint8_t Joystick_GetStatus(void)
{
uint8_t ix, ret = 0;
for (ix = 0; ix < NUM_BUTTONS; ix++) {
if ((Chip_GPIO_GetPinState(LPC_GPIO, portButton[ix],
pinButton[ix])) == false) {
ret |= stateButton[ix];
}
}
return ret;
}
void Serial_CreateStream(void *Stream)
{}
void Board_USBD_Init(uint32_t port)
{
/* VBUS is not connected on the NXP LPCXpresso LPC1769, so leave the
pin at default setting. */
/*Chip_IOCON_PinMux(LPC_IOCON, 1, 30, IOCON_MODE_INACT,
IOCON_FUNC2);*/ /* USB VBUS */
Chip_IOCON_PinMux(LPC_IOCON, 0, 29, IOCON_MODE_INACT, IOCON_FUNC1);
/* P0.29 D1+, P0.30 D1- */
Chip_IOCON_PinMux(LPC_IOCON, 0, 30, IOCON_MODE_INACT, IOCON_FUNC1);
LPC_USB->USBClkCtrl = 0x12; /* Dev, AHB clock enable
*/
while ((LPC_USB->USBClkSt & 0x12) != 0x12);
}
```

Step 4: New source code:

```
/*
```

```
 * @brief FreeRTOS Blinky example
 *
 * @note
 * Copyright(C) NXP Semiconductors, 2014
 * All rights reserved.
 *
 * @par
 * Software that is described herein is for illustrative purposes only
 * which provides customers with programming information regarding the
 * LPC products.  This software is supplied "AS IS" without any warranties of
 * any kind, and NXP Semiconductors and its licensor disclaim any and
 * all warranties, express or implied, including all implied warranties of
 * merchantability, fitness for a particular purpose and non-infringement of
 * intellectual property rights.  NXP Semiconductors assumes no responsibility
 * or liability for the use of the software, conveys no license or rights under any
 * patent, copyright, mask work right, or any other intellectual property rights in
 * or to any products. NXP Semiconductors reserves the right to make changes
 * in the software without notification. NXP Semiconductors also makes no
 * representation or warranty that such application will be suitable for the
 * specified use without further testing or modification.
 *
 * @par
 * Permission to use, copy, modify, and distribute this software and its
 * documentation is hereby granted, under NXP Semiconductors' and its
 * licensor's relevant copyrights in the software, without fee, provided that it
 * is used in conjunction with NXP Semiconductors microcontrollers.  This
 * copyright, permission, and disclaimer notice must appear in all copies of
 * this code.
 */

#include "board.h"
#include "FreeRTOS.h"
#include "task.h"

int red=0;
int green=1;
int blue=2;

static void prvSetupHardware(void)
{
	SystemCoreClockUpdate();
	Board_Init();
	/* Initial LED0 state is off */
	Board_LED_Set(red, false);
	Board_LED_Set(green, false);
	Board_LED_Set(blue, false);
}
/* LED1 toggle thread */
static void vLEDTask1(void *pvParameters)
{
	//bool LedState = false;
	while (1)
```

```
	{
		Board_LED_Set(red, true);
		vTaskDelay(configTICK_RATE_HZ);
		Board_LED_Set(red, false);
		vTaskDelay(5*configTICK_RATE_HZ );
	}
}
static void vLEDTask2(void *pvParameters)
{
	vTaskDelay(configTICK_RATE_HZ + configTICK_RATE_HZ/2 );

	while (1)
	{
		Board_LED_Set(green, true);
		vTaskDelay(configTICK_RATE_HZ);
		Board_LED_Set(green, false);
		vTaskDelay(5 * configTICK_RATE_HZ );
	}
}
static void vLEDTask3(void *pvParameters)
{
	vTaskDelay(3 * configTICK_RATE_HZ);
	while (1)
	{

		Board_LED_Set(blue, true);
		vTaskDelay(configTICK_RATE_HZ);
		Board_LED_Set(blue, false);
		vTaskDelay(5 * configTICK_RATE_HZ );
	}
}

/*****************************************************************************
 * Public functions
 ****************************************************************************/

/**
 * @brief	main routine for FreeRTOS blinky example
 * @return	Nothing, function should not exit
 */
int main(void)
{
	prvSetupHardware();

	/* LED1 toggle thread */
	xTaskCreate(vLEDTask1, (signed char* ) "vTaskLed1",
			configMINIMAL_STACK_SIZE,NULL, (tskIDLE_PRIORITY+3UL),
			(xTaskHandle *) NULL);

	/* LED2 toggle thread */
	xTaskCreate(vLEDTask2, (signed char* ) "vTaskLed2",
			configMINIMAL_STACK_SIZE,NULL, (tskIDLE_PRIORITY+2UL ),
```

```
                        (xTaskHandle *) NULL);

        xTaskCreate(vLEDTask3, (signed char* ) "vTaskLed3",
                        configMINIMAL_STACK_SIZE,NULL, (tskIDLE_PRIORITY+1UL),
                        (xTaskHandle *) NULL);

        /* Start the scheduler */
        vTaskStartScheduler();

        /* Should never arrive here */
        return 1;
}

/**
 * @}
 */
```

Step 5 + 6 +7 + 8: Review the Blinky_LEDs source code that we discussed for Lab #2. Do you need to change the prvSetupHardware() function definition based on the instructions for Lab #3? Why?

Answer:

We do not need to change the prvSetupHardware() function, since the instruction states that all LEDs have to be turned off at the beginning. But the vLEDTask* () functions for each LED have to be changed:

```
static void vLEDTask1(void *pvParameters)
{
        //bool LedState = false;
        while (1)
        {
                Board_LED_Set(red, true); //turn on LED
                vTaskDelay(configTICK_RATE_HZ);
                Board_LED_Set(red, false);//turn off LED
                vTaskDelay(3*configTICK_RATE_HZ + configTICK_RATE_HZ/2 );//delay for 0.5s
        }
}
static void vLEDTask2(void *pvParameters)
{
        vTaskDelay(configTICK_RATE_HZ + configTICK_RATE_HZ /2);

        while (1)
        {
                Board_LED_Set(green, true);//on
                vTaskDelay(configTICK_RATE_HZ);
                Board_LED_Set(green, false);//off
```

```
                vTaskDelay(3 * configTICK_RATE_HZ + configTICK_RATE_HZ/2 );
        }
}
static void vLEDTask3(void *pvParameters)
{
        vTaskDelay(3 * configTICK_RATE_HZ);
        while (1)
        {

                Board_LED_Set(blue, true);//on
                vTaskDelay(configTICK_RATE_HZ);
                Board_LED_Set(blue, false);//off
                vTaskDelay(3 * configTICK_RATE_HZ + configTICK_RATE_HZ/2 );
        }
}
```

Step 9: Changing the xTaskCreate() functions:

```
/* LED1 toggle thread */
        xTaskCreate(vLEDTask1, (signed char* ) "vTaskLed1",
                        configMINIMAL_STACK_SIZE,NULL, (tskIDLE_PRIORITY+3UL),//changed
priority
                        (xTaskHandle *) NULL);

        /* LED2 toggle thread */
        xTaskCreate(vLEDTask2, (signed char* ) "vTaskLed2",
                        configMINIMAL_STACK_SIZE,NULL, (tskIDLE_PRIORITY+2UL ),// changed
priority
                        (xTaskHandle *) NULL);

        xTaskCreate(vLEDTask3, (signed char* ) "vTaskLed3",
                        configMINIMAL_STACK_SIZE,NULL, (tskIDLE_PRIORITY+1UL),//changed
priority
                        (xTaskHandle *) NULL);
```

Step 10 Youtube link:
https://youtu.be/i981IS_efUI