Name: Nivedita Londhe
PRN: 22420003
Batch: EN-4
Practical No. 10

Title: Program to illustrate mailbox.

Objective

- To deeply understand the concept of mailbox.

Mailbox:

uCOS-II (MicroCOS-II) is a real-time operating system (RTOS) designed for embedded systems. In uCOS-II, a mailbox is a communication mechanism that allows tasks to exchange messages or data with each other. The mailbox is a way for tasks to communicate asynchronously, meaning that tasks can send and receive messages without having to be synchronized in time.

- **Purpose:** The purpose of a mailbox is to provide a way for tasks to send messages to each other. A task can post a message to a mailbox, and another task can read or retrieve that message from the mailbox.
- **Data Structure:** In uC/OS-II, a mailbox is typically implemented as a data structure that can hold messages. The structure may include fields for the message itself, the sender's task identifier, and other relevant information.
- **API Functions:** uC/OS-II provides specific API functions for working with mailboxes.

The key functions include:

- **OSMboxCreate:** Creates a mailbox.
- **OSMboxDel:** Deletes a mailbox.
- **OSMboxPost:** Posts a message to a mailbox.
- **OSMboxPend:** Waits for a message to be available in a mailbox and retrieves it.

- **Blocking and Non-Blocking Operations:** Tasks can use OSMboxPend to wait for a message to be available in the mailbox. This function can be set to either block the task until a message is available or return immediately if no message is present (non-blocking).
- **Message Passing:** The messages exchanged through a mailbox can be of any data type, depending on the application's requirements. The sender and receiver tasks need to agree on the format and interpretation of the messages.

Programs to understand use of Mailbox:

Here, there are two tasks namely Task0 and Task1.

The Task0 sends message to Task1 i.e., a variable ‘c’ is sent from Task0 to Task1. The value of variable ‘c’ will define the number of cycles will the port pin related to task1 will have.

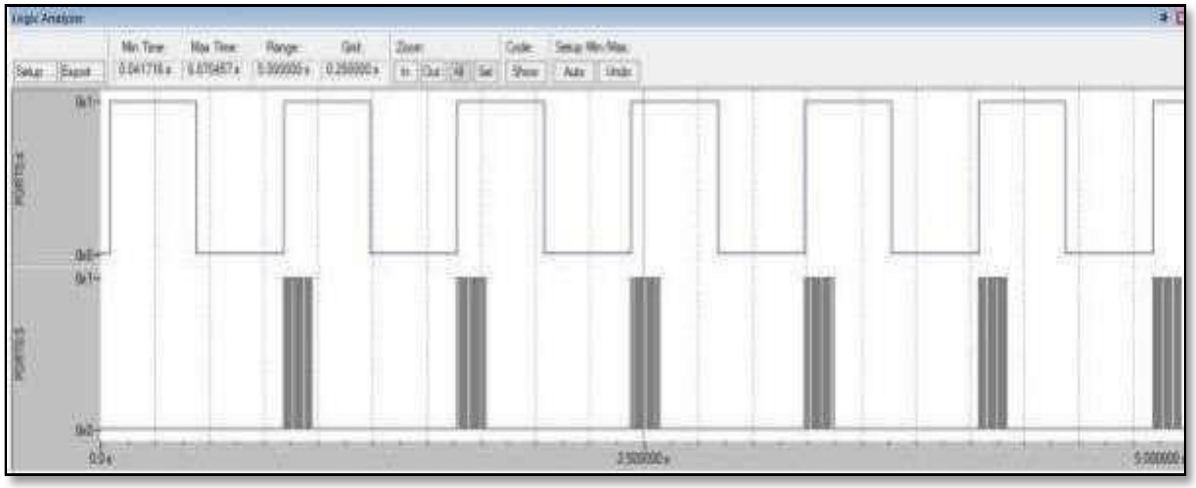
Code

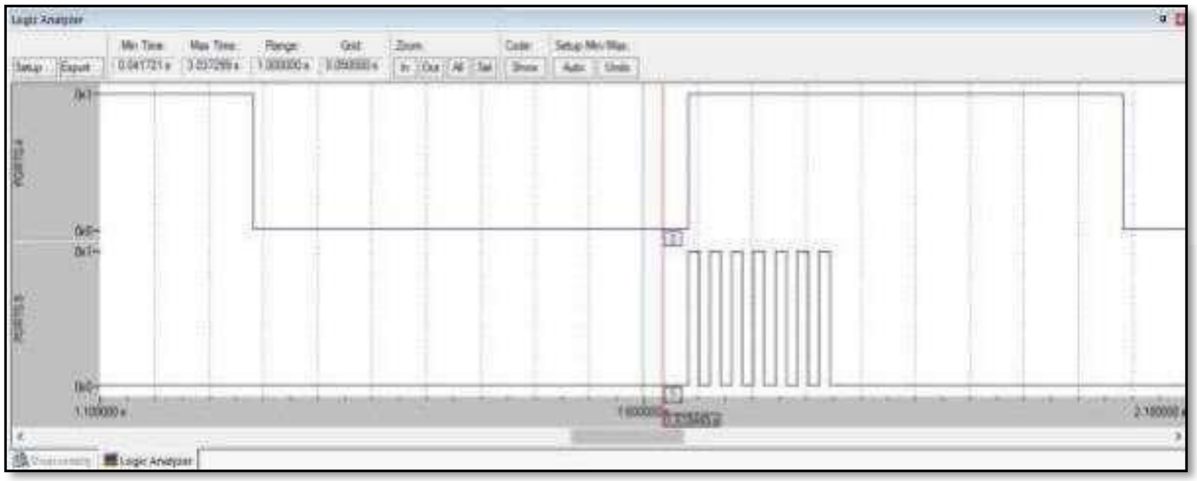
```
#include "config.h"
#include "stdlib.h"
#include <stdio.h>
#define TaskStkLengh 64 //Define the Task0 stack length
OS_STK TaskStk0 [TaskStkLengh];     //Define the Task stack OS_STK TaskStk1 [TaskStkLengh];
void Task0(void *pdata);
void Task1(void *pdata);
OS_EVENT *MyMailBox;    // mail box
uint8 err;
// Required for sending time to serial port
char buffer[25];
int main (void){
    LED_init(); UART0_Init();
    UART0_SendData("\r\n*************************************\r\n");
    UART0_SendData ("* Program for demo of Mailbox *\r\n");
    UART0_SendData ("*************************************\r\n");
    TargetInit();
    OSInit ();
    MyMailBox = OSMboxCreate((void*)0); // create mail box with no message
    OSTaskCreate (Task0,(void *)0, &TaskStk0[TaskStkLengh - 1], 6);
    OSTaskCreate (Task1,(void *)0, &TaskStk1[TaskStkLengh - 1], 7); OSStart();
    return 0;
}


// Task 0
void Task0 (void *pdata)
{
    unsigned int c; int i;
    pdata = pdata; /* Dummy data */
    while(1){
        c = 12; LED_on(0); OSTimeDly(40);
        LED_off(0); OSTimeDly(40);
        OSMboxPost(MyMailBox, &c);
    }
}

//Task 1
void Task1 (void *pdata){
    unsigned int* ptr; int i;
    unsigned int b;
    pdata = pdata; /* Dummy data */
    while(1){
      ptr = OSMboxPend(MyMailBox, 0, &err);
        b=*ptr;
        for(i=0;i<b-5;i++){
            LED_on(1); OSTimeDly(1);
            LED_off(1); OSTimeDly(1);
        }
    }
}
```

Output:





Conclusion:

1) In OS tasks can communicate within themselves.

2) The concept of mailbox is used to send and receive inter task messages.