

# Real-Time Systems and Application-Real-Time Operating System

Dr. J. K. Das

School of Electronics Engineering, KIIT DU

# Table of Contents

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

## 1 Introduction

### ■ Intro

### ■ Difference between GPOS & RTOS

### ■ Defining RTOS

## 2 Characteristics of RTOS

## 3 Task

## 4 Semaphore

## 5 Message Queue

## 6 Pipe

## 7 Exception and Interrupts

## 8 Timers and Timer Services

# Introduction

## Real-Time Systems and Application- Real-Time Operating System

Dr. J. K. Das

### Introduction

#### Intro

#### Difference between GPOS & RTOS

#### Defining RTOS

#### Characteristics of RTOS

#### Task

#### Semaphore

#### Message Queue

#### Pipe

#### Exception and Interrupts

#### Timers and Timer Services

Today most of embedded system/ Real-time systems are using RTOS for managing large complexity to handle different processes/ tasks running on the system. for scheduling these process/task RTOS is used.

# RTOS

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

## Similarity between GPOS & RTOS

- some level of multitasking
- software and hardware resource management
- provides OS services to different applications
- abstracting hardware from software.

# RTOS<sub>contd...</sub>

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

## Key features of RTOS over GPOS

- better reliability
- ability to scale up/down as application needed
- better performance
- reduces memory requirements
- supports scheduling policies for Real-Time Applications.
- supports diskless embedded system, executable/boot runs from ROM/RAM etc.

# RTOS Definition

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

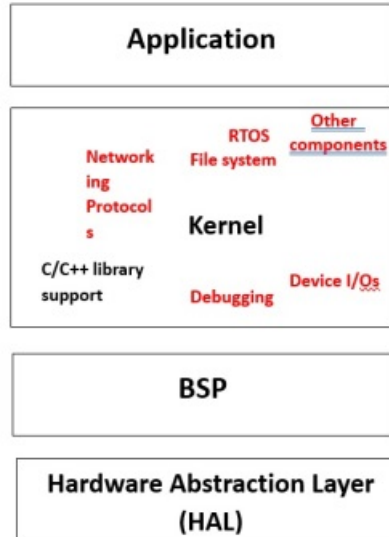
Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

A RTOS is schedules the execution of different task/process timely manner, manages different system resources and provides consistent foundation for development of application code. The RTOS kernel is shown in figure.



# RTOS Kernel

RTOS Kernel contains followings:

- Scheduler- it is a set of algorithm that determines the task to be run when needed.
- Object- it allows developer to create applications for real-time system. Common kernel objects are task, semaphore, message ques, mailbox, pipes.
- Services- kernel performs an object or a operation such as timing or interrupt handling and resource management are called services.

# RTOS Kernel<sub>contd...</sub>

## Real-Time Systems and Application- Real-Time Operating System

Dr. J. K. Das

### Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

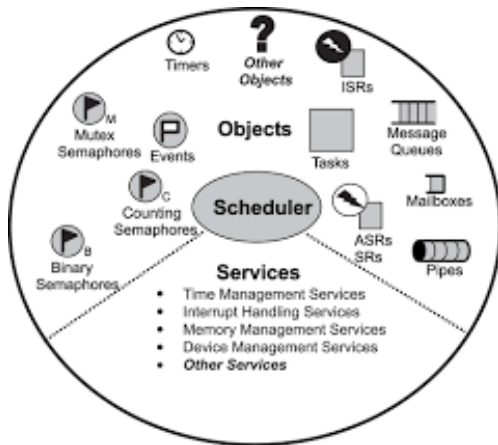
Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services





# RTOS Kernel<sub>contd...</sub>

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

## Scheduler

A scheduler is the heart of every kernel (RTOS) used. the scheduler is an algorithm used to determine which task executes when.

A scheduler has following elements

- schedulable entities
- multitasking
- context switching
- dispatcher
- scheduling algorithm

# RTOS Kernel<sub>contd...</sub>

## Schedullable entities

the schedulable entity is a kernel object that competes for execution time as a system, as per predefined scheduling algorithm.

- a task is an independent thread of execution which contains a sequence of independent schedulable instructions.
- process is another schedulable entity other than task which independently compete for execution time.
- process has better memory protection features at the expenses of memory & performance overhead.

# RTOS Kernel<sub>contd...</sub>

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

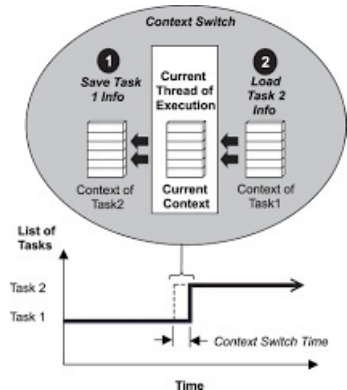
Pipe

Exception and  
Interrupts

Timers and Timer  
Services

## Multitasking

multitasking is the ability of RTOS to handle multiple activities with a set of deadlines. a real-time kernel might have multiple task that has to be scheduled to run.



# RTOS Kernel<sub>contd...</sub>

## Context Switching

- ➡ each task has its own context i.e. state registers
- ➡ context switch occurs when scheduler switches from one task to another as shown in figure [in previous slide].
- ➡ when a new task is created, kernel creates it and maintains an associated TCB(task control block) which is a structure contains taskID, memory state, stack block, priority level etc.
- ➡ whenever context switch occurs, kernel saves the task context from its TCB i.e for example task1 TCB is saved , then it is frozen and task2 start to run and so on.

# RTOS Kernel<sub>contd...</sub>

## Dispatcher


it is the part of scheduler that performs context switching i.e changes flow of execution .


- ➡ when RTOS is in running state, the flow of control may pass through an appropriate task or an ISR.
- ➡ when ISR is raised, control passes through one of the system routine provided by system.
- ➡ when it leaves, dispatcher passes control to one of the task in the user application based on scheduling algorithm.
- ➡ the dispatcher may happen when a task makes a system call and dispatcher helps to move from current state to another.


# RTOS Kernel<sub>contd...</sub>

## Kernel Objects

The different kernel objects are:

 **task**- concurrent and independent threads of execution that can compete for execution time

 **semaphores**- are token like objects that can be incremented or decremented by task synchronization or mutual exclusion.

 **message queue**- are buffer like data structures for synchronization, mutual exclusion, and data exchange between tasks by passing message between tasks.

# RTOS Kernel<sub>c</sub>ontd...

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

## Services

most of kernel provides different services that helps developer to create different applications for real-time embedded system. this consists of a set of API(application programming interface) calls that can be used to perform different operations on the kernel objects or tasks.

# Key Characteristics of RTOS

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

- ✧ Reliability
- ✧ Predictability
- ✧ Scalability
- ✧ Performance
- ✧ Compactness



# Reliability

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

Embedded Systems must be reliable.

Depending on application, system needs to operate for long period without human intervention.

Different degrees of reliability may be required, but system is reliable when it provides continuous service without fail.

# Predictability

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

As most of the systems are rel-time, they must meet time requirement to ensure proper operation.

For this RTOS must be predictable to some degree i.e. RTOS should show a predictable behavior for execution of different tasks soo that they are finished as per their time frame.

# Performance

The embedded systems perform fast to fulfill time requirements, typically the more deadlines to meet and the shorter time requirement between them. that is faster CPU is needed for it.

Typically processor's performance is expressed in terms of MIPS (million instructions per second). The throughput measures the overall sys-

tem performance with hardware and software combined. this throughput measures in terms

# Compactness

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

Application design and cost constraints helps in determining how the compact the embedded system. for example a Cell-Phone must be small, portable and low cost.

# Scalability

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

As RTOS can be used in wide variety of embedded systems, they must be able to scale up or down to meet application specific requirement. Depending on the requirements of different applications, RTOS must support choose different modular component from available file system, process, stacks etc.

# Introduction to Task

- ✧ A task is an independent thread of execution that can compete with other concurrent task for process or execution time.
- ✧ generally developers decomposes the application into multiple small tasks, schedulable and sequential units.
- ✧ when kernel starts first task, it creates a set of system task and allocates appropriate priorities from a set of reserved priority level.

# Introduction to Task<sub>contd....</sub>

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

The system task includes:

- ✳ initialization or startup task
- ✳ idle task- uses processor idle cycles when no other activities are present.
- ✳ logging Task- logs system messages
- ✳ exception handling- handles exceptions
- ✳ debugging agent- allows debugging with host debugger.

# Task States and Scheduling

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

A task can be either in one of the states mentioned below.

- ✳ ready
- ✳ blocked
- ✳ running



# Task States

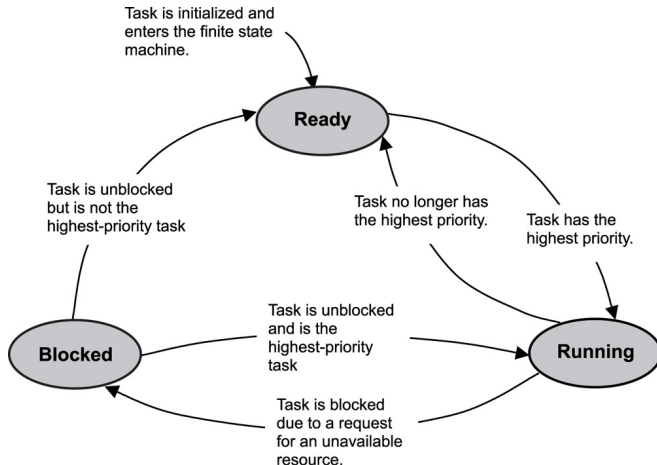


Figure: Task States

# Task States<sub>contd...</sub>

## Ready State

When the task is created, and it is ready to run, kernel put it into ready state.

In this state it competes with other ready tasks for processor execution time.

Highest priority task moves to running state as per scheduling algorithm.

Task is moved to blocking state by making a blocking call whenever resources are not available.

A task is unblocked and moves to running state only if it has got highest priority otherwise moved to ready state.

# Task States<sub>contd...</sub>

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

## Ready State

When kernel supports one task per priority level, highest priority task moves to running state and other ready task after that. In this implementation kernel limits no of task in an application.

But most kernel supports more than one task per priority level, in this situation scheduling algorithm is complex. Figure in next slide shows the how a kernel schedules using a task-ready list using priority based preemptive scheduling.

# Task States<sub>contd...</sub>

## Real-Time Systems and Application- Real-Time Operating System

Dr. J. K. Das

### Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

### Characteristics of RTOS

### Task

### Semaphore

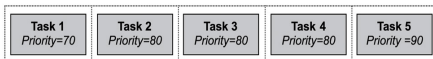
### Message Queue

### Pipe

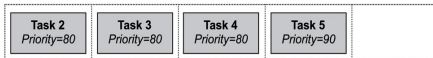
### Exception and Interrupts

### Timers and Timer Services

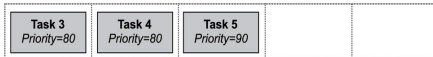
#### 1 First-Step: State of Task-Ready List



#### 2 Second-Step: State of Task-Ready List



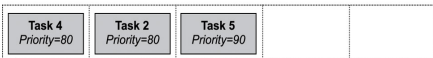
#### 3 Third-Step: State of Task-Ready List



#### 4 Fourth-Step: State of Task - Ready List



#### 5 Fifth-Step: State of Task-Ready List



# Task States<sub>contd...</sub>

## Running State

in single processor one task run a time.

When task moved to running state, processor loads its register with task context.

when a task moves from running to ready when a higher priority task preempted it.

A running task can move to blocked state only when

- ✂ making a call that requests an unavailable resources.
- ✂ making a call that requests to wait for an event to occur.
- ✂ making a call to delay the task for some duration.

# Task States<sub>contd...</sub>




## Blocked State

The possibility of blocked state is important as without it, low priority task can not start. If a high priority task are not blocked, then CPU starvation starts i.e. low priority task will not CPU execution time.

The blocking state attainment is done as per problems discussed in previous slide. When a task is unblocked it moves to ready state and put into task ready list as per priority level.

# Typical task Operations

The kernel provides different task operations are as follows:

-  creating or deleting a task
-  controlling task scheduling
-  obtaining task information.

# Typical task Operations

## Task creation & deletion

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

- Depending upon kernel's API call a task is created or deleted.
- in some kernel, task is created first and then started during this task is put in suspended list till it moved to ready state.
- kernel may provide user configurable hooks to execute programmer specified functions at a specified event time.
- During deletion process, kernel first terminates the task and then frees memory by deleting task's TCB and state.
- inappropriate deletion of task may results in **cor-**



# Typical Task Operations<sub>contd...</sub>

## Task scheduling

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

From task creation to deletion, a task may be in one of task states and the execution of task is done by scheduler kernel provides some API to do this. Different operations provided by task scheduling are:

- suspend
- resume
- Delay
- restart
- Get priority
- set priority
- preemption lock
- preemption unlock.

# Typical Task Operations<sub>contd...</sub>

## Other Task Operations

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

- Obtaining task information: in this task ID and TCB information are collected.
- Synchronization and concurrency: synchronization and communication among different task are done through inter task primitives (one type of kernel object).

# Semaphore

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

Semaphore is just like a **Key** that allows task to carry out some operations or access to shared resources. Whenever a task wants access to the shared resource, it must acquire the semaphore first after completion of operation it is released.

Till this time all other tasks have to wait if they need access to shared resource as semaphore is not available. Even if higher priority task to acquire this, it will be in wait state until semaphore is released by the lower priority task.

# Semaphore

## Use of Semaphore

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

- Managing Shared Resource
- Task Synchronization

Apart from managing shared resource, task synchronization can also be performed with the help of a semaphore. In this case semaphore will be like a flag not key.

- Unilateral synchronization

This is one way synchronization which uses a semaphore as a flag to signal another task.

- Bilateral synchronization

This is two way synchronization performed using two semaphores. A bilateral synchronization is similar to a unilateral synchronization, except both tasks must synchronize with one another before proceeding.

# Semaphore

## Types of semaphore

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

### ■ Binary Semaphore

Binary semaphore is used when there is only one shared resource.

### ■ Counting Semaphore

To handle more than one shared resource of same type, counting semaphore is used.

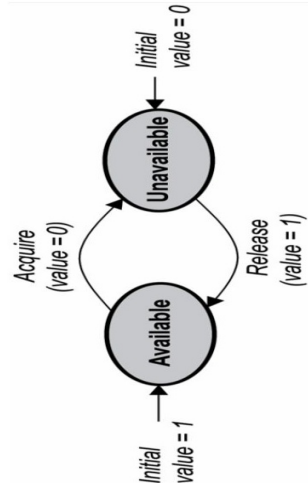
### ■ Mutual Exclusion Semaphore or Mutex

To avoid extended priority inversion, mutexes can be used.

# Semaphore

## Binary Semaphore

The typical state diagram of a binary semaphore given below. The binary semaphore have a value either 0 or 1. For available semaphore the value is 1 else 0.



# Semaphore

## Counting Semaphore

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

The typical state diagram of a counting semaphore given below. Here semaphore is assigned with a counting value  $\text{count} > 0$ . When it is released counting value increased by 1 and acquiring value is decreased by 1. if all semaphore is utilized then counting value becomes 0.

# Semaphore

Counting Semaphore<sub>contd...</sub>

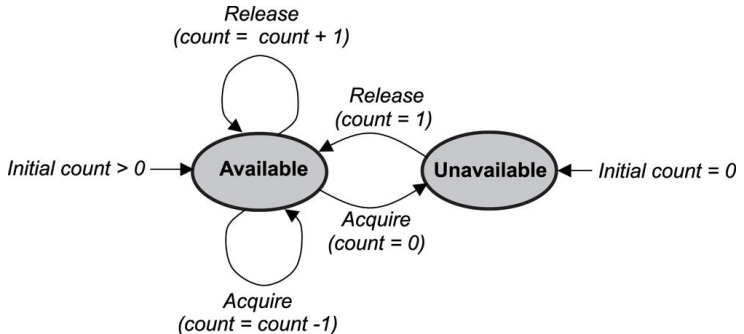


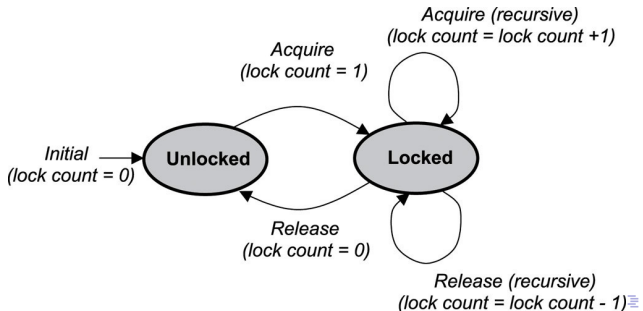
Figure: Counting Semaphore State Diagram



# Semaphore

## Mutual Exclusion or Mutex Semaphore

A mutual exclusion(mutex) semaphore is special binary semaphore that support ownership, recursive access, task deletion safety, and one or more protocols for avoiding inherent mutual exclusion.



# Semaphore

Mutual Exclusion or Mutex Semaphore<sub>contd...</sub>

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

- Ownership of a mutex semaphore gained when a task locks the mutex by acquiring it. i.e. task loses ownership of mutex when it unlocks it by releasing it.
- mutex implementation supports recursive locking which allows a task that owns the mutex to acquire it multiple times in locked state. It is built automatically into mutex or it might need to be enabled explicitly when the mutex is first created.
- recursive locking allow nested attempts to lock semaphore so it causes dead locks.
- mutex implementation have built-in task deletion safety and a premature deletion is prevented by using task deletion locks.

# Semaphore

Mutual Exclusion or Mutex Semaphore<sub>contd...</sub>

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

**Priority Inversion Avoidance:** This happens when a high priority task is blocked and is waiting for a resource being used by a low priority task which preempted itself. This can be avoided by following protocols:

- **priority inheritance protocol:** ensures priority level of low priority task acquired mutex is raised to higher priority task that has requested the mutex when inversion happens. the priority is lowered again to its original value after releasing mutex.
- **ceiling priority protocol:** it ensures that priority level of the task that acquired mutex automatically set to higher priority of all possible tasks that requests that mutex when it first acquired until it is released.

# Semaphore

## Typical Semaphore Operations

Typical operations are:

- creating and deletion semaphore(binary, counting or mutex)
- acquiring and releasing semaphore(wait for ever- task remains blocked, wait with a timeout- task remains unblocked until it is able to acquire or until timeout, do not wait- task requesting takes the semaphore)
- clearing the semaphore's task waiting list(open flush is used clear)
- getting semaphore information(gives information about blocked task)

# Semaphore use

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

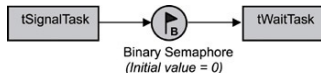
Semaphores are useful either for synchronizing execution of multiple tasks or for coordinating access to a shared resource. The following uses of semaphore

- wait-and-signal synchronization,
- multiple-task wait-and-signal synchronization,
- credit-tracking synchronization,
- single shared-resource-access synchronization,
- recursive shared-resource-access synchronization, and
- multiple shared-resource-access synchronization.

# Semaphore use

## Wait-and-Signal Synchronization

Two tasks can communicate for the purpose of synchronization without exchanging data. For example, a binary semaphore can be used between two tasks to coordinate the transfer of execution control. the binary semaphore is initially unavailable (value of 0). higher priority and runs first. The task makes a request to acquire the semaphore but is blocked because the semaphore is unavailable. Pseudo code is as follows.

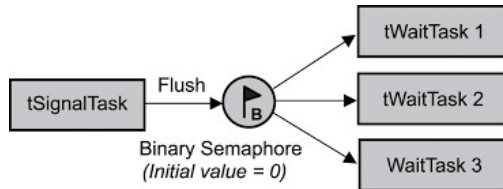


```
tWaitTask ( )  
{  
    :  
    Acquire binary semaphore token  
    :  
}  
tSignalTask ( )  
{  
    :  
    Release binary semaphore token  
    :  
}
```

# Semaphore use

## multiple-task wait-and-signal synchronization

When coordinating the synchronization of more than two tasks, use the flush operation on the task-waiting list of a binary semaphore. As in the previous case, the binary semaphore is initially unavailable (value of 0). The higher priority tWaitTasks 1, 2, and 3 all do some processing; when they are done, they try to acquire the unavailable semaphore and, as a result, block. This action gives tSignalTask a chance to complete its processing and execute a flush command on the semaphore, effectively unblocking the three tWaitTasks.



```
tWaitTask ()
{
    :
    Do some processing specific to task Acquire binary semaphore token
    :
}

tSignalTask ()
```

# Message Queue

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

A message Queue is a buffer-like object through which tasks & ISRs send and receive message to communicate and synchronize with data. It is like a pipeline that holds temporarily the message from sender until receiver receives it. After which message queue is clear to initiate another message.



# Message Queue Diagram

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

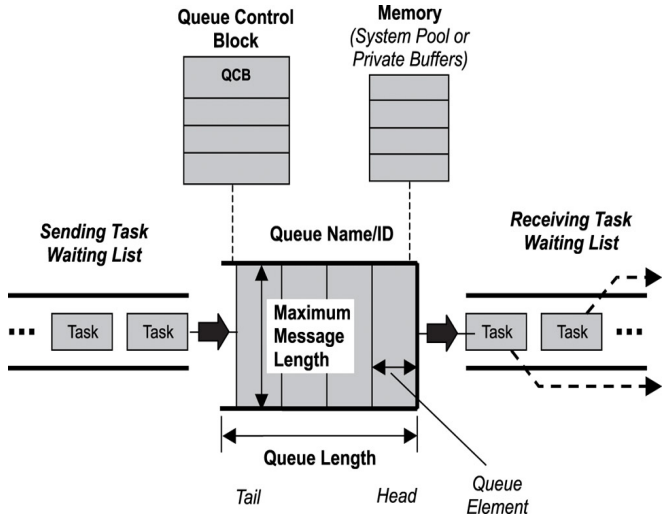
Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services



# Message Queue States

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

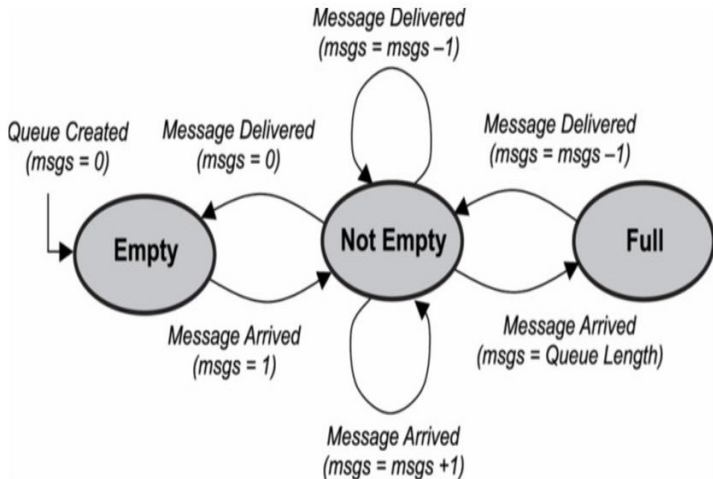
Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services



# Message queue Contents

In order to transmit message greater than buffer length of message queue pointer are used. when the task is sending messages, it is copied 2 times: first the message is copied to buffer from memory and in second time, message is copied from queue to memory of receiver. Depending on kernel implementation, sometimes message may be copied once only.

# Message queue Contents

The message queue can be used to send and receive varieties of data some of which are:

- Temperature value of a sensor
- bitmap to draw on a display
- text message to display in LCD
- keyboard event etc.

# Message Queue Storage

Different kernel stores message queues in different locations which are mentioned below:



**System Pool-** Pool is advantageous when all message queues will never be filled at same time so that it saves memory. Sometimes when a long message queue larger than buffer size, even if queue is not starts rejecting or continues to accept messages.



**Private Buffers-** This requires enough memory to be reserved for the full capacity of message queue that will be created. It also ensures messages are not over written so provides better reliability.

# Message Queue Operations

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

## Message Queue creating and deleting:

When message queues are created treated as global objects and used by all tasks as per ISRs assigned.

While creating message queues, an initial length of message queue and maximum size that it can accommodate message.

When deleting message queues, it automatically unblocks waiting tasks and blocking call in each of these task results an error.

# Message Queue Operations

Typical message queue operations are:

- creating and deleting message queues
- sending and receiving message queue
- obtaining message queue information.

## Message Queue Sending:

While sending the message queue first filled from head an to tail in FIFO order. Sometimes urgent messages directly goes to head in LIFO order. Message queues are sent in following ways:

1. not block(ISRs and Task)
2. block with a timeout.
3. block forever

# Message Queue Operations

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

## Message Queue Receiving:

Messages can be read from head of a message queue in two different ways:

1. destructive read
2. non-destructive read.



# Message Queue information

Real-Time Systems  
and Application-  
Real-Time Operating  
System

Dr. J. K. Das

Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

Characteristics of  
RTOS

Task

Semaphore

**Message Queue**

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

message queue information have following  
application:

1. Show queue information 2. Show queue's  
task-waiting list.

The following are typical ways to use message queues within an application:

- non-interlocked, one-way data communication(without handshaking)
- interlocked, one-way data communication(with handshaking)
- interlocked, two-way data communication
- broadcast communication

# Real-Time Systems and Application- Real-Time Operating System

Dr. J. K. Das

## Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

## Characteristics of RTOS

Task

Semaphore

Message Queue

Pipe

## Exception and Interrupts

Timers and Timer  
Services

kj hkjj

# Real-Time Systems and Application- Real-Time Operating System

Dr. J. K. Das

## Introduction

Intro

Difference between GPOS &  
RTOS

Defining RTOS

## Characteristics of RTOS

Task

Semaphore

Message Queue

Pipe

Exception and  
Interrupts

Timers and Timer  
Services

hhkhkkj jknk