REAL TIME OPERATING SYSTEM (RTOS)

How it works?

When we hear the word “Operating System” the first ones that come to our mind are those we experience/use in our day to day life, say, Windows XP, Linux, Ubuntu, Windows 7 for Computer systems, Android for mobiles and many more . We mainly know that operating systems are for computers. It is a fact that most of the digital electronic devices run some sort of operating systems inside. There are many operating systems developed for micro controllers too. But here it is familiar as REAL TIME OPERATING SYSTEM. The phrase ‘REAL TIME’ indicates that the response of the operating systems is quick. Microcontrollers don’t have much space for code. Thus the operating systems have less scope to be advanced. They try to provide at least the minimum scope of threading, scheduling and monitoring of multiple tasks for small systems.

Usually, Real Time Operating Systems are a segment or a part of the whole program that decides the next task, task priority, handles the task messages and coordinates all of the tasks. An RTOS is a complex concept. I’d like to discuss about the concept of State Machine. Here is an implementation of what you can merrily call a state machine.

```
while(1)
switch(state)
{         case         1:         //Code         for         Task         1;
state=                                                                    2;
case           2:           //Code           for           Task           2;
state=                                                                    3;
case           3:           //Code           for           Task           3;
state=                                                                    4;
case           4:           //Code           for           Task           4;
state=1;
}
```

As you can see from the code, there is a provision of changing the execution sequence. And it can be further modified and made complex. The programmer can modify and place decision making statements (Like if, if-else, switch-case) to switch the task. And the flow of execution can be logically determined.

A Real time operating system handles some tasks or routines to be run. The kernel of the operating system assigns CPU attention to a particular task for a period of time. It also checks the task priority, arranges the massages from tasks and schedules.

The basic functionalities an RTOS are:

- Scheduler
- RTOS Services
- Synchronization and messaging

The Scheduler

Tasks, can have three states.

- **Ready to run:** When task have all the resources to run, but not in running state. It's called a **ready to run**task. It's the state before running.
- **Running** – When a task is executing. It is known as running.
- **Blocked** – When a task doesn't have enough resources to run, it is sent to a **blocked** state.

To schedule a task, three techniques are adapted.

- **Co-operative scheduling:** In this scheme, a task runs, until it completes its execution.
- **Round Robin Scheduling:** Each task is assigned a fixed time slot in this scheme. The task needs to complete its execution. Otherwise, the task may lose its flow, and data generated or it would have to wait for its next turn.
- **Preemptive Scheduling:** This scheduling scheme includes priority dependent time allocation. Usually in programs, 256 priority level is generally used. Each task is assigned an unique priority level. While some system may support more priority levels, and multiple tasks may have same priorities.

The Kernel takes care of the task. It involves the following

- Creating a task
- Deleting a task
- Changing the priority of the task
- Changing state of the task

RTOS Services

The heart of every operating system is called **'kernel'.** Tasks are relived of monitoring the hardware. It's the responsibility of the kernel to manage and allocate the resources. As tasks cannot acquire CPU attention all the time, the kernel must also provide some more services. These includes,

- Interrupt handling services
- Time services
- Device management services
- Memory management services
- Input-output services

Messaging

Messaging provides a means of communication with other system and between the tasks. The messaging services includes

- Semaphores
- Event flags
- Mailboxes

- Pipes
- Message queues

Semaphores are used to synchronize access to shared resources, such as common data areas. Event flags are used to synchronize the inter-task activities. Mailboxes, pipes, and message queues are used to send messages among tasks.

Source : http://www.circuitstoday.com/what-is-real-time-operating-system-rtos