ESE-3025 Embedded Real-Time Operating Systems

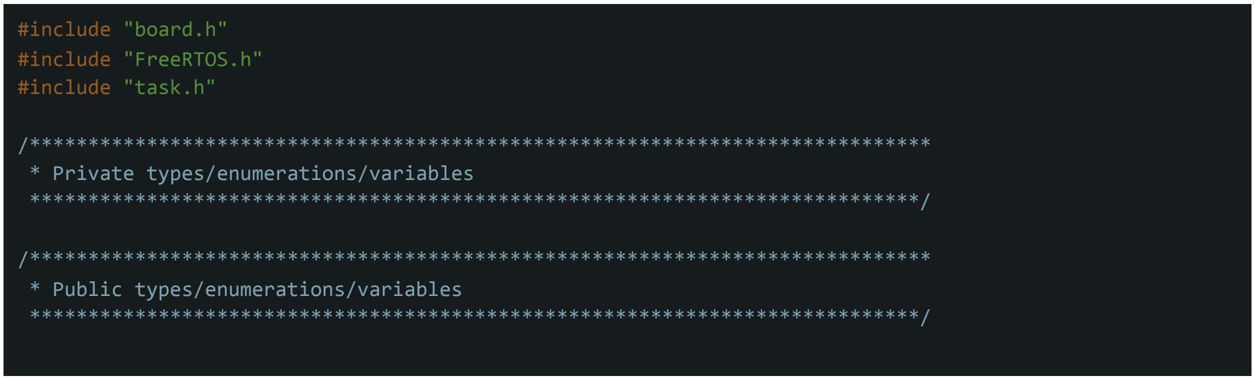
LAB 2

GROUP No. 2

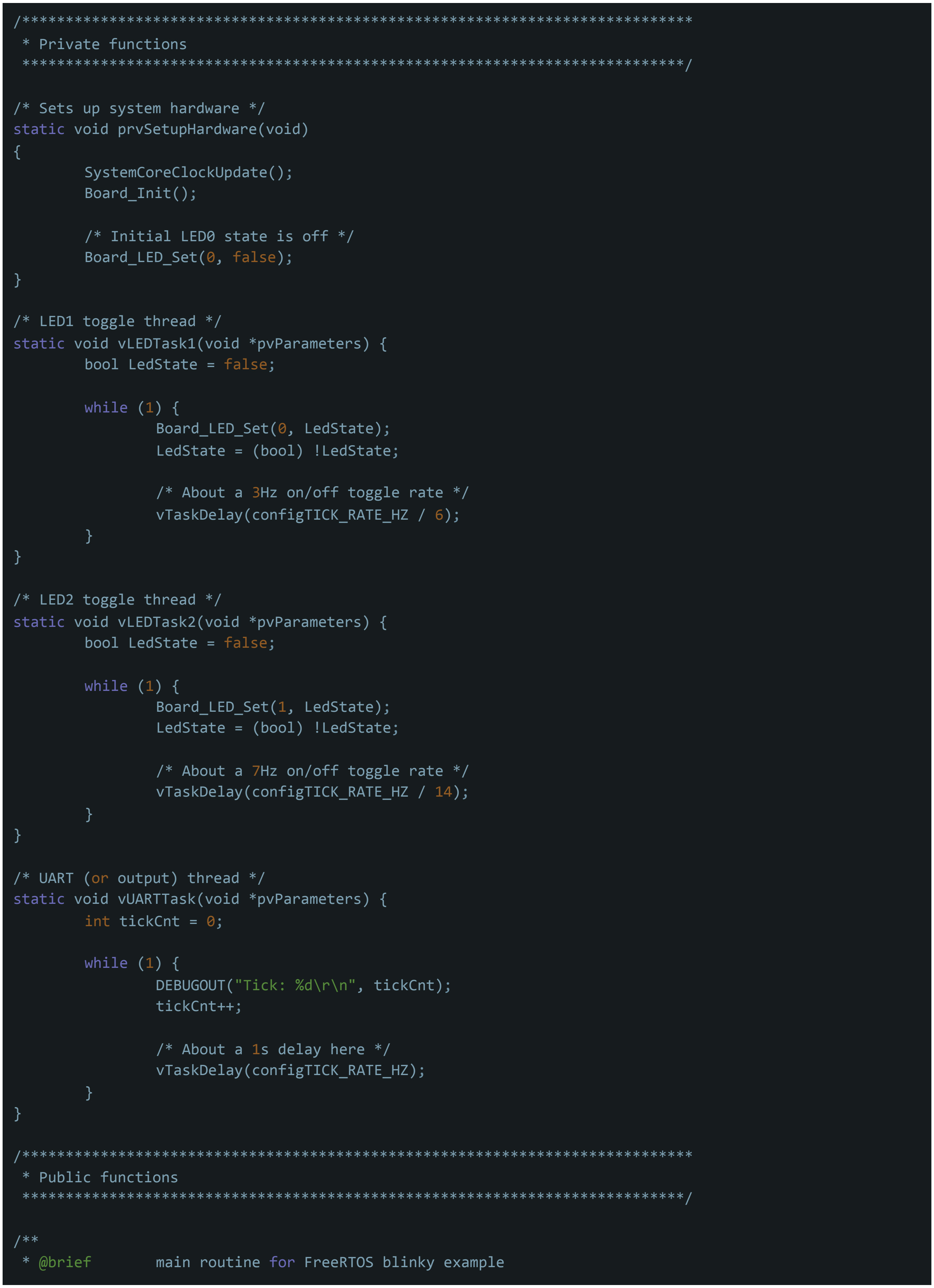
Group Members	Student ID
Christy Rachel Philip	C0765535
Fahad Rahman	C0769871
James M Chacko	C0777192
Premil Prasannan	C0777191
Vy Nguyen	C0776242

LED Blinking on LPC 1769 Using FreeRTOS

Step 1: Compiled and run freertos_blinky.c from the default freertos_blinky project folder



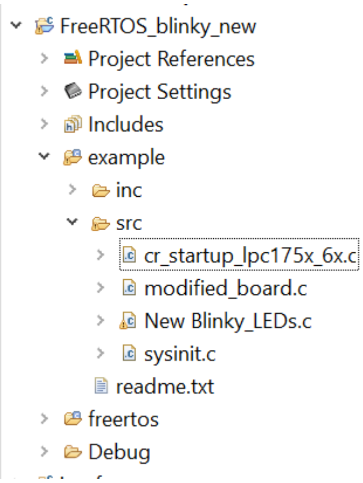
```
#include "board.h"
#include "FreeRTOS.h"
#include "task.h"

/*****************************************************************************
 * Private types/enumerations/variables
 ****************************************************************************/

/*****************************************************************************
 * Public types/enumerations/variables
 ****************************************************************************/
```



```c
/*****************************************************************************
 * Private functions
 ****************************************************************************/

/* Sets up system hardware */
static void prvSetupHardware(void)
{
        SystemCoreClockUpdate();
        Board_Init();

        /* Initial LED0 state is off */
        Board_LED_Set(0, false);
}

/* LED1 toggle thread */
static void vLEDTask1(void *pvParameters) {
        bool LedState = false;

        while (1) {
                Board_LED_Set(0, LedState);
                LedState = (bool) !LedState;

                /* About a 3Hz on/off toggle rate */
                vTaskDelay(configTICK_RATE_HZ / 6);
        }
}

/* LED2 toggle thread */
static void vLEDTask2(void *pvParameters) {
        bool LedState = false;

        while (1) {
                Board_LED_Set(1, LedState);
                LedState = (bool) !LedState;

                /* About a 7Hz on/off toggle rate */
                vTaskDelay(configTICK_RATE_HZ / 14);
        }
}

/* UART (or output) thread */
static void vUARTTask(void *pvParameters) {
        int tickCnt = 0;

        while (1) {
                DEBUGOUT("Tick: %d\r\n", tickCnt);
                tickCnt++;

                /* About a 1s delay here */
                vTaskDelay(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 + 1UL),
                                (xTaskHandle *) NULL);

        /* LED2 toggle thread */
        xTaskCreate(vLEDTask2, (signed char *) "vTaskLed2",
                                configMINIMAL_STACK_SIZE, NULL, (tskIDLE_PRIORITY + 1UL),
                                (xTaskHandle *) NULL);

        /* UART output thread, simply counts seconds */
        xTaskCreate(vUARTTask, (signed char *) "vTaskUart",
                                configMINIMAL_STACK_SIZE, NULL, (tskIDLE_PRIORITY + 1UL),
                                (xTaskHandle *) NULL);

        /* Start the scheduler */
        vTaskStartScheduler();

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

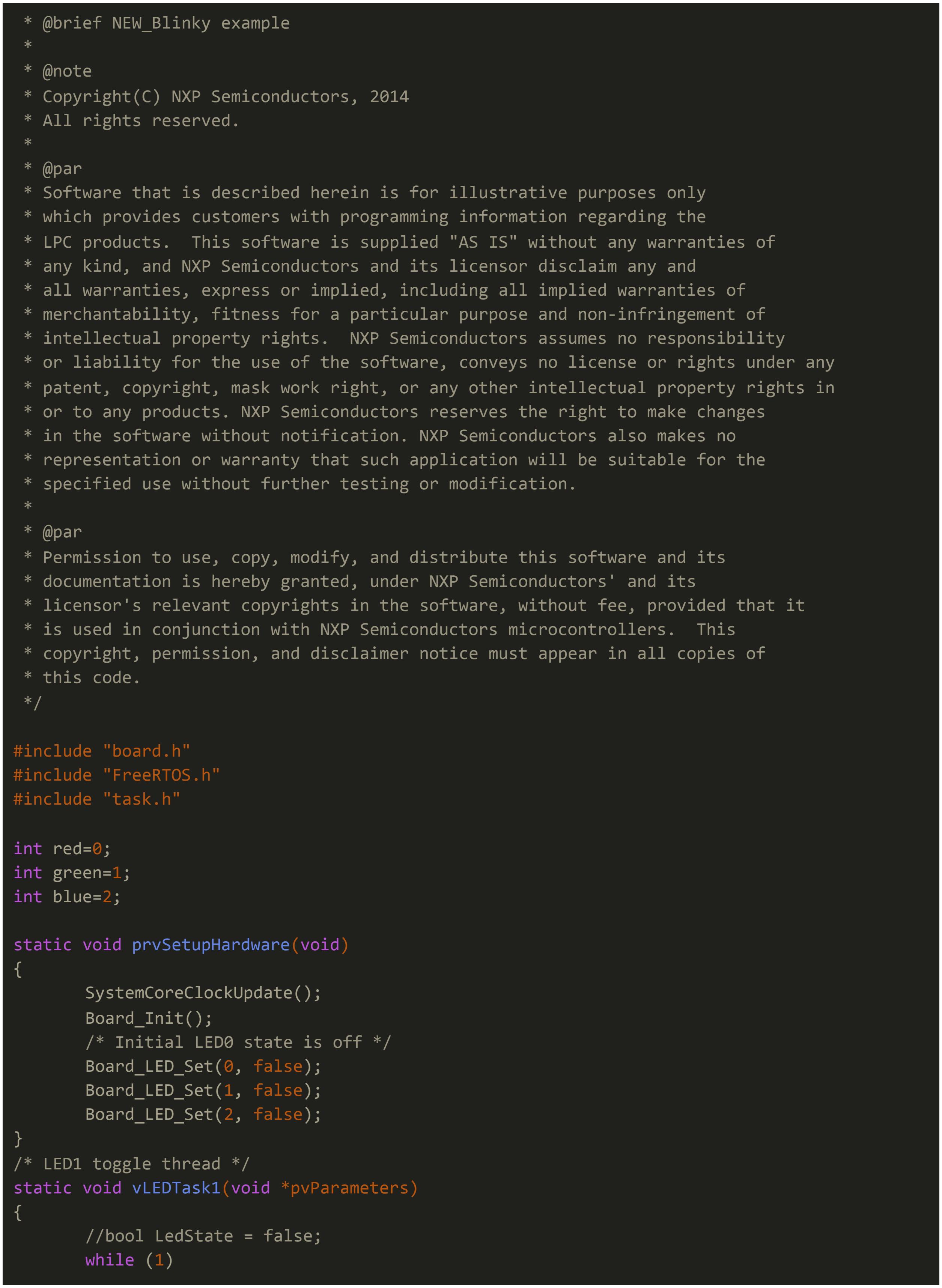
Here we got the result as RED LED blinking.

Step 2: Create a new project folder like freertos_blinky_new, add the new source code called New Blinky_LEDs.c to it.

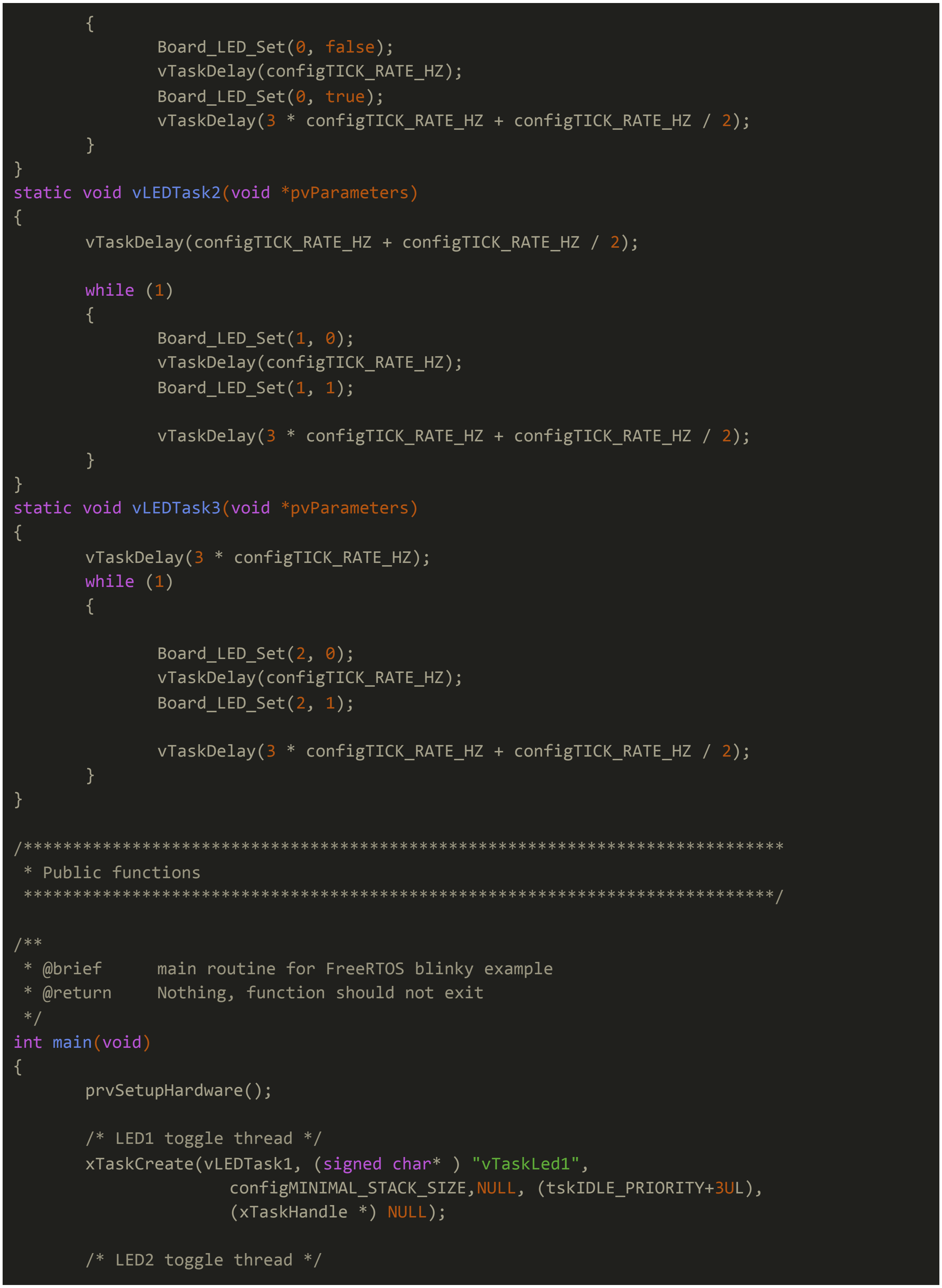


```
FreeRTOS_blinky_new
  Project References
  Project Settings
  Includes
  example
    inc
    src
      cr_startup_lpc175x_6x.c
      modified_board.c
      New Blinky_LEDs.c
      sysinit.c
    readme.txt
  freertos
  Debug
```

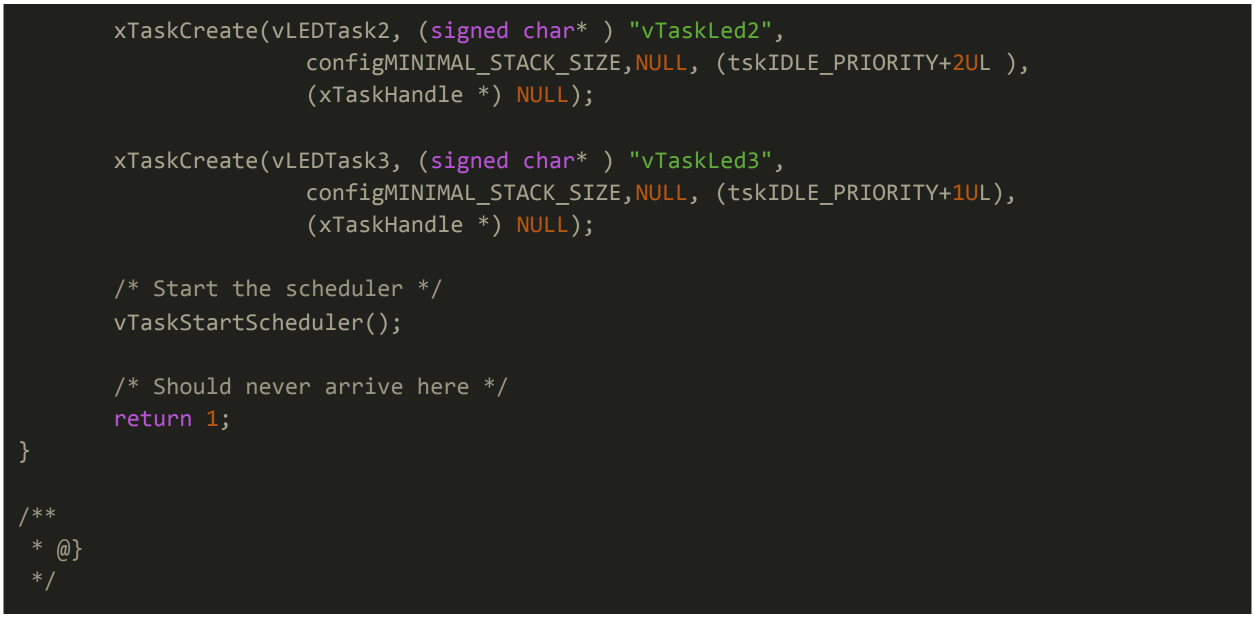
```
/*
```



```c
 * @brief NEW_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(0, false);
       Board_LED_Set(1, false);
       Board_LED_Set(2, false);
}
/* LED1 toggle thread */
static void vLEDTask1(void *pvParameters)
{
       //bool LedState = false;
       while (1)
```



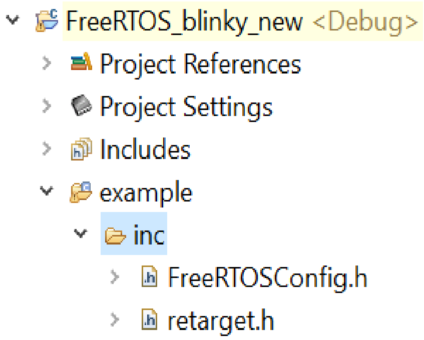
```c
        {
                Board_LED_Set(0, false);
                vTaskDelay(configTICK_RATE_HZ);
                Board_LED_Set(0, true);
                vTaskDelay(3 * configTICK_RATE_HZ + configTICK_RATE_HZ / 2);
        }
}
static void vLEDTask2(void *pvParameters)
{
        vTaskDelay(configTICK_RATE_HZ + configTICK_RATE_HZ / 2);

        while (1)
        {
                Board_LED_Set(1, 0);
                vTaskDelay(configTICK_RATE_HZ);
                Board_LED_Set(1, 1);

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

                Board_LED_Set(2, 0);
                vTaskDelay(configTICK_RATE_HZ);
                Board_LED_Set(2, 1);

                vTaskDelay(3 * configTICK_RATE_HZ + configTICK_RATE_HZ / 2);
        }
}

/*****************************************************************************
 * 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;
}

/**
 * @}
 */
```

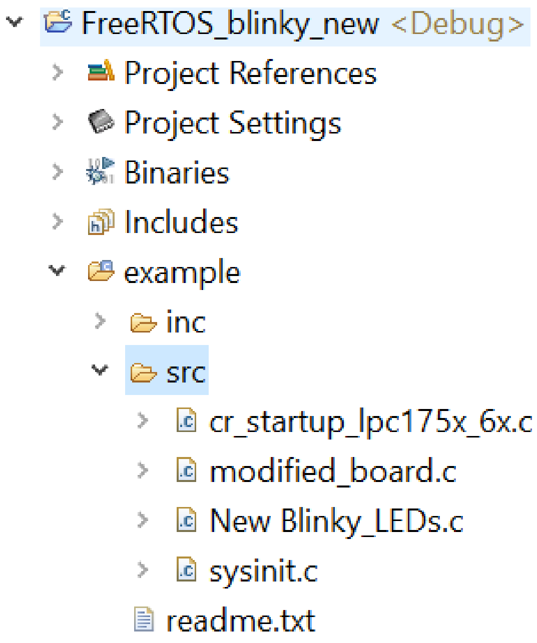
Step 3:

Add retarget.h to the new project folder.



```
FreeRTOS_blinky_new <Debug>
    Project References
    Project Settings
    Includes
    example
        inc
            FreeRTOSConfig.h
            retarget.h
```

Step 4: Clone the modified_board.c to this.



```
FreeRTOS_blinky_new <Debug>
    Project References
    Project Settings
    Binaries
    Includes
    example
        inc
        src
            cr_startup_lpc175x_6x.c
            modified_board.c
            New Blinky_LEDs.c
            sysinit.c
        readme.txt
```

Step 5: Compiled and run the result. Here we got blue and green as output, not red. When we compare the two outputs, we got only red led on the 1st step and two other LEDs blinked on the 5th step.

Step 6: Compare the freertos_blinky.c with the New Blinky_LED.c.

Code in the free_blinky has only two states while the new code has three states.

Freertos_blinky.c

```
static void prvSetupHardware(void)
{
	SystemCoreClockUpdate();
	Board_Init();

	/* Initial LED0 state is off */
	Board_LED_Set(0, false);
}
```

New Blinky_LED.c

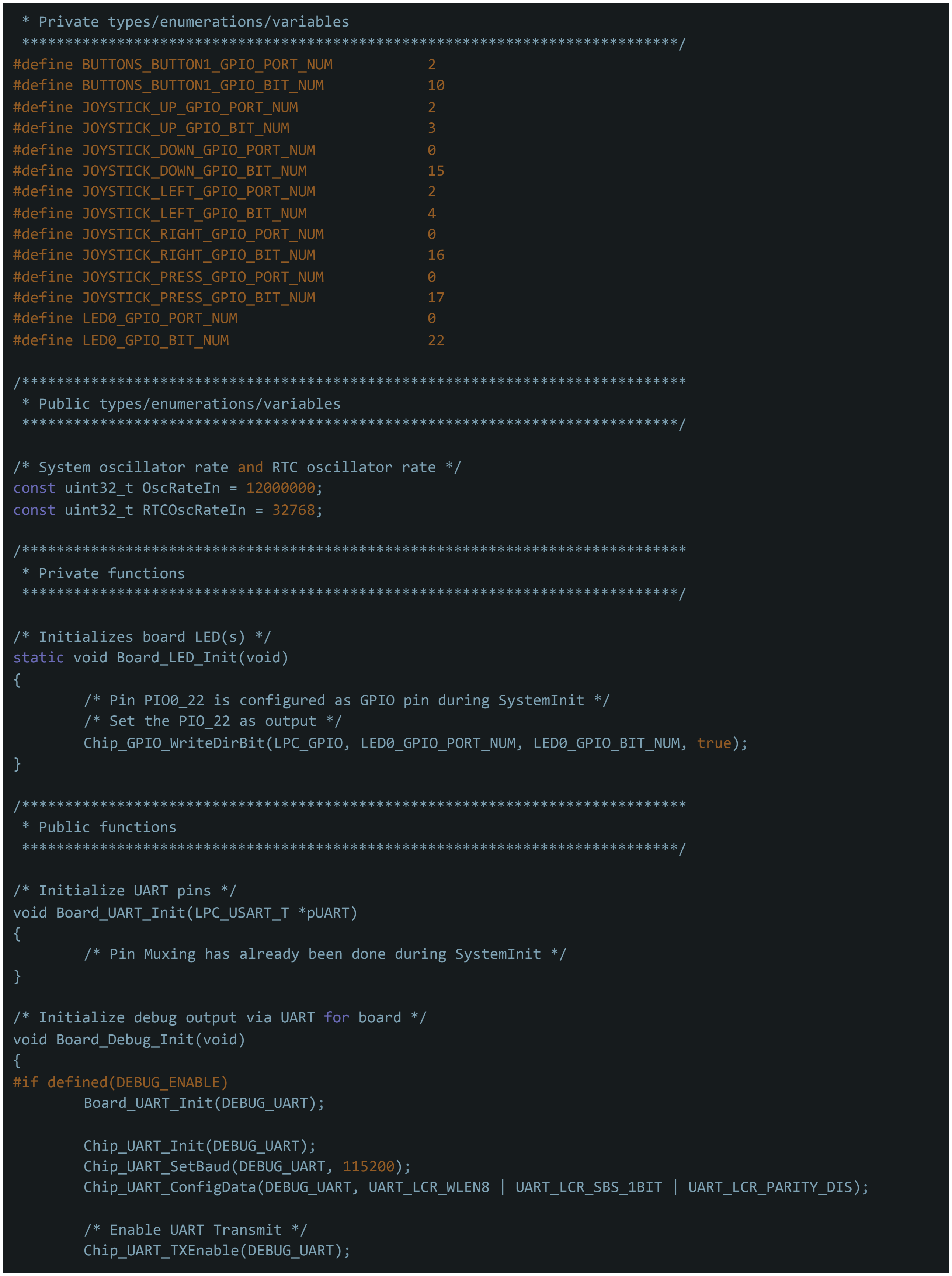
```
static void prvSetupHardware(void)
```

```
{
        SystemCoreClockUpdate();
        Board_Init();
        /* Initial LED0 state is off */
        Board_LED_Set(0, false);
        Board_LED_Set(1, false);
        Board_LED_Set(2, false);
}
```

Similarly, in the new blinky, we have three tasks as well, however, in freeRTOS blinky we have only one task.

Step 7: Compare the original board.c with the modified board.c. Board.c

```
/*
 * @brief NXP LPC1769 LPCXpresso board file
 *
 * @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"
#include "string.h"

#include "retarget.h"

/*****************************************************************************
```

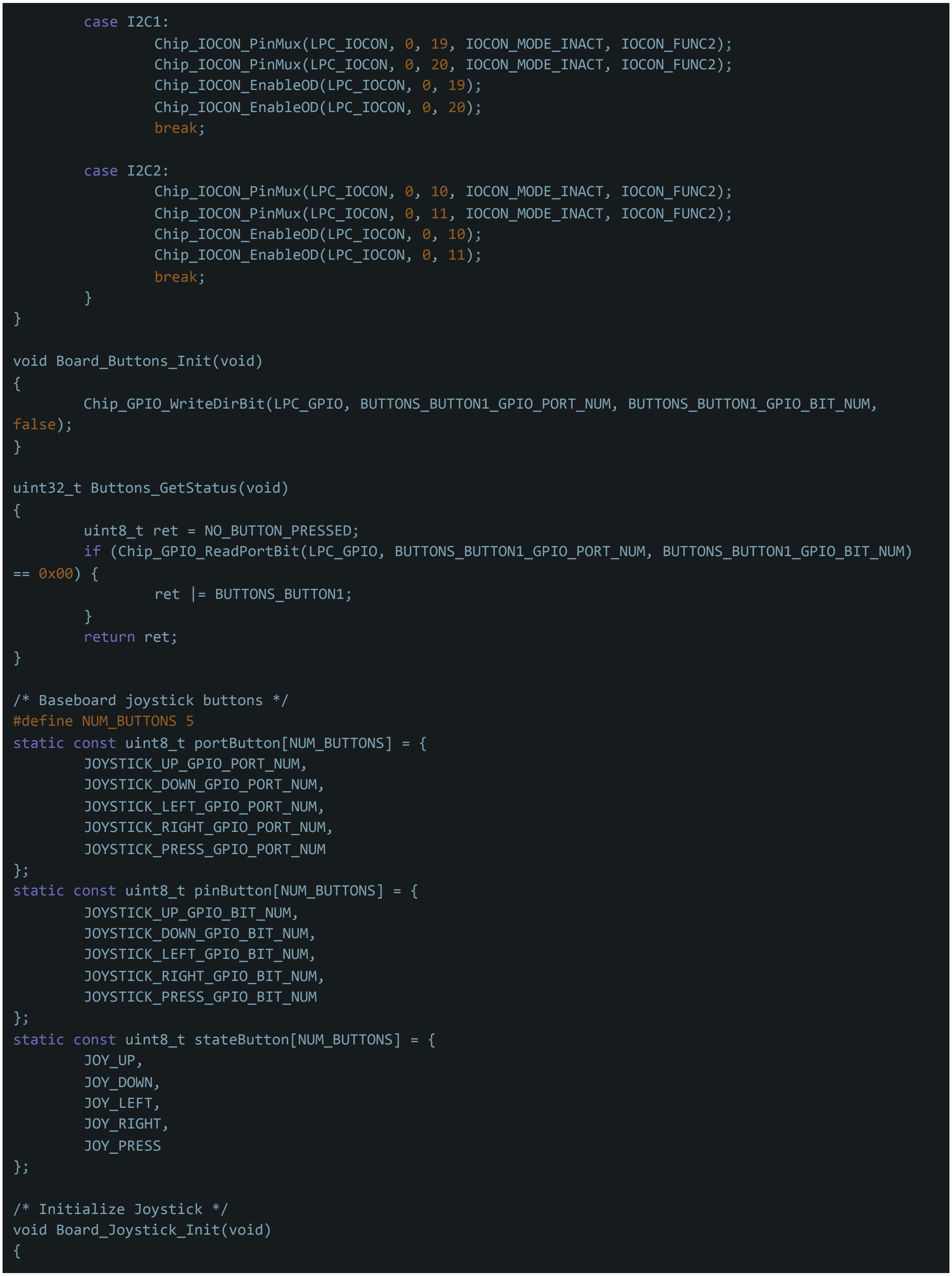


```
 * Private types/enumerations/variables
 ****************************************************************************/
#define BUTTONS_BUTTON1_GPIO_PORT_NUM           2
#define BUTTONS_BUTTON1_GPIO_BIT_NUM            10
#define JOYSTICK_UP_GPIO_PORT_NUM               2
#define JOYSTICK_UP_GPIO_BIT_NUM                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
#define LED0_GPIO_PORT_NUM                      0
#define LED0_GPIO_BIT_NUM                       22

/*****************************************************************************
 * 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);
}

/*****************************************************************************
 * 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)
{
        /* There is only one LED */
        if (LEDNumber == 0) {
                Chip_GPIO_WritePortBit(LPC_GPIO, LED0_GPIO_PORT_NUM, LED0_GPIO_BIT_NUM, On);
        }
}

/* 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);
        }

        return state;
}

void Board_LED_Toggle(uint8_t LEDNumber)
{
        if (LEDNumber == 0) {
                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 */
```

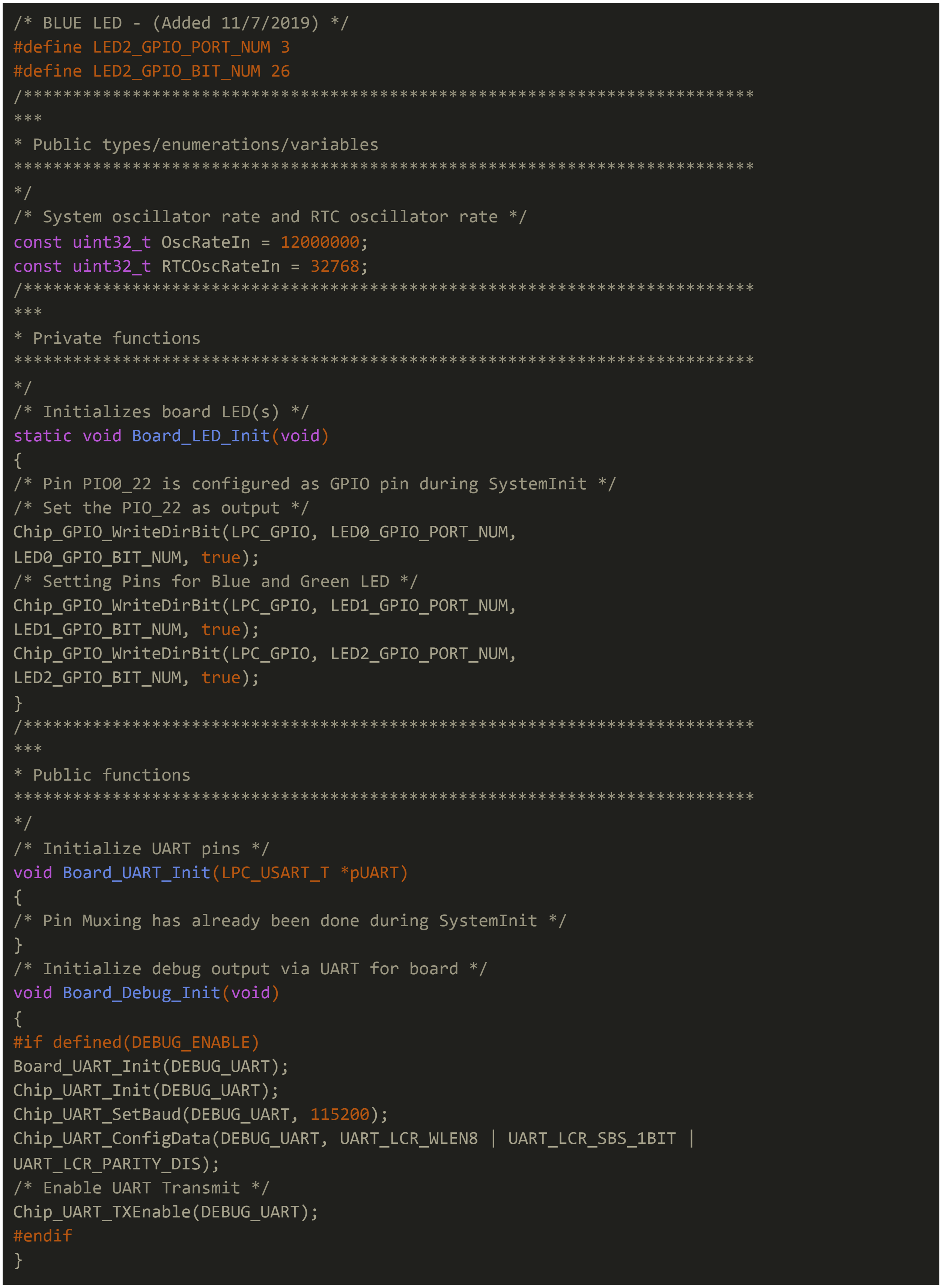
```c
void Board_SPI_Init(bool isMaster)
{
        /* Set up clock and muxing for SSP0 interface */
        /*
         * Initialize SSP0 pins connect
         * P0.15: SCK
         * P0.16: SSEL
         * P0.17: MISO
         * P0.18: MOSI
         */
        Chip_IOCON_PinMux(LPC_IOCON, 0, 15, IOCON_MODE_PULLDOWN, 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);
}
```

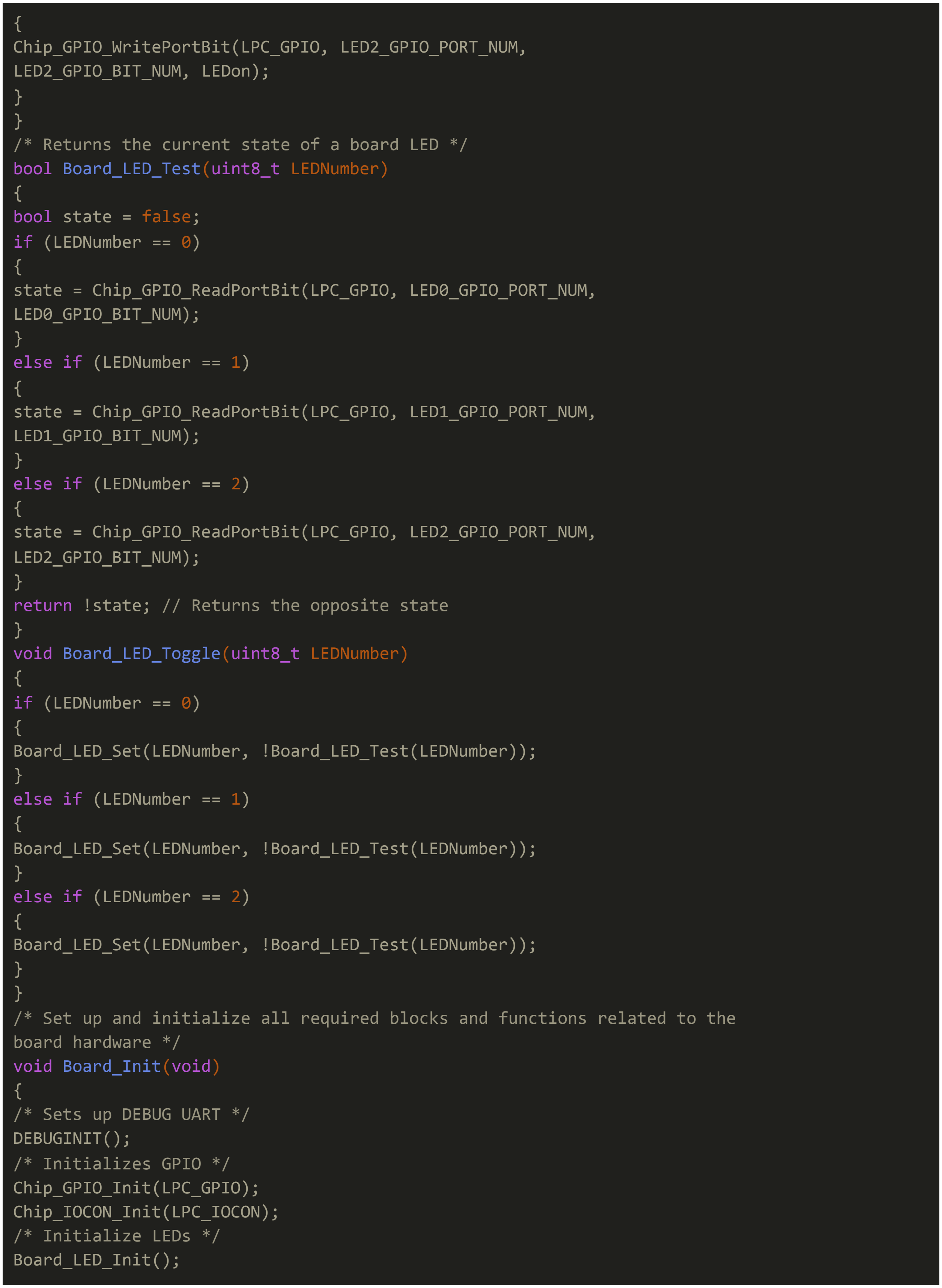
Modified_board.c:

```
/*
* @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
```

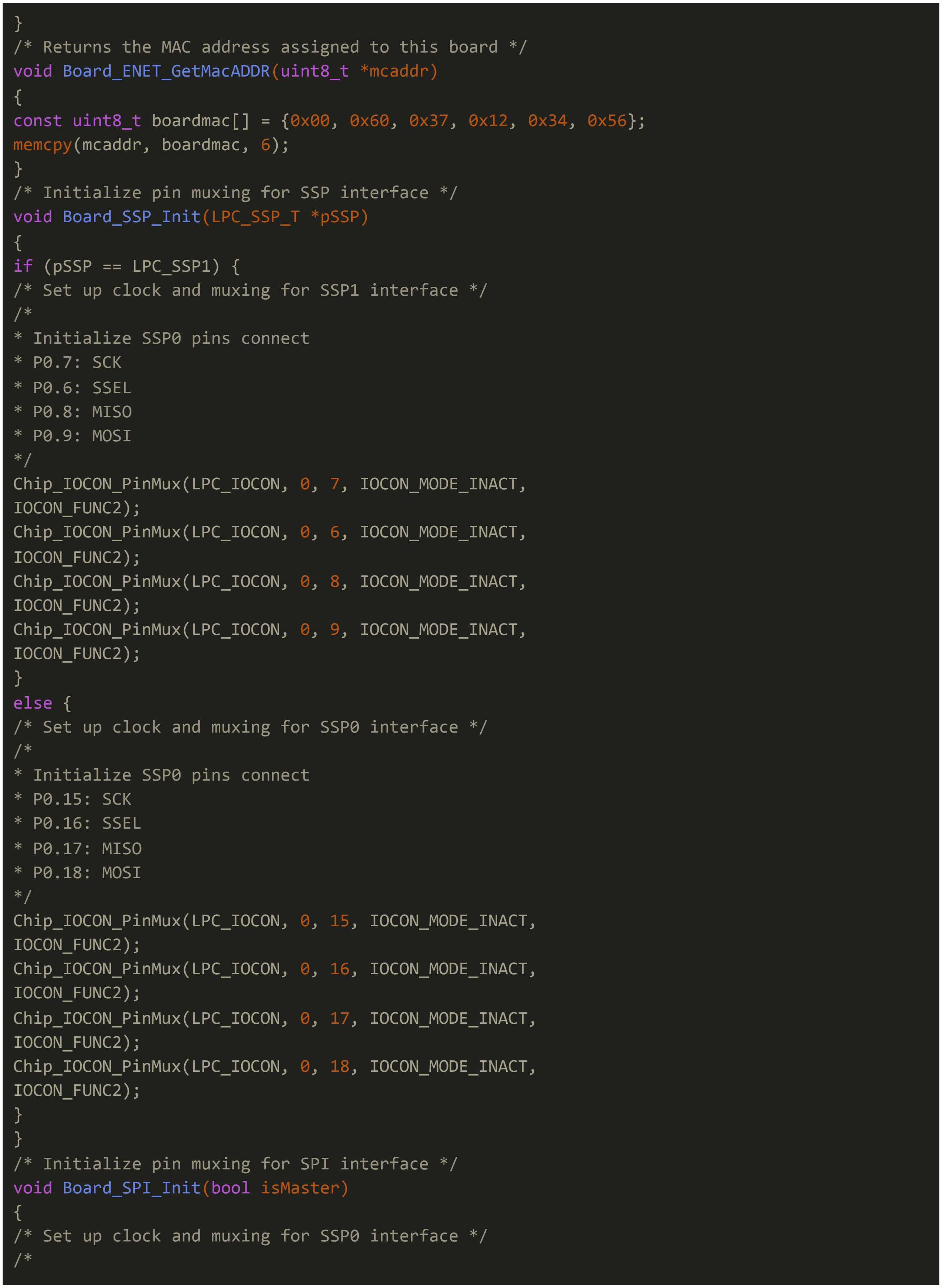


```
/* BLUE LED - (Added 11/7/2019) */
#define LED2_GPIO_PORT_NUM 3
#define LED2_GPIO_BIT_NUM 26
/**************************************************************************
***
* Public types/enumerations/variables
***************************************************************************
*/
/* System 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
}
```

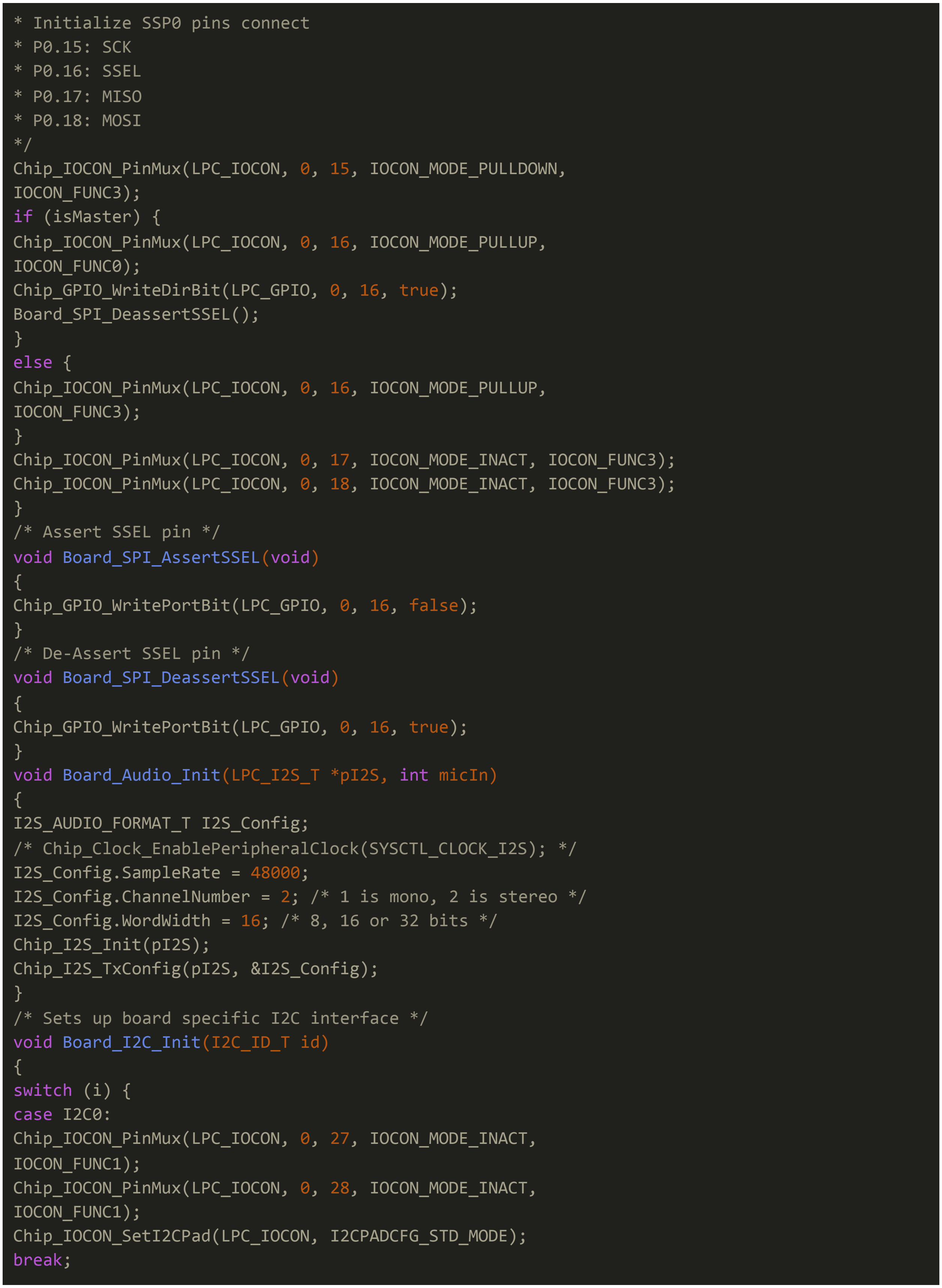
```c
/* Sends a character on the UART */
void Board_UARTPutChar(char ch)
{
#if defined(DEBUG_ENABLE)
while ((Chip_UART_ReadLineStatus(DEBUG_UART) & UART_LSR_THRE) == 0)
{}
Chip_UART_SendByte(DEBUG_UART, (uint8_t) ch);
#endif
}
/* Gets a character from the UART, returns EOF if no character is ready */
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 (i) {
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);
}
```

Youtube video link:

https://youtu.be/hvf6I9RlQUU