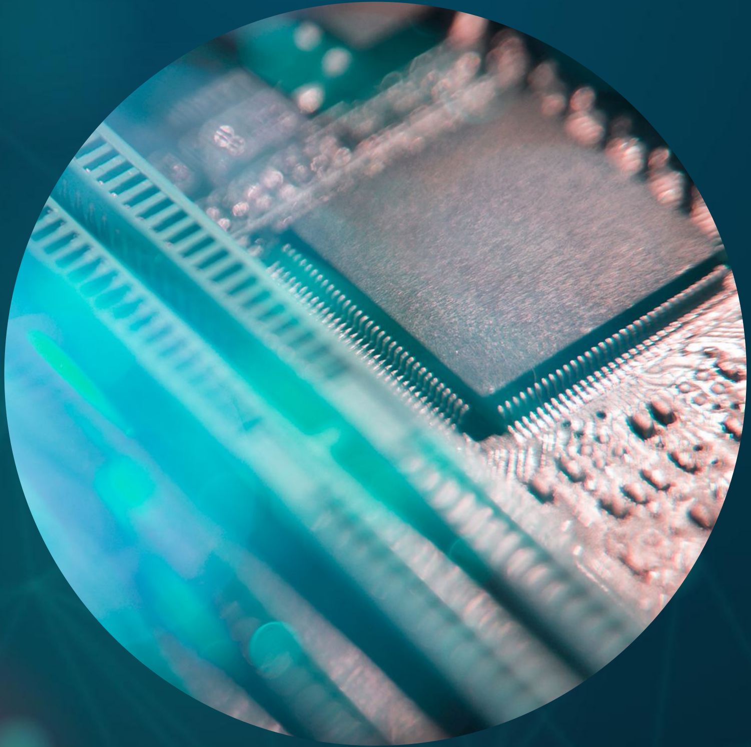
IMT SCHOOL

ARM Based Microcontroller

OS Scheduler

Lecture 7



Advanced RISC Machines

IMT SCHOOL

Introduction 01

Operating System Scheduler

Super Loop Background

Consider we have 5 LEDs and each one has a blinking time (1 , 1.5 , 0.5 ,5 , 3.5)ms respectively what would we do to design this system ... ?

We can use a timer which fires every 0.5 ms. Inside the ISR we can increment a counter. According to the counter value we will blink the LEDs in a super while(1) loop.

What if we have 100 LEDs and every one has a different blinking time ... ?
It will be more complex now to Synchronize between LEDs right !

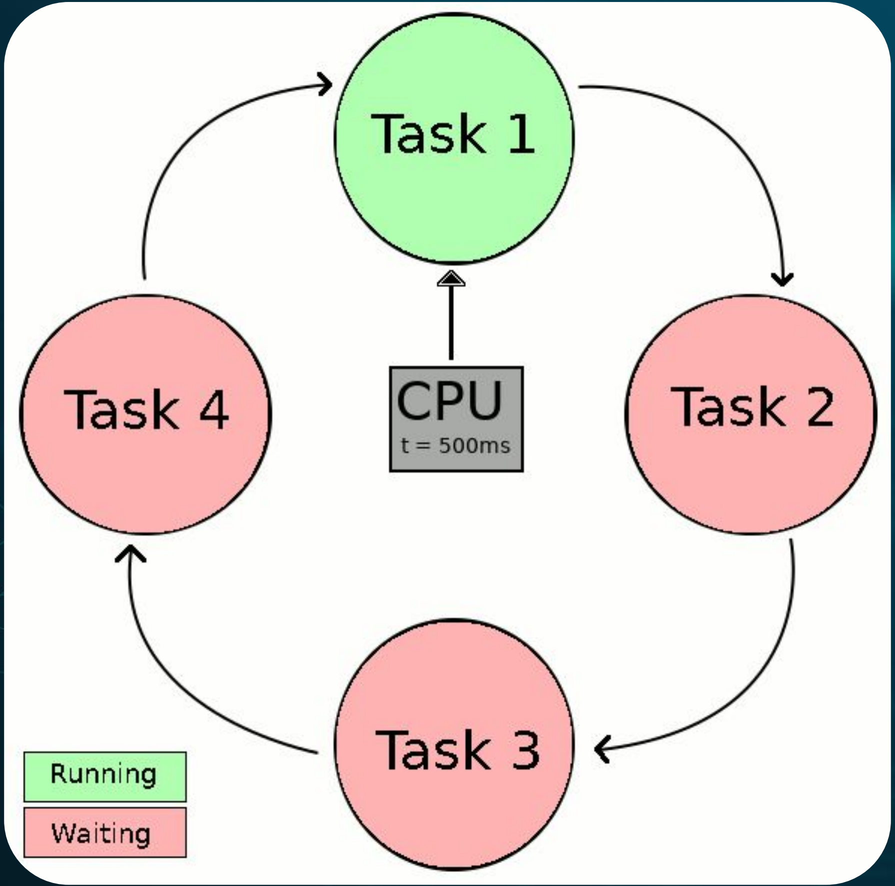
OS(Operating System)

The Operating System is a system which takes number of Tasks and perform them in a specific time called Periodicity. Task is nothing but a piece of code that repeated every specific time. Every task has its own periodicity . The OS will Synchronize between these task Simply.

An **embedded operating system** is an operating system for embedded computer systems. These operating systems are designed to be compact, efficient at resource usage, and reliable, forsaking many functions that standard desktop operating systems provide, and which may not be used by the specialised applications they run.The hardware running an embedded operating system is usually very limited in resources.

Systems made for embedded hardware tend to be very specific, which means that due to the available resources (low if compared to non-embedded systems) these systems are created to cover specific tasks.

OS(Operating System)

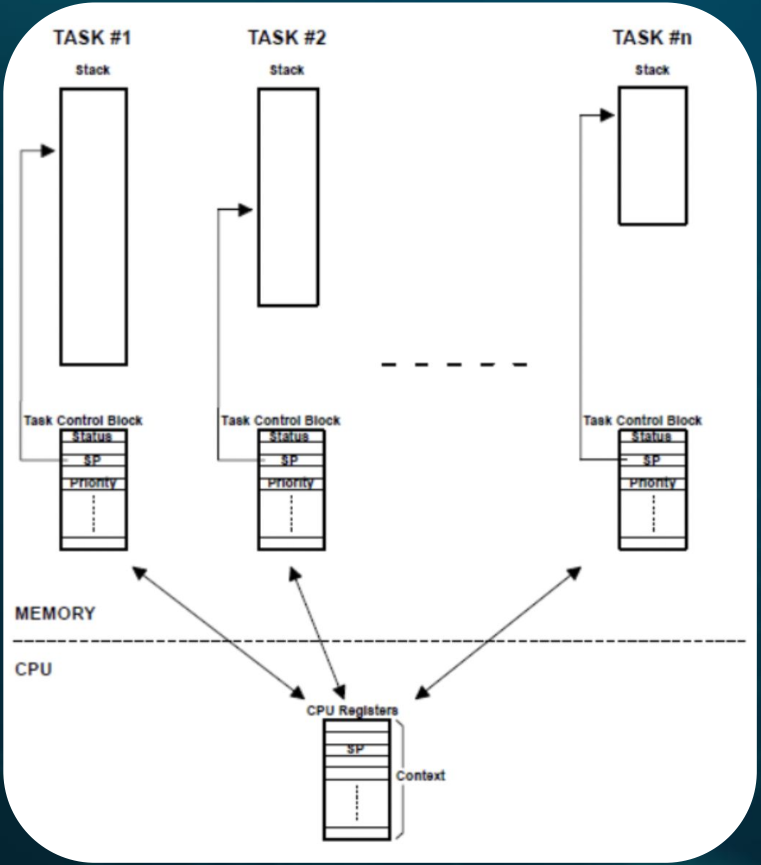


Multitasking

➢Multitasking is a process of scheduling and switching the Central processing unit (CPU) between several tasks .

➢A single CPU switches its attention between several sequential tasks.

➢Application programs are typically easier to design and maintain if multitasking is used.



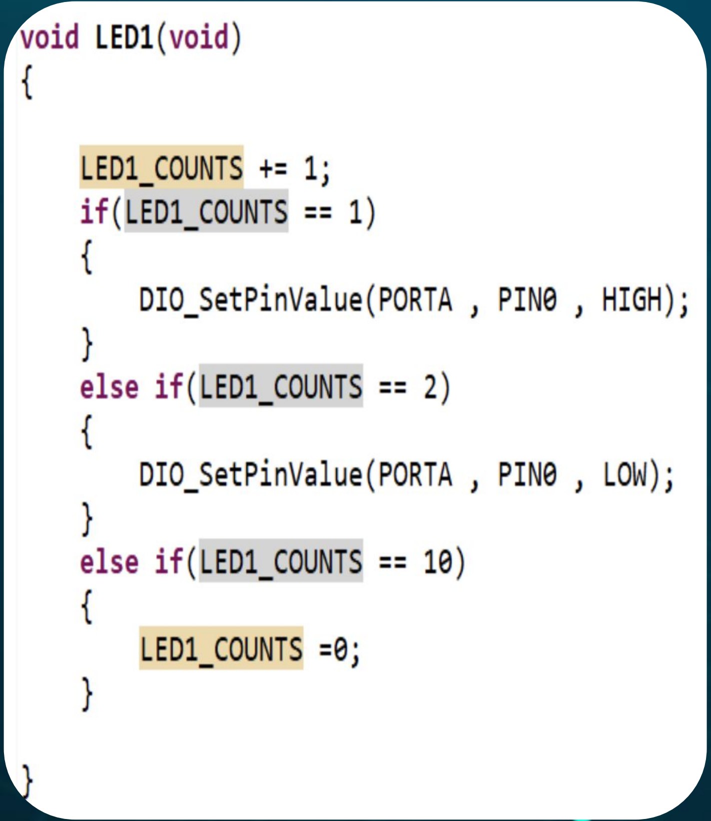
Tasks 02

Operating System Scheduler

Task

Task, or also called a thread, is a simple program that thinks it has the CPU all to itself.
Task is a function which Takes void and returns void only contains some lines of codes.
Every task could be periodic task .

Now every LED will be a Task and the OS will Sync between them.



```
void LED1(void)
{

    LED1_COUNTS += 1;
    if(LED1_COUNTS == 1)
    {
        DIO_SetPinValue(PORTA , PIN0 , HIGH);
    }
    else if(LED1_COUNTS == 2)
    {
        DIO_SetPinValue(PORTA , PIN0 , LOW);
    }
    else if(LED1_COUNTS == 10)
    {
        LED1_COUNTS =0;
    }

}
```

Task Parameters

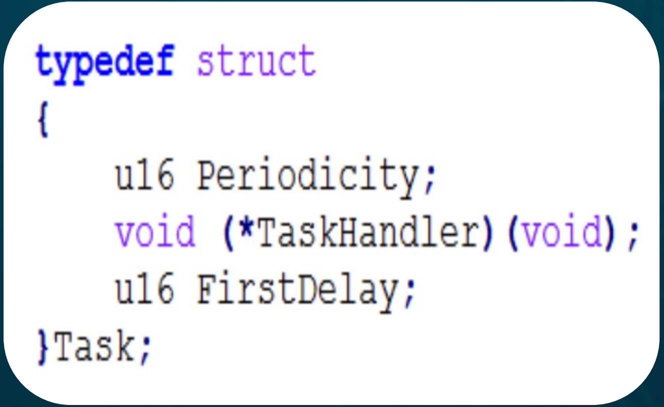
Tasks have main parameters:

- **The function (pointer to function)**
The lines of code that will be executed by CPU.

- **The Periodicity**
The periodic time of the task to be executed .

- **The Priority**
the more importance the task , the higher priority given to it .

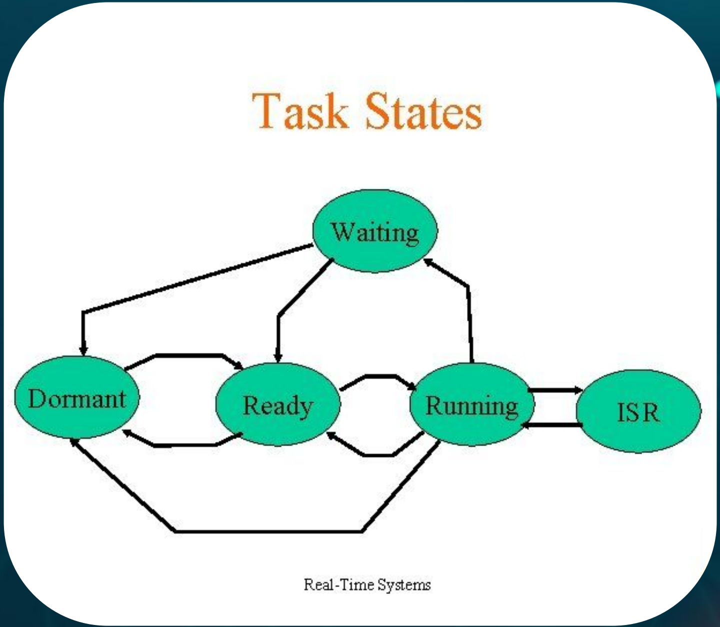
- **The first delay**
The parameter which tells the scheduler that this task should be in the ready state for the first time at which tick.



```
typedef struct
{
    u16 Periodicity;
    void (*TaskHandler)(void);
    u16 FirstDelay;
}Task;
```

Task States

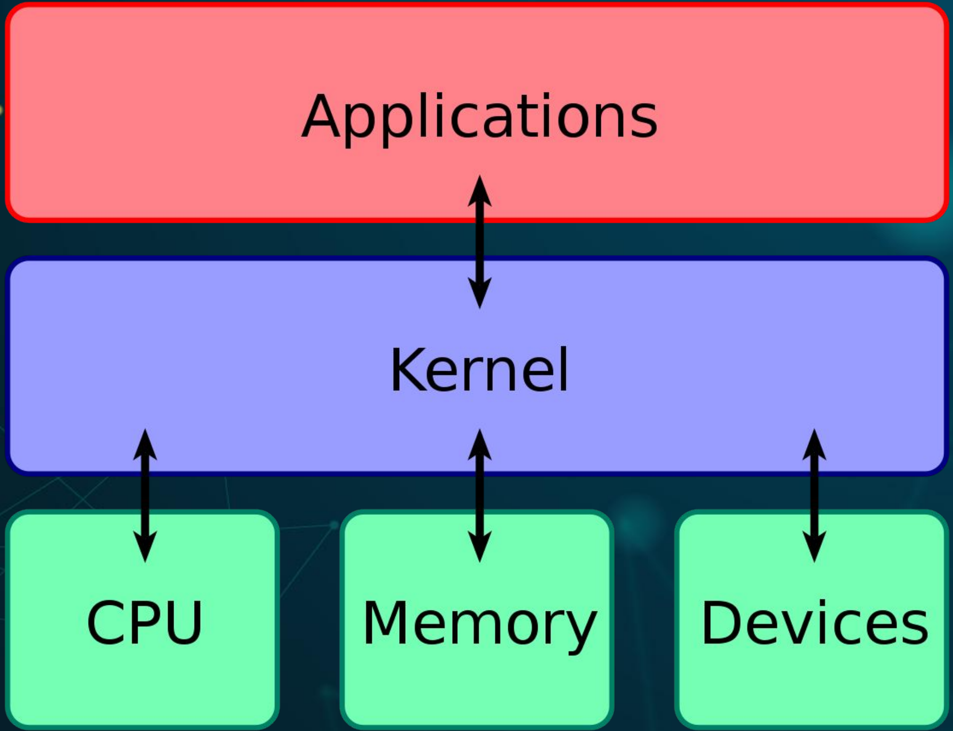
Dormant
Task that resides in memory but has not been passed the RTOS to start scheduling.
Ready
Task that can execute but its priority is less than the currently running task
Running
Task that has the control of CPU.
Waiting
Task that requires the occurrence of an event.(for ex "waiting for an I/o operation")
ISR
When an interrupt has occurred and the CPU is in the process of service the interrupt



Kernel 03

Operating System Scheduler

Kernel



Kernel

Kernel is central component of an operating system that manages operations of computer and hardware. It basically manages operations of memory and CPU time. It is core component of an operating system. Kernel acts as a bridge between applications and data processing performed at hardware level using inter-process communication and system calls.

Kernel loads first into memory when an operating system is loaded and remains into memory until operating system is shut down again. It is responsible for various tasks such as disk management, task management, and memory management.

It decides which process should be allocated to processor to execute and which process should be kept in main memory to execute. It basically acts as an interface between user applications and hardware. The major aim of kernel is to manage communication between software i.e. user-level applications and hardware i.e., CPU and disk memory.

Kernel Functions

Following are the functions of a Kernel:

Access Computer resource: A Kernel can access various computer resources like the CPU, I/O devices and other resources. It acts as a bridge between the user and the resources of the system.
Resource Management: It is the duty of a Kernel to share the resources between various process in such a way that there is uniform access to the resources by every process.
Memory Management: Every process needs some memory space. So, memory must be allocated and deallocated for its execution. All these memory management is done by a Kernel.
Device Management: The peripheral devices connected in the system are used by the processes. So, the allocation of these devices is managed by the Kernel.

Scheduler 03

Operating System Scheduler

IMT SCHOOL

Scheduler

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types –

Long-Term Scheduler
Short-Term Scheduler
Medium-Term Scheduler

Scheduler

Comparison among Scheduler

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Scheduler

Scheduler : also called dispatcher , is the part of the kernel responsible for determining which task runs next.
Most Real time Kernel are priority passed.
Each task is assigned a priority based on it's importance.
The priority of every task is application specific.
In a priority based kernel ,control of the CPU is always given to the highest priority task ready to run.

The Scheduler runs every a specific time called TICK time.

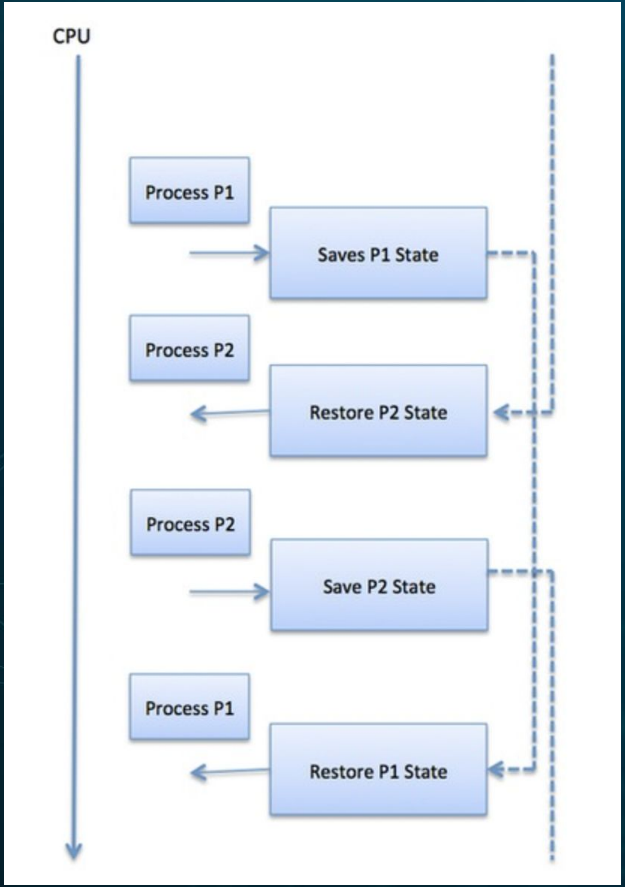
Two types of priority based kernel exist:
1. Preemptive kernel
2. Non-preemptive Kernel

Context Switching

A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.

When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.

Context Switching



Context Switching

Context switches are computationally intensive since register and memory state must be saved and restored. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers. When the process is switched, the following information is stored for later use.

- Program Counter
- Scheduling information
- Base and limit register value
- Currently used register
- Changed State
- I/O State information
- Accounting information

Preemptive scheduling

Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in ready queue till it gets next chance to execute.

A preemptive kernel is used when system responsiveness is important.
The highest priority task ready to run is always given control of the CPU.
The most commercial Real time Kernels are preemptive .

If the ISR a higher priority task ready , when the ISR completes , the interrupt task is suspended, and the new higher priority task is resumed

Non-Preemptive scheduling

Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state. In this scheduling, once the resources (CPU cycles) is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state. In case of non-preemptive scheduling does not interrupt a process running CPU in middle of the execution. Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.

Non-preemptive kernel requires that each task does something to explicitly give up control of the CPU

The ISR can make a higher priority task ready to run , but the ISR always returns to the interrupted task. The new higher priority task gains control of the CPU only when the current task give up the CPU.

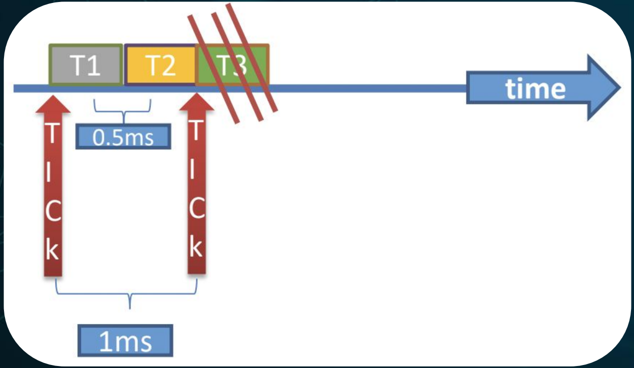
Tick Time 03

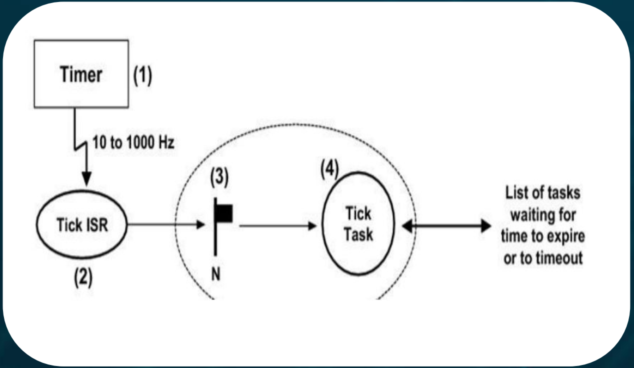
Operating System Scheduler

Tick Time

Tick time is that time which the Scheduler takes place. Tick time must be suitable for all tasks.

Consider that we have a scheduler with Tick time (1ms) and we have three tasks will run at next Tick every one of them takes 500us to be performed . The scheduler will ignore the third task As if it did not exist and perform another one at next tick and that is a Disaster.





CPU Load 04

Operating System Scheduler

Non-Preemptive scheduling

CPU load is one of the most important parameters in the OS .
CPU load is that how much is the CPU busy between two ticks.
CPU load could be determined by a simple equation.

$$CPU\ load = \frac{\sum Exicution\ time\ in\ the\ worst\ case}{Tick\ time}\%$$

CPU load have to be less than 100%.
✓ The lower the CPU load the better the system
✓ It is better for a CPU load to be between 60% and 80%.
✓ The tick time must be greater than the greatest execution time of a task.

Task Priority 04

Operating System Scheduler

Task Priority

Task Priority: A priority is assigned to each task. the more importance the task, the higher priority given to it.

Static Priority: Task priorities are static when the priority of each task does not change during the execution time. Each task is given a fixed priority at a compile time.

Dynamic Priority: Task priorities are dynamic if the priority of the task can be change during the run time this feature is used is used in Real-time kernel to avoid priority inversion (will discussed later)

STM32
Is AWESOME

Session LAb

Create your simple **RTOS**

Time To Code

IMT SCHOOL

THANKS!

Do you have any questions?

www.imtschool.com

www.facebook.com/imaketechnologyschool/