



Batch: A2 Roll No.: 16010322014
Experiment / assignment / tutorial No. 5
Grade: AA / AB / BB / BC / CC / CD / DD

Signature of the Staff In-charge with date

TITLE: Blink LED using FreeRTOS

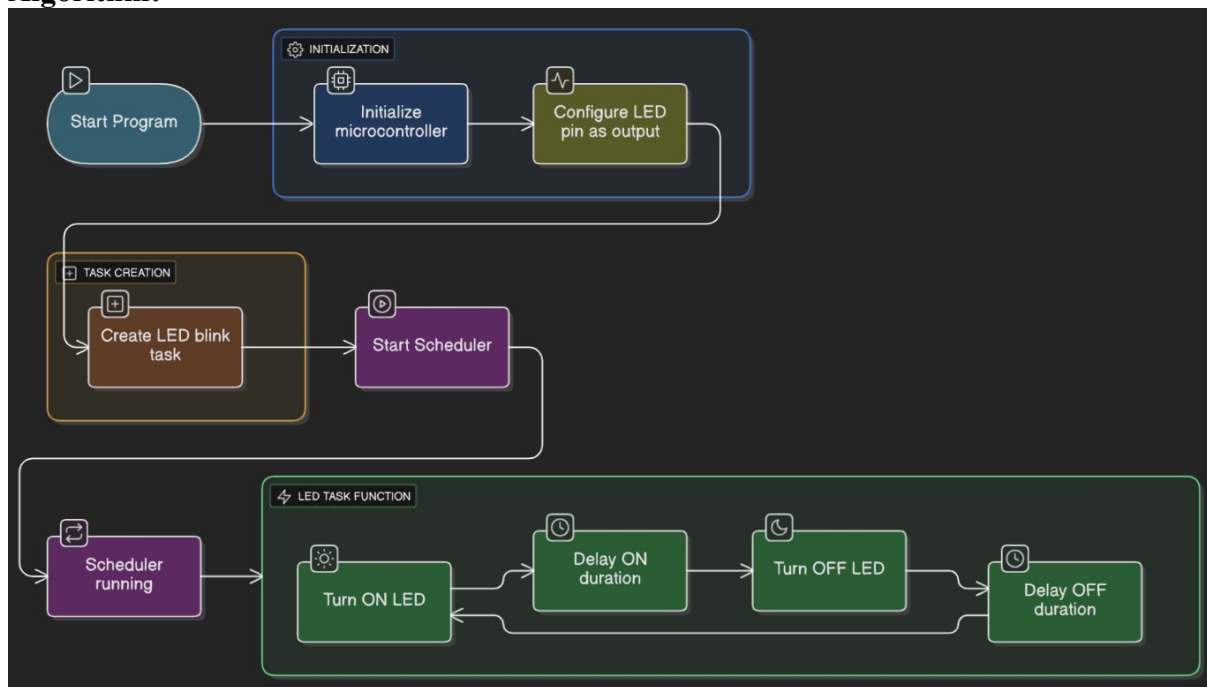
AIM: Write a program to blink LED connected at P0.31 using FreeRTOS API

OUTCOME: Students will be able to understand the open source RTOS and their usage

Tool Used: - Keil uVision, Flash Magic, FreeRTOS Source Code

Component required: - Development Board- LPC2148 ARM7 based board
UART to USB Converter- For UART0 Communication

Algorithm: -



Procedure: -

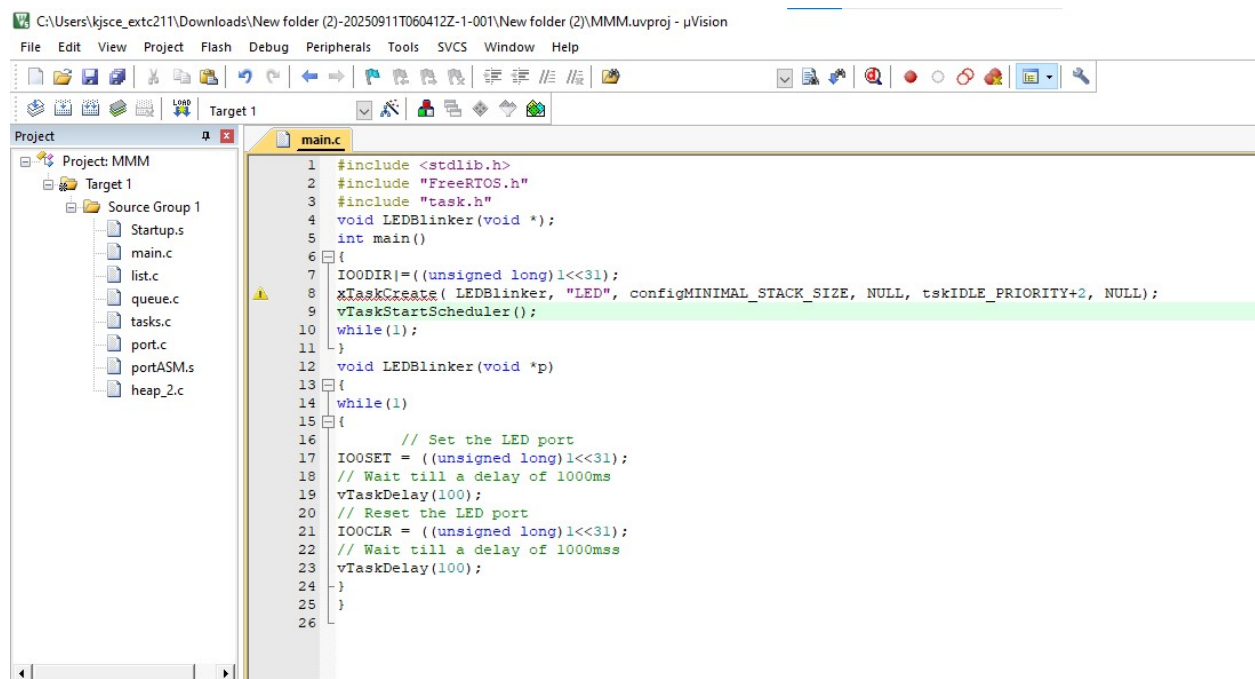
1. **Set up the development environment**
 - Open the project in Keil uVision
 - Include FreeRTOS source code (with correct port for LPC2148).
2. **Configure UART0 on LPC2148**
 - Initialize UART0 for communication.
 - Set baud rate, stop bits, and enable UART interrupts.
3. **Create FreeRTOS tasks**
 - **Task 1:** Waits on a binary semaphore and processes incoming data.
 - **Task 2 (ISR):** Triggered when UART0 receives data and gives the semaphore.
4. **Build and flash the code to LPC2148**
 - Use any LPC programmer to flash the compiled hex file.
5. **Test UART Communication**
 - Connect UART0 to PC via USB-UART adapter.

FreeRTOS API: -

xTaskCreate Used to create a new task.

vTaskStartScheduler Starts the RTOS scheduler, which begins executing the created tasks according to their priorities.

vTaskDelay Puts the calling task into the Blocked state for a specified number of tick periods.



```

C:\Users\kjsce_extc211\Downloads\New folder (2)-20250911T060412Z-1-001\New folder (2)\MMM.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Project: MMM
  Target 1
    Source Group 1
      Startup.s
      main.c
      list.c
      queue.c
      tasks.c
      port.c
      portASM.s
      heap_2.c

main.c
1  #include <stdlib.h>
2  #include "FreeRTOS.h"
3  #include "task.h"
4  void LEDBlinker(void *);
5  int main()
6  {
7      IOODIR |= ((unsigned long)1 << 31);
8      xTaskCreate(LEDBlinker, "LED", configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY+2, NULL);
9      vTaskStartScheduler();
10     while(1);
11 }
12 void LEDBlinker(void *p)
13 {
14     while(1)
15     {
16         // Set the LED port
17         IOOSET = ((unsigned long)1 << 31);
18         // Wait till a delay of 1000ms
19         vTaskDelay(100);
20         // Reset the LED port
21         IOOCLR = ((unsigned long)1 << 31);
22         // Wait till a delay of 1000ms
23         vTaskDelay(100);
24     }
25 }
26
  
```

Observations:

Flash Magic - NON PRODUCTION USE ONLY

File ISP Options Tools Help

Step 1 - Communications

Select Device...: LPC2148

COM Port: COM 28

Baud Rate: 9600

Interface: None (ISP)

Oscillator (MHz): 24

Step 2 - Erase

Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)

☒ Erase all Flash+Code Rd Prot
☐ Erase blocks used by Hex File

Step 3 - Hex File

Hex File: C:\Users\kjsce_extc211\Downloads\New folder (2)-20250911T06(Browse...

Modified: Thursday, September 11, 2025, 12:03:10 [more info](#)

Step 4 - Options

☒ Verify after programming
☐ Fill unused Flash
☐ Gen block checksums
☐ Execute

Step 5 - Start!

Start

Your Training or Consulting Partner: Embedded Systems Academy
www.esacademy.com

6

Build Output

```
*** Using Compiler 'V5.06 update 4 (build 422)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
Rebuild target 'Target 1'
assembling Startup.s...
compiling list.c...
assembling portASM.s...
compiling main.c...
compiling heap_2.c...
compiling queue.c...
compiling port.c...
compiling tasks.c...
linking...
Program Size: Code=5100 RO-data=32 RW-data=88 ZI-data=14752
FromELF: creating hex file...
".\Objects\MMM.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:01
```



Conclusion:

In this experiment, an LED was successfully blinked using the FreeRTOS API. The use of task creation, scheduling, and delay functions demonstrated real-time task management. Students gained practical understanding of open-source RTOS concepts and their applications in embedded systems.

Signature of faculty in-charge with date