|  |  |
|---|---|
| **Batch:** A2 | **Roll No.:**16010322014 |

**Experiment / assignment / tutorial No. 06**
**Grade: AA / AB / BB / BC / CC / CD /DD**
**Signature of the Staff In-charge with date**

**TITLE:** Inter-task communication using Queues
**AIM:** Write a FreeRTOS based program to send & receive message using queues & UART0 of LPC 2148

1. **OUTCOME:** Implement Open Source RTOS for resource sharing using inter task communication

**Components Required: -**

**Hardware:**

- LPC2148 Microcontroller Board
- UART0 interface (for serial communication)
- Serial communication cable (e.g., USB to UART converter)
- PC or terminal software (e.g., PuTTY, Tera Term) for UART communication

**Software:**

- Keil µVision IDE
- ARM Compiler

**Procedure:-**

**Step 1: Setup the Project**

- Create a new Keil project for LPC2148.
- Add the startup files and system initialization code.
- Add FreeRTOS source files to the project.
- Configure the clock and peripheral initialization, especially UART0.

**Step 2: Initialize UART0**

- Configure UART0 baud rate, data bits, parity, stop bits.
- Enable UART0 interrupts if needed.

**Step 3: Create FreeRTOS Tasks**

- Create two tasks:
  - **Sender Task:** Sends messages to a queue.
  - **Receiver Task:** Receives messages from the queue and sends them over UART0.

### Step 4: Create a Queue

- Define and create a queue with `xQueueCreate()` to hold messages

### Step 5: Implement Task Functions

- **Sender Task:**
  - Periodically sends messages to the queue using `xQueueSend()` or `xQueueSendToBack()`.

- **Receiver Task:**
  - Waits on the queue using `xQueueReceive()`.
  - When a message is received, transmits it via UART0.

### Step 6: Start the Scheduler

- Call vTaskStartScheduler() to start the FreeRTOS scheduler.

### Step 7: Debug and Test

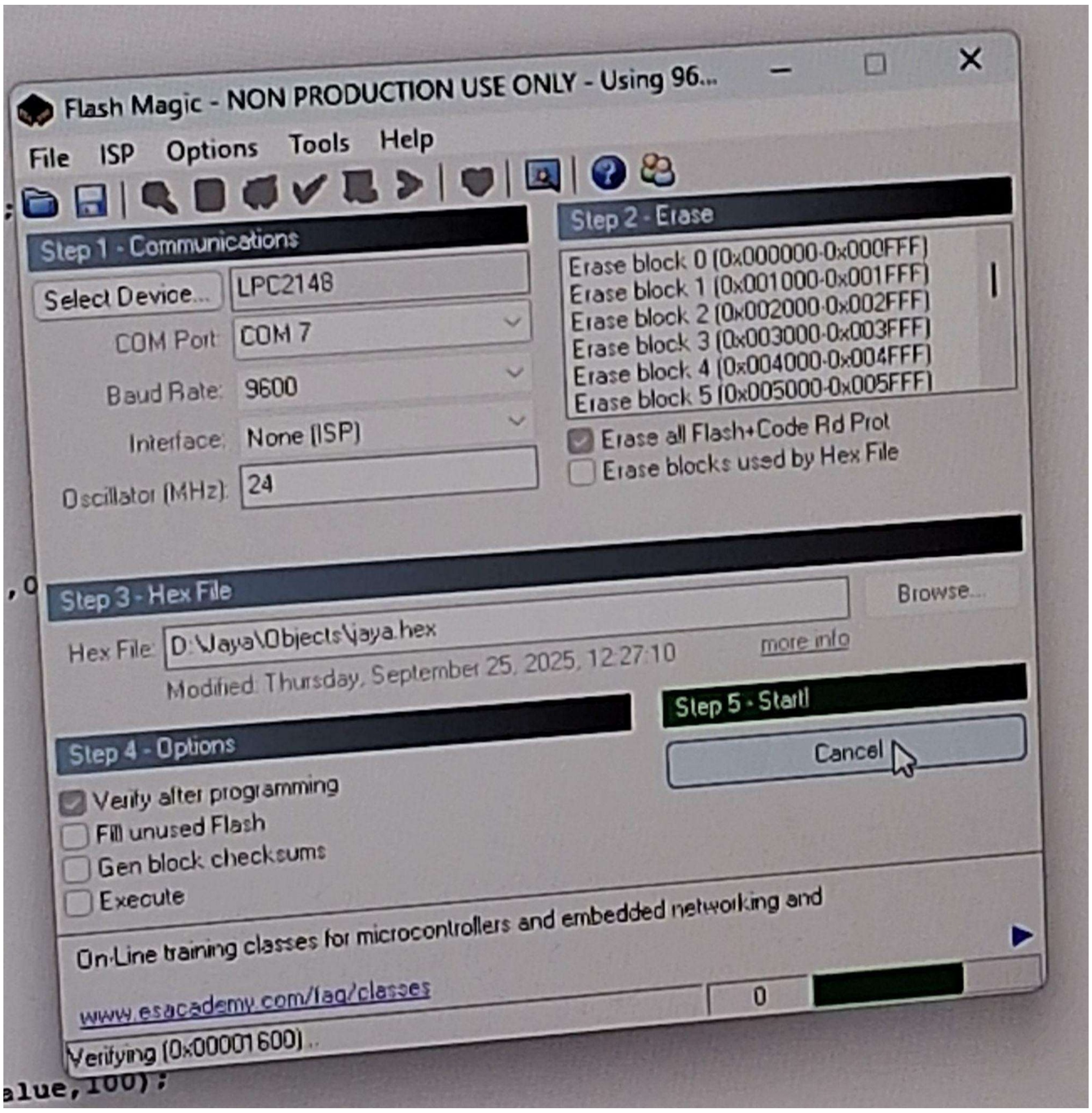
- Connect UART0 to PC terminal software.
- Build and flash the program to LPC2148.
- Observe messages sent from the receiver task on the PC terminal.

## Mention and Describe the FreeRTOS APIs related to Queue

| | |
|---|---|
| xQueueCreate() | Creates a new queue instance. You specify the queue length (number of items) and item size. |
| xQueueSend() | Sends an item to the back of the queue from a task. Blocks or returns immediately based on timeout. |
| xQueueSendToBack() | Alias for xQueueSend(), adds item to the back of the queue. |

| | |
|---|---|
| xQueueSendToFront() | Sends an item to the front of the queue, useful for priority messages. |
| xQueueReceive() | Receives (reads) an item from the queue, optionally blocking until an item becomes available. |
| xQueuePeek() | Peeks at the item at the front of the queue without removing it. |
| uxQueueMessagesWaiting() | Returns the number of messages currently stored in the queue. |
| vQueueDelete() | Deletes a queue and frees up resources. |

## Observations: -



Flash Magic - NON PRODUCTION USE ONLY - Using 96...

File ISP Options Tools Help

Step 1 - Communications

Select Device... LPC2148

COM Port: COM 7

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: D:\Jaya\Objects\jaya.hex Browse...

Modified: Thursday, September 25, 2025, 12:27:10 more info

Step 4 - Options

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

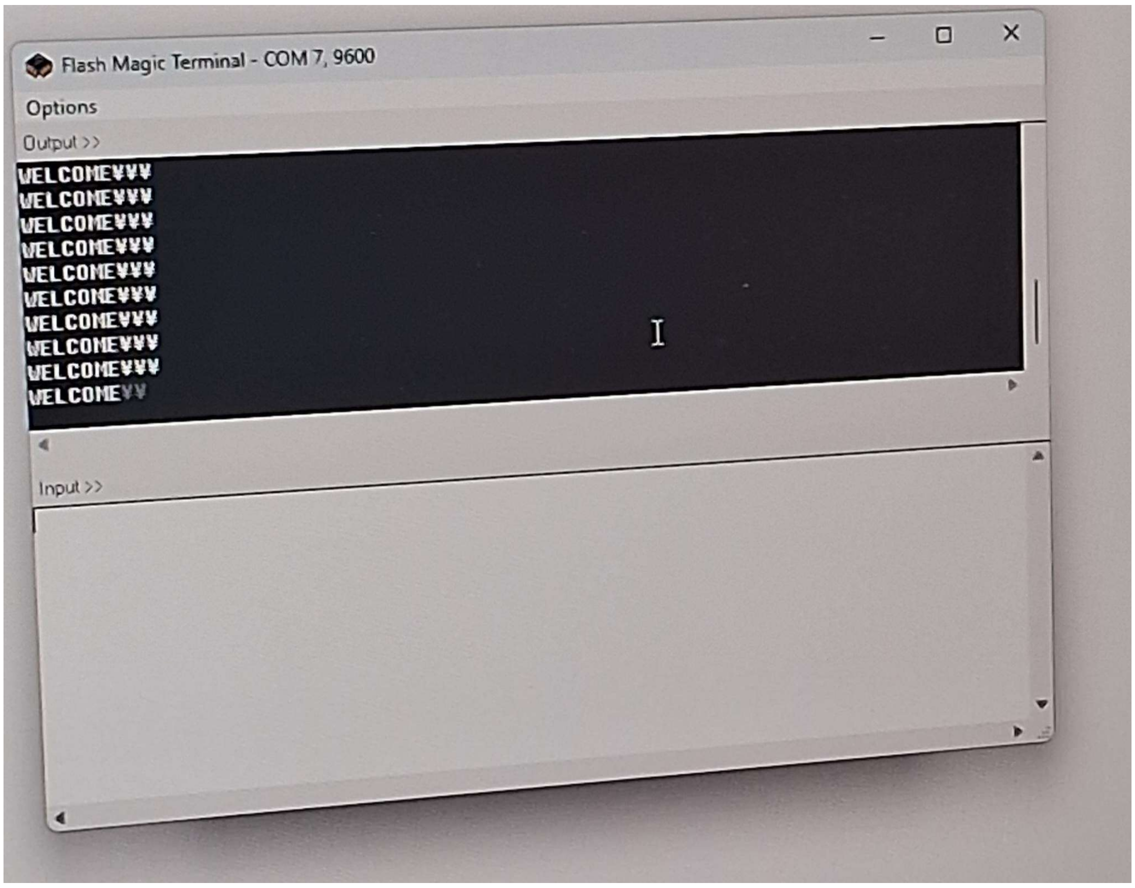
Step 5 - Start!

Cancel

On-Line training classes for microcontrollers and embedded networking and

www.esacademy.com/faq/classes

Verifying (0x00001600)... 0



```
WELCOME¥¥¥
WELCOME¥¥¥
WELCOME¥¥¥
WELCOME¥¥¥
WELCOME¥¥¥
WELCOME¥¥¥
WELCOME¥¥¥
WELCOME¥¥¥
WELCOME¥¥¥
WELCOME¥¥
```

**Conclusion:**

The program successfully demonstrates inter-task communication using FreeRTOS queues, enabling safe and efficient message passing between tasks. This implementation highlights how RTOS facilitates resource sharing and synchronization in embedded systems.

**Signature of faculty in-charge with date**