Embedded system

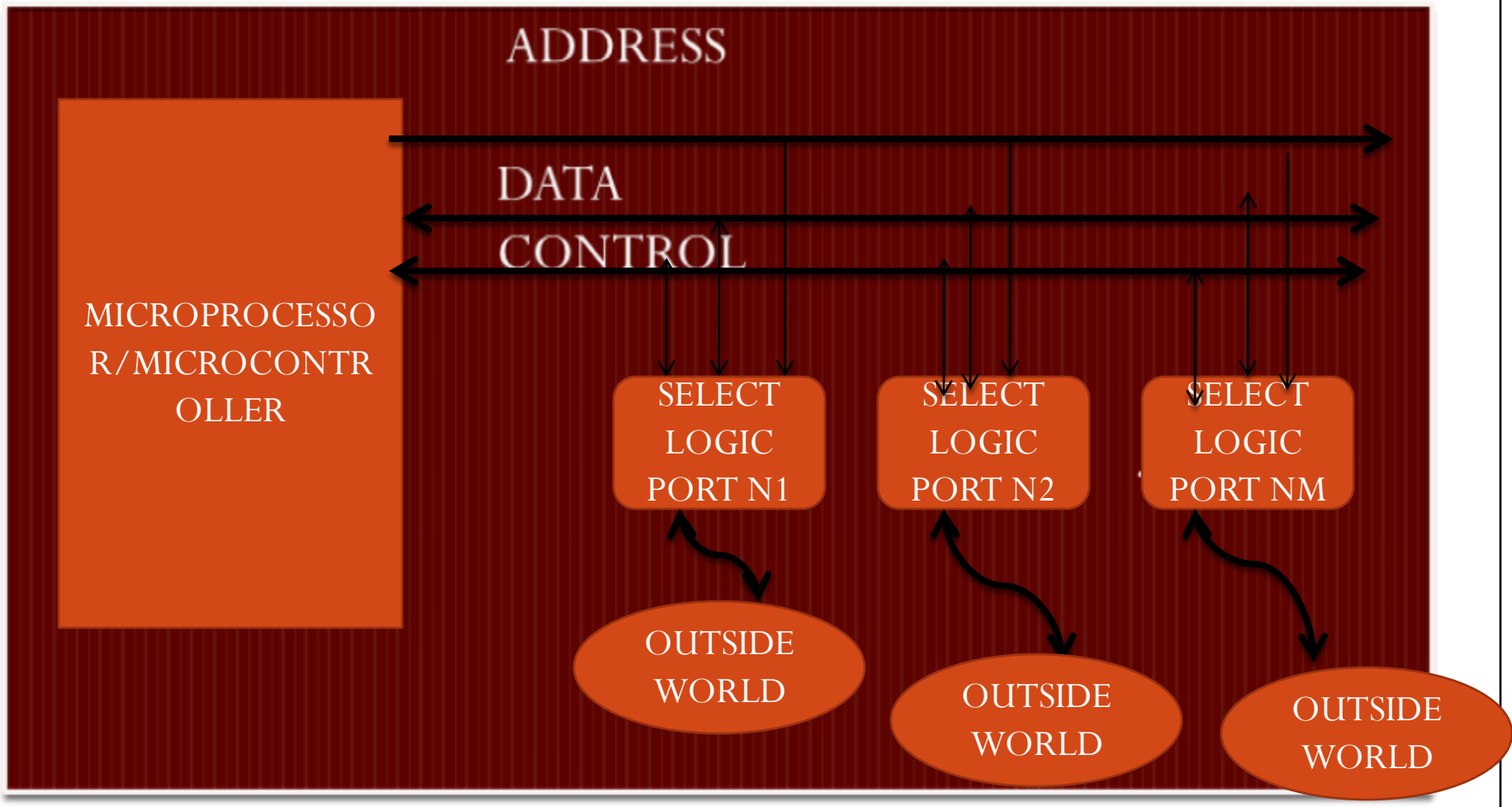
PERIPHERALS IN EMBEDDED SYSTEM

Simplex , duplex and semi duplex

- Simplex communication involves unidirectional data transfer.
- Duplex communication involves bi-directional data transfer
- Full duplex interface have independent channels for transmission and reception.
- Semi duplex communication involves data bi-directional data transfers however at a given time , the data transfer is only possible in one direction.
- Semi duplex interface involves the same communication channel for both transmission and reception.

Control and Status Registers

- The connection of peripherals to the microprocessor is illustrated.
- The microprocessor or microcontroller bus provides address signals (selecting the peripherals in much the same way that memory location are selected), a bi-directional data bus for data transfers, and control signals for controlling the interaction between the peripherals and the microprocessor.


ADDRESS
DATA
CONTROL
MICROPROCESSOR/MICROCONTROLLER
SELECT LOGIC PORT N1
SELECT LOGIC PORT N2
SELECT LOGIC PORT NM
OUTSIDE WORLD
OUTSIDE WORLD
OUTSIDE WORLD

addressing

- A typical microprocessor system will have several peripherals
- Each must be individually addressed . Therefore each peripherals IC has a unique address. Address can be in memory space.

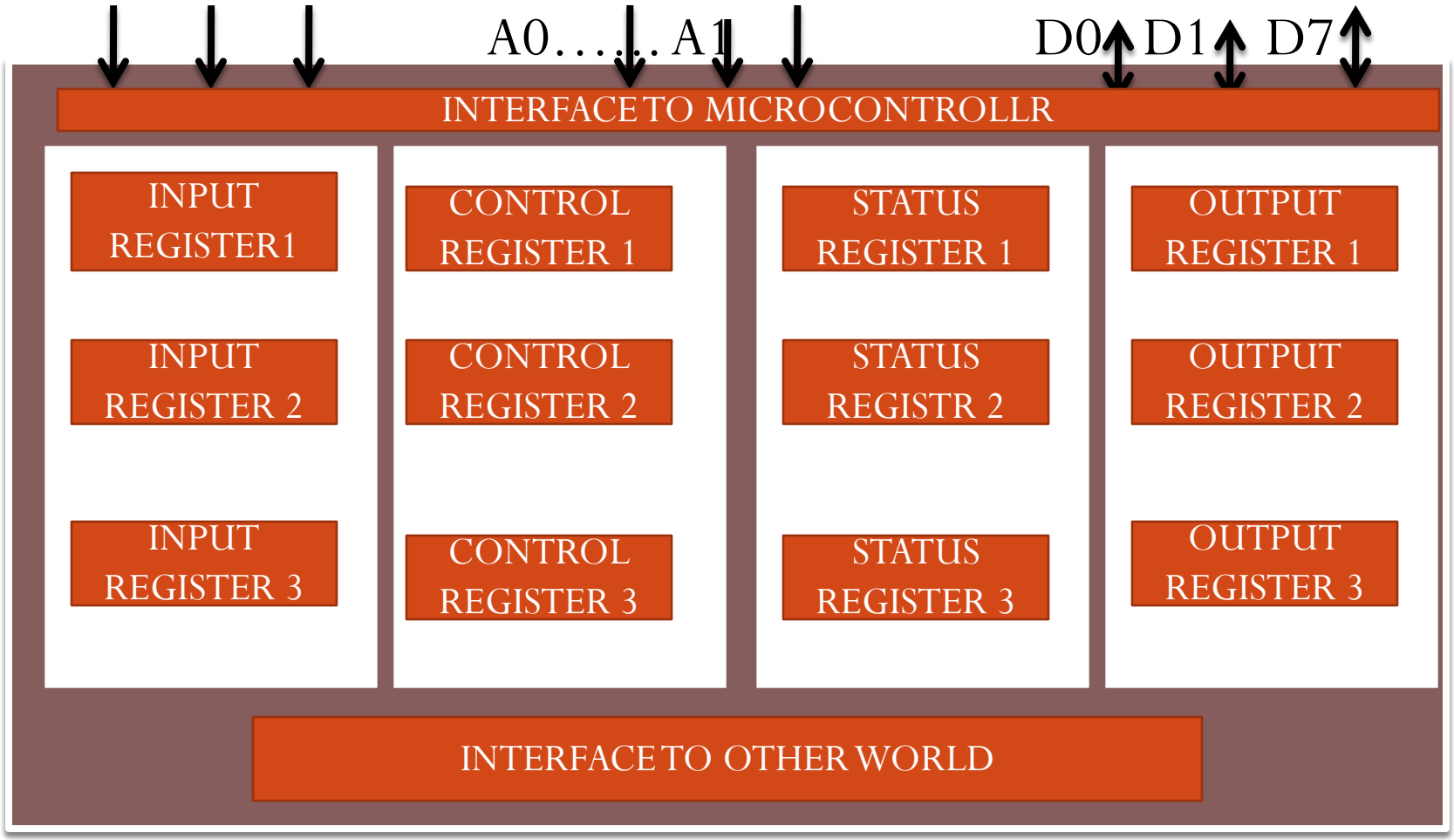
Information exchange

- Each pheriphals IC must correctly communicate with the microprocessor or microcontroller

Internal registers

- Each peripherals IC has an internal structure into which the microprocessor or microcontroller can write information and / or from which the microprocessor or microcontroller can read information.
- This internal structure will be represented by a set of register of four basic types.

Microprocessor side of peripherals

- SELECT READ WRITE ADDRESS DATA BYTE
- 

- **CONTROL REGISTER** : The 'control register' sets up the specific function to be performed by peripherals.
- **STATUS REGISTER** : The 'status register' provide readable information which the microprocessor can access to determine the internal state of the peripherals.
- **DATA REGISTER**: The data is held in the data registers. Data intended to be sent to the external world are found in the "output data register' whereas the data intended to be sent to the 'microprocessor' are found in the "input data register ".

- Thus the control and status registers are written and read through data transfer between the microprocessor / microcontroller and the peripherals.
- Normally , a routine will set up the peripheral , modify control as needed during the program, check on the status information before sending or getting data , and read/write data from and to the peripherals.

DEVICE DRIVER CONCEPTS

- Device drivers are the software that provides an interface between the operating system and the specific hardware device such as terminals , disks, network media.
- A device driver is a hides the details of a particular peripheral and provides a slightly high-level programming interface to it(an **interface** is a tool and concept that refers to a point of interaction between components, and is applicable at the level of both hardware and software)
- Device drivers allow application software to attach to, read and write data from, and change the behavior of the peripheral device.
- A device driver is typically specific to a given operating system

Generic functions

- Generic functions used for the commands to the device are device *create* ( ), *open* ( ), *connect* ( ), *bind* ( ), *read* ( ), *write* ( ), *ioctl* ( ) [for IO control], *delete* ( ) and *close* ( ).

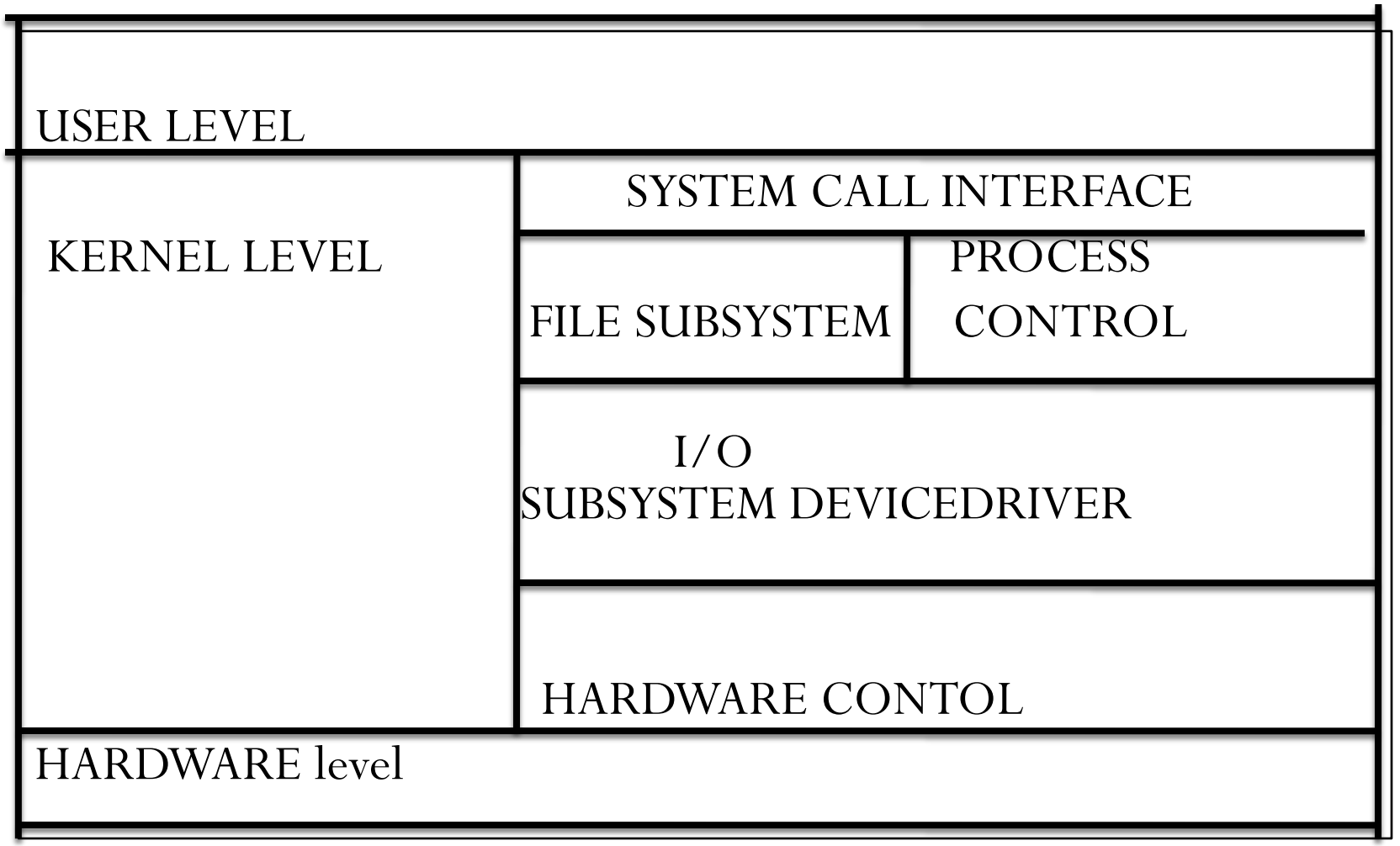
UNDERSTANDING PURPOSE

- USER LEVEL :access control to a file, printer or other network resource based on username
- KERNEL LEVEL :the **kernel** is the main component of most computer operating systems; it is a bridge between applications **Application software**, also known as an **application** or an **app**, is computer software designed to help the user to perform specific tasks. Examples includeenterprise software, accounting software, office suites, graphics software andmedia players.
-)and the actual data processing(**Computer data processing** is any process that uses a computer program to enter data and summarise, analyse or otherwise convert data into usable information. It involves recording, analysing, sorting, summarising, calculating, disseminating and storing data.) done at the hardware level.The kernel's responsibilities include managing the system's resources (the communication between hardware and software components)
- HARDWARE LEVEL :The Hardware Level of the operating system controls the use of physical system resources, such as the memory manager, process manager, disk drivers

- **Process control** is a statistics and engineering discipline that deals witharchitectures, mechanisms and algorithms for maintaining the output of a specific process within a desired range.
- a **system call** is how a program requests a service from an operating system's kernel. This may include hardware related services (e.g. accessing the hard disk), creating and executing new processes, and communicating with integral kernel services (like scheduling). System calls provide an essential interface between a process and the operating system.
- **input/output** or **I/O** is the communication between an information processing system (such as a computer) and the outside world, possibly a human or another information processing system. Inputs are the signals or data received by the system, andoutputs are the signals or data sent from it.

- HARDWARE CONTROL :The control of, and communications between, the various parts of a computer system.

STUCTURE OF DEVICE DRIVER



- FUNCTIONS likes for accessing files , to open ,close , read and written to. These functions are accomplished through standard system calls from a high level program
- The system calls gives the application program access to the kernel service , which identify the device containing the file and the type of I/O requested.
- The kernel then execute the device drive routine provided to perform the function.
- Above figure also elaborate the link between the user level and the hardware level.

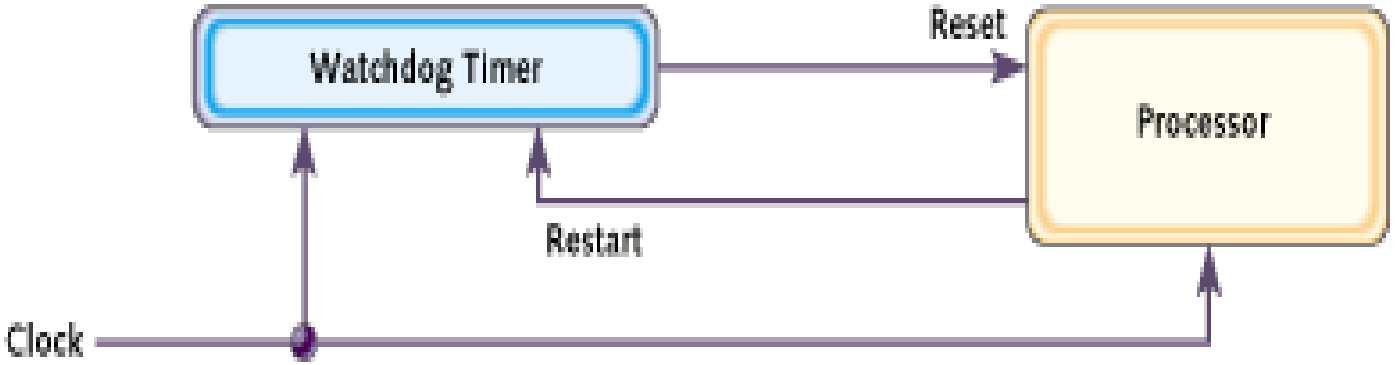
- By issuing the system calls from the user level, a program accesses the file and process control subsystems which in turn access the device driver.
- The device driver brings the device into and out of service , sets the hardware parameter in the device , transmits data from the kernel to the device , receives data from the device and pass it back to the kernel and handles device errors.
- The driver provides and manages a path for the data to or from the hardware device and service interrupts issued by the device controller.

WATCHDOG TIMER

- A watchdog timer is a piece of hardware that can be used to automatically detect software anomalies and reset the processor if any occur(A **watchdog timer** (or **computer operating properly (COP)** timer) is a computer hardware or software timer that triggers a system reset or other corrective action if the main program, due to some fault condition such as a hang,)
- Watchdog timers are commonly found in computer-controlled equipment in cases where humans cannot easily access the equipment or would be unable to react to faults in a timely manner.

- Many embedded systems cannot depend on a human to reboot them if the software hangs; they must be self-reliant. Some embedded systems, such as space probes, are not physically accessible to human operators; these systems might become permanently disabled if they were unable to autonomously recover from faults. In cases such as these, a watchdog timer is usually employed.
- Watchdog timers may also be used when running untrusted code in a sandbox(a **sandbox** is a security mechanism for separating running programs. It is often used to execute untested code, or untrusted programs from unverified third-parties, suppliers, untrusted users and untrusted websites) to limit the CPU time available to the untrusted code and thus prevent some types of denial-of-service attacks.[3]

Figure 1: A typical watchdog setup



- A watchdog timer may initiate any of several types of corrective actions.
- Figure 1 shows a typical arrangement. As shown, the watchdog timer is a chip external to the processor. However, it could also be included within the same chip as the CPU. This is done in many microcontrollers. In either case, the output from the watchdog timer is tied directly to the processor's reset signal
- **Kicking the dog**
- The process of restarting the watchdog timer's counter is sometimes called "kicking the dog." The appropriate visual metaphor is that of a man being attacked by a vicious dog. If he keeps kicking the dog, it can't ever bite him. But he must keep kicking the dog at regular intervals to avoid a bite. Similarly, the software must restart the watchdog timer at a regular rate, or risk being restarted.

```
main(void)
{
        hwinit();

        for (;;)
        {
           *pWatchdog = 10000;
           read_sensors();
           control_motor();
           display_status();
        }
}
```

- Suppose that the loop must execute at least once every five milliseconds. (Say the motor must be fed new control parameters at least that often.) If the watchdog timer's counter is initialized to a value that corresponds to five milliseconds of elapsed time, say 10,000, and the software has no bugs, the watchdog timer will never expire; the software will always restart the counter before it reaches zero

Real time characteristics of embedded operating system.

- Usually some specific characteristics or measurable parameters are highly regarded in RTOSs.
- RESPONSE TIME :
- An important parameter when evaluating an RTOS is its response time.
- Typical parameter falling in this category:
- 1)interrupt latency: the time from an interrupt request and the interrupt servicing(A section of a computer program or of the operating system that takes control when an interrupt is received and performs the operations required to service the interrupt.)the overhead can be caused by extra code inserted by the RTOS into interrupt handler code path (A routine which is executed when an interrupt occurs)

- 2)Thread fly back time: the time from an hardware event usua;;y an interrupt and the restart of the thread (In computer science, a **thread of execution** is the smallest sequence of programmed instructions that can be managed independently by an operating system scheduler. A thread is a light-weight process. The implementation of threads and processes differs from oneoperating system to another, but in most cases, a thread is contained inside a process. Multiple threads can exist within the same process and share resources such as memory, while different processes do not share these resources. In particular, the threads of a process share the latter's instructions (its code) and its context (the values that its variables reference at any given moment).supposed too handle it

- 3)context switch time: the time required to synchrconously switch from the context of one thread to the context of another thread

- JITTER:Jitter is the deviation affect the ability of theprocessor in a personal computer to perform as intended; introduce clicks or other undesired effects in audio signals, and loss of transmitted data between network devices
- Here jitter involved in response time :
- Some factors that determine the system behavior regarding jitter

- Thread priorities assignment
- Interrupt priorities assignement
- Interrupt handlers
- Interaction between threads through shared resources protected by mutual exclusion

- SIZE :
  - In an embedded system the RTOS is an important overhead in terms of occupied memory, a more compact RTOS is preferable being all the other parameters equal because memory cost

- RELIABILITY:
- design choice that makes some system more reliable than other
- Fully static design do not have those limitation

- SYNCHRONIZATION PRIMITIVES:
- Variety in available primitives is also an important factor to be considered
- Having the correct tool for the job can reduce devlopement time and often also helps when integrating with the external code with the RTOS.