Lab(Updated Version of FreeBlinky)

ESE 3025: EmbeddedReal Time Operating Systems

Lambton College in Toronto

Instructor: Takis Zourntos

STUDENT NAME & ID:

Vishal Hasrajani(C0761544)

Parth Patel(C0764929)

Goutham Reddy Alugubelly(C0747981)

Ratnajahnavi rebbapragada(C0762196)

INTRODUCTION :

In this lab,we will use our modified board library to update the freeRTOS-Blinky code.
Further, we will explain how this FreeRtosBlinky code works.

DESCRIPTION :

To begin with,we updated the board library(for reference of the updated library you can look for the week5-updated board library section).Then we finally modified the freeRtosBlinky.c file as named it as **freertos_blinky_modified.c** :

Updates that we made are given below:

1)

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

    /* Initial  state is off */
    Board_LED_Set(red, false);
    Board_LED_Set(green, false);
    Board_LED_Set(blue, false);
}
```

Here this function **prvSetupHardware(void)** sets up the system hardware initially.
Then **SystemCoreUpdate()** is used to update the cpu core clock rate.

Board_Init() is used to set up and initialize all required blocks and functions related to board hardware.

Then we set LEDS(that are red,green and blue) to OFF state initially.

2)

```
/* red LED toggle thread */
static void vLEDTask1(void *pvParameters) {

     while (1) {
          Board_LED_Set(red, true); //red led ON
          vTaskDelay(configTICK_RATE_HZ );//1sec delay
          Board_LED_Set(red, false);//red led OFF
          vTaskDelay(configTICK_RATE_HZ );//1sec delay



     }
}
```

Now this if thread of red led to toggle.
As you can see here,at first we have set the red led to ON state and then we provided the delay that VTaskDelay(1000) which will block this thread for 1sec and then again we set the red LED to OFF state.To continue,we provided a delay of 1sec after the LED is in OFF state.

```
3)
/* green LED toggle thread */
static void vLEDTask2(void *pvParameters) {


     while (1) {

          Board_LED_Set(green, false); //green led ON
          vTaskDelay(configTICK_RATE_HZ );//1sec delay
          Board_LED_Set(green, true); //green led OFF
          vTaskDelay(configTICK_RATE_HZ );//1sec delay


     }
}
```

This is toggle thread for green LED.
What i have done is when the red LED is in OFF state,green LED is in ON state and vice-versa.
Doing so we can see the red and green LED turn ON in turn one after the other.

4)

```
int main(void)
{
     prvSetupHardware();

     /* red LED toggle thread */
     xTaskCreate(vLEDTask1, (signed char *) "vTaskLed1",
                    configMINIMAL_STACK_SIZE, NULL,
(tskIDLE_PRIORITY + 1UL),
                    (xTaskHandle *) NULL);
```

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


    /* Start the scheduler */
    vTaskStartScheduler();

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

So here,we are setting up the hardware and then we created 2 tasks in real time with the priority of 1.These threads can be controlled as per our convenience.

Further moving on ,to schedule these tasks we started the system scheduler by using the function **vTaskStartScheduler()**.

Output :

Youtube link ----)

https://www.youtube.com/watch?v=QslETnEkkQw

CONCLUSION :

Overall, we learned how this freeRtosBlinky code works (The IDE we used here is MCUExpresso IDE).We also modified our code using our updated LPC1769 board library and executed the operations successfully.

APPENDIX:

freertos_blinky_modified.c :

```
/*
 * @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"
```

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

*************************************************************
*****************/

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

*************************************************************
*****************/

/*************************************************************
******************
 * Private functions

*************************************************************
*****************/

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

    /* Initial  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) {

     while (1) {
          Board_LED_Set(red, true); //red led ON
          vTaskDelay(configTICK_RATE_HZ );//1sec delay
          Board_LED_Set(red, false);//red led OFF
          vTaskDelay(configTICK_RATE_HZ );//1sec delay



     }
}

/* LED2 toggle thread */
static void vLEDTask2(void *pvParameters) {


     while (1) {

          Board_LED_Set(green, false); //green led ON
          vTaskDelay(configTICK_RATE_HZ );//1sec delay
          Board_LED_Set(green, true); //green led OFF
          vTaskDelay(configTICK_RATE_HZ );//1sec delay


     }
}


/* UART (or output) thread */
```

```
/*************************************************************
******************
 * 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);


    /* Start the scheduler */
    vTaskStartScheduler();

    /* Should never arrive here */
```

```
    return 1;
}
/**
 * @}
 */
```