

REAL TIME OPERATING SYSTEM PROGRAMMING-I: μ C/OS-II and VxWorks

Lesson-7: **μ C/OS-II Mailbox IPC functions**

1. Mailbox Functions

Mailbox Functions

- Used to communicate a pointer for information.
- μ C/OS-II permits one message-pointer per mailbox.
- At the pointer, there can be a string or data structure of no size limit.

Mailbox Functions...

- Assume an event pointer to the mailbox = *mboxMsg,
- Pointer to the message, *MsgPointer (for retrieving the message itself).

OSMboxCreate (*mboxMsg)

OS_Event *OSMboxCreate
(void *mboxMsg)

—To create a mailbox message pointer ECB of a mailbox message. (Example 11.19 Step 6) – Before initializing for Example 11.19 Step 8)

OSMboxPend(*mboxMsg, timeout, *MboxErr)

- void *OSMboxPend
(OS_Event *mboxMsg, unsigned short
timeOut, unsigned byte *MboxErr)
 - To check if mailbox message not pending
(available) then read *mboxMsg is and empty
mailbox [* mboxMsg = NULL again]. If message
is not available [*mboxMsg points to NULL],
then wait, suspend the task (block further
running) till *mboxMsg not Null or timeout
(Example 11.19 Step 39)

OSMboxAccept (*mboxMsg)

void *OSMboxAccept (OS_EVENT *
mboxMsg)

—To check if mailbox message at the
*MsgPointer, is available at *mboxMsg.

Unlike OSMboxPend function, it does not
block (suspend) the task if message is not
available. If available, it returns the pointer
(Example 11.19 Steps 40 and 52)

OSMboxPost (*mboxMsg, *MsgPointer)

unsigned byte OSMboxPost (OS_EVENT
*mboxMsg, void *MsgPointer)

- Sends a message of task at address MsgPointer by posting the address pointer to the mboxMsg.
- If box is already full (*mboxMsg not Null) , then the message is not placed and error status sent. (Example 11.19 steps 24 and 40)

OSMboxQuery (*mboxMsg, *mboxData)

unsigned byte OSMboxQuery
(OS_EVENT *mboxMsg,
OS_MBOX_DATA *mboxData)

- To get mailbox error information
- pointer Null or Not Null,

2. Macros for mailbox functions to find status after execution of OS mailbox Functions

Macros for mailbox functions

- OS_NO_ERR and
- OS_ERR_EVENT_TYPE

To get mailbox error information, at data structure pointed by *mboxData

3. Example for mailbox functions application

Example for application of for mailbox functions

- Programming Example for communication a message to display task after delivering the chocolate in chocolate vending machine

Step A: Initiating the Mailbox

```
#define OS_MAX_EVENTS 8
/*When total number of IPCs needed in an
application = 8*/
#define OS_MboX_EN 1
/*When the use of semaphores is
contemplated */
```

Step B: Global IPC functions and their parameters declarations

```
OS_EVENT *mboxStrMsgthanks;  
/* When mail box is to be used to send an  
amount message */  
int *i = 0; char [ ] mboxStrMsgthanks;  
mboxStrMsgthanks = OSMboxCreate  
(*i);  
/* When mail box is to be used the initially it  
is to point to null */
```

Step C: Main function

```
void main (void) {  
    OSInit ();  
    /* Create First task */  
    OSTaskCreate (FirstTask, void (*)  
0,(void *)&FirstTaskStack[  
FirstTaskStackSize], FirstTaskPriority);  
    OSStart ();  
}
```

Step D: First task

```
static void FirstTask (void *taskPointer) {  
    /*Create Application Tasks*/  
    OSTaskCreate (ReadTask, void (*) 0,(void  
    *)&ReadTaskStack [ReadTaskStackSize],  
    ReadTaskPriority);  
    OSTaskCreate (DeliveryTask, void (*)  
    0,(void *)&DeliveryTaskStack  
    [DeliveryTaskStackSize],  
    DeliveryTaskPriority);  
}
```

Step D: First task...

```
/*System clock time set */  
OSTimeSet (presetTime);  
OSTickInit (); /* Initiate system timer  
ticking*/  
while (1) {OSTaskSuspend  
(FirstTaskPriority); /* Suspend first task  
indefinitely and Run only the application related  
tasks */  
}
```

Step E: ReadTask

```
static void ReadTask (void *taskPointer)
{...
int *amount;
while (1) {...;
/* Code similar to one given in earlier lessons*/
...;
OSTimeDelay (3000);
}; }
```

Step F: DeliveryTask

```
static void DeliveryTask (void
*taskPointer) {..
while (1) {
...; /* Code similar to one given in earlier
lessons*/
mboxStrMsgthanks = “Collect Nice
Chocolate Thansk for your visit, Visit
Again”; /* define thanks message*/
```

Step F: DeliveryTask...

```
OSMboxPost (mboxStrMsgthanks);  
/* Post the message into the mailbox*/  
OSTimeDelay (3000);  
/* Delay to enable lower priority display task  
run*/  
OSTimeDlyResume (ReadTaskPriority);  
/* Resume to delayed higher priority read task  
*/  
};}
```

Step G: DisplayTask...

```
static void DisplayTask (void *taskPointer)
{..
String displayThanks;;
while (1) {
displayThanks = OSMboxPend
(mboxStrMsgthanks, 0, *MboxErr);
/* Wait for mailbox message, read
mboxStrMsgthanks in displayThanks and then
reset the mboxStrMsgthanks to null*/
```

Step G: DisplayTask...

...; ...; ...;

OSTimeDlyResume

(DeliveryTaskPriority); /* Resume to delayed
higher priority delivery task */
}}

Summary

We learnt

- μ C/OS-II has mailbox functions
- Simple feature of μ C/OS-II mailbox—one message pointer per mailbox.
- Any number of messages or bytes, provided the same pointer accesses them in each mailbox

End of Lesson-7 on μ C/OS-II Mailbox Functions