

# INTER-PROCESS COMMUNICATION AND SYNCHRONISATION:

## Lesson-13:

### OS Functions for Queue (of Messages)

# 1. IPC Queue functions

# Queue and Mailbox

- Some OSes provide the mailbox and queue both IPC functions
- Every OS provides queue IPC functions.
- When the IPC functions for mailbox are not provided by an OS, then the OS employs queue for the same purpose.

# Queue IPC features

- OS provides for inserting and deleting the message-pointers or messages.
- Each queue for a message need initialization (creation) before using the functions in the scheduler for the message queue.

# Queue IPC features

- There may be a provision for multiple queues for the multiple types or destinations of messages. Each queue have an ID.
- Each queue either has a user definable size (upper limit for number of bytes) or a fixed pre-defined size assigned by the scheduler.

# Queue IPC features

- When an RTOS call is to insert into the queue, the bytes are as per the pointed number of bytes.
- For example, for an integer or float variable as a pointer, there will be four bytes inserted per call. If the pointer is for an array of 8 integers, then 32 bytes will be inserted into the queue.

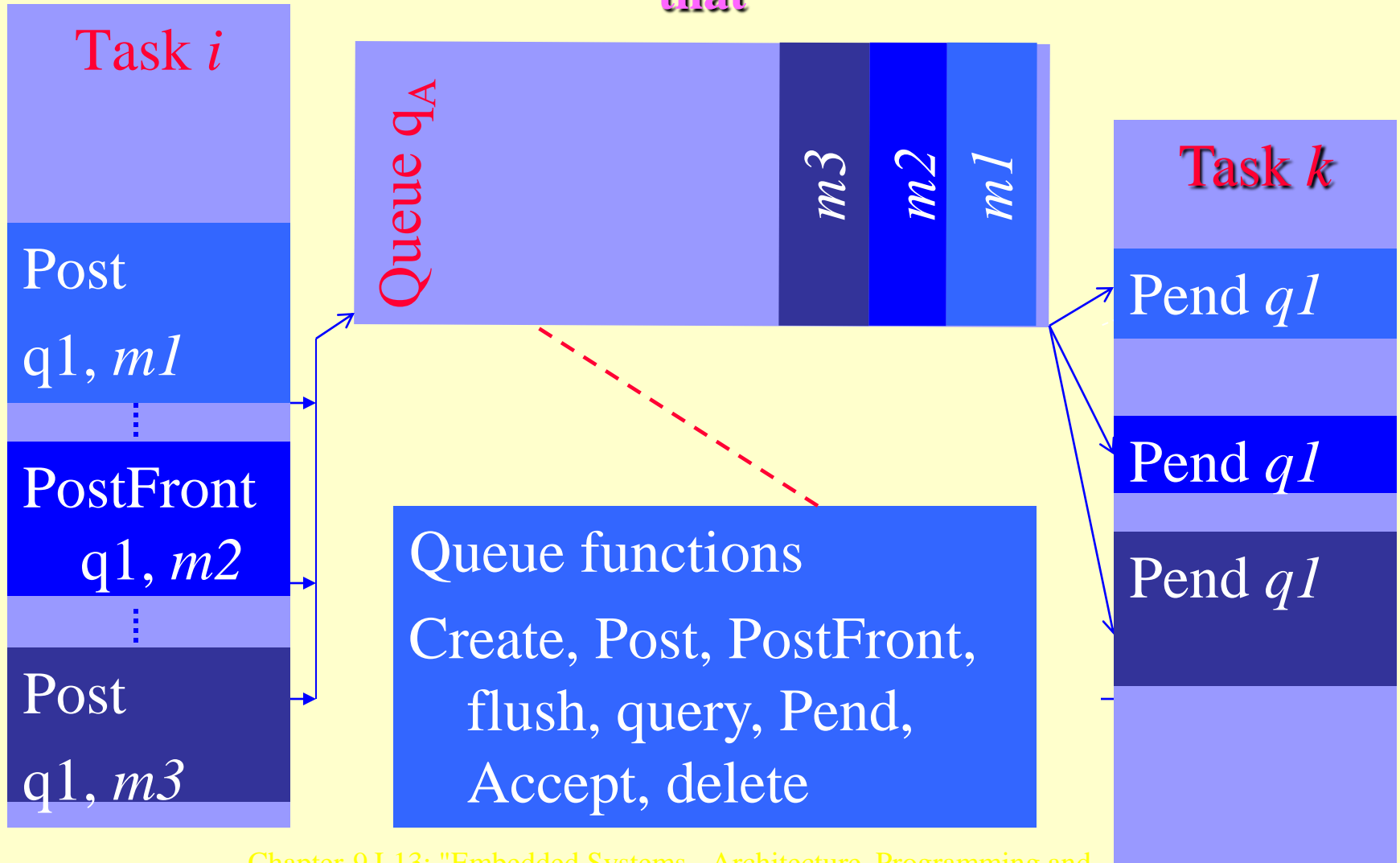
# Queue IPC features

- When a queue becomes full, there may be a need for error handling and user codes for blocking the task(s). There may not be self-blocking.

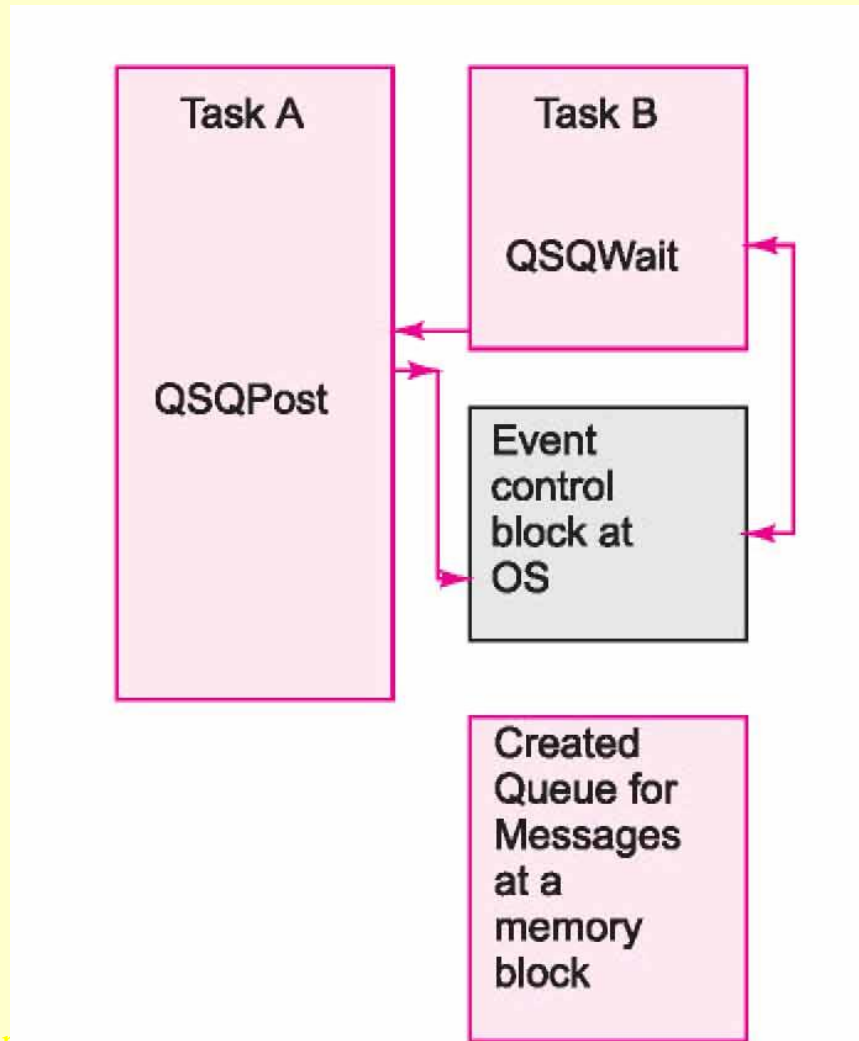
## **2. Queue Related Functions at the OS**



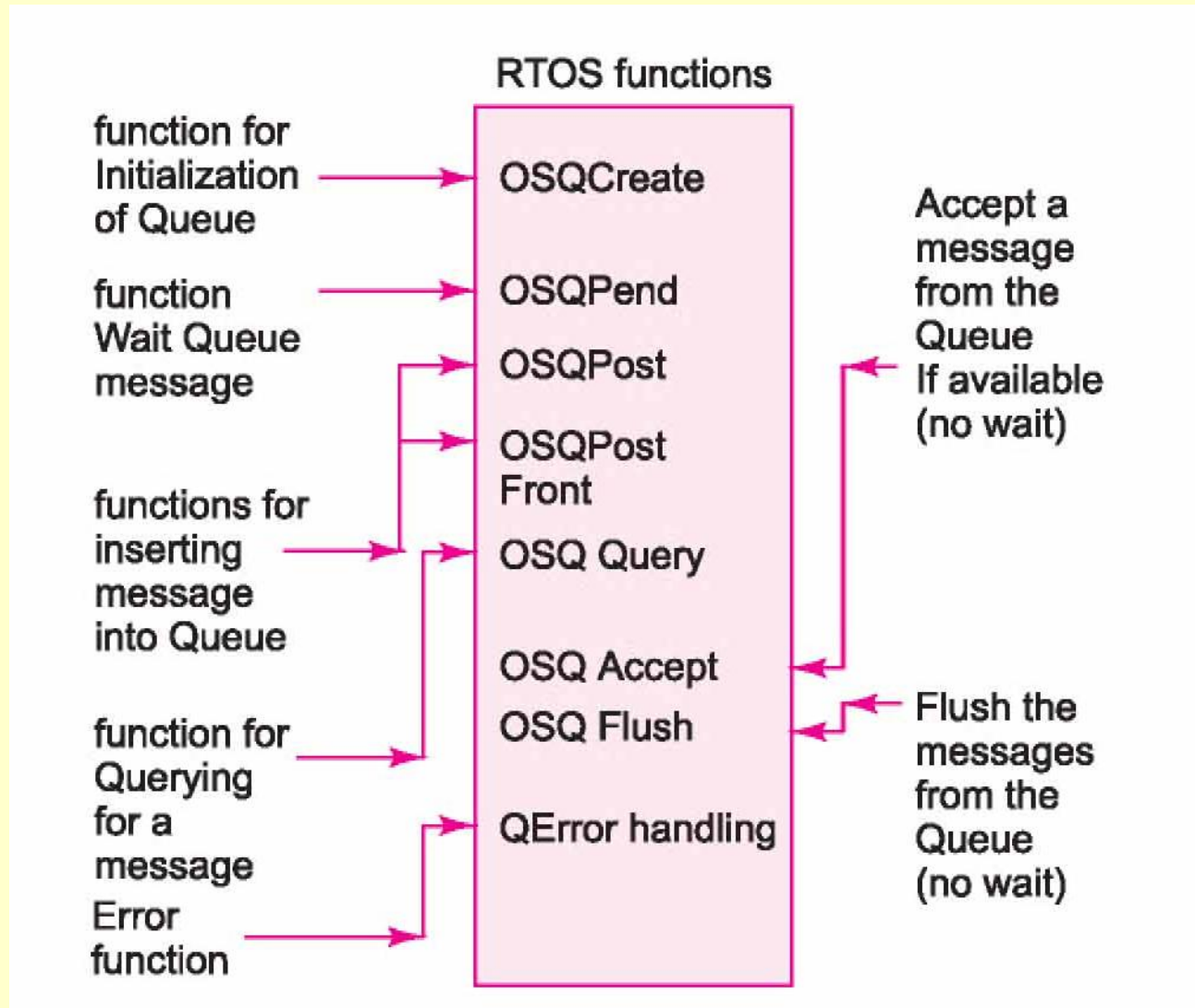
# Tasks $i$ sending messages into a queue $q1$ and task $k$ receiving that



# Memory-blocks at OS — for queue inserting, deleting and other functions

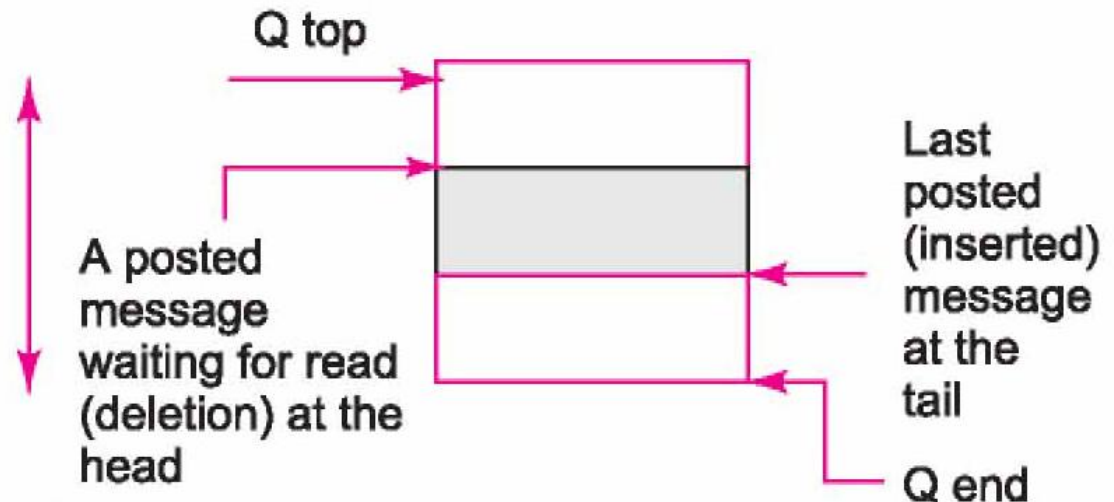


# Memory Block for OS Queue IPC functions



# Queue messages block

Message Queue in memory for posting (insertion), accept (deletion without wait) and wait (deletion after wait)



# Queue IPC functions

- OSQCreate— to create a queue and initialize the queue message, blocks the contents with front and back as queue-top pointers, \*QFRONT and \*QBACK, respectively.
- OSQPost — to post a message to the message block as per the queue back pointer, \*QBACK. (Used by ISRs and tasks)

## Queue IPC functions...

- OSQPend — to wait for a queue message at the queue and reads and deletes that when received. (Wait, Used by tasks.)

# Queue IPC functions...

- OSQAccept — to read the present queue front pointer after checking its presence yes or no and after the read the queue front pointer increments (No wait. Used by ISRs and tasks)
- OSQFlush — to read queue from front to back, and deletes the queue block, as it is not needed later after the *flush* the queue front and back points to QTop, pointer to start of the queue. (Used by ISRs and tasks)

# Queue IPC functions...

- OSQQuery— to query the queue message-block when *read* and but the message is not deleted. The function returns pointer to the message queue \*QFRONT if there are the messages in the queue or else null. It return a pointer to data structure of the queue data structure which has \*QFRONT, number of queued messages, size of the queue and. table of tasks waiting for the messages from the queue. [Query is used by tasks.]



# Queue IPC functions...

- OSQPostFront — to send a message as per the queue front pointer, \*QFRONT. Use of this function is made in the following situations. A message is urgent or is of higher priority than all the previously posted message into the queue (Used in ISRs and tasks)
- OSQDelete— to delete a queue

## 3. IPC Queue functions Application Example

# Task\_Director\_Output in Robot Orchestra

```
static void Task_Director_Output (void
    *taskPointer) {
.
while (1) {
.
/* Codes for inserting musical notes into the
   queue */
for (OSQEntries = 0; OSQEntries < OSQSize;
    OSQEntries ++)
```

# Task\_Director\_Output in Robot Orchestra

```
{OSQPost (QDirector, note)} /* Post for the  
    Queue QDirector messages upto the  
    OSQSize /*
```

```
.  
};
```

# Task Player Input in Robot Orchestra

```
static void Task_Player_Input (void  
    *taskPointer) {
```

```
.  
while (1) {  
.  
/* Codes for deleting notes from the queue */  
for (OSQEntries = OSQSize; OSQEntries >0 ;  
    OSQEntries --)
```

# Task Player Input in Robot Orchestra

```
note (i) = OSQPend (QDirector, 0, err) /* Post  
    for the mailbox message and userInput,  
    which equaled null now equals userInput  
    message pointer*/
```

```
.  
};
```

# Summary

# We learnt

- An OS provides the IPC functions for creating and using queues as the messages in FIFO and priority message) modes



# We learnt

- OS provides the IPC functions
- Create, Post, PostFront, Pend, Accept, Flush, Query and Delete or using message queues.
- The time out and error handling function can be provided with Pend function argument.

# End of Chapter 9 Lesson-13: OS Functions for Queue (of Messages)