Embedded System Design & Application

Real time operating system (RTOS)

Dr. Sarwan Singh

NIELIT Ropar

agenda

- About Operating system
- Type of Operating system
- Real-time Operating System (RTOS)
  - Types, working, advantages
  - RTOS on microcontroller

- Micro-Controller Operating Systems (MicroC/OS, stylized as µC/OS
- FreeRTOS

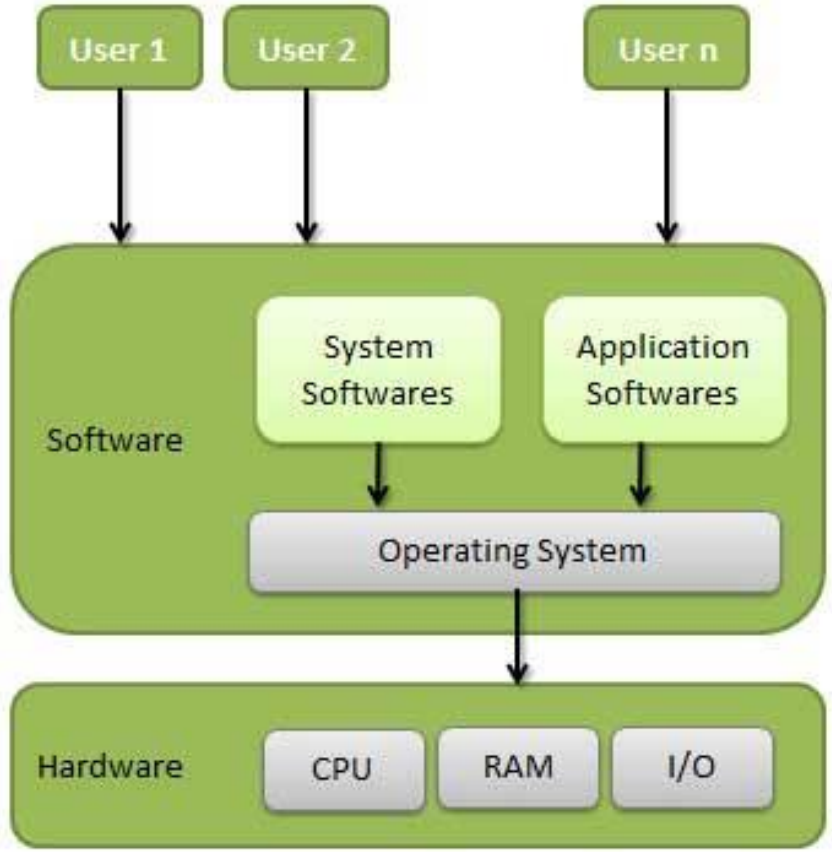
Refrences

- Websites : google.com, medium.com, freeRTOS.org, circuitdigest.com, microcontrollerslab.com

Operating System

- In technical terms, It is a software which manages hardware.

- An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently.



Important functions of operating System

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Type of Operating System

- Single-user operating system
  - MS-DOS, WindowsXp, etc
- Multi-user operating system
  - UNIX, Linux, etc
- Network operating System
  - Microsoft Windows Server 2003, Microsoft Windows Server 2008, etc
- Real Time Operating System
  - For example : Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, and home-appliance controllers, Air traffic control system etc.

Real-Time Operating System (RTOS)

- RTOS is an operating system (OS) intended to serve real-time applications which process data as it comes in, typically without buffering delays.

Processing time requirements (including any OS delay) are measured in tenths of seconds or shorter increments of time.

They either are **event driven** or **time sharing**. Event driven system switches between task based on their priorities while time sharing switch the task based on clock interrupts.

www.wikipedia.org

RTOS - Introduction

- In embedded devices, real-time tasks are critical where timing plays a very important role.
- It is able to process data as comes in, typically without buffering delays.
- Real-time tasks are **Time Deterministic** means the response time to any event is always constant so that it can be guaranteed that any particular event will occur at a fixed time.
- RTOS is designed to run applications with very precise timing and a high degree of reliability.
- RTOS also helps in multi-tasking with a single core.

Real-Time Operating System (RTOS)

- Real-time OS (RTOS) is an intermediate layer between hardware devices and software programming
- “Real-time” means keeping deadlines, not speed
- Advantages of RTOS in SoC design
  - Shorter development time
  - Less porting efforts
  - Better reusability
- Disadvantages
  - More system resources needed
  - Future development confined to the chosen RTOS

RTOS Types

- Soft real-time
  - Tasks are performed by the system as fast as possible, but tasks don’t have to finish by specific times
  - Priority scheduling
  - Multimedia streaming
- Hard real-time
  - Tasks have to be performed correctly and on time
  - Deadline scheduling
  - Aircraft controller, Nuclear reactor controller

RTOS Types

- Firm real-time
  - bound of a deadline but if they miss the deadline it is acceptable but not in the case of Hard RTOS.

Real Time operating System

- Real time system is defines as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment.
- **Real time** processing is always **on line** whereas **on line** system need not be **real time**.
- The key factors in Real Time Operating Systems are minimum interrupt latency and minimum threads switching latency.
- The Real Time Operating System is valued more for how quickly and how predictably it responds to complete the tasks in a given period of time.

Example of RTOS

- Keil RTX Real-Time Kernel
  - http://www.keil.com/arm/rl-arm/kernel.asp
- Micrium µC/OS-II
  - http://www.micrium.com/products/rtos/kernel/rtos.html
- CMX-RTX
  - http://www.cmx.com/rtx.htm
- Express Logic ThreadX
  - http://www.rtos.com/
- Segger ebmOS
  - http://www.segger.com/embos_general.html
- uClinux
  - As distributed with the EA LPC2468 board
- FreeRTOS.org
  - http://www.freertos.org/

Rtos features

- Abstract out timing information
- Maintainability/Extensibility
- Modularity
- Cleaner interfaces
- Easier testing (in some cases)
- Code reuse
- Improved efficiency
- Idle time
- Flexible interrupt handling
- Mixed processing requirements
- Easier control over peripherals

RTOS ADVANTAGES

- Low Priority Tasks
- Precision of code
- Limited Tasks
- Complex Algorithms
- Device driver and interrupt signals
- Thread Priority
- Expensive
- Not easy to program

The µC/OS-II RTOS

DR. SARWAN SINGH

NIELIT CHANDIGARH

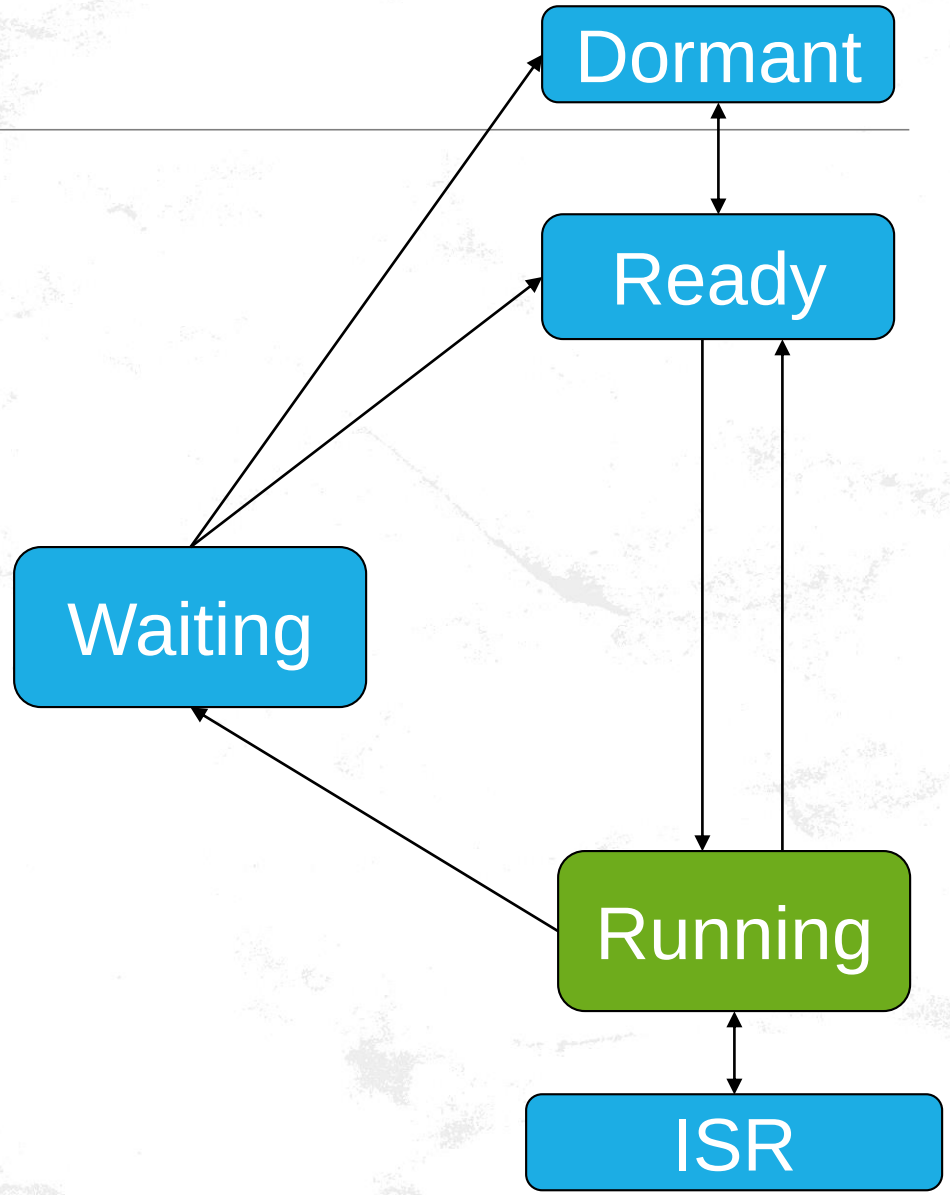
- Micro-Controller Operating Systems (MicroC/OS, stylized as µC/OS) is **a real-time operating system (RTOS) designed by Jean J. Labrosse in 1991**.
- It is a priority-based preemptive real-time kernel for microprocessors, written mostly in the programming language C.
- It is intended for use in embedded systems.

µC/OS-II

- Real-time kernel
  - Portable, scalable, preemptive RTOS
  - Ported to over 90 processors
- Pronounced “microC OS two”
- Written by Jean J. Labrosse of Micrium, http://ucos-ii.com
- Extensive information in **MicroC/OS-II: The Real-Time Kernel (A complete portable, ROMable scalable preemptive RTOS),** Jean J. LaBrosse, CMP Books

Task States

- Five possible states for a task to be in
  - Dormant – not yet visible to OS (use OSTaskCreate(), etc.)
  - Ready
  - Running
  - Waiting
  - ISR – preempted by an ISR



Getting Started with Arduino FreeRTOS

https://www.freertos.org/

- FreeRTOS is specially designed for microcontrollers.
- Because Microcontrollers come with limited resources, therefore, we need an operating system as per the available resources of microcontrollers.
- It is an open-source Kernel that means it can download free of cost and be used in RTOS-based applications.
- Although, it has two commercial variants also such as OpenRTOS and SafeRTOS.

Mastering the FreeRTOS Real Time Kernel - a Hands On Tutorial Guide

FreeRTOS V10.0.0 Reference Manual

Book companion source code

About FreeRTOS

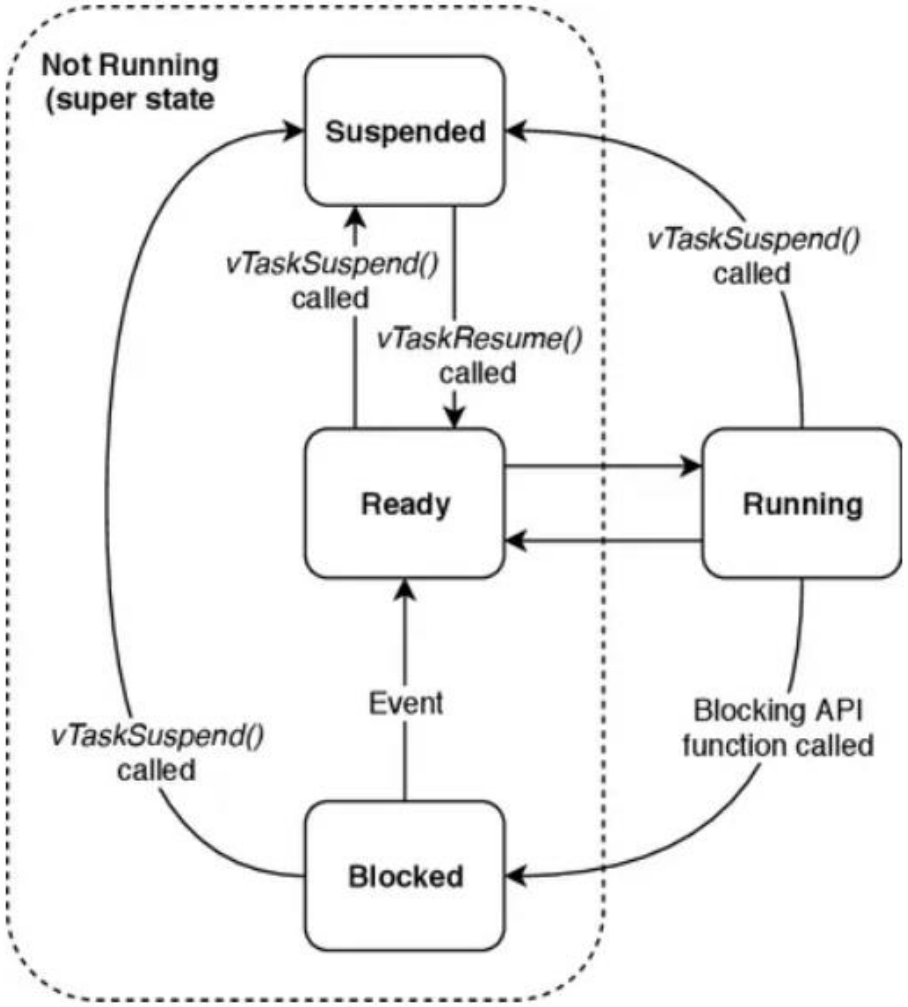
- FreeRTOS is developed by Real Time Engineers Ltd. It is an open-source popular Real-Time Operating System kernel.
- Furthermore, it is used for embedded devices which as microcontrollers, Arduino.
- It is mostly written in C but some functions are written in assembly.
- There are also SafeRTOS and OpenRTOS available online which are similar to FreeRTOS.

About FreeRTOS

- FreeRTOS is a class of RTOS for embedded devices which is small enough to be run on 8/16-bit microcontrollers, although its use is not limited to these microcontrollers.
- It is a completely open-source and it's code is available on github.
- As FreeRTOS can run on 8-bit MCU so it can also be run on Arduino Uno board. We have to just download the FreeRTOS library and then start implementing the code using APIs.

FreeRTOS Task Management

- using a single-core processor, then only one task can run on the processor at any given time. Hence, only one task will be in the running state and all other tasks will be in the not running state.
- In RTOS based applications, tasks can be either in running state or not running state.
- The running state of a task can be further divided into three sub-states such as blocked state, ready state, and suspended state.
- figure shows the transition lifetime of a task in a multitasking system.



Frequently used terms in RTOS

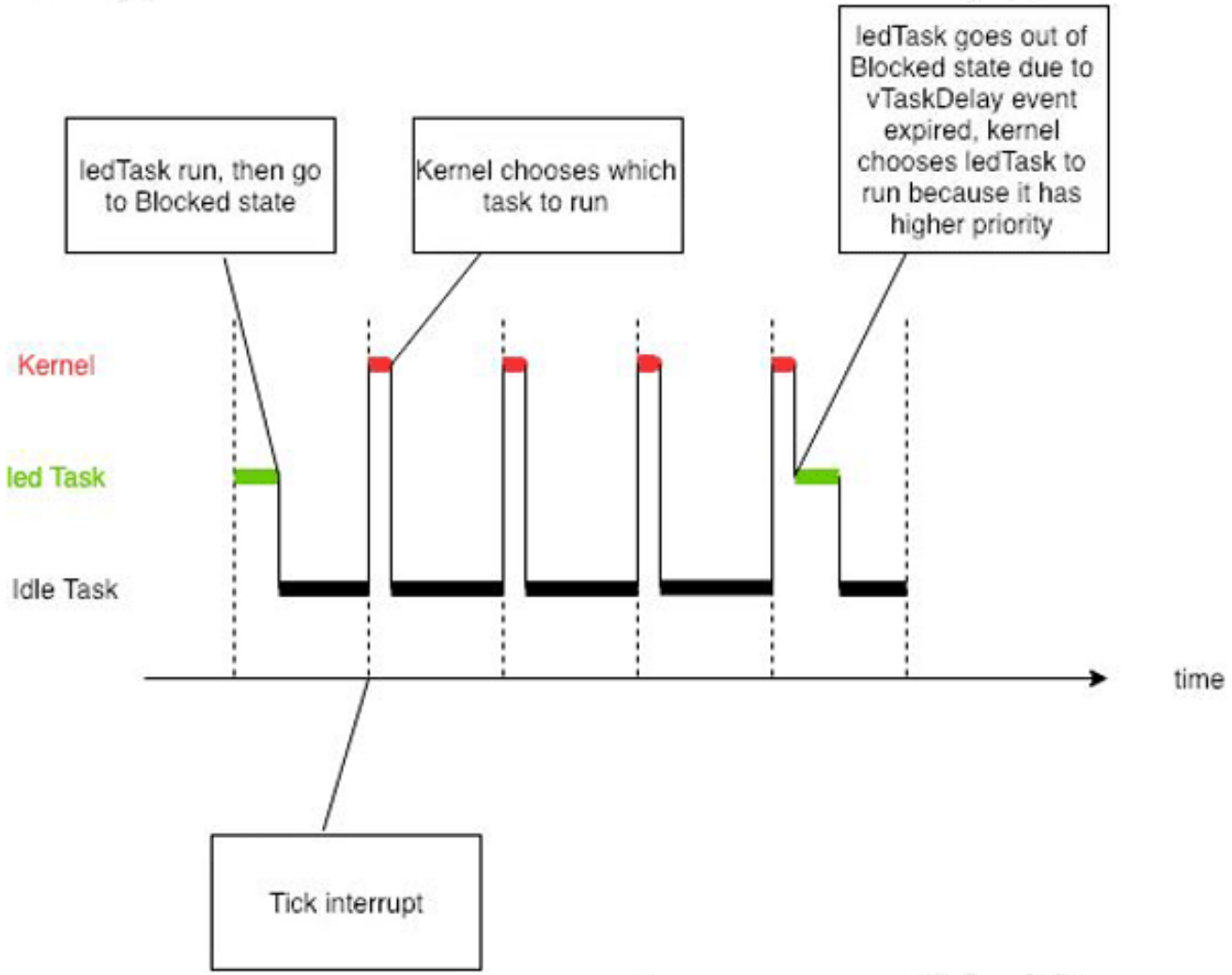
- **Task:** It is a piece of code that is schedulable on the CPU to execute.
- **Scheduler:** It is responsible for selecting a task from the ready state list to the running state. Schedulers are often implemented so they keep all computer resources busy (as in load balancing).
- **Preemption:** It is the act of temporarily interrupting an already executing task with the intention of removing it from the running state without its co-operation.
- **Context Switching:** In priority-based preemption, the scheduler compares the priority of running tasks with a priority of ready task list on every *systick* interrupt. If there is any task in the list whose priority is higher than running task then context switch occurs. Basically, in this process contents of different tasks get saved in their respective stack memory.

Frequently used terms in RTOS

- **Types of Scheduling policies:**
  - **Preemptive Scheduling:** In this type of scheduling, tasks run with equal time slice without considering the priorities.
  - **Priority-based Preemptive:** High priority task will run first.
  - **Co-operative Scheduling:** Context switching will happen only with the co-operation of running tasks. Task will run continuously until task yield is called.
- **Kernel Objects:** For signaling the task to perform some work, the synchronization process is used. To perform this process Kernel objects are used. Some Kernel objects are Events, Semaphores, Queues, Mutex, Mailboxes, etc.

How RTOS works

- Task is a piece of code that is schedulable on the CPU to execute. So, if you want to perform some task, then it should be scheduled using kernel delay or using interrupts.
- This work is done by Scheduler present in the kernel. In a single-core processor, the scheduler helps tasks to execute in a particular time slice but it seems like different tasks are executing simultaneously.
- Every task runs according to the priority given to it.
- Now, let's see what happens in the RTOS kernel if we want to create a task for LED blinking with a one-second interval and put this task on the highest priority.


ledTask run, then go to Blocked state
Kernel chooses which task to run
ledTask goes out of Blocked state due to vTaskDelay event expired, kernel chooses ledTask to run because it has higher priority
Kernel
led Task
Idle Task
time
Tick interrupt

Happy Coding

- Github Repo

https://github.com/sarwansingh/IoT/tree/master/ClassExamples/CEPTAM_ESDANov23