Chapter 11

Real Time Operating System
Lesson 03

Inter process Communication (IPC)
Inter process communication

• Inter Task Message
• A task can not call another task
• A task can only put information or message at a task control block
• The OS examines task control block at each tick of the system clock
• How the signal or token or message sent to another task?
Inter-process Communication using a Signal

- Signal is like software interrupt
- Signal communicated when a software interrupt instruction executes
Inter-process Communication using a Signal

• Signal means a call for action on an event after which the system runs a signal-handler task, similar to running of an interrupt service routine on an interrupt

• Several times, when one task finishes certain set of codes then only the system should start other task

• The signal can be used for doing this
Signal - A simplest IPC

IPC: Stands for Inter-process communication

A task executes a function ‘send-signal’

Another task waits for the signal and is started by OS on execution of the send-signal function

Example- A function os_send_signal (2) executes at task 1

so that a task 2 with at a function os_wait (K_SIG,0, 0) starts after OS gets signal ‘2’
Two tasks Example for the ECG recorder

- Tasks run at two different times
- One task $i$ on finishing a set of actions, sends a signal to the system using a signalling instruction
- The system passes the signal to waiting second task $j$
- Then the $j$ runs
Signal between Two Tasks

Task 1

Task 2

$t_2 - t_1 = t_{\text{switch}}$ (task switching time)

signals 2 to enable the OS to start waiting task2
Example of 3 tasks

Example - Task 1 and 3 Synchronisation

Task 1

1

signals 3 to enable the RTOS to start waiting task3

Task 2

Task 3

waiting for signal from 1

RTOS

2

3
Four tasks of Example for the digital camera

• All tasks run in four different time slots
• First task executes when system sends it a signal sw1, for example, from the interrupt service routine
• The routine executes on the user pressing a switch to start taking the picture
The digital camera first three tasks

- Must execute in sequence one after one
- Second starts when first task sends signal $st_2$
- Third task executes when second task sends it a signal $st_3$
- Fourth task executes when system sends it a message $md_4$ to start or the task 3 sends signal $st_4$
Task Synchronisation in a sequence among three tasks

Example- Three Task Synchronisation in a digital camera
Semaphore— A token for an IPC

A task executes a function ‘send-token’

Another task blocked at certain point, waits for the token and is started (unblocks) by OS on execution of the send-token function

Example- A function os_send_token (K_SEM) executes at task 1

so that a task 2 with at a function os_wait (K_SEM,0, 0) starts after OS gets token ‘K_SEM’
Task Synchronisation among three tasks waiting for token s

Example - Two Tasks 1 and 3 Synchronisation
Critical Section

• Section of codes which have some action or variable, which must complete before other section of the codes run

• For example, a section updating time in hrs. min. sec. and date is a critical section. Unless all five parameters update, the other section for display should not update
Wait for an event-flag

A task section \( B \) waits for an event to occur at task section \( A \) and unblocks and runs after that event

Example– Time-date display section waiting for update for new time-date
Two Critical sections A and B

- Section A having a set of codes that should run un-interrupted, such that other section(s) B should block during A’s run
Task critical sections Synchronisation among two tasks waiting for token s

Example- Two Tasks 1 and 3 critical sections Synchronisation
Task Synchronisation in a sequence among three tasks

Example- Three Task Synchronisation in a digital camera
Wait for a message

A task section $B$ section waits for a message from at task section $A$ and unblocks and runs after that message

Example- Time-date display section waiting for updated new time-date
Message—A string or pointer address for an IPC

A task executes a function ‘send-message’

Another task blocked at certain point, waits for the message K_MSG and is started (unblocks) by OS on execution of the send-message function

Example- A function os_send_msg (K_MSG) executes at task 1

so that a task 2 with at a function os_wait (K_MSG,0, 0) starts after OS gets message ‘K_MSG’
Task Synchronisation among three tasks waiting for K.MSG

Example - Two Tasks 1 and 3 Synchronisation

1. Task 1
2. Message K_MSG
3. RTOS
4. Task 3
Wait for an *exception* (condition) message

An exception handling task section B section waits for an error message (exception) from task section A and unblocks and runs after that message

Example- handling of exceptions thrown by a task
A hypothetical example of a toffee vending machine system (ITMS)

- Inter process Communication among four tasks
- Task, *taskkeyparsing* parses the action needed from the keys entered by the customer
A hypothetical example of a toffee vending machine system (ITMS)

- Task, *taskdsply* displays the idle time message(s) and the messages during response to customer queries and customer selection of a specific toffee and to facilitate the graphic user interactions (GUIs)
A hypothetical example of a toffee vending machine system (ITMS)

- Task, *taskmoney* collects the cost of toffee in terms of coins (or debit card) from a customer
A hypothetical example of a toffee vending machine system (ITMS)

- Task, tasktoffee delivers the selected toffee, whether chocolate or almond or mango flavoured toffee
Summary
We learnt

- A *signal* permits a task or ISR program run after a certain task signals and notifies to OS that signal
We learnt

- A *semaphore* permits unblocking of a task section after a certain task sends and RTOS notifies that *semaphore*
We learnt

- A *message* passes and unblocks a task section after a certain task sends the message and RTOS notifies that *message*
End of Lesson 03 on

Inter process Communication (IPC)