SVPM'S

COLLEGE OF ENGINEERING, MALEAON (BK)

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

Experiment No:	**SUBJECT: EMBEDDED SYSTEMS & RTOS**
Class: BE E&TC	**Title: MESSAGE MAILBOX**

TITLE: Message Mailbox using microcontroller ARM 7TDMI family as device LPC2148

OJECTIVE: To Study Mailbox implementation 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 g

SOFTWARE USED:

- SCARM(IDE)
- Flash Magic.

HARDWARE USED:

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

THEORY:

Message mailbox (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'. Basically mailbox is called as message exchanger.Mailbox is an object of kernel Each task has its own Mailbox. μC/OS-II provides five services to access mailboxes:

***Creating a Mailbox*-OSMboxCreate()**,

Waiting for a message at a Mailbox-

OSMboxPend(),

Sending a message to a mailbox-

OSMboxPost(),

Getting a message without waiting-

OSMboxAccept()

Obtaining the status of a mailbox-

Messaging is done through kernel services. When one task wants to communicates with other, it writes the corresponding message into the Mailbox.

Mailbox is widely used for inter-task communication

When message is deposited in the mailbox either the highest priority task waiting for message is given the message (priority based) or the first task to request a message is given the message (FIFO) The mailbox message includes header information to identify message type &specification. The task always sends the message pointer to the mail box. Initially the mailbox has null pointer.

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 .Message transmitted after every 5 seconds

2 .Message received

3. Message not received

CONCLUSION:

PROGRAM:

```
#include "includes.h"

OS_STK Task1Stk[1024];
OS_STK Task2Stk[1024];
OS_EVENT  *CommMbox;

int  main (void)
{
   BSP_Init();                                    /* Initialize BSP functions             */
   InitUart0();
   OSInit();                              /* Initialize "uC/OS-II, The Real-Time Kernel"     */
          CommMbox = OSMboxCreate((void *)0);
          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);
}


INT16U value;
INT8U err;
INT8U key = 3;


void  App_Task1 (void *p_arg)
{
   (void)p_arg;

   while(1)
   {   /* repeate forever */

                     err = OSMboxPost(CommMbox, (void *)key);
       printf("Message is transmitted after every 5 seconds \n");
       OSTimeDlyHMSM(0,0,5,0);
   }
}





void  App_Task2 (void *p_arg)
{
    (void)p_arg;
    INT8U  *msg;
       while(1)
    {
           msg = (INT8U *)(OSMboxPend(CommMbox, 10, &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);
     }
}
```