SVPM'S

COLLEGE OF ENGINEERING, MALEAON (BK)

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

SUBJECT: EMBEDDED SYSTEMS & RTOS

Experiment No: **Title: MESSAGE QUEUE**

Class: BE E&TC

TITLE: Message Queue using Ucos II RTOS and microcontroller ARM 7TDMI family as device LPC2148

OJECTIVE: To Study Message Queue using Ucos II RTOS for message passing on ARM7 family as device LPC2148

AIM: To write a C program using U-COS-II RTOS that create two tasks generating outputs using Message Queue functions of Ucos II

SOFTWARE USED:

- SCARM(IDE)
- Flash Magic.

HARDWARE USED:

- Educational practice board for LPC2148(ARM7 kit)
- Adapter(9V DC,500mA)
- RS232
- PC

THEORY:

Message queue (or simply a mailbox) is a µC/OS-II object that allows a task or an ISR to send a pointer size variable to another task.The pointer would typically be initialized to point to some application specific data structure containing a 'message'. A message queue is used to send one or more messages to a task.

A message queue is basically an array of mailboxes.

Through a service provided by the kernel, a task or an ISR can deposit a message (the pointer) into a message queue.

Similarly, one or more tasks can receive messages through a service provided by the kernel.

Both the sending task and receiving task or tasks have to agree as to what the pointer is actually pointing to.

Generally, the first message inserted in the queue will be the first message extracted from the queue (FIFO).

In addition, to extract messages in a FIFO fashion, µC/OS-II allows a task to get messages Last-In-First-Out (LIFO).

Kernels typically provide the following mailbox services.

- Initialize the contents of a queue. The queue initially may or may not contain a message.
- Deposit a message into the queue (POST).
- Wait for a message to be deposited into the queue (PEND).
- Get a message from a queue if one is present,

PROCEDURE:

1. Connect 9V DC Power Supply to the trainer kit.
2. Connect RS232 to evaluation board and com port of PC.
3. Open Side_arm software.

4. Create a new project. => Select manufacture- Phillips
    - ⇨ select microcontroller- LPC 2148
5. Then in editor window type main program.
6. If header files are included then add those files in ‘c files’ tab in workspace window.
7. Build and compile (Check if any errors are present and remove them)
8. Hex file will be created.
9. Now open flash magic and switch on the board
10. Step 1 => microcontroller ARM7 family, LPC2148
    - ⇨ =>Com port- Select proper com port
    - ⇨ Baud rate- 19200 (min) , 38400(max)
    - ⇨ Interface- none (ISP)
    - ⇨ Oscillator (MHz)- 12
11. Step 2 => Check ‘Erase all blocks used by Hex file’
12. Step 3 => Add respective Hex file using ‘Browse’ option
13. Click On ‘start’.
14. Thus the Hex file will get downloaded in the memory of the device.
15. Switch on respective switches present on the board
16. You will see the output
17. **Included files:**

Header files	Source files
includes.h	APP.C

OUTPUT:

1 .Task 1 Deposit message into the Queue and displays on terminal after depositing each message

2 .Task 2 reads the message and displays it on terminal

CONCLUSION:

Program:

```
#include "includes.h"


OS_STK Task1Stk[1024];
OS_STK Task2Stk[1024];

OS_EVENT  *CommQ;
void *CommMsg[10];

int  main (void)
{
    BSP_Init();                          /* Initialize BSP functions
*/
    //InitUart0();
    OSInit();                            /* Initialize "uC/OS-II, The
Real-Time Kernel"              */

      CommQ = OSQCreate(&CommMsg[0],10);          /* Create Comm Q */

      OSTaskCreate(App_Task1,(void *)0,&Task1Stk[1023],6);
    OSTaskCreate(App_Task2,(void *)0,&Task2Stk[1023],7);

    OSStart();                                  /* Start multitasking (i.e.
give control to uC/OS-II)       */
    while(1);

}

INT8U i;
INT8U err;
INT8U CommRxBuf[10] = {1,2,3,4,5,6,7,8,9,0};


void  App_Task1 (void *p_arg)
{


    while(1)
    {
      for(i=0;i<12;i++)
      {
            err = OSQPost(CommQ, (void *)&CommRxBuf[i]);

         switch (err) {
            case OS_NO_ERR:
                  printf("Message was deposited into queue \n");
```

```
                break;

            case OS_Q_FULL:
                printf("Queue is full \n");
                break;
        }
        }
        OSTimeDlyHMSM(0,0,10,0);
     }
}



void  App_Task2 (void *p_arg)
{
    INT8U  *msg;
    while(1)
    {
     msg = OSQPend(CommQ, 100, &err);
        if (err == OS_NO_ERR) {
            printf("Message received \n");
            printf("Received Message =%d \n", *msg);
        }
        else
            printf("Message not received \n");
        OSTimeDlyHMSM(0,0,1,0);
     }
}
```