[Definition:]
An embedded system has embedded software and computer hardware which makes a system dedicated for an application or specific part of an application or product or part of a larger system.

[An embedded system is]:

- A microprocessor / microcontroller based system
- Software driven
- Reliable
- Real time control system
- Autonomous or human or network interactive
- Operating on diverse physical variables and in diverse environments
- Sold into competitive and cost conscious market.

[Components of Embedded System]

An embedded system has 3 main components embedded into it:

1. Hardware - The hardware consists of :-
    1. CPU
    2. Read Only Memory
    3. Random Access Memory
    4. I/O ports
    5. Serial Communication ports
    6. Timers
    7. Interrupt Controller
    8. I/O devices
    9. Networking units / bus drivers
2. **Application Software** - An application software may concurrently perform a series of tasks or processes or threads
3. Real Time Operating System -
    1. The OS supervises the application software running on hardware
    2. Organizes access to resource according to priorities of tasks in the system.
    3. Provides a mechanism to let the processor run the process as scheduled and context switch between various processes.

[Layered architecture of Embedded System]

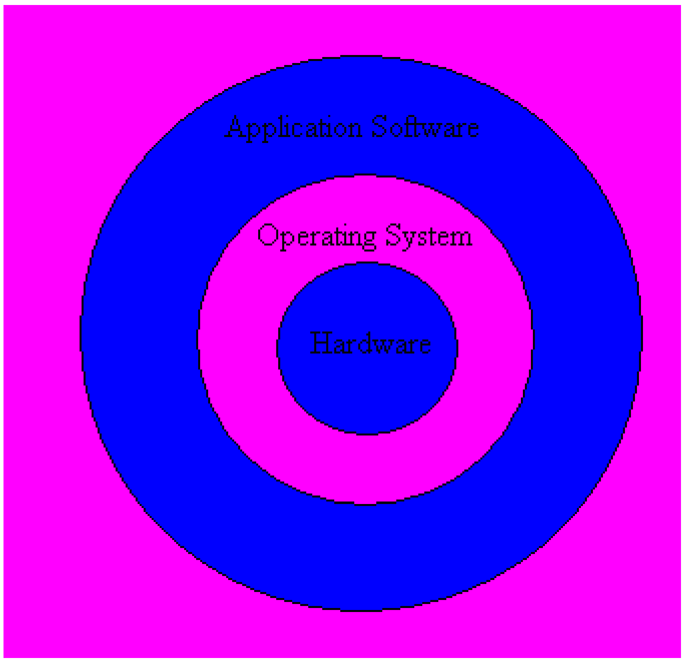
Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU). This hardware also contains memory chips onto which the software ,called ‘firmware’, is loaded

The operating system runs above the hardware, and the application software runs above the operating system. The same architecture is applicable to any computer including a desktop computer.

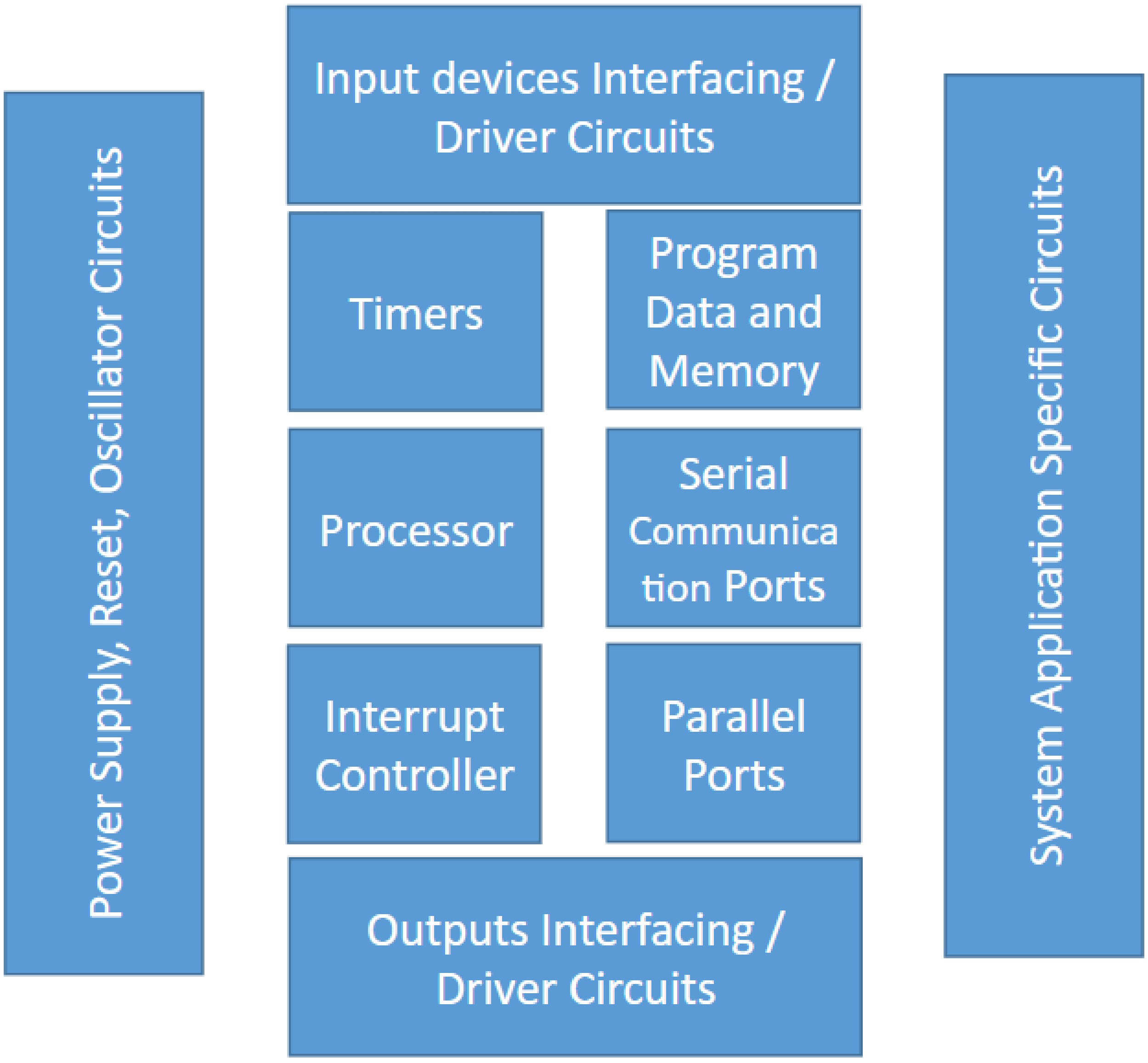
However, there are significant differences. It is not compulsory to have an operating system in every embedded system.

For small appliances such as remote control units, air-conditioners, toys etc., there is no need fir an operating system and we can write only the software specific to that application.

For applications involving complex processing, it is advisable to have an operating system.eg smart appliances with complex user interfaces



[Components of Embedded System Hardware]


Power Supply, Reset, Oscillator Circuits
Input devices Interfacing / Driver Circuits
Timers
Program Data and Memory
Processor
Serial Communication Ports
Interrupt Controller
Parallel Ports
Outputs Interfacing / Driver Circuits
System Application Specific Circuits

[An embedded system processor can be one of the following:]

1. General Purpose Processor (GPP) – Is a general purpose processor with instruction set designed not specific to the application e.g., Microprocessor
2. Digital Signal Processor (DSP)

The GPP is further classified as microcontrollers and microprocessors. Microcontrollers have memory and other peripherals on the chip itself, and hence is the best choice for small embedded systems.

[An Application Specific Instruction Set Processor (ASIP)]
ASIP Is a processor with instruction set designed for specific applications e.g., Microcontroller, Digital Signal Processor

[Single Purpose Processors]
these are additional processors which are co-processors used along with main processor for additional functions – e.g., graphic processing, encrypting.

[System On Chip (SOC)]
system on a VLSI chip that has all the necessary analog as well as digital circuits, processors and hardware.

[Memory]

- Memory is divided into program memory and data memory.
- Program memory is a Read memory and stores firmware (program code) permanently where as data memory is read / write holds the data dynamically.
- In a microcontroller, both program memory and data memory are internal, if the internal memory is not sufficient, external memory chips can be used.
- EEPROM memory can be additionally used as programmable memory.

[Clock]

- A clock or oscillator circuit is required to be provided as input for clock signal.
- In some processors, the clock is inbuilt.
- All events of the processor are related to the clock.
- Higher the clock frequency, greater the speed of the processor.

[Watchdog timer]

- A watchdog timer helps in reset function of the processor.
- The timer is set to a large value and is decremented slowly. When it reaches zero, the processor is reset through a reset signal

[I/O Devices]

- I/O devices can be classified as programmed I/O or interrupt driven I/O.
- In programmed I/O, the processor sends the data to the device by itself.
- In interrupt driven I/O, the processor is driven by an interrupt signal and an ISR is executed. The ISR transfers the data from input device to memory or memory to output device.
- Generally the data transfer between I/O devices and memory is coordinated by the CPU. In cases where this transfer is not efficient, the function is carried out by a special device called DMA controller.
- Sensors and Transducers:
    - Embedded systems need to convert real life information into digital signals. This is achieved through sensors and transducers.
    - E.g., Temperature sensors convert temperature into electrical voltage (used in air conditioners, boilers, ovens etc.).
    - Light sensors convert light intensity into electrical voltage
    - Accelerometer converts acceleration into voltage
    - Pressure sensors convert pressure level into voltage
    - Microphone and speakers convert acoustic energy into voltage level
- ADC & DAC: The analog signal can be converted to digital signal through sampling and quantization technique (ADC). The reverse process is called DAC.

[Features of an Embedded System]
Embedded systems do a very specific task, they cannot be programmed to do different things.

- Embedded systems have very limited resources, particularly the memory. Generally, they do not have secondary storage devices such as the CDROM or the floppy disk.
- Embedded systems have to work against some deadlines. A specific job has to be completed within a specific time. In some embedded systems, called real-time systems, the deadlines are stringent. Missing a dead line may cause a catastrophe - loss of life or damage to property.

- Embedded systems are constrained for power, As many embedded systems operate through a battery, the power consumption has to be very low.
- Embedded systems need to be highly reliable. Once in a while, computers can be reset, but you cannot afford to reset your embedded system.
- Some embedded systems have to operate in extreme environmental conditions such as very high temperatures and humidity.
- Embedded systems that address the consumer market (for example electronic toys) are very cost-effective.

Unlike desktop computers in which the hardware platform is dominated by Intel and the operating system is dominated by Microsoft, there is a wide variety of processors and operating systems for the embedded systems. So, choosing the right platform is the most complex task.

[Characteristics & Constraints of an Embedded System]

- Characteristics
  - Real Time & Multi rate operations define the way in which system works, reacts to events, interrupts and schedules
  - Follows a plan to control latencies and to meet deadlines.
  - Different operations take place at different rates
  - Complex Algorithms
  - Complex graphic and other user interfaces
  - Dedicated functions
- Constraints
  - Available System Memory
  - Available processor speed
  - Need to limit power dissipation when the system is running in different cycles

[Classification of embedded systems]

1. Small Scale embedded systems:
   Designed with 8 or 16 bit microcontroller. They have little hardware or software complexities. The system may be battery operated and will utilize low memory (mostly that provided by processor on chip). The software design is generally through assembly language and considers low power dissipation requirements.
2. Medium Scale embedded systems:
   Designed with single or few 16 or 32 bit microcontrollers, DSPs or RISCs. They may also employ single purpose processors for various functions. Medium scale embedded systems have both hardware and software complexities. Tools like High level programming languages - C/C++, RTOS, debugger, simulator etc. are used for the complex design.
3. Sophisticated embedded systems:
   These systems have enormous complexities and may need several ASIPs, scalable or configurable processors and programmable logic arrays. They are used for cutting edge applications that need software and hardware co-design and components that have to be integrated. Certain software functions such as encryption, TCP/IP protocol etc., the software implements some of the functions of the hardware resources. Development tools for these systems may not be readily available at a reasonable cost or may not be available at all.

RTOS

- The software in an embedded system consists of - an operating system - Optional, if not present, need to write software routines to access hardware - Embedded systems are constrained by memory, hence can not use OS such as Windows or Linux - the application software - Services Provided by an operating System - Every computing device needs a piece of software using which the user interacts with the hardware. This software is known as the Operating System (OS). - A computer or embedded system is designed to perform different things simultaneously. Each such job is called a process in case of a computer and in case of embedded system, it is called as a task. - Each task needs memory and access to I/O devices. Managing these multiple tasks is done by the OS.

- An OS has to do the following functions:
  - Process / Task Management
  - Memory Management
  - Input / Output Management including file services
  - Providing services to applications
  - Providing user interface so that user need not be concerned about underlying hardware details.
- An OS is said to be
  - Of type Hard, when the tasks to be completed are to be completed absolutely on time without missing any deadline e.g., pacemaker, defense systems etc.
  - Of type Soft, when the tasks to be completed are to be completed can miss few deadlines without compromising overall application performance. E.g., music system, automated toys etc.

[Process]

- An application program consists of number of processes
- Each process runs under the control of OS
  - Each process consists of sequentially executable program codes and state controlled by OS.
  - The state of a process is represented by information of process state(created, running, blocked, finished), process structure(data, objects, resources) and Process Control Block(PCB)
  - A process runs on scheduling by OS, which gives control of CPU to the process.
- Process Control Block:
  - Is a data structure having the information using which the OS controls the process state. The PCB consists of
  - Process Id, Process priority, parent process, child process(if any) and address to the next process PCB.
  - It also holds allocated memory information of ROM, stack and CPU registers and other context related information.

[MULTIPLE THREADS]

- A thread consists of sequentially executable code under state control by OS.
- The state information of a thread is represented by
  - a thread state (started, running, blocked, finished)
  - thread structure (data, object and resources)
- A thread is a process or a sub process within a process that has
  - its own PC, SP and stack
  - Priority parameter for scheduling

    - Variables that load into processor register on context switching
    - Signal mask when unmasked allows the tread to activate and run
    - When masked, the thread is put into a queue of pending threads
- A thread stack is at a memory block allocated by OS.

[Multithreaded Process]
A multiprocessing OS runs more than one process.
A process consisting of multiple threads is called multithreaded process.
A thread defines a minimum unit of a multithreaded process that an OS schedules onto the CPU and allocates other system resources.
Difference between task and thread:
A thread can be a sub process within a process or a process within an application program
A task is a process and the OS does the multitasking.
Task is kernel controlled entity, where as thread is process controlled entity.
A task and a thread are analogous in the aspect that
A task or a thread do not call another task or thread
Both need schedulers – Task scheduler / thread scheduler.

[Tasks]
A task is similar to a process. Some OSes use the term task and some use process.
A task consists of a sequentially executable program(codes) under a state control by an OS.
The tasks include the OS tasks as well as application specific tasks.
The task object contains all information related to the task:
The state information of a task is represented by the task state, task structure and task control block.
Task Scheduling:
Since only one CPU can handle multiple tasks, the tasks have to share the CPU time in disciplined manner to ensure one task does not get too much time while other task is waiting for ever.
Each task has to be assigned a priority and a mechanism for deciding which task will get CPU time next. This is known as task scheduling.
In addition to CPU time, the tasks have to share system resources such as CPU registers, external memory and input / output devices.
Also, one task should not corrupt data of another task.

[Context Switching]
Each task has a context (CPU registers, parameters and pointer to the stack top of called function).
The context is continuously updated during running of the task and is saved before switching to another task.
Hence the context reflects the state just before the OS blocks one task and initiates another task into running state.
Context Switching
Only after saving the registers and pointers, the CPU control switches to other process or task. The context will retrieve the PC and register information once the CPU is back to the task again, on the OS unblocking its state and allowing it to enter running state again.
Context switching happens each time the schedule blocks one task and runs another task.
Each task also has initial context - context-init . This has initial parameters of the task: 1) Pointer to a start up function – a task starts from this address 2) Pointer

to task data structure
Task Control Block
The TCB is a data structure having the information using which the OS controls the task state.
TCB is a memory block stored in protected memory area of the kernel.