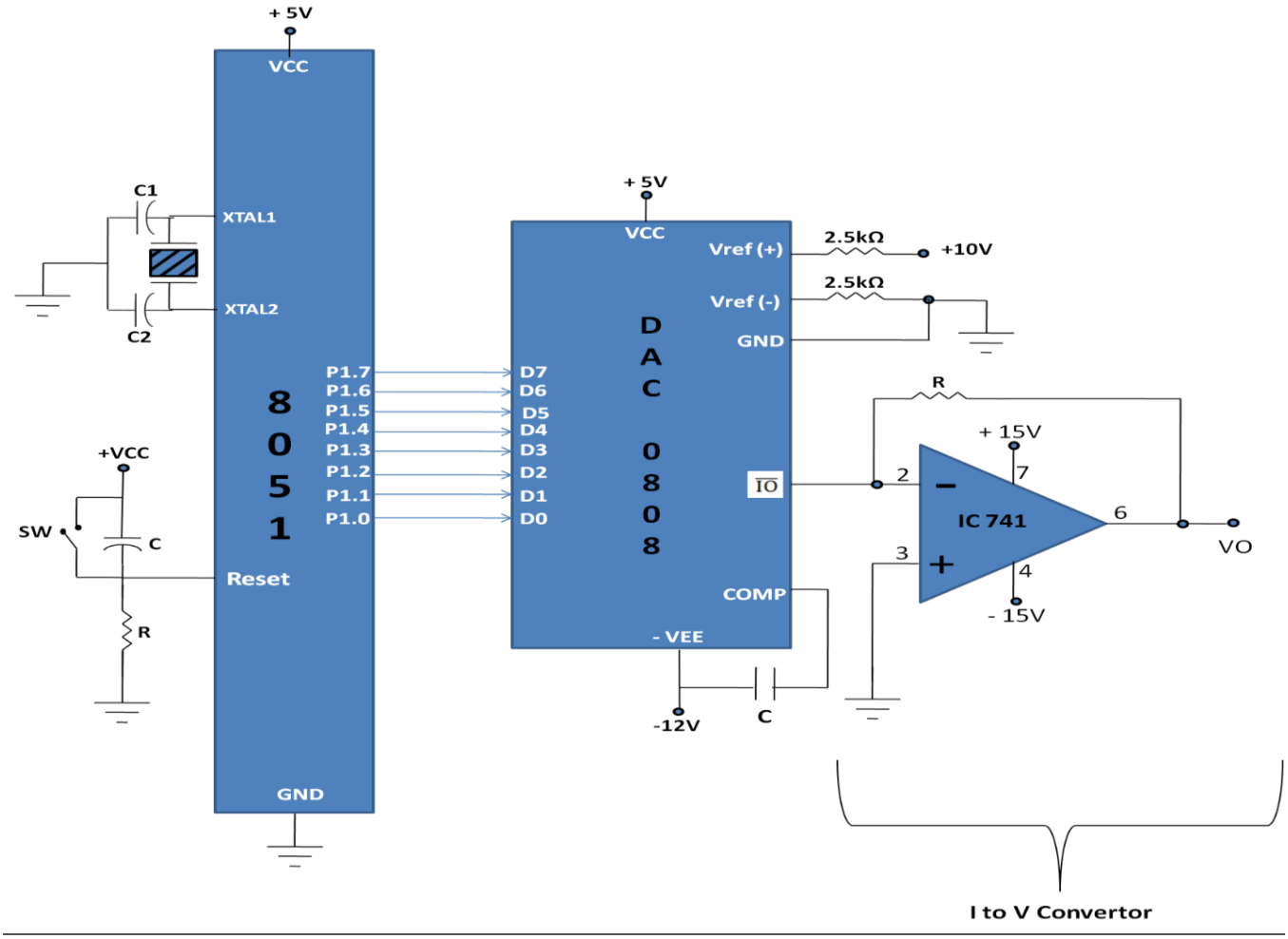
Notes by: Dr Dinesh V Kala

Head, Department of Physics

Guru Nanak Khalsa College Autonomous

Embedded Systems & RTOS

PG SEM III; Paper-IV; U-III & IV



Peripherals @ Embedded Systems

This is an era of artificial intelligence. The intelligence of any device depends & decided by the quality of coding & sometimes on the complexity of the coding in its software. However, the whole intention is to replace human intelligence, however, whatever best possible, there cannot be replacement of extraordinary humans' intelligence & a device cannot have faster baud rate than humans' speed of thoughts, the speed at which the thoughts originates and propagates in human mind has no comparability & compatibility. Humans can never be substituted, however, they can be relieved from lot of rigorous physical tasks where robots & smart machines can be helpful.

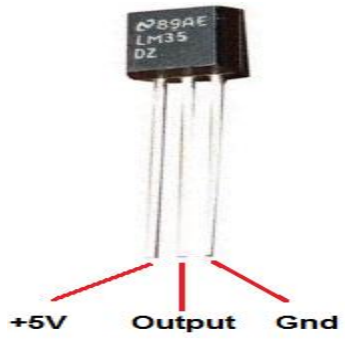
Ultimately, sensing and response to it depends on the extent of intelligence of a device. For example, in a microprocessor-based water heating system, if the temperature is within the permissible range the GREEN LED would glow and if temperature reaches beyond the upper limit than RED LED would glow, now somebody has to switch off the water heater manually, after seeing the glow of RED LED.

However, with some more coding, it is possible that even the heater on / off can also be controlled by the microprocessor, the heater should be automatically becomes off when temperature reaches above certain upper limit and automatically becomes on when temperature reaches below certain lower limit.

The heating system will be further more intelligent, if even the lower & upper limits of temperature can be SET / RESET remotely either by a remote or through your mobile.

Every intelligent system needs a sensor at its input and a control system or display at its output. After sensing any parameter as per its intelligence, the device will process the data and send as the control signal or display to the output. Now all smart systems are digitally controlled, the inputs are usually analog in nature, so we need an ADC. The data gets processed in the digital form & before display it has to be converted back to analog form, we need a DAC. ADC & DAC are the integral part of any intelligent system. However, the input to the ADC has to be in the form of analog voltage, which is possible through a transducer / sensor. Before learning about embedded system, lets sense few peripherals, namely Transducer, Sensors, ADCs & DACs etc.

Temperature Sensor LM 35:



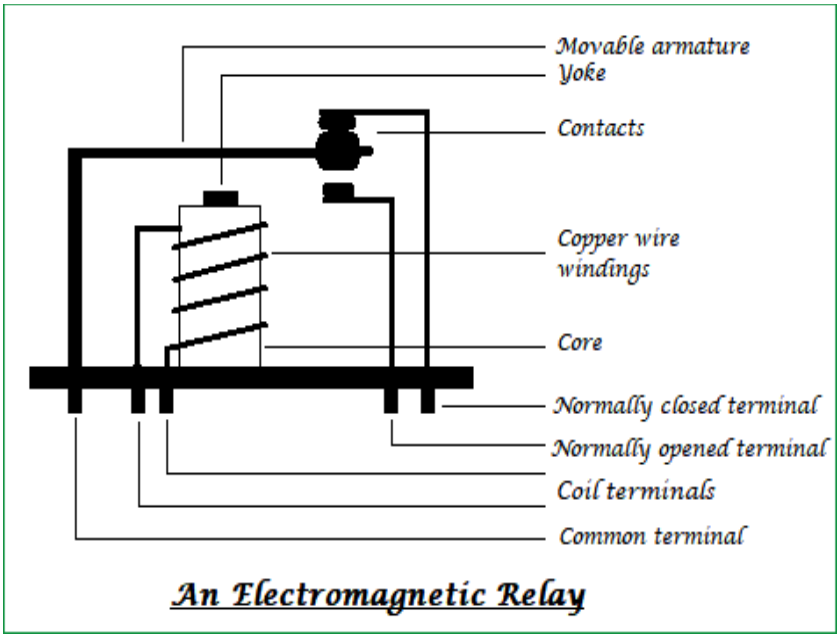
The LM35 is an IC sensor that can be used to measure temperature with an electrical output proportional to the temperature in °C, it converts heat energy into electrical energy in appropriately calibrated form. The sensor circuitry is sealed and not subject to oxidation. The LM35 has an output

voltage that is proportional to the celsius temperature. The scale factor is 0.01V/°C or 10mV /°C .

The LM35 does not require any external trimming for any calibration or adjustment. Another important characteristic of the LM35 is that it draws only 60 micro amps from its supply and exhibit a low self-heating capability. With the help of suitable signal conditioning & appropriate calibration, one can display the exact value of the temperature sensed by IC LM 35.

Relay: The main operation of this device is to make or break the contact with the help of a signal without any human involvement in order to switch it ON or OFF. It is mainly used to control a High-Powered circuit using a Low Power signal exactly the way an automotive driver at his seat with small steering can control the movement of even a very big bus or a truck. Generally a DC signal from a μP/μC is used to control circuit, which is driven by high voltage, like controlling AC home appliances. An electromechanical relay is basically designed using few mechanical parts like Electromagnet, a Movable Armature, Contacts, Yoke & a Spring/Frame/Stand etc. An electromagnet plays a major role in the working of a relay. It is a metal which doesn't have magnetic property but it can be converted into a magnet with the help of an electrical signal. We know that when current passes through the conductor it acquires the properties of a magnet. So, when a metal winded with a copper wire and driven by the sufficient power supply, that metal can act as a magnet & can attract the metals within its range. Movable armature is a simple metal piece which is balanced on a pivot or a stand. It helps in making or

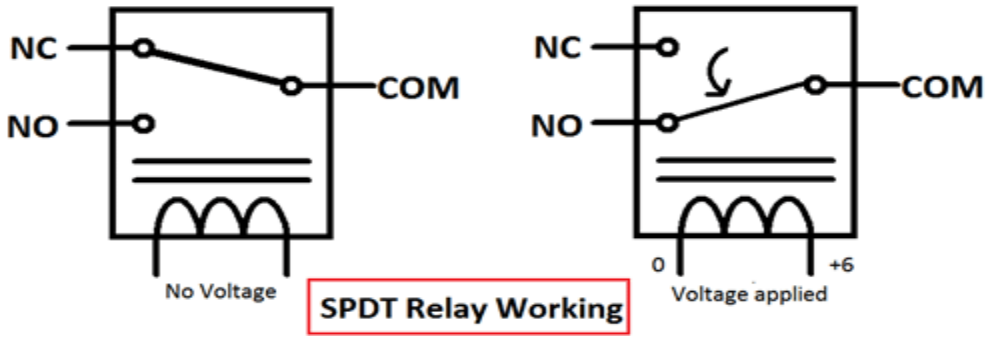
breaking the connection with the contacts connected to it. The following figure shows how a relay looks internally and how it can be constructed, on a casing, a core with copper windings (forms a coil) winded on it is placed. A movable armature consists of a spring support or stand like structure connected to one end and a metal contact connected to another side, all these arrangements are placed over the core such that, when the coil is energized, it attracts the armature. The movable armature is generally considered as a common terminal which is to be connected to the external circuitry.



An Electromagnetic Relay

The relay also has two pins namely *normally closed and normally opened (NC & NO),* the normally closed pin is connected to the armature or the common terminal whereas the normally opened pin is left free (when the coil is not energized). When the coil is energized the armature moves and is get connected to the normally opened contact till there exists flow of current through the coil. When it is de-energized it goes to its initial

position. The general circuit representation of the relay is as shown in the figure below

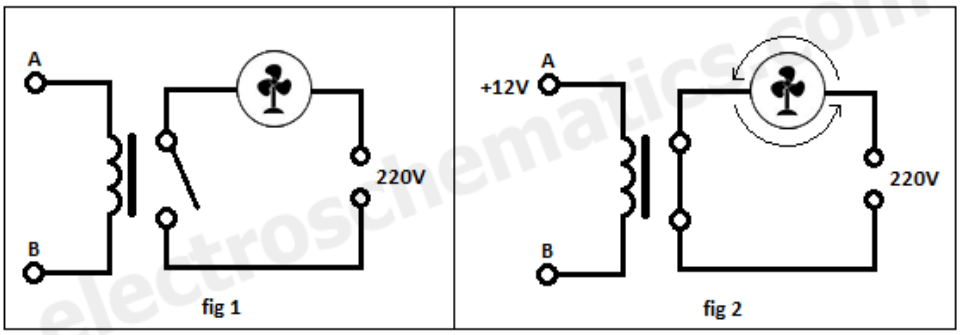


Relay in NC Condition: When no voltage is applied to the core, it cannot generate any magnetic field and it doesn't act as a magnet. Therefore, it cannot attract the movable armature. Thus, the initial position itself is the armature connected in normally closed position (NC).

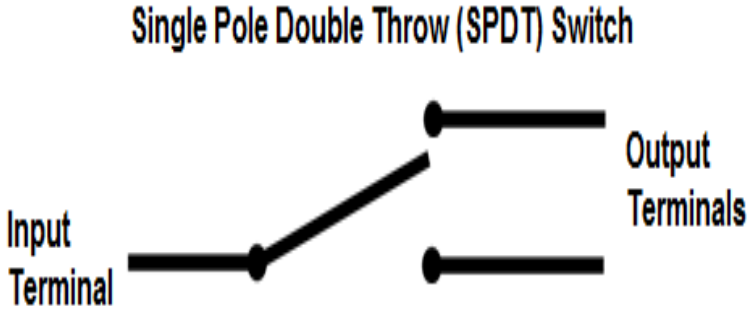
Relay in NO Condition: When sufficient voltage is applied to the core it starts to create a magnetic field around it and acts as a magnet. Since the movable armature is placed within its range, it gets attracted to that magnetic field created by the core, thus the position of the armature is being altered. It is now connected to the normally opened pin of the relay and external circuit connected to it function in a different manner. So finally, we can say that when a coil is energized the armature is attracted and the switching action can be seen, if the coil is de-energized it loses its magnetic property and the armature goes back to its initial position.

Single Pole, Single Throw (SPST): It consists of only one pole and one throw. Generally, the path is either closed or opened (remains untouched to any terminal). A push button is the best example of this type. When we push the button, the contact is in the closed position & when

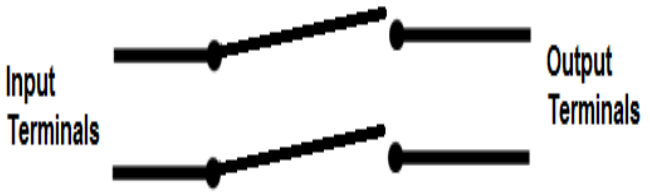
released the contact is in the open position, which can be understood from the below image.



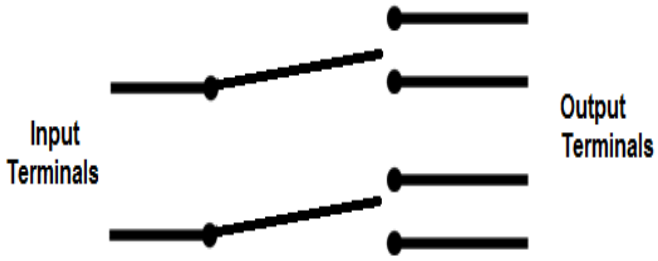
Single Sole, Double throw (SPDT): This type of switches consists of only one pole but has two throws. So, the contact is always made to either of the terminals. A slide switch can be considered as its example. The slider is always connected to either of the contacts i.e., a closed path always exists all the time if both the terminals are connected to a circuit.



Double Pole, Single Throw (DPST): It has two poles and a throw. The contacts of it are either opened or closed which is done simultaneously. Toggle switch works on this property. When the switch is toggled from one position to another, both the contacts are moved simultaneously.



Double Pole, Double Throw (DPDT): **This type of switches has two poles but the individual pole has two throws. So, it is named as double throw & the switching action is done similarly and simultaneously for both the poles. A switch on a standard trimmer is of DPDT because while we are charging the trimmer & when the switch on the trimmer is in the ON state, it automatically stops charging means the switches are internally opened in the charging circuit.**



Applications of the relay are limitless; its main function is to control the high voltage circuit (230V circuit AC) with the low voltage power supply (a DC voltage).

1. **Relays are not only used in the large electrical circuits but also used in the computer circuits in order to perform the Arithmetic & Mathematical Operations.**
2. **Used to control the Electric Motor switches. To turn ON an electric motor we need 230V AC supply but in few cases/applications, there**

may be a situation to switch ON the motor with a DC supply voltage. In those cases, a relay can be used.

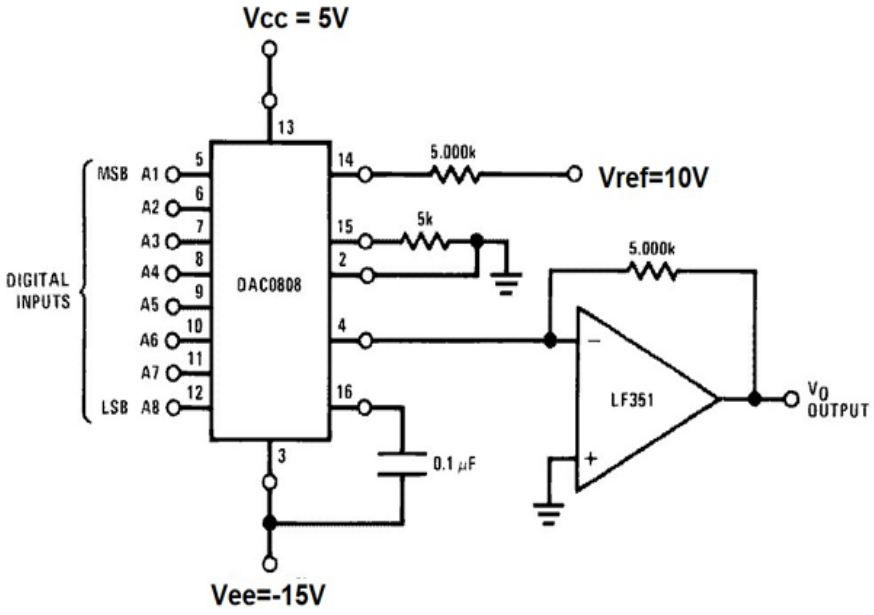
3. **Automatic Stabilizers are one of its applications where a relay is used. When the supply voltage is other than the rated voltage, set of relays sense the voltage variations & controls the load circuit with the help of circuit breakers.**
4. **Used for the circuit selection if exists more than one circuit in a system.**
5. **Used in Televisions. An old picture tube television's internal circuitry works with the DC voltage but the picture tube needs a very high AC voltage, in order to turn on the picture tube with a DC supply we can use a relay.**

DAC0808 IC: The chip used to be popular a decade ago & its applications used to be many but at present there are many modern IC which can perform DAC with higher Resolution, Speed & Efficiency. But used for reference and beginner applications. For advance applications microcontroller with DAC feature or DAC IC with Serial Interface is preferred at present.

How to use DAC0808 IC in Circuits:

- **For DAC0808 we need two voltage sources +5V & -15V as shown in diagram. Eight digital inputs are given to the chip and are supposed to be in order from MSB to LSB.**
- **A +10V power source is connected as reference voltage for the device and the negative reference is grounded.**

Working: **The device takes in parallel 8-bit data from μC or μP & converts that data in to analog signal at the output. The analog output from DAC is a current & this needed to be converted in to voltage for further applications. Op-Amp Circuit is to convert Current into Voltage. The IC output is Binary Weighted Current of the I_{REF} & the weight is decided by the Digital Data available at the input. Analog Voltage from Op-Amp is in linear relation with input Digital Value & hence DAC conversion with DAC0808 is achieved.**



Electrical Characteristics of DAC0808:

- **8-bit parallel digital data input**
- **Fast settling time (typical value): 150 ns**
- **Relative accuracy at ±0.19% (maximum error)**
- **Full scale current match: ±1 LSB**
- **Digital inputs are TTL & CMOS compatible**

- **High speed multiplying input slew rate: 8 mA/µs**
- **Power supply voltage range: ±4.5V to ±18V**
- **Low power consumption: 33 mW@ ±5V**
- **Maximum Power Dissipation: 1000 mW**
- **Operating Temperature Range: 0°C to +75°C**

Features of ADC0804:

- **Easy to Interface with all µP & µC / Works Standalone as well.**
- **Single channel 8-bit ADC module. On chip Clock available**
- **Digital Output various from 0 to 255; step Size is 19.53V**
- **V_{ref} = 5V, for every 19.53mV of analog value there will be rise of one bit on Digital Side**
- **Available in 20-pin PDIP, SOIC package**

Where to use an ADC0804: The ADC0804 is a commonly used ADC module, for projects were an external ADC is required. It is a 20-pin Single channel 8-bit ADC module. Meaning it can measure one ADC value from 0V to 5V and the precision when voltage reference (V_{ref}) is +5V is 19.53mV.That is for every increase of 19.53mV on input side there will be an increase of 1 bit at the output side. This IC comes with its own internal clock.

How to use ADC0804: Since the IC comes with an internal clock, we do not need many components to make it work. However, to make the internal clock to work we have to use a RC circuit. The IC should be powered by +5V & the both ground pins should be tied to circuit ground.

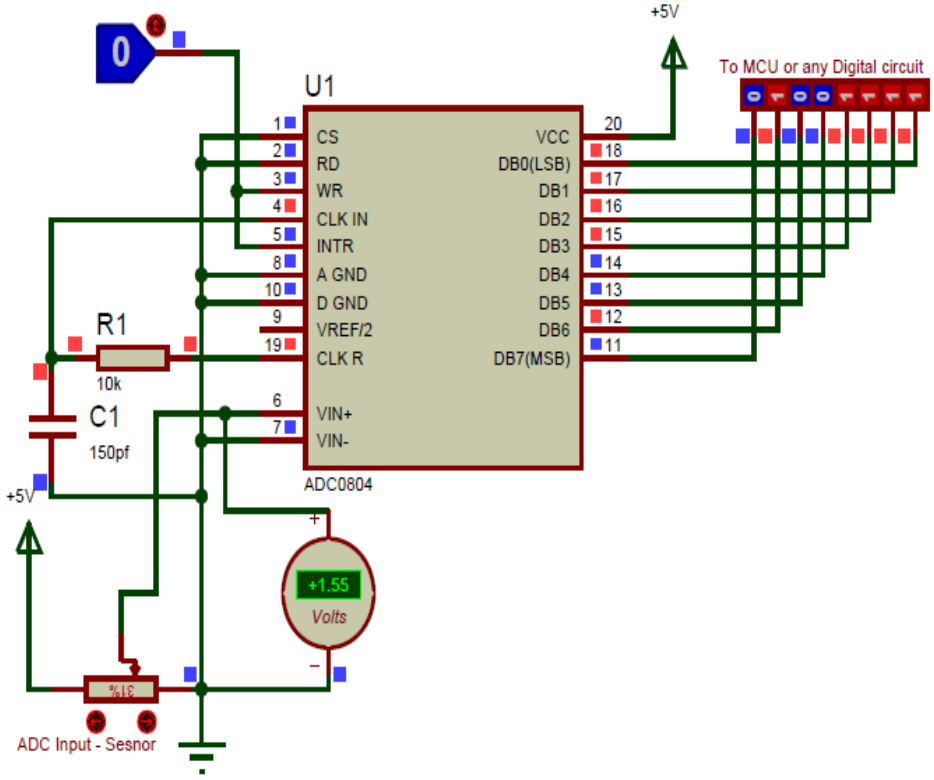
To design the RC circuit simply use a resistor of value 10k and capacitor of 100pf and connect them to CLK R and CLK IN pins as shown in the circuit below. The chip selects (CS) and Read/OE (R) pin should also be grounded. The V_{ref} pin is left free because by default without any connection it will work for max input of +5V and itself set to 2.5 V.

The digital output will be obtained from the pins DB0 to DB7 and the analog voltage should be connected to V_{in} (+) pin as shown in the circuit. Also note that another end of the voltage source (sensor/module) should also be grounded to the circuit for the ADC conversion to work. For the ADC Conversion to start we have made the Write /SOC (WR) pin to go high momentary this can be done connecting the pin to I/O of MPU & toggling it high before every ADC read. Only if this is done the ADC value on the output side will be update.

In the circuit below, a potentiometer is used to feed in a variable voltage of 0V to 5V to the V_{in} pin and the present Voltage is read using a voltmeter. As you can see in the image the voltage value is 1.55V and the resulting binary value is 01001111. Let us see how this binary value can be converted to Analog value, since we will need it while programming/designing. Binary Value = 01001111, Converting to Decimal = (0*128) +(1*64) +(0*32) +(0*16) +(1*8) +(1*4) +(1*2) +(1*1) = 79

Analog Voltage = Decimal Value * Step size = 79 * 19.53mV = 1.54V

The obtained value is 1.54V and the measured voltage is 1.55V which are very much close. So, this is how you use an ADC0804 IC.



Applications:

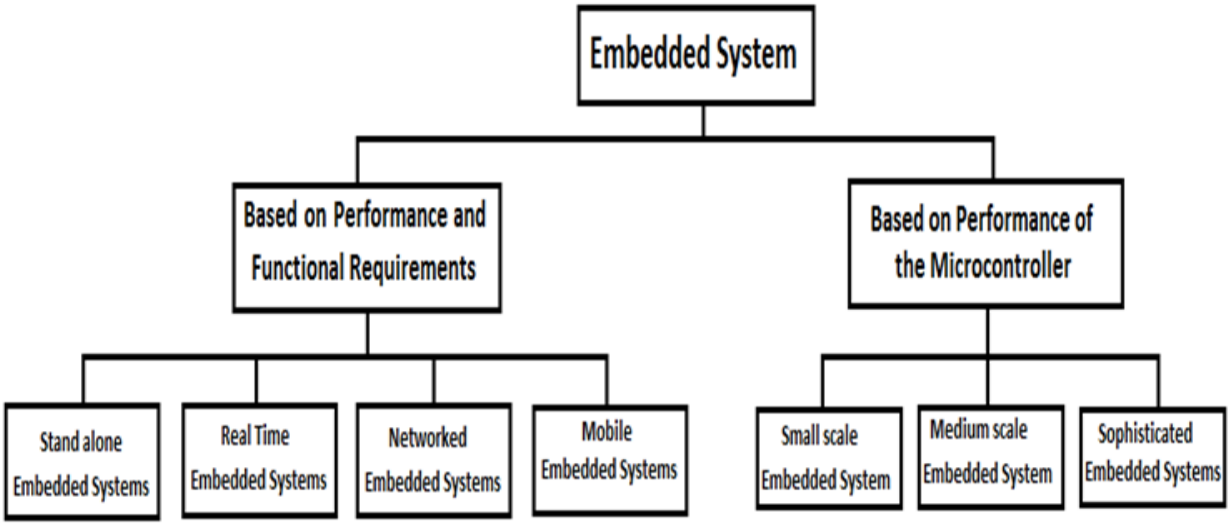
- **Operates with Any 8-Bit µP Processors or as a Stand-Alone Device**
- **Widely used with Raspberry Pi, Beagle Bone & other MPU**
- **Interface to Temp Sensors, Voltage Sources & Transducers**

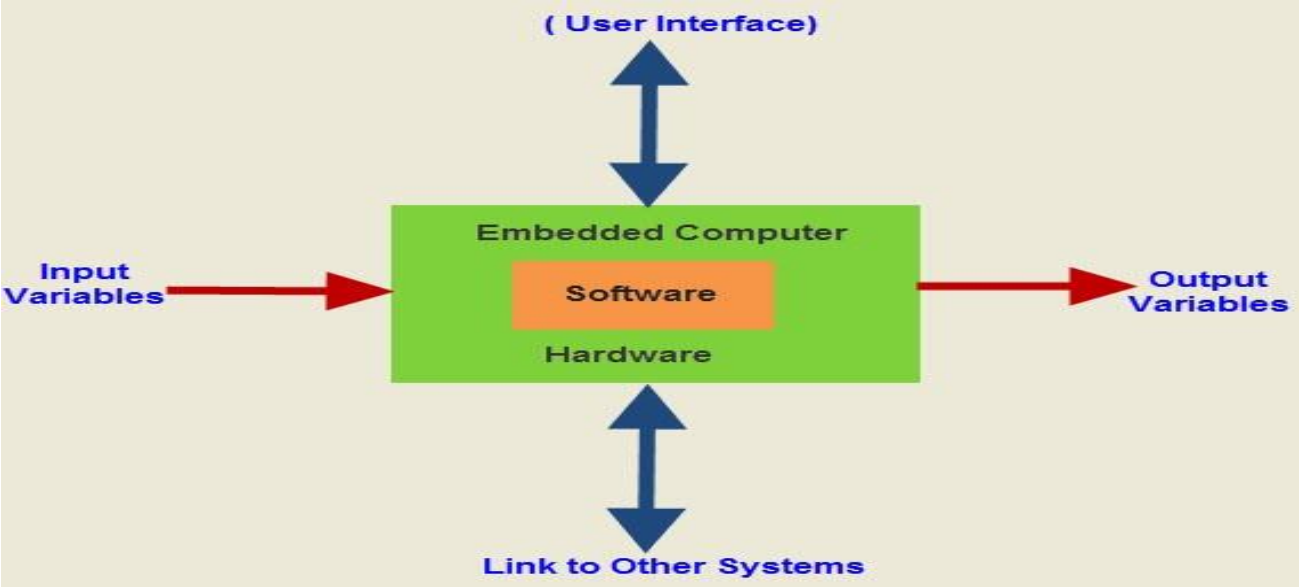
Introduction to Embedded Systems

ESs can be discussed under the below sub-topics

1. **ES v/s GCS (General Computing System)**
2. **Major Applications areas of ES**
3. **Classification of Embedded Systems**
4. **Purpose of Embedded Systems**
5. **Application Specific User Interface: Smart Running Shoes.**
6. **Elements of ES**
7. **Core of ES**
8. **Characteristics of an ES**
9. **Operational Quality Attributes Quality Attributes of an ES**
10. **Non-Operational Quality Attributes of an ES**
11. **Washing Machine: Application Specific Embedded System**
12. **Automotive: Domain Specific ES**
13. **ACVM (Automotive Chocolate Vending Machine)**
14. **Digital Camera**
15. **Mobile Phones**
16. **A Set of Robots**

1. **An Embedded System is an Electronic System that has a Software and is embedded in Computer Hardware. It is programmable or non-programmable depending on the application. An ES is defined as a way of Working, Organizing, Performing single or multiple tasks according to a set of rules. In an ES all the units assemble & work together according to the program. Examples of embedded systems include numerous products such as Microwave Ovens, Washing Machine, Printers, Automobiles, Cameras, etc. These systems use μPs, μCs as well as processors like DSPs. The important characteristics of an ES are Speed, Size, Power, Reliability, Accuracy & Adaptability.**





Embedded System

2. **ESs are classified into four categories based on their Performance & Functional requirements:**

a. **Standalone ESs**
b. **Real time ESs**
c. **Networked ESs**
d. **Mobile ESs**

✓ **Stand Alone Embedded Systems: Standalone ESs do not require a host system like a computer, it works by itself. It takes the input from the input ports either analog or digital and processes, calculates and converts the data and gives the resulting data through the connected device-which either controls, drives or displays the connected devices. Examples for the stand alone ESs are MP3**

Players, Digital Cameras, Video Game Consoles, Microwave Ovens & Temperature Measurement systems.

- ✓ **Real Time Embedded Systems: A real time ES is defined as, a system which gives a required o/p in a particular time. These types of ESs follow the time deadlines for completion of a task. Real time embedded systems are classified into two types such as Soft & Hard real time systems.**
- ✓ **Networked Embedded Systems: These types of ESs are related to a network to access the resources. The connected network can be LAN, WAN or the Internet. The connection can be any wired or wireless. This type of ES is the fastest growing area in ES applications. The embedded web server is a type of system wherein all embedded devices are connected to a web server and accessed & controlled by a web browser. Example for the LAN networked embedded system is a home security system wherein all Sensors are connected and run on the protocol TCP/IP**
- ✓ **Mobile Embedded Systems: Mobile embedded systems are used in portable embedded devices like Cell Phones, Mobiles, Digital Cameras, MP3 Players & PDA etc. The basic limitation of these devices is the other resources and limitation of memory.**

3. **Embedded Systems are classified into three types based on the Performance of the μC such as**

a. **Small Scale ESs**
b. **Medium Scale ESs**
c. **Sophisticated ESs**

- ✓ **Small Scale Embedded Systems: These types of ESs are designed with a single 8 or 16-bit μC that may even be activated by a battery. For developing embedded software for small scale embedded systems, the main programming tools are an Editor, Assembler, Cross Assembler & Integrated Development Environment (IDE).**
- ✓ **Medium Scale Embedded Systems: These types of embedded systems design with a single or 16 or 32 bit μC, RISCs or DSPs. These types of ESs have both hardware and software complexities. For developing embedded software for medium scale embedded systems, the main programming tools are C, C++, JAVA, Visual C++, RTOS, Debugger, Source Code Engineering Tool, Simulator & IDE.**
- ✓ **Sophisticated Embedded Systems: These types of ESs have enormous hardware & software complexities that may need ASIPs, IPs, PLAs, Scalable or Configurable processors. They are used for cutting-edge applications that need Hardware & Software Co-design and components which have to assemble in the final system.**

4. **Applications of Embedded Systems: ESs are used in different applications like Automobiles, Telecommunications, Smart Cards, Missiles, Satellites, and Computer Networking & Digital Consumer Electronics. An Embedded System is a Controller with a dedicated function within a larger Mechanical or Electrical system, often with real-time computing constraints. E*mbedded system* is part of a**

complete device often including Hardware & Mechanical parts. Embedded systems control many devices in common use today. Modern ESs are often based on μC but ordinary Microprocessors (using external chips for memory & peripheral interface circuits are also common, especially in more complex systems. In either case, the processor(s) used may be types ranging from general purpose to those specialized in certain class of computations or even custom designed for the application at hand. A common standard class of dedicated processors is the Digital Signal Processor (DSP).

Since the ES is dedicated to specific tasks, design engineers can Optimize it to Reduce the Size & Cost of the product and increase the Reliability & Performance. ES are commonly found in Consumer, Industrial, Automotive, Medical, Commercial & Military applications. Embedded systems range from portable devices such as:

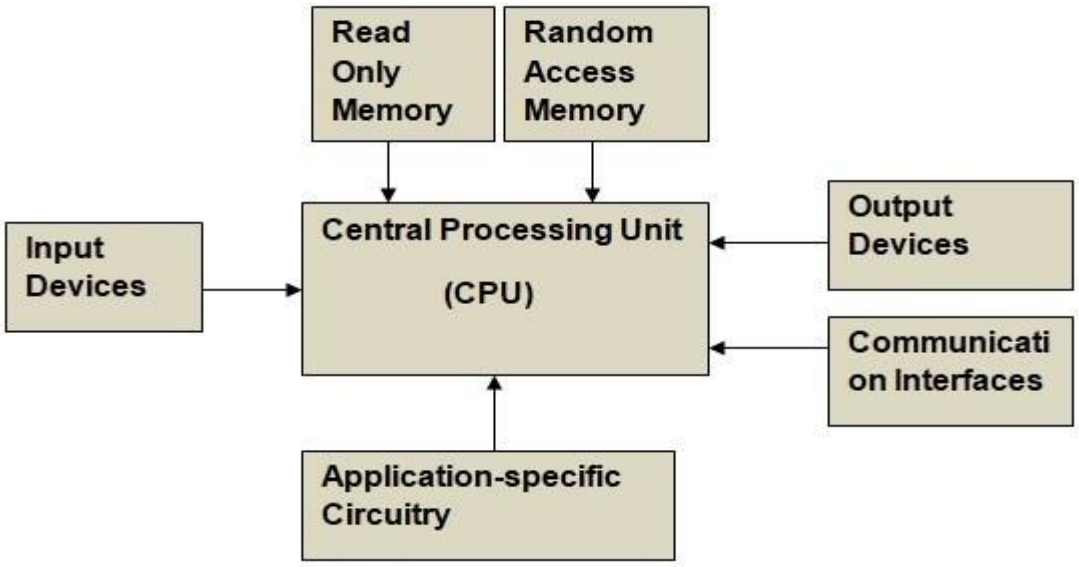
a) **Digital watches & digital cameras**
b) **Mp3 players**
c) **Traffic light controllers**
d) **Programmable logic controllers**
e) **Hybrid vehicles, anti-lock braking system**
f) **Telephone switches, cell phones & GPS**
g) **Routers & network bridges**
h) **Video game consoles**
i) **Printers**

j) **Microwave ovens, washing machines , dishwashers**

k) **Electronic stethoscopes , medical imaging.**

l) **Home automation can be used for security, surveillance, etc.**

5. **Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs. User interface embedded systems range from no user interface at all, in systems dedicated only to one task, to complex graphical user interfaces that resemble modern computer desktop operating systems. Simple embedded devices use buttons, LEDs, graphic or character LCDs (HD44780 LCD for example) with a simple menu system. More sophisticated devices that use a graphical screen with touch sensing or screen-edge buttons provide flexibility while minimizing space used: the meaning of the buttons can change with the screen, and selection involves the natural behavior of pointing at what is desired. Some systems provide user interface remotely with the help of a serial (e.g. RS-232, USB, I²C, etc.) or network (e.g. Ethernet) connection.**



6. **<u>Processors in Embedded Systems:</u> Typical embedded computers when compared with general-purpose counterparts are Low Power Consumption, Small Size, Rugged Operating Ranges & Low per-unit Cost. This comes at the price of limited processing resources, which make them significantly more difficult to program and to interact with. However, by building intelligence mechanisms on top of the hardware, taking advantage of possible existing sensors and the existence of a network of embedded units, one can both optimally manage available resources at the unit and network levels as well as provide augmented functions, well beyond those available. For example, intelligent techniques can be designed to manage power consumption of embedded systems.**
7. **Embedded processors can be broken into two broad categories. Ordinary microprocessors (µP) use separate integrated circuits for memory and peripherals. Microcontrollers (µC) have on-chip**

peripherals, thus reducing power consumption, size and cost. In contrast to the personal computer market, many different basic CPU architectures are used since software is custom-developed for an application and is not a commodity product installed by the end user. RISC as well as non-RISC processors are found. Word lengths vary from 4-bit to 64-bits & beyond, although the most typical remain 8/16-bit. Most architectures come in a large number of different variants and shapes, many of which are also manufactured by several different companies.

Peripherals:Embedded systems talk with the outside world via PERIPHERALS such as:

- **Serial Communication Interfaces (SCI): RS-232, RS-422, RS-485, etc.**
- **Synchronous Serial Communication Interface: I2C, SPI, SSC**
- **ESSI (Enhanced Synchronous Serial Interface)**
- **Universal Serial Bus (USB)**
- **Multi Media Cards (SD cards, Compact Flash, etc.)**
- **Networks: Ethernet, LonWorks, etc.**
- **Fieldbuses: CAN-Bus, LIN-Bus, PROFIBUS, etc.**
- **Timers: PLL(s), Capture/Compare and Time Processing Units**
- **Analog to Digital/Digital to Analog (ADC/DAC)**

8. **Embedded Software Architectures: There are several different types of software architecture in common use.**

Simple Control Loop: In this design, the software simply has a loop. The loop calls subroutines, each of which manages a part of the hardware or software. Hence it is called a simple control loop or control loop.

Interrupt-Controlled System: Some embedded systems are predominantly controlled by interrupts. This means that tasks performed by the system are triggered by different kinds of events; an interrupt could be generated, for example, by a timer in a predefined frequency, or by a serial port controller receiving a byte.

These kinds of systems are used if event handlers need low latency, and the event handlers are short and simple. Usually, these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays.

Objectives:

1) **Learn the characteristics describing an embedded system & Learn the non-functional requirements that needs to be addressed in the design of an embedded system**
2) **Learn the important quality attributes of the embedded system that needs to be addressed for the operational mode /online mode of the system. This includes Response, Throughput, Reliability, Maintainability, Security, Safety, etc.**
3) **Learn the important quality attributes of the embedded system that needs to be addressed.**

The characteristics of embedded system are different from those of a general-purpose computer and so are its Quality metrics. Unlike General Purpose Computing Systems, embedded systems possess certain specific characteristics & these characteristics are unique to each embedded system.

9. Important Characteristics of an Embedded System:

a. **Application & Domain Specific: An ES is designed for a Specific Purpose only. It will not do any other task. Ex. Air conditioner's embedded control unit, it cannot replace microwave oven. Ex. A washing machine can only wash, it cannot cook. Because the embedded control units of microwave oven & air conditioner are specifically designed to Perform Certain Specific tasks. Certain ESs are specific to a domain: ex. A hearing aid is an application that belongs to the domain of signal processing & telecom with another control unit designed to serve another domain like consumer electronics.**

b. **Reactive & Real Time: Certain ESs are designed to react to the events that occur in the nearby environment. These events also occur real-time. Ex. Flight control systems, Antilock Brake Systems (ABS), etc. are examples of Real Time systems. Ex. An air conditioner adjusts its mechanical parts as soon as it gets a signal from its sensors to increase or decrease the temperature when the user operates it using a remote control. An embedded**

system uses Sensors to take inputs & has actuators to bring out the required functionality.

c. **Operates in Harsh Environments: Certain embedded systems are designed to operate in harsh environments like a dusty one or a high temperature zone or an area subject to vibrations & shock or very high temperature of the deserts or very low temperature of the mountains or extreme rains. These embedded systems have to be capable of sustaining the environmental conditions it is designed to operate.**

d. **Distributed: The term distributed means that embedded systems may be a part of a larger system. These components are independent of each other but have to work together for the larger system to function properly. Ex. An automatic vending machine is a typical example for this. The vending machine contains a card reader (for pre-paid vending systems), a vending unit, etc. Each of them are independent embedded units but they work together to perform the overall vending function. Ex. Automatic Teller Machine (ATM) contains a card reader embedded unit, responsible for reading and validating the user's ATM card, transaction unit for performing transactions, a currency counter for dispatching/vending currency to the authorized person and a printer unit for printing the transaction details. This can visualize these as independent embedded systems. But they work together to achieve a common goal.**

e. **Small Size & Weight: An embedded system that is compact in size and has light weight will be desirable or more popular than one that is bulky & heavy. Ex. Currently available cell phones. The cell phones that have the maximum features are popular but also their size & weight is an important characteristic.**

f. **Power Concerns: It is desirable that the power utilization & heat dissipation of any embedded system be low. If more heat is dissipated then additional units like heat sinks or cooling fans need to be added to the circuit. Ex. The production of high amount of heat demands cooling requirements like cooling fans which in turn occupies additional space and make the system bulky. Nowadays ultralow power components are available in the market. Select the design according to the low power components like low dropout regulators & controllers/processors with power saving modes. Also power management is a critical constraint in battery operated application. The more the power consumption the less is the battery life.**

g. **Quality Attributes of Embedded Systems: Quality attributes are the non-functional requirements that need to be documented properly in any system design. If the quality attributes are more concrete & measurable, it will give a positive impact on the system development process and the end product. The various quality attributes that needs to be**

addressed in any embedded system development are broadly classified into two namely:

a) Operational Quality Attributes b) Non-Operational Quality Attributes

Operational Quality Attributes: The operational quality attributes represent the relevant quality attributes related to the embedded system when it is in the operational mode or 'online' mode. The important quality attributes coming under this category are listed below:

1. Response: Response is a measure of quickness of the system. It gives you an idea about how fast your system is tracking the input variables. Most of the embedded system demand fast response which should be real-time. Ex. An embedded system deployed in flight control application should respond in a Real Time manner. Any response delay in the system will create potential damages to the safety of the flight as well as the passengers. It is not necessary that all embedded systems should be Real Time in response. For example, the response time requirement for an electronic toy is not at all time-critical.

2. Throughput: Throughput deals with the efficiency of system. It can be defined as rate of production or process of a defined process over a stated period of time. The rates can be expressed in terms of units of products, batches produced or any other meaningful measurements. In case of card reader like the ones used in buses, throughput means how much transactions the Reader can perform in a minute or hour or day. Throughput is generally measured in terms of 'Benchmark'. A 'Benchmark' is a reference point by which something can be measured.

Benchmark can be a set of performance criteria that a product is expected to meet or a standard product that can be used for comparing other products of the same product line.

3. Reliability: Reliability is a measure of how much percentage you rely upon the proper functioning of the system or what is the % susceptibility of the system to failure. Mean Time Between Failures (MTBF) & Mean Time To Repair (MTTR) are the terms used in defining system reliability. MTBF gives the frequency of failures in hours/weeks/months. MTTR specifies how long the system is allowed to be out of order following a failure. For an embedded system with critical application need, it should be of the order of minutes.

4. Maintainability: Maintainability deals with support & maintenance to the end user or client in case of technical issues and product failures or on the basis of a routine system checkup. Reliability & Maintainability are considered as two complementary disciplines. A more reliable system means a system with less corrective maintainability requirements and vice versa. Maintainability can be classified into two types: 1. Scheduled or Periodic Maintenance (Preventive Maintenance) An inkjet printer uses ink cartridges, which are consumable components and as per the printer manufacturer the end use should replace the cartridge after each 'n' number of printouts to get quality prints. 2. Maintenance to Unexpected Failures (Corrective Maintenance). If the paper feeding part of the printer fails the printer fails to print and it requires immediate repairs to rectify this problem. Hence it is obvious that maintainability is simply an indication of the availability of the product for use. In any embedded

system design, the ideal value for availability is expressed as $Ai = MTBF/(MTBF + MTTR)$ Where Ai=Availability in the ideal condition, MTBF=Mean Time Between Failures, and MTTR= Mean Time To Repair.

5. Security: 'Confidentially', 'Integrity' & 'Availability' are three major measures of information security. 'Confidentially' deals with the protection of data and application from unauthorized disclosure. 'Integrity' deals with the protection of data and application from unauthorized modification. 'Availability' deals with protection of data and application from unauthorized users. Certain embedded systems have to make sure they conform to the security measures. Ex. An electronic safety Deposit Locker can be used only with a pin number like a password.

6. Safety: Safety deals with the possible damages that can happen to the operators, public and the environment due to the breakdown of an embedded system or due to the emission of radioactive or hazardous materials from the embedded products. The breakdown of an embedded system may occur due to a hardware failure or a firmware failure. Safety analysis is a must in product engineering to evaluate the anticipated damages and determine the best course of action to bring down the consequences of the damages to an acceptable level.

Non-Operational Attributes: The quality attributes that needs to be addressed for the product 'not' on the basic of operational aspects are grouped under this category. The important quality attributes coming under this category are listed below:

1. Testability & Debug-ability: Testability deals with how easily one can test his/her design, application and by which means he/she can test it. For an embedded product, testability is applicable to both the embedded hardware and firmware. Debug-ability is a means of debugging the product as such for figuring out the probable sources that create unexpected behavior in the total system. Debug-ability has two aspects in the embedded system development context, namely, hardware level debugging and firmware level debugging. Hardware debugging is used for figuring out the issues created by hardware problems whereas firmware debugging is employed to figure out the probable errors that appear as a result of flaws in the firmware.

2. Evolve-ability: Evolve-ability is a term which is closely related to Biology. Evolve-ability is referred as the non-heritable variation. For an embedded system, the quality attribute 'Evolve-ability' refers to the ease with which the embedded product (including firmware and hardware) can be modified to take advantage of new firmware or hardware technologies.

3. Portability: Portability is a measure of 'system independence'. An embedded product can be called portable if it is capable of functioning in various environments, target processors/controllers and embedded operating systems. A standard embedded product should always be flexible and portable.

4. Time to Prototype & Market: Time-to-Prototype & Market: Time-to-market is the time elapsed between the conceptualization of a product and the time at which the product is ready for selling (for commercial product) or use (for non-commercial products). The commercial embedded product

market is highly competitive and time to market the product is a critical factor in the success of a commercial embedded product. Product prototyping helps a lot in reducing time-to-market.

5. Unit Cost & Total Cost: Cost is a factor which is closely monitored by both end user (those who buy the product) and product manufacturer (those who build the product). Cost is a highly sensitive factor for commercial products. Proper market study and cost benefit analysis should be carried out before taking decision on per unit cost of the embedded product. When the product is introduced in the market, for the initial period the sales and revenue will be low. There won't be much competition when the product sales and revenue increase. During the maturing phase, the growth will be steady and revenue reaches highest point and at retirement time there will be a drop in sales volume.

Examples of Embedded Systems

Real life examples of Embedded Systems which we use in our daily Routine. Embedded Systems perform Specific Tasks. They have μC as the main part which controls all the Operations required through them. Almost every Device that we use today is an example of ES. Embedded Systems examples can be seen at our Homes, Offices and Industries & Automation systems.

Digital Camera (DC)

- **D.C that we use today are Smart, all because of Embedded System.**
- **D. C has its own Sensors, Processor, Memory & Actuators.**
- **D.C has basically three functions to Capture Image which we call DATA, to Store Image data in MEMORY to represent this data PRINT.**
- **Images are Stored & Processed in form of Digital Data in bits, no Film.**
- **Increased Storage Capacity & easy to Transfer Images.**
- **In a D.C, first image is Captured & Converted to Digital form.**
- **D.C are able to capture scene in details & analyze the images & are able to detect humans, motion, faces etc. from the whole image.**
- **This Digital Image is stored in Internal Memory.**
- **D.C attached to personal computer; it transfers the stored data.**
- **For detection of objects, some processing is required in cameras.**
- **Usually, image processing includes low level processing and high-level processing & various algorithms are available that are employed for this purpose.**

- **Components of a Smart Camera:**
  - **Image Sensor that may be a CCD (Charge Coupled Device) or a CMOS (Complementary Metal Oxide Semiconductor)**
  - **Analog to Digital converter (A2D)**
  - **Image Processor**
  - **Memory**
  - **Lens**
  - **Led or Illuminating device**
  - **Communication Interface etc.**

Automotive Embedded Systems

- **Today Cars use ESs replacing old traditional systems.**
- **Electronic Control Units (ECU) are used in Automotive.**
- **These units contain Microcontroller, Switches, Sensors, Drivers, etc.**
- **All the Sensors & Actuators are connected to ECU.**
- **Automobiles embedded based may consists of hundreds of μC.**

- **Each μC performs its own Dedicated Task, some of them control engine, some runs dashboard devices.**
- **The whole system is actually Comprised of Several Small Systems.**
- **Using ESs in Automotive has reduced the Cost Factor.**
- **It has improved the overall Performance & Increased Functionality.**
- **It has also reduced Weight & made Automobile more Safe & Reliable.**
- **Applications of Automotive ESs include:**
  - **Automatic Stability Control**
  - **Traction Control System**
  - **Pre-crash Safety System**
  - **Air Bag**
  - **Car Navigation System**

Home Security System

- **Home Security Systems are used largely today.**
- **These systems have several features just as checking for Fire or Gas Leakages.**
- **Detecting if someone suspicious tries to enter the house.**
- **A μC is used for controlling all the Operations.**
- **Sensors give Data & if something wrong happens than Safety Alarms get Activated.**
- **Sensors Include: Gas Sensors, Smoke Sensors, Temperature Sensors, IR Sensors etc.**

- **Keypad is included in such systems for entering password at the gate.**
- **If correct password is entered then this embedded system opens the gate & if someone tries to enter wrong password than alarm is set on & gates remain closed.**
- **The O/P is received from alarms or some display & can also be sent to distant location.**
- **Even not being at home the activities can be monitored going on in their house.**
- **Security System also used at Shops, Stores & Industries.**
- **Industry & Office security systems can recognize the workers from their face or IDs.**

Automatic Washing Machine

- **Washing Clothes is not a difficult Task now owing to ES.**
- **Add Clothes & Leave it to the Machine, rest Operations are done by the WM.**
- **Machines have a μC for controlling all the Tasks.**
- **Sensors & Actuators: Level Sensors, Valves, Motor & a Display & Keypad at I/P.**
- **The whole process consists of three cycles. Washing, Rinsing & Spinning.**
- **You just have to enter information for Hot or Cold water & Press Start Button.**
- **During Washing & Rinsing cycle, water is added to the drum by pipes.**

- **Closing & Opening of Valves is checked through Level Sensors by PIC µC.**
- **Rotation of Drum starts for pre-set time & later water is drained out through pipes.**
- **During Spinning Cycle, water is not added & drum rotates for a set time.**
- **The timings for each cycle can be changed through the keypad.**

Personal Digital Assistant

- **PDA is just like a PC in hand. It was used before Smartphones.**
- **PDA is used as information manager & has the ability to connect to Internet.**
- **It has a display mostly touchscreen for the user to Interact with the device.**
- **The display for entering Data, Memory Card is used for Storing Data & it is provided with Bluetooth or WiFi for Connectivity.**
- **Some of the PDA use keypads instead of touchscreen to input information.**
- **This device is very handy in Managing & Sorting personal information.**
- **It is very light in Weight & Serves multiple functions.**

Robots

- **Industrial Robots: An industrial Robot is an ES that comes in a variety of forms.**
- **These Robots have increased the Productivity.**
- **Industrial Robots are used for moving Parts, Tools and Materials etc.**
- **Some are used in Assembly Operations while some of them are used in Manufacturing.**
- **They are widely used where Precise Operations are required & places which are difficult to Access for Humans.**

- **Painting Robots have a wide Application Area.**
- **They are replacing humans as they require less time for the whole Operation & Ensure best result.**
- **All of the Activities are controlled through the Program.**
- **Timing for the whole Process & Amount of Paint is preset.**
- **Assembly Robot is another example.**
- **The task of such Robot is to create an Assembly from all the parts.**

- **All parts are Collected & Assembled in Correct Sequence to form Final Product.**

Automated Teller Machine

- **An Automated Teller Machine (ATM) is also an Embedded System.**
- **It is a Computerized Device used in Banking.**
- **A customer can Access & Perform his transactions without going to the Bank.**
- **Consists of a Card Reader for detecting Card & Accessing information of the Person.**
- **Also it has a Keypad so the user can enter his Commands & Password.**
- **A Screen displays information, a Printer prints the Receipts & Cash is received from Cash Dispenser.**
- **A network is present between the Bank Computer & ATM machine through a Host Computer.**
- **All the data is verified with the Bank Computer & all transactions are stored in it.**
- **All these I/P and O/P Operations are carried with the help of μC.**

Calculator

- **Calculator is also an example of ES, it is one of very earlier Embedded System that is used widely.**
- **Function: Take input from the Keypad, Perform the required Operation & show the Results on LCD.**

- **Embedded Scientific Calculator has a High Performance Processor. A number of mathematical complex calculations can be performed by these calculators. One can also Program such Scientific Calculators.**

ACVM

Automatic Chocolate Vending Machine: In a big shopping mall, it is difficult to manage the shop with chocolates. Dealers have to spend more time for less profitable product like chocolates. For that an Automatic Chocolate Vending Machine is necessary. The machine already in malls are working based on weight comparison. In addition to that we are going to introduce a comparison on coin with image processing. Each coin will be compared with already stored images. In the previous method there is a possibility of inserting any metals equal to that of a coin's weight. By including this technique, these malpractices can be eliminated.

An RTOS has to schedule the buying tasks from start to finish. Let µC/OS-II be the RTOS used in ACVM. The machine has a slot for inserting the coin. There is a slot into which a buyer inserts the coins for buying a chocolate. The chocolate costs Rs. 8. A coin can be in one of the three possible denominations: Rs. 1, 2 and 5. Whenever a coin is inserted, a

mechanical system directs each coin to its appropriate port, Port_1, Port_2 and Port_5. The machine should have an LCD display matrix as 'User Interface'. Let the interface be called Port Display. It displays the message strings in three lines. It should have a bowl from where the buyer collects the chocolate through a Port for Delivery. Let this port be called Port Deliver. All ports, Port Deliver, Port Refund & Port Excess Refund communicate to Port Collect. This port is a common mechanical interface to the bowl. An RTOS has to schedule the processes (tasks) for buying from start to finish. Usually it is possible to reprogram the codes & relocation of the codes.

RTOS

Operating Systems are there from the very first Computer Generation & they keep evolving with time. Let us discuss some of the important types of OSs which are most commonly used. The operating system is the most important software & the heart of the computer which not only manages the memory & processes inside a computer but also allows the users to run application software. It's a collective set of programs which abstract the hardware of the system & present the users with a complete virtual machine. Apart from the basic tasks such as Tracking Files or Directories, Controlling Peripheral Devices, Sending Output to the Display Screen, the OS also Serves Higher purposes such as Multiprogramming & Multitasking to make sure programs running concurrently do not interfere with each other. The modern OSs not only facilitate parallel processing but also timesharing which is just a concept of

Multiprogramming. Multitasking is just a rudimentary form of Multiprogramming used in a different context.

Batch Operating System: The users of a BOSs do not interact with the computer directly. Each user prepares his job on an off-line device like Punch Cards & submits it to the computer operator. To speed up processing, jobs with similar needs are batched together & run as a group. The programmers leave their programs with the operator & the operator then sorts the programs with similar requirements into Batches.

The Problems with Batch Systems are as follows:

- **Lack of Interaction between the User & the Job.**
- **CPU is often Idle, the speed of the mechanical I/O devices is slower than the CPU.**
- **Difficult to provide the Desired Priority.**

Time-Sharing Operating Systems: Time-Sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-Sharing or Multitasking is a logical extension of Multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multi-Programmed Batch Systems & Time-Sharing Systems is that in case of Multi-Programmed batch systems, the objective is to maximize processor use, whereas in Time-Sharing Systems, the objective is to minimize response time.

Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, the processor executes each user program in a short burst or quantum of computation. That is, if n users are present, then each user can get a time quantum. When the user submits the command, the response time is in few seconds at most.

The operating system uses CPU Scheduling & Multiprogramming to provide each user with a small portion of a time. Computer Systems that were designed primarily as batch systems have been modified to time-sharing systems.

Advantages of Time-Sharing Operating systems are as follows:

- **Provides the Advantage of Quick Response.**
- **Avoids Duplication of Software.**
- **Reduces CPU Idle time.**

Disadvantages of Time-Sharing Operating systems are as follows:

- **Problem of Reliability.**
- **Question of Security & Integrity of user Programs & Data.**
- **Problem of Data Communication.**

Distributed Operating System: Distributed Systems use multiple central processors to serve multiple real-time applications & multiple users. Data processing jobs are distributed among the processors accordingly. The processors communicate with one another through various

communication lines (such as high-speed buses or telephone lines). These are referred as loosely coupled systems or distributed systems. Processors in a distributed system may vary in size & function. These processors are referred as sites, nodes, computers, and so on.

Advantages of distributed systems are as follows:

- **A user at one site may be able to use the resources available at another.**
- **Speedup the exchange of data with one another via electronic mail.**
- **If one site fails in a distributed system, the remaining sites potentially continue.**
- **Better service to the Customers.**
- **Reduction of the load on the Host Computer.**
- **Reduction of delays in Data Processing.**

Network Operating System: A Network Operating System runs on a server & provides the server the capability to Manage Data, Users & Groups, Security, Applications & other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN) and a private network or to other networks. Examples of network operating systems include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

Advantages of NOS are as follows:

- **Centralized Servers are Highly Stable.**
- **Security is Server Managed.**
- **Upgrades to new Technologies & Hardware can be easily Integrated.**
- **Remote Access to servers is possible from different locations & types of systems.**

Disadvantages of NOS are as follows:

- **High Cost of buying & Running a Server.**
- **Dependency on a Central Location for most Operations.**
- **Regular Maintenance & Updates are required.**

Real Time Operating System: A real-time system is defined as a data processing system in which the time interval required to process & respond to inputs is so small that it controls the environment. The time taken by the system to respond to an input and display of required updated information is termed as the response time. So in this method, the response time is very less as compared to online processing. Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data & real-time systems can be used as a control device in a dedicated application. A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, Scientific Experiments, Medical Imaging Systems, Industrial Control Systems, Weapon Systems, Robots, Air Traffic Control Systems, etc. Processing time requirements (including any OS delay) are

measured in tenths of seconds or shorter increments of time. A real-time system is a time bound system which has well defined fixed time constraints. Processing must be done within the defined constraints or the system will fail. They either are Event Driven or Time Sharing.

Event Driven or Time Sharing: Event driven systems switch between tasks based on their priorities while time sharing systems switch the task based on clock interrupts. Most RTOSs use a Pre-Emptive Scheduling Algorithm.

A key characteristic of an RTOS is the level of its consistency concerning the amount of time it takes to accept & complete an application's task. A 'hard' real-time operating system has less jitter than a 'soft' real-time operating system. The chief design goal is not high throughput, but rather a guarantee of a Soft or Hard performance category. An RTOS that can usually or generally meet a deadline is a soft real-time OS, but if it can meet a deadline deterministically it is a hard real-time OS.

An RTOS has an advanced algorithm for scheduling. Scheduler flexibility enables a wider, computer-system orchestration of process priorities, but a real-time OS is more frequently dedicated to a narrow set of applications. Key factors in a real-time OS are minimal interrupt latency & minimal thread switching latency; a real-time OS is valued more for how quickly or how predictably it can respond than for the amount of work it can perform in a given period of time. There are two types of real-time operating systems:

Hard Real-Time Systems: **Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.**

Soft Real-Time Systems: **Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers, etc.**

Thread, Process & Task: **A process invokes or initiates a program. It is an instance of a program that can be multiple & running the same application. A thread is the smallest unit of execution that lies within the process. A process can have multiple threads running. An execution of thread results in a task. Hence, in a multithreading environment, multithreading takes place. A program in execution is known as 'process'. A program can have any number of processes. Every process has its own address space.**
Threads uses address spaces of the process. The difference between a thread & a process is, when the CPU switches from one process to another the current information needs to be saved in Process Descriptor and load the information of a new process. Switching from one thread to another is simple. A task is simply a set of instructions loaded into the memory. Threads can split themselves into two or more simultaneously running tasks.

The most common designs are:

- **Event-Driven: Switches tasks only when an event of higher priority needs servicing; called preemptive priority, or priority scheduling.**
- **Time-Sharing: Switches tasks on a regular clocked interrupt, and on events; called round robin.**

Time sharing designs switch tasks more often than strictly needed, but give smoother multitasking, giving the illusion that a process or user has sole use of a machine.

Early CPU designs needed many cycles to switch tasks during which the CPU could do nothing else useful. For example, with a 20 MHz 68000 processor (typical of the late 1980s), task switch times are roughly 20 microseconds. (In contrast, a 100 MHz ARM CPU (from 2008) switches in less than 3 microseconds.) Because of this, early OSes tried to minimize wasting CPU time by avoiding unnecessary task switching.

Scheduling: A task has three states:

1. **Running (executing on the CPU);**
2. **Ready (ready to be executed);**
3. **Blocked (waiting for an event, I/O for example).**

Most tasks are blocked or ready most of the time because generally only one task can run at a time per CPU. The number of items in the ready queue can vary greatly, depending on the number of tasks the system needs to perform and the type of scheduler that the system uses. On simpler non-preemptive but still multitasking systems, a task has to give

up its time on the CPU to other tasks, which can cause the ready queue to have a greater number of overall tasks in the ready to be executed state (resource starvation).

Usually the data structure of the ready list in the scheduler is designed to minimize the worst-case length of time spent in the scheduler's critical section, during which preemption is inhibited, and, in some cases, all interrupts are disabled, but the choice of data structure depends also on the maximum number of tasks that can be on the ready list.

If there are never more than a few tasks on the ready list, then a doubly linked list of ready tasks is likely optimal. If the ready list usually contains only a few tasks but occasionally contains more, then the list should be sorted by priority. That way, finding the highest priority task to run does not require iterating through the entire list. Inserting a task then requires walking the ready list until reaching either the end of the list, or a task of lower priority than that of the task being inserted. Care must be taken not to inhibit preemption during this search. Longer critical sections should be divided into small pieces. If an interrupt occurs that makes a high priority task ready during the insertion of a low priority task, that high priority task can be inserted and run immediately before the low priority task is inserted.

The critical response time, sometimes called the fly-back time, is the time it takes to queue a new ready task and restore the state of the highest priority task to running. In a well-designed RTOS, readying a new task will take 3 to 20 instructions per ready-queue entry, and restoration of the highest-priority ready task will take 5 to 30 instructions.

In more advanced systems, real-time tasks share computing resources with many non-real-time tasks, and the ready list can be arbitrarily long. In such systems, a scheduler ready list implemented as a linked list would be inadequate.

In a modern computing system, there are usually several concurrent application processes which compete for (few) resources like, for instance, the CPU. As we have already introduced, the Operating System (OS), amongst other duties, is responsible for the effective & efficient allocation of those resources. Generally speaking, the OS module which handles resource allocation is called scheduler. On the basis of the type of OS to be realized, different scheduling policies may be implemented.

Multiprogramming is the ability for more than one user to use the computer at a time using a single CPU. The idea is to effectively utilize the processor to create multiple ready-to-run processes with each process belongs to different user. If the current process stalls for some reason, because it has to wait for some particular event, the operating system allocates the CPU to another process in the queue. The whole operation is facilitated by multiprogramming operating systems to maximize CPU utilization so that to reduce the idle time of the CPU. The idea is to keep the CPU busy for as long as possible. Multitasking means concurrent execution of multiple processes by one user on the same computer utilizing multiple CPUs. For example, in a multitasking operating system, you may work on a word document with one program while listening to music as the same time with another program. Multitasking is effective when programs on a compute require a high degree of parallelism. It is

based on the concept of time sharing because multiple processes or tasks can be switched accordingly at a regular interval of time, so that the users get the idea that they are performed concurrently. The term multiprogramming is a rudimentary form of parallel processing meaning multiple processes run concurrently at the same time on a single processor. The term is used in modern operating systems when multiple programs or processes run on a single processor and it's the job of the OS to manage all the processes effectively and efficiently. Multitasking refers to the ability of the OS to execute multiple tasks at a time using multiple CPUs. It basically uses two or more CPUs within a single system for allocation of tasks which share common resources including CPU & Memory. Multiprogramming is based on the concept of context switching which is a standard procedure that facilitates switching of the CPU from one process of thread to another utilizing a single CPU. It stores the state of an active process for the CPU in the Process Control Block (PCB) so that the process resumes from the same state. Multitasking, on the other hand, is based on the concept of time sharing which is a technique used to provide each user with a portion of the time-shared system allowing users to share the same resources simultaneously. It's a logical extension of multiprogramming.

Multiprogramming: In a multiprogramming system there are one or more programs loaded in main memory which are ready to execute. Only one program at a time is able to get the CPU for executing its instructions (i.e., there is at most one process running on the system) while all the others are waiting their turn. The main idea of multiprogramming is to maximize the

use of CPU time. Indeed, suppose the currently running process is performing an I/O task (which, by definition, does not need the CPU to be accomplished). Then, the OS may interrupt that process and give the control to one of the other in-main-memory programs that are ready to execute (i.e. *process context switching*). In this way, no CPU time is wasted by the system waiting for the I/O task to be completed, and a running process keeps executing until either it voluntarily releases the CPU or when it blocks for an I/O operation. Therefore, the ultimate goal of multiprogramming is to keep the CPU busy as long as there are processes ready to execute.

Note that in order for such a system to function properly, the OS must be able to load multiple programs into separate areas of the main memory and provide the required protection to avoid the chance of one process being modified by another one. Other problems that need to be addressed when having multiple programs in memory is *fragmentation* as programs enter or leave the main memory. Another issue that needs to be handled as well is that large programs may not fit at once in memory which can be solved by using *pagination* and *virtual memory*. Finally, note that if there are N ready processes and all of those are highly CPU-bound (i.e., they mostly execute CPU tasks and none or very few I/O operations), in the very worst case one program might wait all the other N-1 ones to complete before executing.

<u>Multiprocessing:</u> Multi-processing sometimes refers to executing multiple processes (programs) at the same time. This might be misleading because we have already introduced the term "multiprogramming" to describe that

before. In fact, multiprocessing refers to the *hardware* (i.e., the CPU units) rather than the *software* (i.e., running processes). If the underlying hardware provides more than one processor then that is multiprocessing. Several variations on the basic scheme exist, e.g., multiple cores on one die or multiple dies in one package or multiple packages in one system. Anyway, a system can be both multi-programmed by having multiple programs running at the same time and multiprocessing by having more than one physical processor.

Multitasking: Multi-tasking has the same meaning of multiprogramming but in a more general sense, as it refers to having multiple (programs, processes, tasks, threads) running at the same time. This term is used in modern operating systems when multiple tasks share a common processing resource (e.g., CPU and Memory). At any time, the CPU is executing one task only while other tasks waiting their turn. The illusion of parallelism is achieved when the CPU is reassigned to another task (i.e. *process* or *thread context switching*).

There are subtle differences between multitasking & multiprogramming. A *task* in a multitasking operating system is not a whole application program but it can also refer to a "thread of execution" when one process is divided into sub-tasks. Each smaller task does not hijack the CPU until it finishes like in the older multiprogramming but rather a fair share amount of the CPU time called quantum.

Just to make it easy to remember, both multiprogramming and multitasking operating systems are (CPU) time sharing systems. However, while in multiprogramming (older OSs) one program as a whole

keep running until it blocks, in multitasking (modern OSs) time sharing is best manifested because each running process takes only a fair quantum of the CPU time.

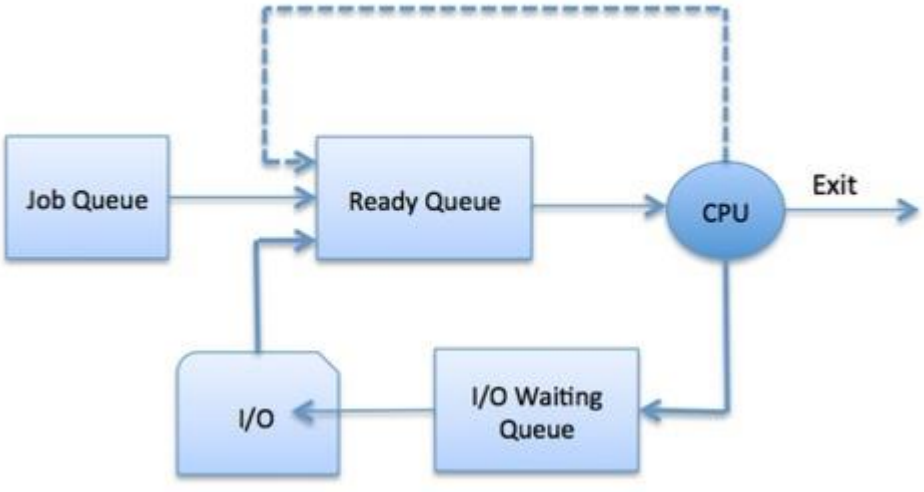
Multithreading: Up to now, we have talked about multiprogramming as a way to allow multiple programs being resident in main memory and (apparently) running at the same time. Then, multitasking refers to multiple tasks running (apparently) simultaneously by sharing the CPU time. Finally, multiprocessing describes systems having multiple CPUs. So, where does multithreading come in?

Multithreading is an execution model that allows a single process to have multiple code segments (i.e., *threads*) run concurrently within the "context" of that process. You can think of threads as child processes that share the parent process resources but execute independently. Multiple threads of a single process can share the CPU in a single CPU system or (purely) run in parallel in a multiprocessing system. Why should we need to have multiple threads of execution within a single process context?

Well, consider for instance a GUI application where the user can issue a command that require long time to finish (e.g., a complex mathematical computation). Unless you design this command to be run in a separate execution thread you will not be able to interact with the main application GUI (e.g., to update a progress bar) because it is going to be unresponsive while the calculation is taking place.

Of course, designing multithreaded/concurrent applications requires the programmer to handle situations that simply don't occur when developing

single-threaded, sequential applications. For instance, when two or more threads try to access and modify a shared resource (*race conditions*), the programmer must be sure this will not leave the system in an inconsistent or deadlock state. The concept of *context switching* applies to multiprogramming as well as to multitasking yet at a different level of granularity. In the former, the "context" refers to that of a whole process whereas in the latter the "context" may be that of a lighter thread. In fact, process context switching involves switching the virtual memory address space: this includes memory addresses, mappings, page tables, and kernel resources. On the other hand, thread context switching is context switching from one thread to another in the same process (i.e. there is no need of switching the virtual memory address space as the "switcher".



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready & run queues which can only have one entry

per processor core on the system; in the above diagram, it has been merged with the CPU.

Scheduling: The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy. Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

Schedulers: 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 & to decide which process to run. Schedulers are of three types:

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

Long Term Scheduler: It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound & processor bound. It also controls the degree of multiprogramming.

On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler: It is also called as CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute & allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler: Medium-term scheduling is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended process cannot make any progress towards completion. In this condition, to remove the process from memory & make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

Context Switch: A context switch is the mechanism to store & 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.

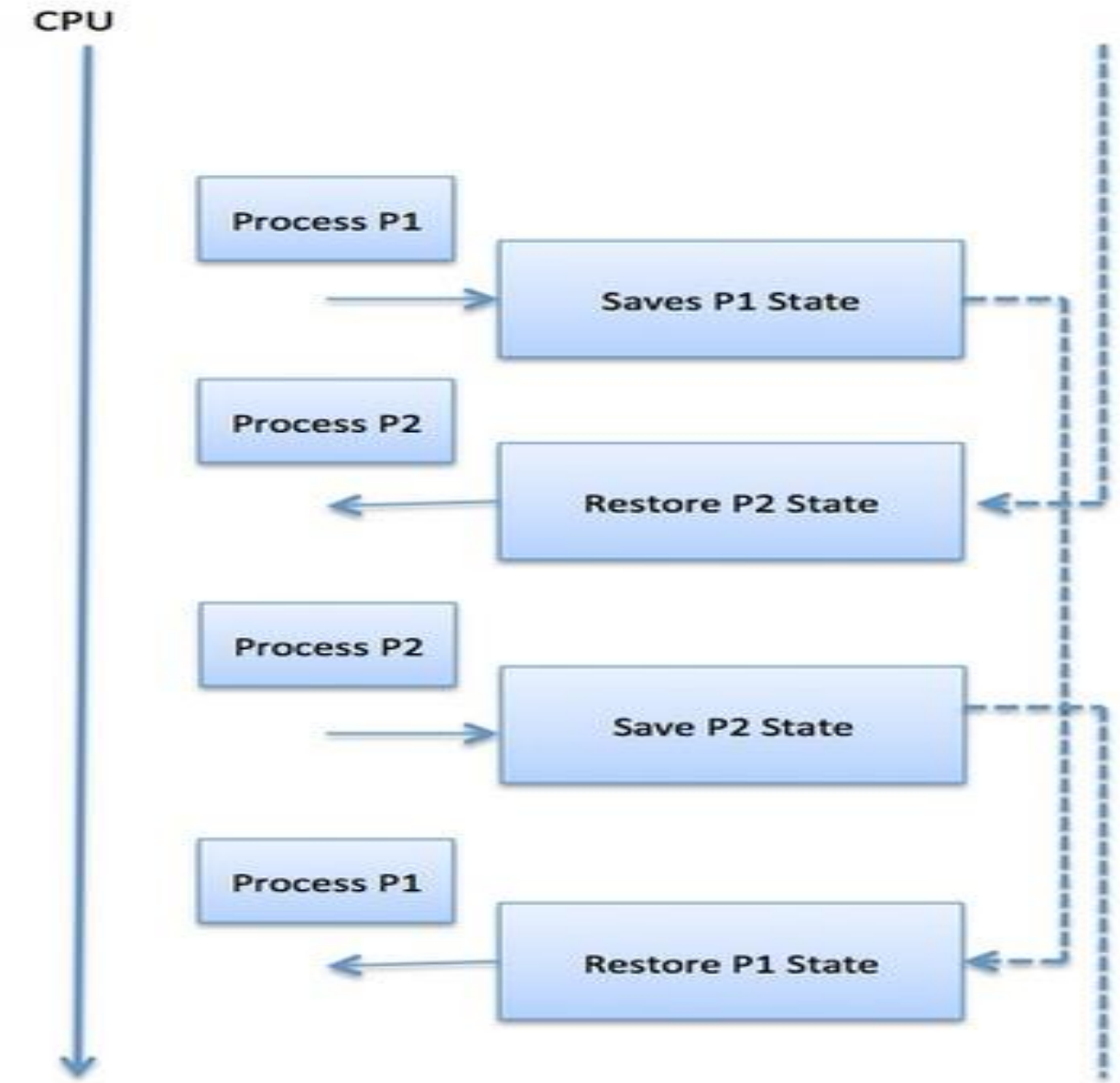
Context switches are computationally intensive since register and memory state must be saved & restored. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers. When the process is switched, the following information is stored for later use.

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

Device Driver: In computing, a device driver is a computer program that operates or controls a particular type of device that is attached to a computer. Typical devices are Keyboards, Printers, Scanners, Digital Cameras, External Storage Devices etc. Each of these need a driver in order to work properly. A driver provides a software interface to hardware devices, enabling operating systems & other computer programs to access hardware functions without needing to know precise details about the hardware being used.

A driver communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program. Drivers are hardware dependent & operating-system-specific.

Synchronization and messaging provide the necessary communication between tasks in one system to tasks in another system. The event flag is used to synchronize internal activities while message queues and mailboxes are used to send text messages between systems. Common data areas utilize semaphores. These are independent kernel objects that are designed to offer flagging mechanisms required to control access to resources. There are two types of semaphores; counting semaphores that feature a random number of states and binary semaphores that feature two states.


CPU
Process P1
Saves P1 State
Process P2
Restore P2 State
Process P2
Save P2 State
Process P1
Restore P1 State