Jaipur Engineering College & Research Centre, Jaipur

Department of Electronics & Communication Engineering

Session (2020-21)

Notes File

MECHATRONIC SYSTEMS (5ME3-07)

Prepared By:
Mr.Devedra Sharma
(AP, Dept. of ECE)

RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Syllabus

3rd Year - V Semester: B.Tech. : Mechanical Engineering

5ME3-01: MECHATRONIC SYSTEMS

Credit: 2 **Max. Marks: 100(IA: 20, ETE:80)**
2L+0T+0P **End Term Exam: 2 Hours**

SN	CONTENTS	Hours
1	Introduction: Objective, scope and outcome of the course.	1
2	Overview of Mechatronics: Historical perspective, Definition, Applications, Block diagram of Mechatronic system, Functions of Mechatronics Systems, Systems Engineering, Verification Vs Validation, Benefits of mechatronics in manufacturing.	2
	Electrical and Electronic Systems: Electrical circuits and Kirchhoff's laws, Network Theorems and AC circuit Analysis, Transformers, Analog Devices, Signal Conditioning, Digital Electronics, Data Acquisition systems.	3
3	Modeling, Analysis and Control of Physical Systems: Basics of System Modeling: LTI and LTV systems, Need for modeling, Types of modeling, Steps in modeling, Building blocks of models, Modelling of one and two degrees of freedom systems, Modeling of Electro-mechanical systems, Mechanical Systems, Fluid systems, Thermal systems; Dynamic Responses, System Transfer Functions, State Space Analysis and System Properties, Stability Analysis using Root Locus Method, Stability Analysis using Bode Plots, PID Controllers (with and without Time Delay)	5
4	Sensors and Actuators: Static characteristics of sensors and actuators, Position, Displacement and Proximity Sensors, Force and torque sensors, Pressure sensors, Flow sensors, Temperature sensors, Acceleration sensors, Level sensors, Light sensors, Smart material sensors, Micro and Nano sensors, Selection criteria for sensors, Actuators: Electrical Actuators (Solenoids, Relays, Diodes, Thyristors, Triacs, BJT, FET, DC motor, Servo motor, BLDC motor, AC motor, Stepper motors), Hydraulic and Pneumatic actuators, Design of Hydraulic and Pneumatic circuits, Piezoelectric actuators, Shape memory alloys.	7
5	Microprocessors, Microcontrollers and Programmable Logic Controllers: Logic Concepts and Design, System Interfaces, Communication and Computer Networks, Fault Analysis in Mechatronic Systems, Synchronous and Asynchronous Sequential Systems, Architecture, Microcontrollers.	3
6	Programmable Logic Controllers (PLCs): Architecture, Number Systems Basics of PLC Programming, Logics, Timers and Counters, Application on real time industrial automation systems.	4
	Case Studies: Design of pick and place robot, Car engine management system, Automated manufacturing system, Automatic camera, Automatic parking system, Safety devices and systems.	3
	TOTAL	28

UNIT 1 Overview of Mechatronics:

Historical perspective, Definition, Applications, Block diagram of Mechatronic system, Functions of Mechatronics Systems, Systems Engineering, Verification Vs Validation, Benefits of mechatronics in manufacturing.

Electrical and Electronic Systems: Electrical circuits and Kirchhoff's laws, Network Theorems and AC circuit Analysis, Transformers, Analog Devices, Signal Conditioning, Digital Electronics, Data Acquisition systems

UNIT 2

Modeling, Analysis and Control of Physical Systems: Basics of System Modeling: LTI and LTV systems, Need for modeling, Types of modeling, Steps in modeling, Building blocks of models, Modelling of one and two degrees of freedom systems, Modeling of Electro-mechanical systems, Mechanical Systems, Fluid systems, Thermal systems; Dynamic Responses, System Transfer Functions, State Space Analysis and System Properties, Stability Analysis using Root Locus Method, Stability Analysis using Bode Plots, PID Controllers (with and without Time Delay

UNIT 3

Sensors and Actuators: Static characteristics of sensors and actuators, Position, Displacement and Proximity Sensors, Force and torque sensors, Pressure sensors, Flow sensors, Temperature sensors, Acceleration sensors, Level sensors, Light sensors, Smart material sensors, Micro and Nano sensors, Selection criteria for sensors, Actuators: Electrical Actuators (Solenoids, Relays, Diodes, Thyristors, Triacs, BJT, FET, DC motor, Servo motor, BLDC motor, AC motor, Stepper motors), Hydraulic and Pneumatic actuators, *Design of Hydraulic and Pneumatic circuits, Piezoelectric actuators, Shape memory alloys.*

UNIT 4

Microprocessors, Microcontrollers and Programmable Logic Controllers: Logic Concepts and Design, System Interfaces, Communication and Computer Networks, Fault Analysis in Mechatronic Systems, Synchronous and Asynchronous Sequential Systems, Architecture, **Microcontrollers.**

Unit 5

Programmable Logic Controllers (PLCs): Architecture, Number Systems Basics of PLC Programming, Logics, Timers and Counters, Application on real time industrial automation systems .Case Studies: Design of pick and place robot, Car engine management system, Automated manufacturing system, Automatic camera, Automatic parking system, Safety devices and systems

UNIT 1 Overview of Mechatronics:

Historical perspective, Definition, Applications, Block diagram of Mechatronic system, Functions of Mechatronics Systems, Systems Engineering, Verification Vs Validation, Benefits of mechatronics in manufacturing.

Electrical and Electronic Systems: Electrical circuits and Kirchhoff's laws, Network Theorems and AC circuit Analysis, Transformers, Analog Devices, Signal Conditioning, Digital Electronics, Data Acquisition systems

UNIT- 1

Overview of Mechatronics

INTRODUCTION: The word Mechatronics was first introduced by the senior engineer of a Japanese company, Yaskawa, in 1969. The company was granted trademark rights on the word in 1971. The word soon received broad acceptance in industry and, in order to allow its free use, Yaskawa decided to abandon his rights on the word in 1982. The Mechatronics systems consist of components from different physical domains, which are closely coupled and therefore interact with one another. Moreover, the behavior of the components is highly non-linear.

Today it means mechatronics engineering activities including designing, testing and operation of machinery and equipment, in which there is a high level of functional integration of mechanical systems with electronics and computer control.Mechatronics is an interdisciplinary field, combining in a synergistic manner the classical knowledge of mechanical engineering, hydraulics, pneumatics, electronics, optics and computer science. A typical mechatronic system picks up signals from the environment, processes them to generate output signals, transforming them for example into forces, motions and actions. The aim of Mechatronics is to improve the functionality of technical systems and the creation of new concepts of machinery and equipment with built-in "artificial intelligence".Mechatronics provides an opportunity, not only humanization of machines, but also it changes the mindset and the approach to technological issues and most importantly teaching new technologies and ways of acquiring knowledge and skills. The most important feature of mechatronic devices is the ability to process and communicate information accurately in a form of different types of signals (mechanical, electrical, hydraulic, pneumatic, optical, chemical, biological), with high level of automation of these devices.Mechatronics DefinitionMechatronics is the synergistic integration of sensors, actuators, signal conditioning, power electronics, decision and control algorithms, and computer hardware and software to manage complexity, uncertainty, and communication in engineered systems.IEEE (Institute of Electrical and Electronics Engineers) and ASME (American Society

Mechanical Engineers) promote the following definition: Mechatronics is the synergistic integration of mechanical engineering with the electronic control and the intelligent, PC- based control, in the design and manufacturing of goods and processes.

Objectives Of Mechatronics-echatronics has mainly the objective to improve technicalproperties, i.e., to make machines work faster and to make manufacturing cheaper. In industry, products and processes are designed fromscratch, and therefore they are known, and dealing with them is a kind of straightforward actionwhere the behavior can be predicted, at least in principle. Even there, however, the complexity oftasks and situations is increasing, leading already o the use of unconventional tools like fuzzy control,neural networks, expert systems, and their combinations. Figure shown the mechatronics industry system

Mechatronics Design Step-echatronic products are commonly used in industry and everyday life. Designing of mechatronic products requires aedicated approach that takes into account: interdisciplinary design, market related constraints, multifunctionality, user- friendly operation and demand of minimization of the cost of the whole product operation period. Thus designers who create mechatronic products should possess comprehensive interdisciplinary knowledge, ability to co-operate in an interdisciplinary designing team as well as team management skills, and knowledge how to use the up-to-date tools of computer aided engineering. Additionally the know-how in scheduling and carrying out prototyping of mechatronic systems is very useful.

Type of Elements

Electromechanical elements: –

Sensors, A variety of physical variables can be measured using sensors. Actuators, DCservomotor, stepper motor, relay, solenoid, speaker, light emitting diode (LED), shape memory alloy, electromagnet, and pump apply commanded action on thephysical process.IC-based sensors and actuatorsdigital-compass, potentiometer, etc.

Fig: – Mechatronics Design Process ectrical elements:-

Electrical components e.g., resistor (R), capacitor (C), inductor (L), transformer, etc

Electronic elements:-

Analog/digital electronics, transistors, thyristors, opto-isolators, operationalamplifiers, power electronics, and signal conditioning.

Control interface /computing hardware elements:-

Analog-to-digital (A2D) converter, digital-to-analog (D2A) converter, digitalinput/output (I/O), counters, timers,microprocessor, microcontroller, dataacquisition and control (DAC) board, and digital signal processing (DSP) board

Computer/Information System:-

Computer elements refer to hardware/software utilized to perform, computer- aided dynamic system analysis, optimization, design, and simulation, virtual instrumentation, rapid control prototyping, hardware-in-the-loop simulation, PC-based data acquisition and control

Elements of Mechatronics:-

Typical knowledgebase for optimal design and operation of mechatronic systems comprises of: Dynamic system modelling and analysis. Thermo-fluid, tructural, hydraulic, electrical, chemical, biological, etc. Decision and control theory, Sensors and signal conditioning, Actuators and power electronics, Data acquisition. A2D, D2A, digital I/O, counters, timers, etc. Hardware interfacing, Rapid control prototyping, Embedded computing.Balance theory, simulation, hardware, and software.

Mechatronics is the result of applying information systems to physicalsystems. The physical system, the rightmost dotted block, consists of mechanical, electrical, andcomputer (electronic) systems as well as actuators, sensors, and real time interfacing. Sensorsand actuators are used to transduce energy from high power, usually the mechanical side, to lowpower, the electrical and computer or electronic side. The block labeled mechanical systemsfrequently consists of more than just mechanical components and may include fluid, pneumatic,thermal, acoustic, chemical, and other disciplines as well.

New Research Challenges

These interactively cooperating, intelligent machines lead to new research topics in the controltechniques of mechatronics and in other areas as well. It will be important that a machine and itscomponents have learning capabilities, self- adaptation and self-calibration. Techniques such as thecombination of neural networks, and fuzzy control with expert systems will further emphasize theimportance of software.

Fig: – Elements of Mechatronics

Mechatronic Engineering is the engineering discipline concerned with the research, design, implementation and maintenance of intelligent engineered products and processes enabled by the integration of mechanical, electronic, computer, and software engineering technologiesSpecific expertise areas can include:

- Artificial Intelligence Techniques
- Avionics Computer Hardware and Systems Control Systems
- Data Communications and Networks

- Dynamics of Machines and Mechanisms
- Electromagnetic Energy Conversion
- Electronics
- Embedded & Real-time Systems
- Fluid Power and other Actuation Devices
- Human-Machine Interface Engineering and Ergonomics
- Industrial Automation
- Measurement, Instrumentation and Sensors
- Mechanical Design and Material Selection
- Mechatronic Design and System Integration
- Modelling and Simulation
- Motion Control
- Power Electronics
- Process Management, Scheduling, Optimization, and Control
- Process Plant and Manufacturing Systems

- Robotics
- Signal Processing
- Smart Infrastructure
- Software Engineering
- Systems Engineering
- Thermofluids
- Haptic Interfaces
- Medicals
- Automation
- Robotics
- Control
- Mechanical Vibrations
- Mechatronics In Medicine

In 1985 the Department of Mechanical Engineering at Imperial College began research into medical robotics for neurosurgery. Further research into a robot for prostatectomy, commencing in 1988 culminated, in 1991, in a "World First" with the demonstration of robotic prostate

surgery. This robot was the first to actively remove tissue from a human patient in an operating theatre. With the expansion of robotic surgery applications, the Mechatronics in Medicine Laboratory was set up in 1993, as part of the Computer Aided Systems Engineering Section, to research and develop mechatronic aids to surgery. The group has developed mechatronic applications in fields as diverse as neurosurgery, magnetic resonance imaging (MRI) compatible robotics, haptic training systems for surgeons, urological

surgery and orthopaedics, high intensity focused ultrasound and blood sampling.

New Applications

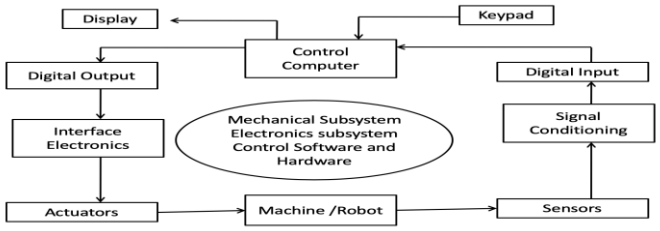
i-VTEC, Micro Air Vehicles, Pistonless Pump, Multi Air Engine, Six Stroke Engine, Solar Cars, Thermo Acoustic Refrigeration, Biodeisel, Digital Twin Spark Ignition, Nano Enabled Coating Makes Aircraft Invisible, Automatic Transmission In Cars, Nitro Shock Absorbers, Hypercar, 3d Machine Vision Systems, Geothermal Energy, Cryogenic Heat Treatment, Next Generation 2-Stroke Engine, Microturbines, Next Generation Engines, Tidal Energy, Air Craft Hydraulic System, Cruise Missile Technology, Camless Engine, Automatic Vehicle Locator, Autonomous Car, Solar Energy Through Solar Space Stations, Anti-Lock Braking Sensors, Air Muscles, SkyBus Technology, Scramjet Engine, Pollution Less Engine, Paper Battery, Nano IC Engine, Liquid Nitrogen, Gasoline Direct Injection, Emulsified Ethanol, Direct Injection Diesel Engine, Vehicle Dynamics, Valvetronics, Tidal Power, Hovercraft, Infrared Curing and Convection Curing, Aeroplane Propulsion System, Running gearing, Fuel Energizer, GPS And Applications, Selective Laser Sintering, Agile Manufacturing, Cryocar, Cylinder deactivation, Dyna-cam engine, Apache Helicopter, CAMM Systems, Friction Stir Welding, HEMI engines, Just In Time Manufacturing, Lean manufacturing, Quality improvement tool "poka yoke", MEMS for Space, Personal Protection ,, Mine Detection Using Radar Bullets, Overall Equipment Effectiveness, Predictive Maintenance using Thermal Imaging, Methanol Fueled Marine Diesel Engine, Quality function deployment, Quasi turbine, Robots In Radioactive Environments, Sidewinder Missile, Smart Materials, Transit mxer & Concrete Pump, Turbofan Engine, Solar Sails, Ultrasonic Metal

Welding, The Hy-Wire Car, Thermal Barrier Coatings, Ultrasonic Techniques for hidden corrosion detection, Solar-powered vehicles , Two Stroke Engine Using Reed Valves, Vacuum Braking System, Variable Valve Timing In I.C. Engines, Space Elevator, Supercavitation, Thermal shock on interfacial adhesion of thermally conditioned, Total Productive Maintenance, Welding Robots , Air powered cars, Biomechatronic Hand , Computer Aided Process Planning , Re-entry of Space Vehicle, Sensotronic Brake Control, Skid Steer Loader and Multiterrain Loader, Space Robotics, Space Shuttles and its Advancements, Continuously variable transmission (CVT), Cryogenic grinding, Design, Analysis, Fabrication And Testing Of A Composite Leaf Spring, F1 Track Design and Safety, Green Engine, Head And Neck Support (HANS), Hydro Drive , Fractal Robots, Smart Bombs,, Military Radars, Stealth Fighter, Handfree Driving, Solar Power Satellites (SPS), Nano Technology, Iontophoresis, Aerodynamics, Micro- Electromechanical Systems.

Sensors to measure the controlled variables. Sensors of all types provide the computer with information. it needs to perform monitoring and control tasks.

Actuators convert digital signals into physical or chemical quantities as required to correct errors in meeting the required performance.

Instrumentation subsystem.



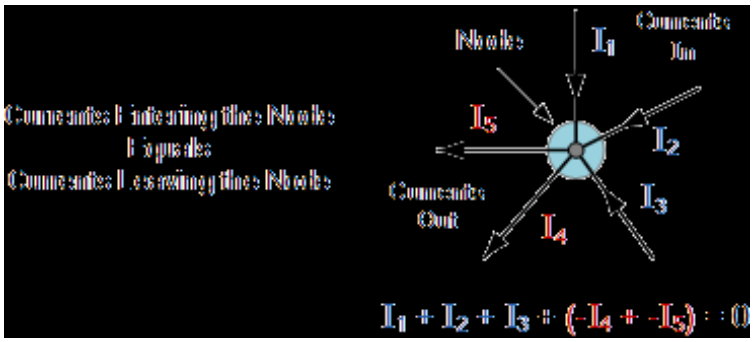
This involves all necessary signal-conditioning (SC) circuits to interface actuators, sensors, and other devices to the control computer.

Control computer. Proper control of a process requires intense processing of information representing a complex interplay between sensing the environmental conditions that affect the outcome of the process, interpretation of the sensed values, recognising effect of these values on the integrity of the operation, and initiating a response by signalling the actuators to take the necessary corrective actions. Coordinating all these activities is the responsibility of the control computer, which involves embedded microprocessor systems. it handles the flow of information between the various components by integrating the software and hardware dedicated to controlling the process in question. The hardware, also referred to as the target system, consists of all computers needed to manage the inputs and outputs of the mechatronic device. This includes sensors, actuators, mechanical components, power supplies, control computer, and the SC circuits interfacing the computer with the sensors and actuators. As a first step, a prototype of the hardware may be built using wire-wrapped boards to test the application software before the final product is realised. The software manages all activities of the target system. It consists of the program (or code), which is developed to meet the application requirement. Development of the application code can begin concurrently with the hardware prototype development once the system specifications have been completed. The code could be modified as it is developed to reflect design changes and/or new requirements as they emerge. The application code may be developed using machine language, assembly language, or a high-level language such as C. The machine code, or object code, contains binary codes that represent the instructions to the computer. It is the only language the microprocessor can understand and execute. High-level languages such as C use words and statements that are easily understood. A program written in a high-level language must be translated into executable machine language using a program called a compiler. Assembly language represents a middle ground between the extremes of machine language and the high-level language. An assembly language program is written using

mnemonics in which each statement corresponds to a machine instruction. The assembly program must be translated into machine language by a special program called an assembler.

Kirchhoffs First Law – The Current Law, (KCL)

Kirchhoffs Current Law or KCL, states that the "total current or charge entering a junction or node is exactly equal to the charge leaving the node as it has no other place to go except to leave, as no charge is lost within the node". In other words the algebraic sum of ALL the currents entering and leaving a node must be equal to zero, I(exiting) + I(entering) = 0. This idea by Kirchhoff is commonly known as the Conservation of Charge.

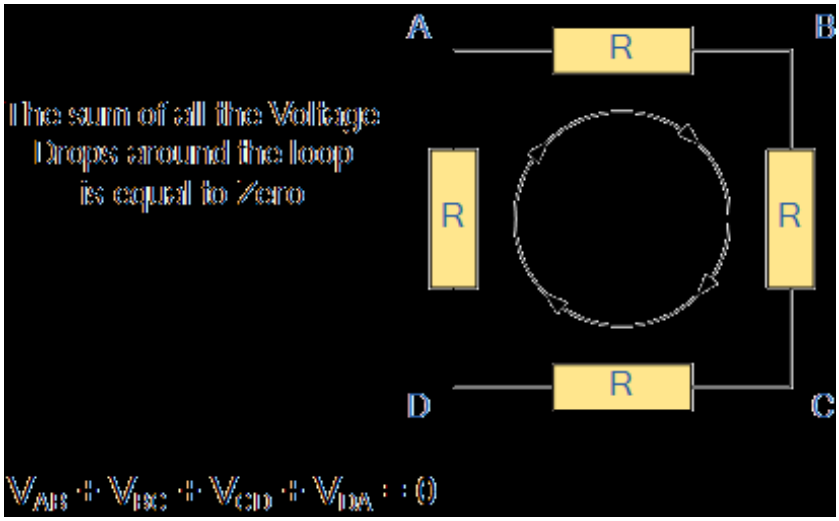


Here, the three currents entering the node, I1, I2, I3 are all positive in value and the two currents leaving the node, I4 and I5 are negative in value. Then this means we can also rewrite the equation as;I1 + I2 + I3 – I4 – I5 = 0

The term Node in an electrical circuit generally refers to a connection or junction of two or more current carrying paths or elements such as cables and components. Also for current to flow either in or out of a node a closed circuit path must exist. We can use Kirchhoff's current law when analysing parallel circuits.

Kirchhoffs Second Law – The Voltage Law, (KVL)

Kirchhoffs Voltage Law or KVL, states that "in any closed loop network, the total voltage around the loop is equal to the sum of all the voltage drops within the same loop" which is also equal to zero. In other words the algebraic sum of all voltages within the loop must be equal to zero. This idea by Kirchhoff is known as the Conservation of Energy.



Starting at any point in the loop continue in the same direction noting the direction of all the voltage drops, either positive or negative, and returning back to the same starting point. It is important to maintain the same direction either clockwise or anti-clockwise or the final voltage sum will not be equal to zero. We can use Kirchhoff's voltage law when analysing series circuits.When analysing either DC circuits or AC circuits using Kirchhoffs Circuit Laws a number of definitions and terminologies are used to describe the parts of the circuit being analysed such as: node, paths, branches, loops and meshes. These terms are used frequently in circuit analysis so it is important to understand them.

Common DC Circuit Theory Terms:

• Circuit – a circuit is a closed loop conducting path in which an electrical current flows.

• Path – a single line of connecting elements or sources.

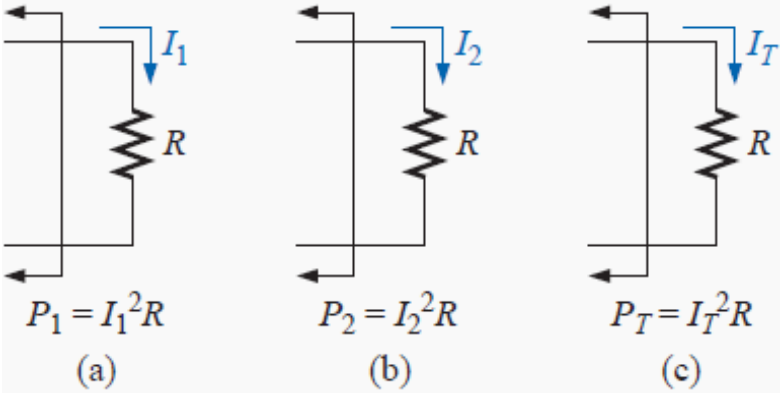
• Node – a node is a junction, connection or terminal within a circuit were two or more circuit elements are connected or joined together giving a connection point between two or more branches. A node is indicated by a dot.

• Branch – a branch is a single or group of components such as resistors or a source which are connected between two nodes.

• Loop – a loop is a simple closed path in a circuit in which no circuit element or node is encountered more than once.

• Mesh – a mesh is a single closed loop series path that does not contain any other paths. There are no loops inside a mesh.

Superposition Theorem



The current through, or voltage across, an element in a linear bilateral network is equal to the algebraic sum of the currents or voltages produced independently by each source.

To consider the effects of each source independently requires that sources be removed and replaced without affecting the final result.

To remove a voltage source when applying this theorem, the difference in potential between the terminals of the voltage source must be set to zero (short circuit); removing a current source requires that its terminals be opened (open circuit).Any internal resistance or conductance associated with the displaced sources is not eliminated but must still be considered.

THÉVENIN'S THEOREM

The next theorem to be introduced, Thévenin's theorem, is probably oneof the most interesting in that it permits the reduction of complex

networks to a simpler form for analysis and design.In general, the theorem can be used to do the following:

• Analyze networks with sources that are not in series or parallel.

• Reduce the number of components required to establish the same

characteristics at the output terminals.Investigate the effect of changing a particular component on the behavior of a network without having to analyze the entire network after each change.All three areas of application are demonstrated in the examples to follow.

Thévenin's theorem states the following:

Any two-terminal dc network can be replaced by an equivalent circuit consisting solely of a voltage source and a series resistor as shown in

NORTON'S THEOREM

every voltage source with a series internal resistance has a current source equivalent. The current source equivalent can be determined by Norton's theorem (Fig. 9.64). It can also be

found through the conversions of Section 8.2.The theorem states the following: Any two-terminal linear bilateral dc network can be replaced by an quivalent circuit consisting of a current source and a parallel rsistor,

Norton's Theorem Procedure

1. Remove that portion of the network across which the Norton euivalent circuit is found.

2. Mark the terminals of the remaining two-terminal networ

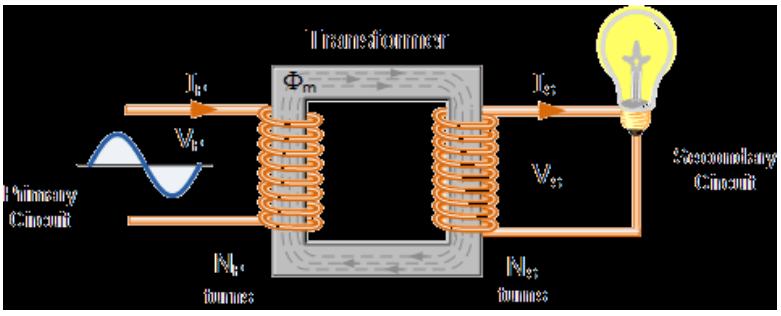
3. Calculate RN by first setting all sources to zero (voltage sources are rplaced with short circuits and current sources with openrcuits) and then finding the resultant resistance between the twomarked terminals. (If the internal resistance of the voltage and/orcurrent sources is included in the original network, it must remainwhen the sources are set to zero.) Since RN = RTh, the procedureand value obtained using the approach described for Thévenin'stheorem will determine the proper value of RN.IN:

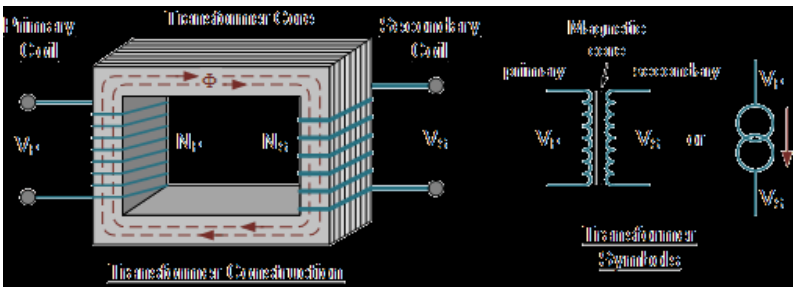
4. Calculate IN by first returning all sources to their original position and then finding the short-circuit current between the markedterminals. It is the same current that would be measured by anammeter placed between the marked terminals.Conclusion:

5. Draw the Norton equivalent circuit with the portion of the circuitpreviously removed replaced between the terminals of the equivalent

Transformer Basics

Transformers are electrical devices consisting of two or more coils of wire used to transfer electrical energy by means of a changing magnetic field oneof the main reasons that we use alternating AC voltages and currents in our homes and workplace's is that AC supplies can be easily generated at a convenient voltage, transformed (hence the name transformer) into much higher voltages and then distributed around the country using a national grid of pylons and cables over very long distances.The reason for transforming the voltage to a much higher level is that higher distribution voltages implies lower currents for the same power and therefore lower I2*R losses along the networked grid of cables. These higher AC transmission voltages and currents can then be reduced to a much lower, safer and usable voltage level where it can be used to supply electrical equipment in our homes and workplaces, and all this is possible thanks to the basic Voltage Transformer.





The systems, used for data acquisition are known as data acquisition systems. These data acquisition systems will perform the tasks such as conversion of data, storage of data, transmission of data and processing of data.

Data acquisition systems.Analog signals, which are obtained from the direct measurement of electrical quantities such as DC & AC voltages, DC & AC currents, resistance and etc.Analog signals, which are obtained from transducers such as LVDT, Thermocouple & etc.

Types of Data Acquisition Systems

Data acquisition systems can be classified into the following two types.

Analog Data Acquisition Systems

Digital Data Acquisition Systems

Now, let us discuss about these two types of data acquisition systems one by one.

Analog Data Acquisition Systems

The data acquisition systems, which can be operated with analog signals are known as analog data acquisition systems. Following are the blocks of analog data acquisition systems.

Transducer − It converts physical quantities into electrical signals.

Signal conditioner − It performs the functions like amplification and selection of desired portion of the signal.

Display device − It displays the input signals for monitoring purpose.

Graphic recording instruments − These can be used to make the record of input data permanently.

Magnetic tape instrumentation − It is used for acquiring, storing & reproducing of input data.

Digital Data Acquisition Systems

The data acquisition systems, which can be operated with digital signals are known as digital data acquisition systems. So, they use digital components for storing or displaying the information.

Mainly, the following operations take place in digital data acquisition.

Acquisition of analog signals

Conversion of analog signals into digital signals or digital data

Processing of digital signals or digital data

Following are the blocks of Digital data acquisition systems.

Transducer − It converts physical quantities into electrical signals.

Signal conditioner − It performs the functions like amplification and selection of desired portion of the signal.

Multiplexer − connects one of the multiple inputs to output. So, it acts as parallel to serial converter.

Analog to Digital Converter − It converts the analog input into its equivalent digital output.

Display device − It displays the data in digital format.

Digital Recorder − It is used to record the data in digital format.Data acquisition systems are being used in various applications such as biomedical and aerospace. So, we can choose either analog data acquisition systems or digital data acquisition systems based on the requirement.

UNIT 2

Modeling, Analysis and Control of Physical Systems: Basics of System Modeling: LTI and LTV systems, Need for modeling, Types of modeling, Steps in modeling, Building blocks of models, Modelling of one and two degrees of freedom systems, Modeling of Electro-mechanical systems, Mechanical Systems, Fluid systems, Thermal systems; Dynamic Responses, System Transfer Functions, State Space Analysis and System Properties, Stability Analysis using Root Locus Method, Stability Analysis using Bode Plots, PID Controllers (with and without Time Delay

UNIT-2

Modeling :

control systems can be represented with a set of mathematical equations known as mathematical model. These models are useful for analysis and design of control systems. Analysis of control system means finding the output when we know the input and mathematical model. Design of control system means finding the mathematical model when we know the input and the output.

The following mathematical models are mostly used.

Differential equation model

Transfer function model

State space model

Let us discuss the first two models in this chapter.

Differential Equation Model

Differential equation model is a time domain mathematical model of control systems. Follow these steps for differential equation model.

Apply basic laws to the given control system.

Get the differential equation in terms of input and output by eliminating the intermediate variable(s).

Example

Consider the following electrical system as shown in the following figure. This circuit consists of resistor, inductor and capacitor. All these electrical elements are connected in series. The input voltage applied to this circuit is vi and the voltage across the capacitor is the output voltage vo.

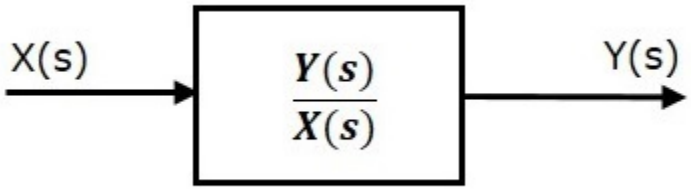
Tansfer Function Model

Transfer function model is an s-domain mathematical model of control systems. The Transfer function of a Linear Time Invariant (LTI) system is defined as the ratio of Laplace transform of output and Laplace transform of input by assuming all the initial conditions are zero.

If x(t) and y(t) are the input and output of an LTI system, then the corresponding Laplace transforms are X(s) and Y(s)

herefore, the transfer function of LTI system is equal to the ratio of Y(s) and X(s)

The transfer function model of an LTI system is shown in the following figure



Here, we represented an LTI system with a block having transfer function inside it. And this block has an input X(s) & output Y(s)

d2vodt2+(RL)dvodt+(1LC)vo=(1LC)vi

Apply Laplace transform on both sides.

s2Vo(s)+(sRL)Vo(s)+(1LC)Vo(s)=(1LC)Vi(s)

⇒{s2+(RL)s+1LC}Vo(s)=(1LC)Vi(s)

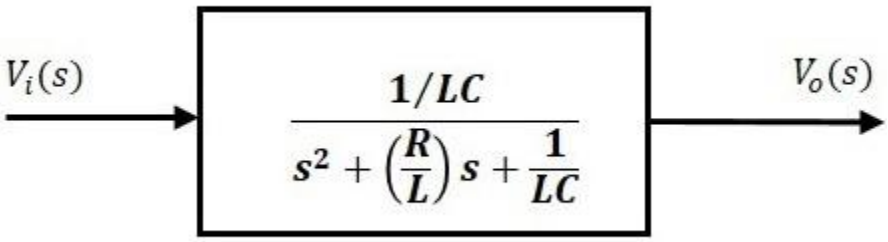
⇒Vo(s)Vi(s)=1LCs2+(RL)s+1LC

Where,

vi(s) is the Laplace transform of the input voltage vi

vo(s) is the Laplace transform of the output voltage vo

The above equation is a transfer function of the second order electrical system. The transfer function model of this system is shown below.



ere, we show a second order electrical system with a block having the transfer function inside it. And this block has an input Vi(s) & an output Vo(s).

here are two types of mechanical systems based on the type of motion.

Translational mechanical systems

Rotational mechanical systems

Modeling of Translational Mechanical Systems

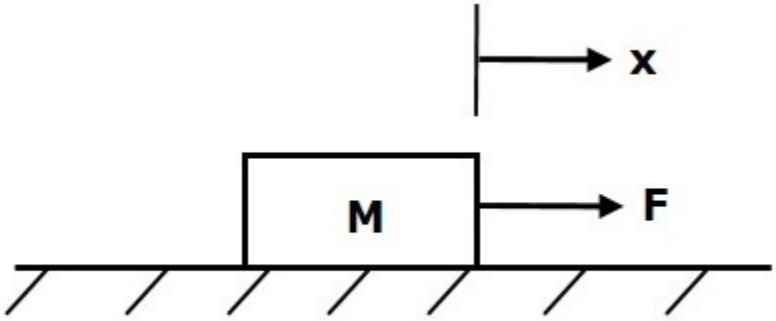
Translational mechanical systems move along a straight line. These systems mainly consist of three basic elements. Those are mass, spring and dashpot or damper.

If a force is applied to a translational mechanical system, then it is opposed by opposing forces due to mass, elasticity and friction of the system. Since the applied force and the opposing forces are in opposite directions, the algebraic sum of the forces acting on the system is zero. Let us now see the force opposed by these three elements individually.

Mass

Mass is the property of a body, which stores kinetic energy. If a force is applied on a body having mass M, then it is opposed by an opposing force due to mass. This opposing force is

proportional to the acceleration of the body. Assume elasticity and friction are negligible



$Fm \propto a$

$\Rightarrow Fm = Ma = M\frac{d^2x}{dt^2}$

$F = Fm = M\frac{d^2x}{dt^2}$

Where,

F is the applied force

Fm is the opposing force due to mass

M is mass

a is acceleration

x is displacement

Spring

Spring is an element, which stores potential energy. If a force is applied on spring K, then it is opposed by an opposing force due to elasticity of spring. This opposing force is proportional to the displacement of the spring. Assume mass and friction are negligible.

Modeling of Rotational Mechanical Systems

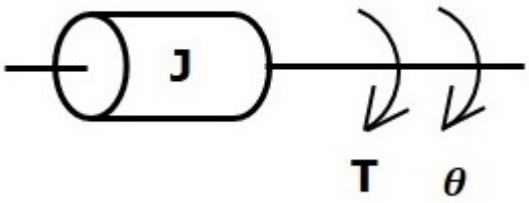
Rotational mechanical systems move about a fixed axis. These systems mainly consist of three basic elements. Those are moment of inertia, torsional spring and dashpot.

If a torque is applied to a rotational mechanical system, then it is opposed by opposing torques due to moment of inertia, elasticity and friction of the system. Since the applied torque and the opposing torques are in opposite directions, the algebraic sum of torques acting on the system is zero. Let us now see the torque opposed by these three elements individually.

Moment of Inertia

In translational mechanical system, mass stores kinetic energy. Similarly, in rotational mechanical system, moment of inertia stores kinetic energy.

If a torque is applied on a body having moment of inertia J, then it is opposed by an opposing torque due to the moment of inertia. This opposing torque is proportional to angular acceleration of the body. Assume elasticity and friction are negligible.



$Tj \propto \alpha$

$\Rightarrow Tj = J\alpha = J\frac{d^2\theta}{dt^2}$

$T = Tj = J\frac{d^2\theta}{dt^2}$

Where,

T is the applied torque

Tj is the opposing torque due to moment of inertia

J is moment of inertia

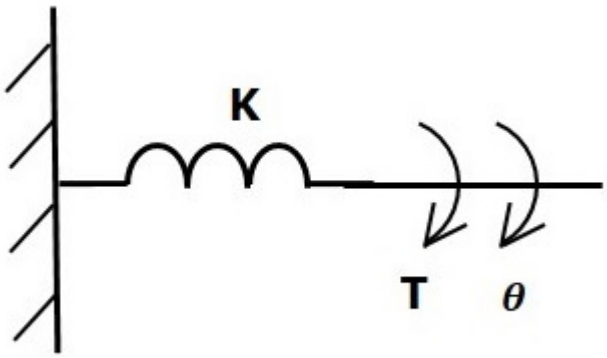
α is angular acceleration

θ is angular displacement

Torsional Spring

In translational mechanical system, spring stores potential energy. Similarly, in rotational mechanical system, torsional spring stores potential energy.

If a torque is applied on torsional spring K, then it is opposed by an opposing torque due to the elasticity of torsional spring. This opposing torque is proportional to the angular displacement of the torsional spring. Assume that the moment of inertia and friction are negligible.



$T_k \propto \theta$

$\Rightarrow T_k = K\theta$

$T = T_k = K\theta$

Where,

T is the applied torque

T_k is the opposing torque due to elasticity of torsional spring

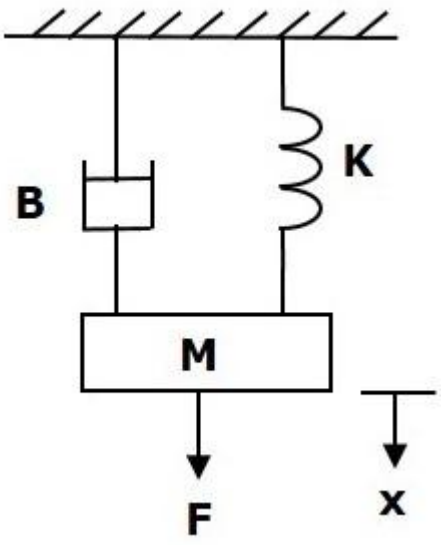
K is the torsional spring constant

θ is angular displacement

Force Voltage Analogy

In force voltage analogy, the mathematical equations of translational mechanical system are compared with mesh equations of the electrical system.

Consider the following translational mechanical system as shown in the following figure.

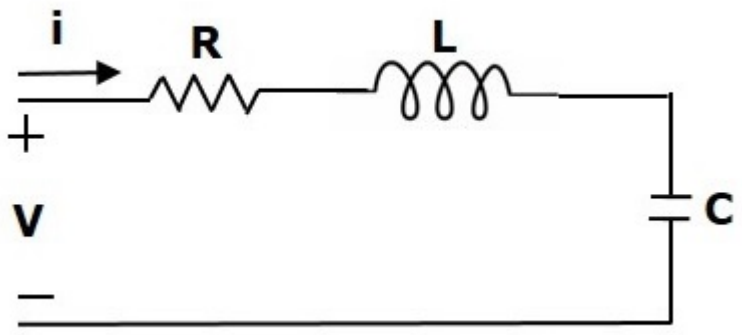


e force balanced equation for this system is

F=Fm+Fb+Fk

⇒F=Md2xdt2+Bdxdt+Kx (Equation 1)

Consider the following electrical system as shown in the following figure. This circuit consists of a resistor, an inductor and a capacitor. All these electrical elements are connected in a series. The input voltage applied to this circuit is V volts and the current flowing through the circuit is i Amps.



Mesh equation for this circuit is

V=Ri+Ldidt+1c∫idt (Equation 2)

Substitute, i=dqdt in Equation 2.

V=Rdqdt+Ld2qdt2+qC

⇒V=Ld2qdt2+Rdqdt+(1c)q (Equation 3)

By comparing Equation 1 and Equation 3, we will get the analogous quantities of the translational mechanical system and electrical system. The following table shows these analogous quantities.

Translational Mechanical System	**Electrical System**
Force(F)	Voltage(V)
Mass(M)	Inductance(L)
Frictional Coefficient(B)	Resistance(R)
Spring Constant(K)	Reciprocal of Capacitance (1c)
Displacement(x)	Charge(q)
Velocity(v)	Current(i)

Similarly, there is torque voltage analogy for rotational mechanical systems. Let us now discuss about this analogy.

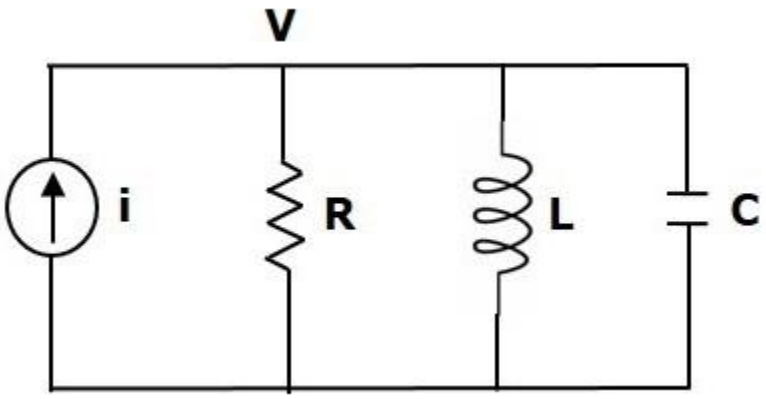
rotational Mechanical System	**Electrical System**
Torque(T)	Voltage(V)
Moment of Inertia(J)	Inductance(L)
Rotational friction coefficient(B)	Resistance(R)
Torsional spring constant(K)	Reciprocal of Capacitance (1c)
Angular Displacement(θ)	Charge(q)

Angular Velocity(ω)	Current(i)

Force Current Analogy

In force current analogy, the mathematical equations of the translational mechanical system are compared with the nodal equations of the electrical system.

Consider the following electrical system as shown in the following figure. This circuit consists of current source, resistor, inductor and capacitor. All these electrical elements are connected in parallel.



The nodal equation is

$i=\frac{V}{R}+\frac{1}{L}\int V dt+C\frac{dV}{dt}$ (Equation 5)

Substitute, $V=\frac{d\Psi}{dt}$ in Equation 5.

$i=\frac{1}{R}\frac{d\Psi}{dt}+(\frac{1}{L})\Psi+C\frac{d^2\Psi}{dt^2}$

$\Rightarrow i=C\frac{d^2\Psi}{dt^2}+(\frac{1}{R})\frac{d\Psi}{dt}+(\frac{1}{L})\Psi$ (Equation 6)

By comparing Equation 1 and Equation 6, we will get the analogous quantities of the

translational mechanical system and electrical system. The following table shows these analogous quantities.

Translational Mechanical System	Electrical System
Force(F)	Current(i)
Mass(M)	Capacitance(C)
Frictional coefficient(B)	Reciprocal of Resistance(1R)
Spring constant(K)	Reciprocal of Inductance(1L)
Displacement(x)	Magnetic Flux(ψ)
Velocity(v)	Voltage(V)

Similarly, there is a torque current analogy for rotational mechanical systems. Let us now discuss this analogy.

Torque Current Analogy

In this analogy, the mathematical equations of the rotational mechanical system are compared with the nodal mesh equations of the electrical system.

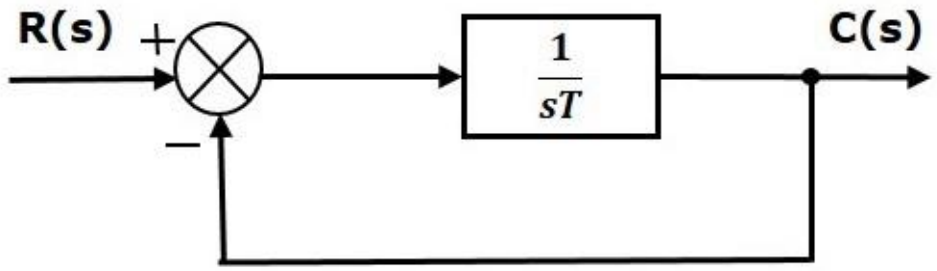
By comparing Equation 4 and Equation 6, we will get the analogous quantities of rotational mechanical system and electrical system. The following table shows these analogous quantities.

Rotational Mechanical System	Electrical System
Torque(T)	Current(i)
Moment of inertia(J)	Capacitance(C)
Rotational friction coefficient(B)	Reciprocal of Resistance(1R)
Torsional spring constant(K)	Reciprocal of Inductance(1L)
Angular displacement(θ)	Magnetic flux(ψ)
Angular velocity(ω)	Voltage(V)

In this chapter, we discussed the electrical analogies of the mechanical systems. These analogies are helpful to study and analyze the non-electrical system like mechanical system from analogous electrical system.

Response of the First Order System

In this chapter, let us discuss the time response of the first order system. Consider the following block diagram of the closed loop control system. Here, an open loop transfer function, 1sT is connected with a unity negative feedback.



we know that the transfer function of the closed loop control system has unity negative feedback as,

C(s)R(s)=G(s)1+G(s)

Substitute, G(s)=1sT in the above equation.

C(s)R(s)=1sT1+1sT=1sT+1

The power of s is one in the denominator term. Hence, the above transfer function is of the first order and the system is said to be the first order system.

We can re-write the above equation as

C(s)=(1sT+1)R(s)

Where,

C(s) is the Laplace transform of the output signal c(t),

R(s) is the Laplace transform of the input signal r(t), and

T is the time constant.

Follow these steps to get the response (output) of the first order system in the time domain.

Take the Laplace transform of the input signal r(t).

Consider the equation, C(s)=(1sT+1)R(s)

Substitute R(s) value in the above equation.

Do partial fractions of C(s) if required.

Apply inverse Laplace transform to C(s).

In the previous chapter, we have seen the standard test signals like impulse, step, ramp and parabolic. Let us now find out the responses of the first order system for each input, one by one. The name of the response is given as per the name of the input signal. For example, the response of the system for an impulse input is called as impulse response.

Impulse Response of First Order System

Consider the unit impulse signal as an input to the first order system.

So, $r(t)=\delta(t)$

Apply Laplace transform on both the sides.

$R(s)=1$

Consider the equation, $C(s)=(1sT+1)R(s)$

Substitute, $R(s)=1$ in the above equation.

$C(s)=(1sT+1)(1)=1sT+1$

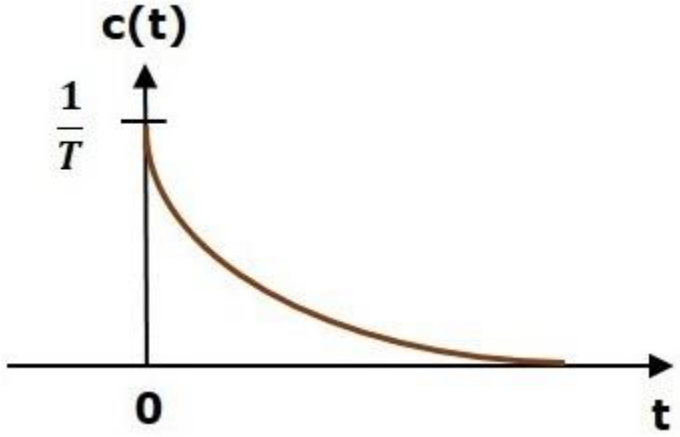
Rearrange the above equation in one of the standard forms of Laplace transforms.

$C(s)=1T(s+1T)\Rightarrow C(s)=1T(1s+1T)$

Apply inverse Laplace transform on both sides.

$c(t)=1Te(-tT)u(t)$

The unit impulse response is shown in the following figure.



The unit impulse response, c(t) is an exponential decaying signal for positive values of 't' and it is zero for negative values of 't'.

Step Response of First Order System

Consider the unit step signal as an input to first order system.

So, r(t)=u(t)

Apply Laplace transform on both the sides.

R(s)=1s

Consider the equation, C(s)=(1sT+1)R(s)

Substitute, R(s)=1s in the above equation.

C(s)=(1sT+1)(1s)=1s(sT+1)

Do partial fractions of C(s).

C(s)=1s(sT+1)=As+BsT+1

⇒1s(sT+1)=A(sT+1)+Bss(sT+1)

On both the sides, the denominator term is the same. So, they will get cancelled by each other. Hence, equate the numerator terms.

1=A(sT+1)+Bs

By equating the constant terms on both the sides, you will get A = 1.

Substitute, A = 1 and equate the coefficient of the s terms on both the sides.

0=T+B⇒B=−T

Substitute, A = 1 and B = −T in partial fraction expansion of C(s).

C(s)=1s−TsT+1=1s−TT(s+1T)

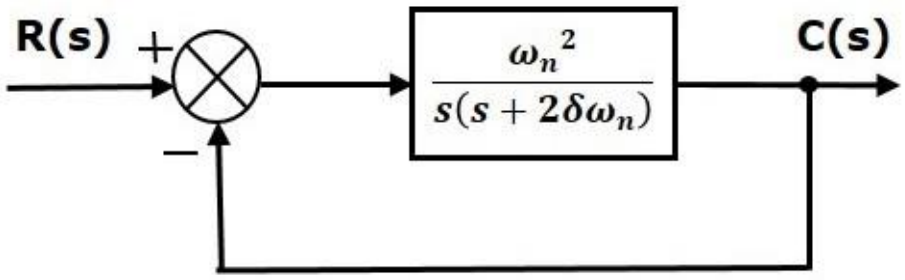
⇒C(s)=1s−1s+1T

Apply inverse Laplace transform on both the sides.

c(t)=(1−e−(tT))u(t)

Response of Second Order System

let us discuss the time response of second order system. Consider the following block diagram of closed loop control system. Here, an open loop transfer function, ω2ns(s+2δωn) is connected with a unity negative feedback



e know that the transfer function of the closed loop control system having unity negative feedback as

C(s)R(s)=G(s)1+G(s)

Substitute, G(s)=ω2ns(s+2δωn) in the above equation.

C(s)R(s)=(ω2ns(s+2δωn))1+(ω2ns(s+2δωn))=ω2ns2+2δωns+ω2n

The power of 's' is two in the denominator term. Hence, the above transfer function is of the second order and the system is said to be the second order system.

The characteristic equation is -

s2+2δωns+ω2n=0

The roots of characteristic equation are -

s=−2ωδn±(2δωn)2−4ω2n−−−−−−−−−−−√2=−2(δωn±ωnδ2−1−−−−−√)2

⇒s=−δωn±ωnδ2−1−−−−−√

The two roots are imaginary when δ = 0.

The two roots are real and equal when δ = 1.

The two roots are real but not equal when δ > 1.

The two roots are complex conjugate when 0 < δ < 1.

We can write C(s) equation as,

C(s)=(ω2ns2+2δωns+ω2n)R(s)

Where,

C(s) is the Laplace transform of the output signal, c(t)

R(s) is the Laplace transform of the input signal, r(t)

ωn is the natural frequency

δ is the damping ratio.

Follow these steps to get the response (output) of the second order system in the time domain.

Take Laplace transform of the input signal, r(t).

Consider the equation, $C(s)=(\omega 2ns2+2\delta\omega ns+\omega 2n)R(s)$

Substitute R(s) value in the above equation.

Do partial fractions of C(s) if required.

Apply inverse Laplace transform to C(s).

Step Response of Second Order System

Consider the unit step signal as an input to the second order system.

Laplace transform of the unit step signal is,

$R(s)=1s$

We know the transfer function of the second order closed loop control system is,

$C(s)R(s)=\omega 2ns2+2\delta\omega ns+\omega 2n$

Case 1: $\delta = 0$

Substitute, $\delta=0$ in the transfer function.

$C(s)R(s)=\omega 2ns2+\omega 2n$

$\Rightarrow C(s)=(\omega 2ns2+\omega 2n)R(s)$

Substitute, $R(s)=1s$ in the above equation.

$C(s)=(\omega 2ns2+\omega 2n)(1s)=\omega 2ns(s2+\omega 2n)$

Apply inverse Laplace transform on both the sides.

$c(t)=(1-\cos(\omega nt))u(t)$

So, the unit step response of the second order system when /delta=0 will be a continuous time signal with constant amplitude and frequency.

Case 2: $\delta = 1$

Substitute, /delta=1 in the transfer function.

$C(s)R(s)=\omega 2ns2+2\omega ns+\omega 2n$

$\Rightarrow C(s)=(\omega 2n(s+\omega n)2)R(s)$

Substitute, R(s)=1s in the above equation.

$C(s)=(\omega 2n(s+\omega n)2)(1s)=\omega 2ns(s+\omega n)2$

Do partial fractions of C(s).

$C(s)=\omega 2ns(s+\omega n)2=As+Bs+\omega n+C(s+\omega n)2$

After simplifying, you will get the values of A, B and C as $1,-1and-\omega n$ respectively. Substitute these values in the above partial fraction expansion of C(s).

$C(s)=1s-1s+\omega n-\omega n(s+\omega n)2$

Apply inverse Laplace transform on both the sides.

$c(t)=(1-e-\omega nt-\omega nte-\omega nt)u(t)$

So, the unit step response of the second order system will try to reach the step input in steady state.

Case 3: $0 < \delta < 1$

We can modify the denominator term of the transfer function as follows −

$s2+2\delta\omega ns+\omega 2n=\{s2+2(s)(\delta\omega n)+(\delta\omega n)2\}+\omega 2n-(\delta\omega n)2$

$=(s+\delta\omega n)2+\omega 2n(1-\delta 2)$

The transfer function becomes,

$C(s)R(s)=\omega 2n(s+\delta\omega n)2+\omega 2n(1-\delta 2)$

$\Rightarrow C(s)=(\omega 2n(s+\delta\omega n)2+\omega 2n(1-\delta 2))R(s)$

Substitute, R(s)=1s in the above equation.

$C(s)=(\omega 2n(s+\delta\omega n)2+\omega 2n(1-\delta 2))(1s)=\omega 2ns((s+\delta\omega n)2+\omega 2n(1-\delta 2))$

Do partial fractions of C(s).

$C(s)=\omega 2ns((s+\delta\omega n)2+\omega 2n(1-\delta 2))=As+Bs+C(s+\delta\omega n)2+\omega 2n(1-\delta 2)$

After simplifying, you will get the values of A, B and C as $1,-1and-2\delta\omega n$ respectively. Substitute these values in the above partial fraction expansion of C(s).

$C(s)=1s-s+2\delta\omega n(s+\delta\omega n)2+\omega 2n(1-\delta 2)$

$C(s)=1s-s+\delta\omega n(s+\delta\omega n)2+\omega 2n(1-\delta 2)-\delta\omega n(s+\delta\omega n)2+\omega 2n(1-\delta 2)$

$C(s)=1s-(s+\delta\omega n)(s+\delta\omega n)2+(\omega n1-\delta 2\surd)2-\delta 1-\delta 2\surd(\omega n1-\delta 2\surd(s+\delta\omega n)2+(\omega n1-\delta 2\surd)2)$

Substitute, $\omega n1-\delta 2------\surd$ as ωd in the above equation.

$C(s)=1s-(s+\delta\omega n)(s+\delta\omega n)2+\omega 2d-\delta 1-\delta 2------\surd(\omega d(s+\delta\omega n)2+\omega 2d)$

Apply inverse Laplace transform on both the sides.

$c(t)=(1-e-\delta\omega ntcos(\omega dt)-\delta 1-\delta 2------\surd e-\delta\omega ntsin(\omega dt))u(t)$

$c(t)=(1-e-\delta\omega nt1-\delta 2------\surd((1-\delta 2------\surd)cos(\omega dt)+\delta sin(\omega dt)))u(t)$

If $1-\delta 2------\surd=sin(\theta)$, then ‘δ’ will be $cos(\theta)$. Substitute these values in the above equation.

$c(t)=(1-e-\delta\omega nt1-\delta 2------\surd(sin(\theta)cos(\omega dt)+cos(\theta)sin(\omega dt)))u(t)$

$\Rightarrow c(t)=(1-(e-\delta\omega nt1-\delta 2------\surd)sin(\omega dt+\theta))u(t)$

So, the unit step response of the second order system is having damped oscillations (decreasing amplitude) when ‘δ’ lies between zero and one.

Case 4: $\delta > 1$

We can modify the denominator term of the transfer function as follows −

$s2+2\delta\omega ns+\omega 2n=\{s2+2(s)(\delta\omega n)+(\delta\omega n)2\}+\omega 2n-(\delta\omega n)2$

$=(s+\delta\omega n)2-\omega 2n(\delta 2-1)$

The transfer function becomes,

$C(s)R(s)=\omega 2n(s+\delta\omega n)2-\omega 2n(\delta 2-1)$

$\Rightarrow C(s)=(\omega 2n(s+\delta\omega n)2-\omega 2n(\delta 2-1))R(s)$

Substitute, $R(s)=1s$ in the above equation.

$C(s)=(\omega 2n(s+\delta\omega n)2-(\omega n\delta 2-1\surd)2)(1s)=\omega 2ns(s+\delta\omega n+\omega n\delta 2-1\surd)(s+\delta\omega n-\omega n\delta 2-1\surd)$

Do partial fractions of C(s).

$C(s)=\omega 2ns(s+\delta\omega n+\omega n\delta 2-1\sqrt{})(s+\delta\omega n-\omega n\delta 2-1\sqrt{})$

$=As+Bs+\delta\omega n+\omega n\delta 2-1\sqrt{}+Cs+\delta\omega n-\omega n\delta 2-1\sqrt{}$

After simplifying, you will get the values of A, B and C as 1, $12(\delta+\delta 2-1\sqrt{})(\delta 2-1\sqrt{})$ and $-12(\delta-\delta 2-1\sqrt{})(\delta 2-1\sqrt{})$ respectively. Substitute these values in above partial fraction expansion of C(s).

$C(s)=1s+12(\delta+\delta 2-1\sqrt{})(\delta 2-1\sqrt{})(1s+\delta\omega n+\omega n\delta 2-1\sqrt{})-(12(\delta-\delta 2-1\sqrt{})(\delta 2-1\sqrt{}))(1s+\delta\omega n-\omega n\delta 2-1\sqrt{})$

Apply inverse Laplace transform on both the sides.

$c(t)=(1+(12(\delta+\delta 2-1\sqrt{})(\delta 2-1\sqrt{}))e-(\delta\omega n+\omega n\delta 2-1\sqrt{})t-(12(\delta-\delta 2-1\sqrt{})(\delta 2-1\sqrt{}))e-(\delta\omega n-\omega n\delta 2-1\sqrt{})t)u(t)$

Since it is over damped, the unit step response of the second order system when $\delta > 1$ will never reach step input in the steady state.

Impulse Response of Second Order System

The impulse response of the second order system can be obtained by using any one of these two methods.

Follow the procedure involved while deriving step response by considering the value of R(s) as 1 instead of 1s.

Do the differentiation of the step response.

The following table shows the impulse response of the second order system for 4 cases of the damping ratio.

Construction of Root Locus The root locus is a graphical representation in s-domain and it is symmetrical about the real axis. Because the open loop poles and zeros exist in the s-domain having the values either as real or as complex conjugate pairs. In this chapter, let us discuss how to construct (draw) the root locus.

Rules for Construction of Root Locus

Follow these rules for constructing a root locus.

Rule 1 − Locate the open loop poles and zeros in the ‘s’ plane.

Rule 2 − Find the number of root locus branches.

We know that the root locus branches start at the open loop poles and end at open loop zeros. So,

the number of root locus branches N is equal to the number of finite open loop poles P or the number of finite open loop zeros Z, whichever is greater.

Mathematically, we can write the number of root locus branches N as

$N=P$ if $P \geq Z$

$N=Z$ if $P<Z$

Rule 3 − Identify and draw the real axis root locus branches.

If the angle of the open loop transfer function at a point is an odd multiple of 1800, then that point is on the root locus. If odd number of the open loop poles and zeros exist to the left side of a point on the real axis, then that point is on the root locus branch. Therefore, the branch of points which satisfies this condition is the real axis of the root locus branch.

Rule 4 − Find the centroid and the angle of asymptotes.

If P=Z, then all the root locus branches start at finite open loop poles and end at finite open loop zeros.

If P>Z , then Z number of root locus branches start at finite open loop poles and end at finite open loop zeros and P−Z number of root locus branches start at finite open loop poles and end at infinite open loop zeros.

If P<Z , then P number of root locus branches start at finite open loop poles and end at finite open loop zeros and Z−P number of root locus branches start at infinite open loop poles and end at finite open loop zeros.

So, some of the root locus branches approach infinity, when $P \neq Z$. Asymptotes give the direction of these root locus branches. The intersection point of asymptotes on the real axis is known as centroid.

We can calculate the centroid α by using this formula,

$$\alpha=\frac{\sum Real\,part\,of\,finite\,open\,loop\,poles-\sum Real\,part\,of\,finite\,open\,loop\,zeros}{P-Z}$$

The formula for the angle of asymptotes θ is

$$\theta=\frac{(2q+1)1800}{P-Z}$$

Where,

$q=0,1,2,....,(P-Z)-1$

Rule 5 − Find the intersection points of root locus branches with an imaginary axis.

We can calculate the point at which the root locus branch intersects the imaginary axis and the

value of K at that point by using the Routh array method and special case (ii).

If all elements of any row of the Routh array are zero, then the root locus branch intersects the imaginary axis and vice-versa.

Identify the row in such a way that if we make the first element as zero, then the elements of the entire row are zero. Find the value of K for this combination.

Substitute this K value in the auxiliary equation. You will get the intersection point of the root locus branch with an imaginary axis.

Rule 6 − Find Break-away and Break-in points.

If there exists a real axis root locus branch between two open loop poles, then there will be a break-away point in between these two open loop poles.

If there exists a real axis root locus branch between two open loop zeros, then there will be a break-in point in between these two open loop zeros.

Note − Break-away and break-in points exist only on the real axis root locus branches.

Follow these steps to find break-away and break-in points.

Write K in terms of s from the characteristic equation 1+G(s)H(s)=0.

Differentiate K with respect to s and make it equal to zero. Substitute these values of s in the above equation.

The values of s for which the K value is positive are the break points.

Rule 7 − Find the angle of departure and the angle of arrival.

The Angle of departure and the angle of arrival can be calculated at complex conjugate open loop poles and complex conjugate open loop zeros respectively.

The formula for the angle of departure ϕd is

$$\phi d=1800-\phi$$

The formula for the angle of arrival ϕa is

$$\phi a=1800+\phi$$

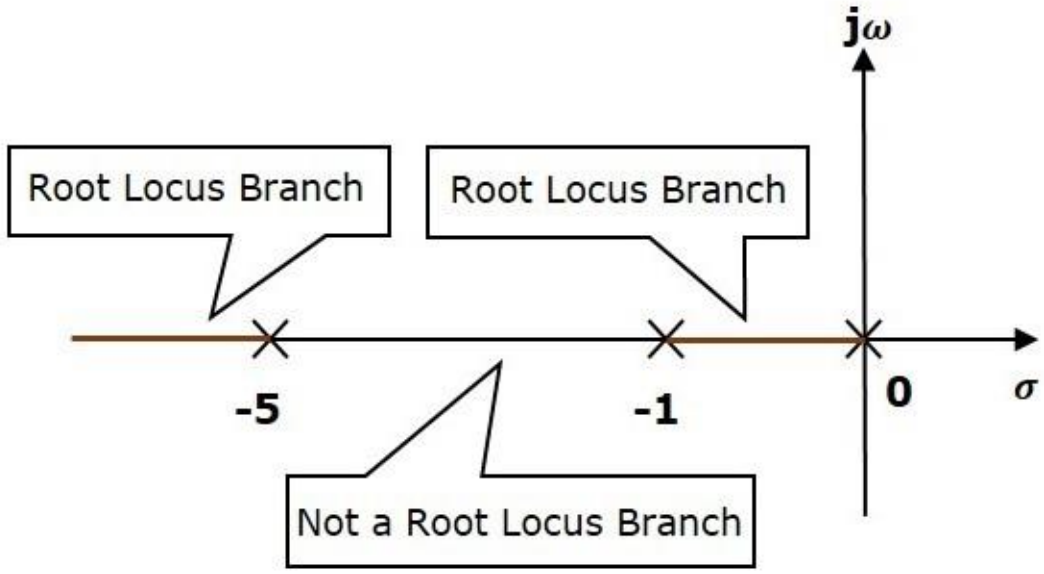
Where,

$$\phi=\sum\phi P-\sum\phi Z$$

Example

Let us now draw the root locus of the control system having open loop transfer function, G(s)H(s)=Ks(s+1)(s+5)

Step 1 − The given open loop transfer function has three poles at s=0,s=−1 and s=−5. It doesn't have any zero. Therefore, the number of root locus branches is equal to the number of poles of the open loop transfer function.



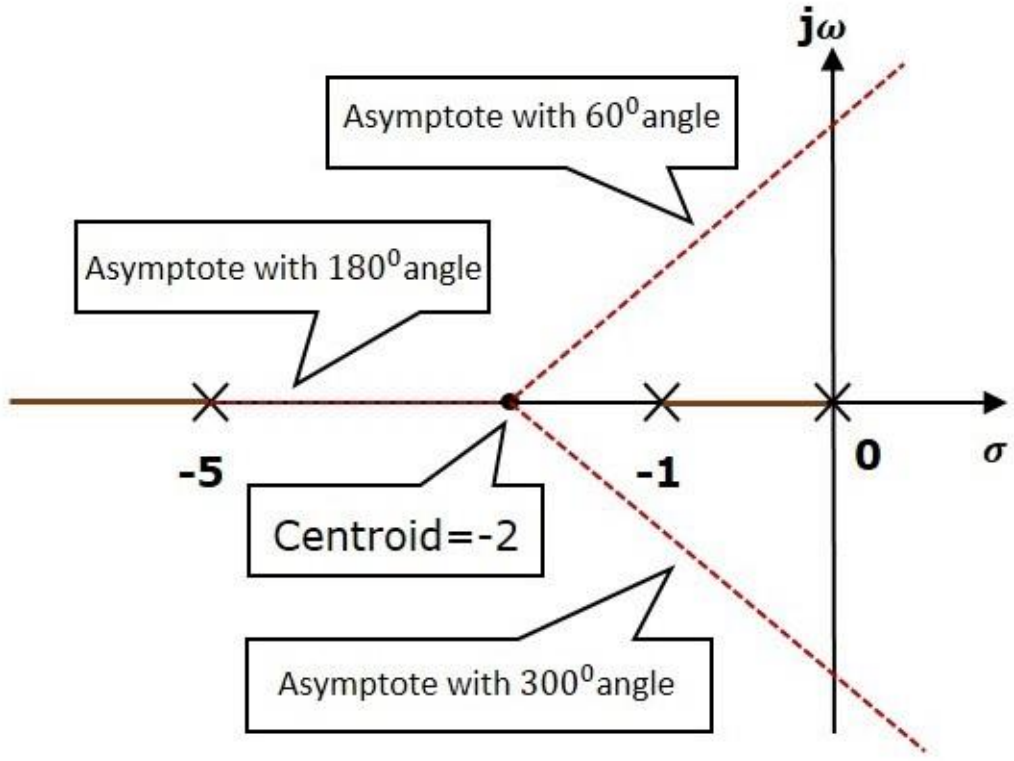
he three poles are located are shown in the above figure. The line segment between s=−1 and s=0 is one branch of root locus on real axis. And the other branch of the root locus on the real axis is the line segment to the left of s=−5.

Step 2 − We will get the values of the centroid and the angle of asymptotes by using the given formulae.

Centroid α=−2

The angle of asymptotes are θ=600,1800 and 3000.

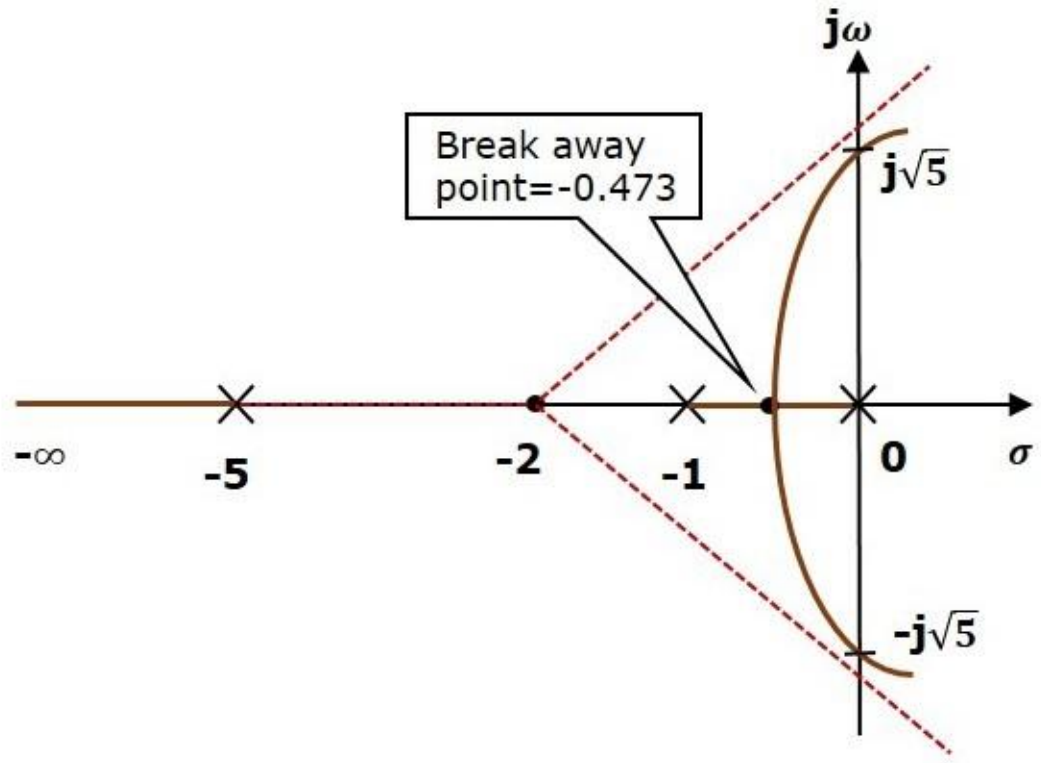
The centroid and three asymptotes are shown in the following figure.



Step 3 − Since two asymptotes have the angles of 600 and 3000, two root locus branches intersect the imaginary axis. By using the Routh array method and special case(ii), the root locus branches intersects the imaginary axis at j5–√ and −j5–√.

There will be one break-away point on the real axis root locus branch between the poles s=−1 and s=0. By following the procedure given for the calculation of break-away point, we will get it as s=−0.473.

The root locus diagram for the given control system is shown in the following figure.



n this way, you can draw the root locus diagram of any control system and observe the movement of poles of the closed loop transfer function.

From the root locus diagrams, we can know the range of K values for different types of damping.

Effects of Adding Open Loop Poles and Zeros on Root Locus

The root locus can be shifted in 's' plane by adding the open loop poles and the open loop zeros.

If we include a pole in the open loop transfer function, then some of root locus branches will move towards right half of 's' plane. Because of this, the damping ratio δ decreases. Which implies, damped frequency ωd increases and the time domain specifications like delay time td, rise time tr and peak time tp decrease. But, it effects the system stability.

If we include a zero in the open loop transfer function, then some of root locus branches will move towards left half of 's' plane. So, it will increase the control system stability. In this case, the damping ratio δ increases. Which implies, damped frequency ωd decreases and the time domain specifications like delay time td, rise time tr and peak time tp increase.

So, based on the requirement, we can include (add) the open loop poles or zeros to the transfer function.

Control Systems - Controllers

Proportional Controller

The proportional controller produces an output, which is proportional to error signal.

$u(t) \propto e(t)$

$\Rightarrow u(t) = KPe(t)$

Apply Laplace transform on both the sides -

$U(s) = KPE(s)$

$U(s)E(s) = KP$

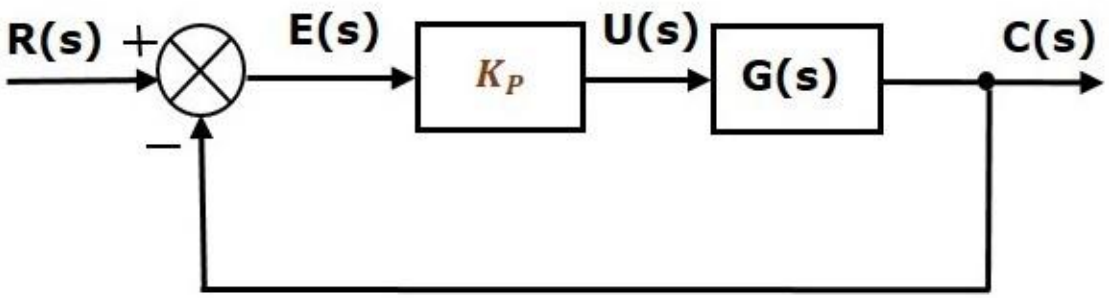
Therefore, the transfer function of the proportional controller is KP.

Where,

U(s) is the Laplace transform of the actuating signal u(t)

E(s) is the Laplace transform of the error signal e(t)

KP is the proportionality constant



the block diagram of the unity negative feedback closed loop control system along with the proportional controller is shown in the following figure**.**

Derivative Controller

The derivative controller produces an output, which is derivative of the error signal.

u(t)=KDde(t)dt

Apply Laplace transform on both sides.

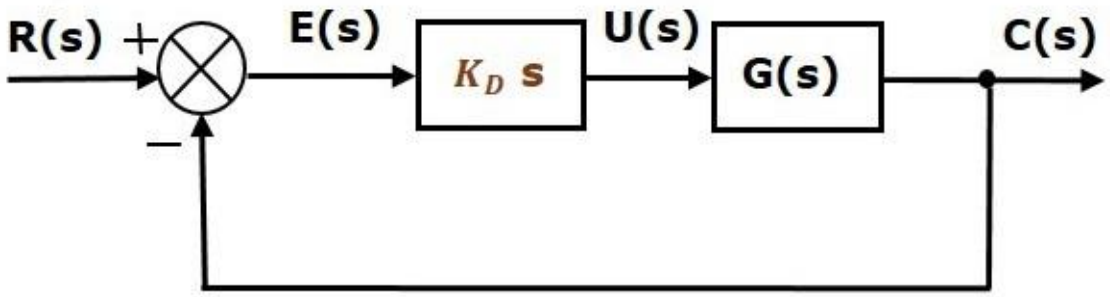
U(s)=KDsE(s)

U(s)E(s)=KDs

Therefore, the transfer function of the derivative controller is KDs.

Where, KD is the derivative constant.

The block diagram of the unity negative feedback closed loop control system along with the derivative controller is shown in the following figure.



Integral Controller

The integral controller produces an output, which is integral of the error signal.

u(t)=KI∫e(t)dt

Apply Laplace transform on both the sides -

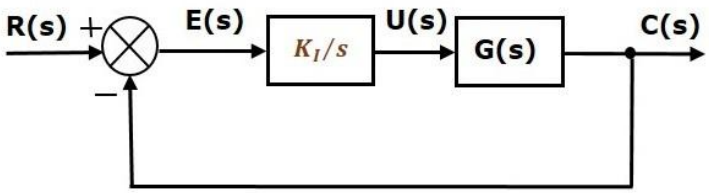
U(s)=KIE(s)s

U(s)E(s)=KIs

Therefore, the transfer function of the integral controller is KIs.

Where, KI is the integral constant.

The block diagram of the unity negative feedback closed loop control system along with the integral controller is shown in the following figure.


R(s)
+
−
E(s)
K_I/s
U(s)
G(s)
C(s)

UNIT 3

Sensors and Actuators: Static characteristics of sensors and actuators, Position, Displacement and Proximity Sensors, Force and torque sensors, Pressure sensors, Flow sensors, Temperature sensors, Acceleration sensors, Level sensors, Light sensors, Smart material sensors, Micro and Nano sensors, Selection criteria for sensors, Actuators: Electrical Actuators (Solenoids, Relays, Diodes, Thyristors, Triacs, BJT, FET, DC motor, Servo motor, BLDC motor, AC motor, Stepper motors), Hydraulic and Pneumatic actuators, *Design of Hydraulic and Pneumatic circuits, Piezoelectric actuators, Shape memory alloys.*

UNIT -3

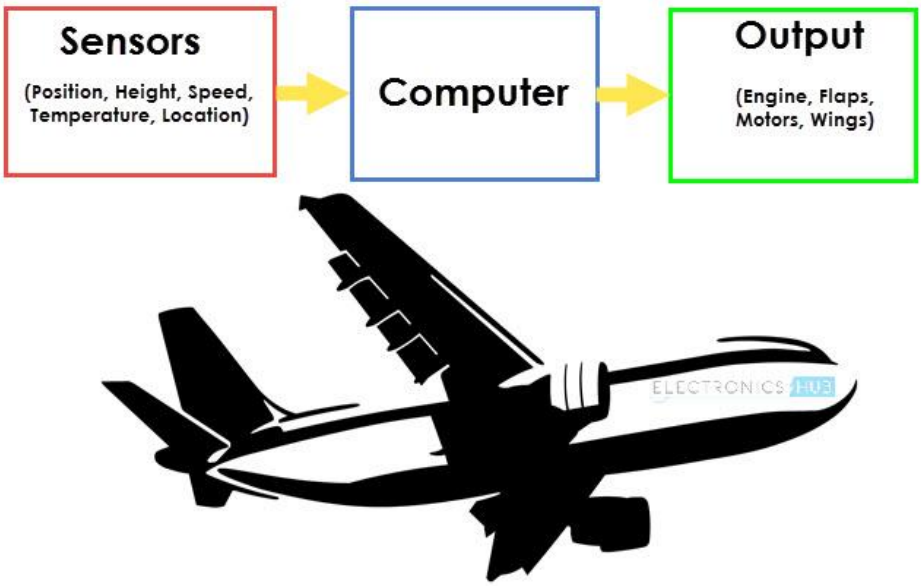
Sensors

We live in a World of Sensors. You can find different types of Sensors in our homes, offices, cars etc. working to make our lives easier by turning on the lights by detecting our presence, adjusting the room temperature, detect smoke or fire, make us delicious coffee, open garage doors as soon as our car is near the door and many other tasks.All these and many other automation tasks are possible because of Sensors. Before going in to the details of What is a Sensor, What are the Different Types of Sensors and Applications of these different types of Sensors, we will first take a look at a simple example of an automated system, which is possible because of Sensors (and many other components as well).

Real Time Application of Sensors

The example we are talking about here is the Autopilot System in aircrafts. Almost all civilian and military aircrafts have the feature of Automatic Flight Control system or sometimes called as

Autopilot.



An Automatic Flight Control System consists of several sensors for various tasks like speed control, height, position, doors, obstacle, fuel, maneuvering and many more. A Computer takes data from all these sensors and processes them by comparing them with pre-designed values.

The computer then provides control signal to different parts like engines, flaps, rudders etc. that help in a smooth flight. The combination of Sensors, Computers and Mechanics makes it possible to run the plane in Autopilot Mode.

All the parameters i.e. the Sensors (which give inputs to the Computers), the Computers (the brains of the system) and the mechanics (the outputs of the system like engines and motors) are equally important in building a successful automated system.

But in this tutorial, we will be concentrating on the Sensors part of a system and look at different concepts associated with Sensors (like types, characteristics, classification etc.).

What is a Sensor?

There are numerous definitions as to what a sensor is but I would like to define a Sensor as an input device which provides an output (signal) with respect to a specific physical quantity (input).

The term “input device” in the definition of a Sensor means that it is part of a bigger system which provides input to a main control system (like a Processor or a Microcontroller).

Another unique definition of a Sensor is as follows: It is a device that converts signals from one energy domain to electrical domain. The definition of the Sensor can be understood if we take an

example in to consideration.

Classification of Sensors

There are several classifications of sensors made by different authors and experts. Some are very simple and some are very complex. The following classification of sensors may already be used by an expert in the subject but this is a very simple classification of sensors.

In the first classification of the sensors, they are divided in to Active and Passive. Active Sensors are those which require an external excitation signal or a power signal.

Passive Sensors, on the other hand, do not require any external power signal and directly generates output response.

The other type of classification is based on the means of detection used in the sensor. Some of the means of detection are Electric, Biological, Chemical, Radioactive etc.

The next classification is based on conversion phenomenon i.e. the input and the output. Some of the common conversion phenomena are Photoelectric, Thermoelectric, Electrochemical, Electromagnetic, Thermooptic, etc.

The final classification of the sensors are Analog and Digital Sensors. Analog Sensors produce an analog output i.e. a continuous output signal with respect to the quantity being measured.

Digital Sensors, in contrast to Analog Sensors, work with discrete or digital data. The data in digital sensors, which is used for conversion and transmission, is digital in nature.

Different Types of Sensors

The following is a list of different types of sensors that are commonly used in various applications. All these sensors are used for measuring one of the physical properties like Temperature, Resistance, Capacitance, Conduction, Heat Transfer etc.

Temperature Sensor

Proximity Sensor

Accelerometer

IR Sensor (Infrared Sensor)

Pressure Sensor

Light Sensor

Ultrasonic Sensor

Smoke, Gas and Alcohol Sensor

Touch Sensor

Color Sensor

Humidity Sensor

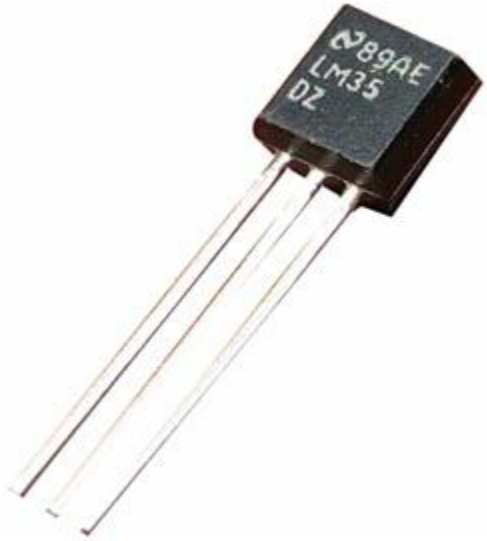
Tilt Sensor

Flow and Level Sensor

We will see about few of the above mentioned sensors in brief. More information about the sensors will be added subsequently. A list of projects using the above sensors is given at the end of the page.

Temperature Sensor

One of the most common and most popular sensor is the Temperature Sensor. A Temperature Sensor, as the name suggests, senses the temperature i.e. it measures the changes in the temperature.



LM35 - Temperature Sensor IC

10KΩ NTC Thermistor

Proximity Sensors

A Proximity Sensor is a non-contact type sensor that detects the presence of an object. Proximity Sensors can be implemented using different techniques like Optical (like Infrared or Laser), Ultrasonic, Hall Effect, Capacitive, etc.

Some of the applications of Proximity Sensors are Mobile Phones, Cars (Parking Sensors), industries (object alignment), Ground Proximity in Aircrafts, etc.

Proximity Sensor in Reverse Parking is implemented in this Project: REVERSE PARKING SENSOR CIRCUIT.

Infrared Sensor (IR Sensor)

IR Sensors or Infrared Sensor are light based sensor that are used in various applications like Proximity and Object Detection. IR Sensors are used as proximity sensors in almost all mobile phones.

There are two types of Infrared or IR Sensors: Transmissive Type and Reflective Type. In Transmissive Type IR Sensor, the IR Transmitter (usually an IR LED) and the IR Detector (usually a Photo Diode) are positioned facing each other so that when an object passes between them, the sensor detects the object.

The other type of IR Sensor is a Reflective Type IR Sensor. In this, the transmitter and the detector are positioned adjacent to each other facing the object. When an object comes in front of the sensor, the sensor detects the object.

Different applications where IR Sensor is implemented are Mobile Phones, Robots, Industrial assembly, automobiles etc.

A small project, where IR Sensors are used to turn on street lights: STREET LIGHTS USING IR SENSORS.

Ultrasonic Sensor

An Ultrasonic Sensor is a non-contact type device that can be used to measure distance as well as velocity of an object. An Ultrasonic Sensor works based on the properties of the sound waves with frequency greater than that of the human audible range

Using the time of flight of the sound wave, an Ultrasonic Sensor can measure the distance of the object (similar to SONAR). The Doppler Shift property of the sound wave is used to measure the velocity of an object.

Arduino based Range Finder is a simple project using Ultrasonic Sensor: PORTABLE ULTRASONIC RANGE METER.

The following is a small list of projects based on few of the above mentioned Sensors.

Light Sensor – LIGHT DETECTOR USING LDR

Smoke Sensor – SMOKE DETECTOR ALARM CIRCUIT

Alcohol Sensor – HOW TO MAKE ALCOHOL BREATHALYZER CIRCUIT?

Touch Sensor – TOUCH DIMMER SWITCH CIRCUIT USING ARDUINO

Color Sensor – ARDUINO BASED COLOR DETECTOR

Humidity Sensor – DHT11 HUMIDITY SENSOR ON ARDUINO

Tilt Sensor – HOW TO MAKE A TILT SENSOR WITH ARDUINO?

In this article, we have seen about What is a Sensor, what are the classification of sensors and different Types of Sensors along with their practical applications.

Actuator

An actuator is a component of a machine that is responsible for moving and controlling a mechanism or system, for example by opening a valve. In simple terms, it is a "mover".

An actuator requires a control signal and a source of energy. The control signal is relatively low energy and may be electric voltage or current, pneumatic, or hydraulic fluid pressure, or even human power. Its main energy source may be an electric current, hydraulic pressure, or pneumatic pressure. When it receives a control signal, an actuator responds by converting the source's energy into mechanical motion. In the electric, hydraulic, and pneumatic sense, it is a form of automation or automatic control.

An actuator is a mechanism by which a control system acts upon to perform an operation or task. The control system can be simple (a fixed mechanical or electronic system), software-based (e.g. a printer driver, robot control system), a human, or any other input.

Types of actuators

Hydraulic

Main article: Hydraulic actuator

The hydraulic actuator consists of cylinder or fluid motor that uses hydraulic power to facilitate mechanical operation. The mechanical motion gives an output in terms of linear, rotatory or oscillatory motion. As liquids are nearly impossible to compress, a hydraulic actuator can exert a large force. The drawback of this approach is its limited acceleration.

The hydraulic cylinder consists of a hollow cylindrical tube along which a piston can slide. The term single acting is used when the fluid pressure is applied to just one side of the piston. The piston can move in only one direction, a spring being frequently used to give the piston a return stroke. The term double acting is used when pressure is applied on each side of the piston; any difference in force between the two sides of the piston moves the piston to one side or the other.

Pneumatic

Pneumatic actuators enable considerable forces to be produced from relatively small pressure changes. Pneumatic energy is desirable for main engine controls because it can quickly respond in starting and stopping as the power source does not need to be stored in reserve for operation. Moreover, pneumatic actuators are cheaper, and often more powerful than other actuators. These forces are often used with valves to move diaphragms to affect the flow of air through the valve.

The advantage of pneumatic actuators consists exactly in the high level of force available in a relatively small volume. While the main drawback of the technology consists in the need for a compressed air network composed of several components such as compressors, reservoirs, filters, dryers, air treatment subsystems, valves, tubes, etc. which makes the technology energy inefficient with energy losses that can sum up to 95%

Electric

Since 1960, several actuator technologies have been developed, Electric actuators can be classified in the following groups:

Electromechanical Actuator

It converts the rotational force of an electric rotary motor into a linear movement to generate the requested linear movement through a mechanism either a belt (Belt Drive axis with stepper or servo) or a screw (either a ball or a lead screw or planetary mechanics)

The main advantages of electromechanical actuators are their relatively good level of accuracy respect to pneumatics, their possible long lifecycle and the little maintenance effort required (might require grease). It is possible to reach relatively high force, until order of 100 kN.

The main limitation of these actuators are the reachable speed, the important dimensions and weight they require.

Electrohydraulic Actuator

Another approach is an electrohydraulic actuator, where the electric motor remains the prime mover but provides torque to operate a hydraulic accumulator that is then used to transmit actuation force in much the same way that diesel engine/hydraulics are typically used in heavy equipment.

Electrical energy is used to actuate equipment such as multi-turn valves, or electric-powered construction and excavation equipment.

When used to control the flow of fluid through a valve, a brake is typically installed above the motor to prevent the fluid pressure from forcing open the valve. If no brake is installed, the actuator gets activated to reclose the valve, which is slowly forced open again. This sets up an oscillation (open, close, open ...) and the motor and actuator will eventually become damaged.

Stepper motor

A stepper motor, also known as step motor or stepping motor, is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any position sensor for feedback (an open-loop controller), as long as the motor is carefully sized to the application in respect to torque and speed. Switched reluctance motors are very large stepping motors with a reduced pole count, and generally are closed-loop commutated.

Fundamentals of operation

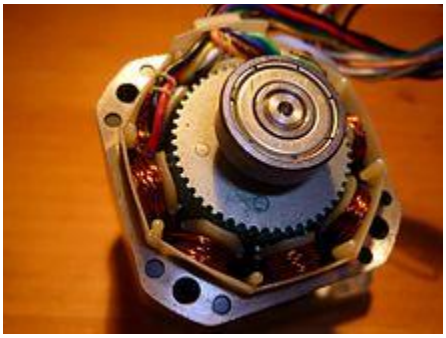
Brushed DC motors rotate continuously when DC voltage is applied to their terminals. The stepper motor is known for its property of converting a train of input pulses (typically square waves) into a precisely defined increment in the shaft’s rotational position. Each pulse rotates the shaft through a fixed angle.

Stepper motors effectively have multiple "toothed" electromagnets arranged as a stator around a central rotor, a gear-shaped piece of iron. The electromagnets are energized by an external driver circuit or a micro controller. To make the motor shaft turn, first, one electromagnet is given power, which magnetically attracts the gear's teeth. When the gear's teeth are aligned to the first electromagnet, they are slightly offset from the next electromagnet. This means that when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one. From there the process is repeated. Each of those rotations is called a "step", with an integer number of steps making a full rotation. In that way, the motor can be turned by a precise angle.

The circular arrangement of electromagnets is divided into groups, each group called a phase, and there is an equal number of electromagnets per group. The number of groups is chosen by

the designer of the stepper motor. The electromagnets of each group are interleaved with the electromagnets of other groups to form a uniform pattern of arrangement. For example, if the stepper motor has two groups identified as A or B, and ten electromagnets in total, then the grouping pattern would be ABABABABAB.

Electromagnets within the same group are all energized together. Because of this, stepper motors with more phases typically have more wires (or leads) to control the



AC motor

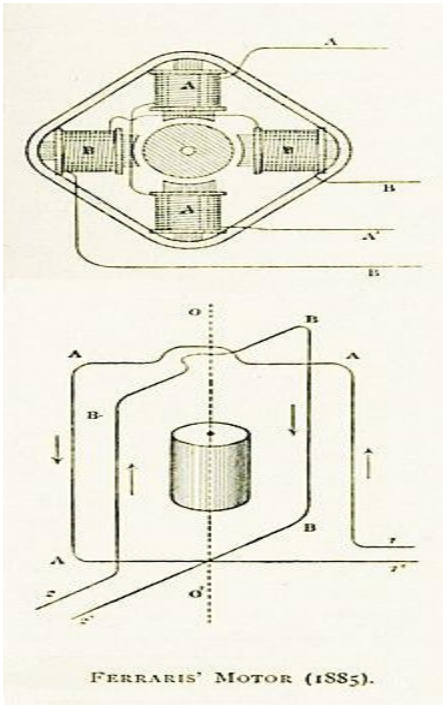
An AC motor is an electric motor driven by an alternating current (AC). The AC motor commonly consists of two basic parts, an outside stator having coils supplied with alternating current to produce a rotating magnetic field, and an inside rotor attached to the output shaft producing a second rotating magnetic field. The rotor magnetic field may be produced by permanent magnets, reluctance saliency, or DC or AC electrical windings.

Less common, AC linear motors operate on similar principles as rotating motors but have their stationary and moving parts arranged in a straight line configuration, producing linear motion instead of rotation.

Operating principles

The two main types of AC motors are induction motors and synchronous motors. The induction motor (or asynchronous motor) always relies on a small difference in speed between the stator rotating magnetic field and the rotor shaft speed called slip to induce rotor current in the rotor AC winding. As a result, the induction motor cannot produce torque near synchronous speed where induction (or slip) is irrelevant or ceases to exist. In contrast, the synchronous motor does not rely on slip-induction for operation and uses either permanent magnets, salient poles (having projecting magnetic poles), or an independently excited rotor winding. The synchronous motor produces its rated torque at exactly synchronous speed. The brushless wound-rotor doubly fed synchronous motor system has an independently excited rotor winding that does not rely on the principles of slip-induction of current. The brushless wound-rotor doubly fed motor is a synchronous motor that can function exactly at the supply frequency or sub to super multiple of the supply frequency.

Other types of motors include eddy current motors, and AC and DC mechanically commutated machines in which speed is dependent on voltage and winding connection.



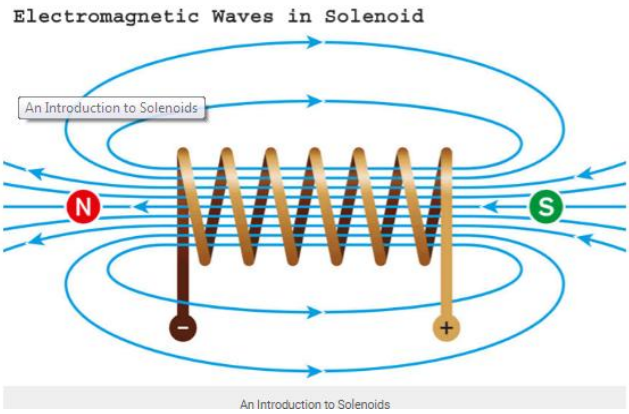
FERRARIS' MOTOR (1885).

Solenoid?

What is a solenoid?

Solenoid is the generic term for a coil of wire used as an electromagnet. It also refers to any device that converts electrical energy to mechanical energy using a solenoid. The device creates a magnetic field from electric current and uses the magnetic field to create linear motion. Common applications of solenoids are to power a switch, like the starter in an automobile, or a valve, such as in a sprinkler system.

A solenoid is a device comprised of a coil of wire, the housing and a moveable plunger (armature). When an electrical current is introduced, a magnetic field forms around the coil which draws the plunger in. More simply, a solenoid converts electrical energy into mechanical work.



The coil is made of many turns of tightly wound copper wire. When an electrical current flows through this wire, a strong magnetic field/flux is created.

The housing, usually made of iron or steel, surrounds the coil concentrating the magnetic field generated by the coil.

The plunger is attracted to the stop through the concentration of the magnetic field providing the mechanical force to do work.

How a Solenoid Works

A solenoid is a coil of wire in a corkscrew shape wrapped around a piston, often made of iron.

As in all electromagnets, a magnetic field is created when an electric current passes through the wire. Electromagnets have an advantage over permanent magnets in that they can be switched on and off by the application or removal of the electric current, which is what makes them useful as switches and valves and allows them to be entirely automated.

Like all magnets, the magnetic field of an activated solenoid has positive and negative poles that will attract or repel material sensitive to magnets. In a solenoid, the electromagnetic field causes the piston to either move backward or forward, which is how motion is created by a solenoid coil.

How Does a Solenoid Valve Work?

In a direct-acting valve, electric current activates the solenoid, which in turn pulls a piston or plunger that would otherwise block air or fluid from flowing. In some solenoid valves, the electromagnetic field does not act directly to open the conduit. In pilot-operated valves, a solenoid moves the plunger, which creates a small opening, and pressure through the opening is what operates the valve seal. In both types, solenoid valves require a constant flow of electrical current to remain open because once the current is stopped, the electromagnetic field disperses and the valve returns to its original closed position.

Electric Solenoids

In an automobile ignition system, the starter solenoid acts as a relay, bringing metal contacts into place to close a circuit. The starter solenoid receives a small electric current when the car's ignition is activated, usually by the turn of the key. The magnetic field of the solenoid then pulls on the contacts, closing the circuit between the car's battery and the starter motor. The starter solenoid requires a constant flow of electricity in order to maintain the circuit, but because the engine is self-powering once started, the solenoid is inactive for most of the time.

Uses for Solenoids

Solenoids are incredibly versatile and extremely useful. They're found in everything from automated factory equipment to paintball guns and even doorbells. In a chime doorbell, the audible chime is produced when a metal piston strikes a tone bar. The force that moves the

piston is the magnetic field of a solenoid that receives electric current when the doorbell is pushed.

Relay

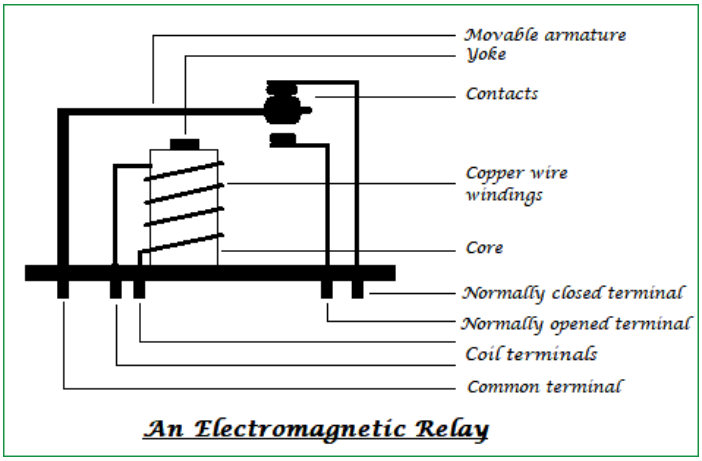
What is Relay?

A Relay is an electromechanical device that can be used to make or break an electrical connection. It consists of a flexible moving mechanical part which can be controlled electronically through an electromagnet, basically, a relay is just like a mechanical switch but you can control it with an electronic signal instead of manually turning it on or off. Again this working principle of relay fits only for the electromechanical relay.

There are many types of relay and each relay has its own application, a standard, and generally used relay is made up of electromagnets which in general used as a switch. Dictionary says that relay means the act of passing something from one thing to another, the same meaning can be applied to this device because the signal received from one side of the device controls the switching operation on the other side. So relay is a switch which controls (open and close) circuits electromechanically. The main operation of this device is to make or break 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. Generally, a DC signal is used to control the circuit which is driven by high voltage like controlling AC home appliances with DC signals from microcontrollers.

Construction of Relay and its operation:

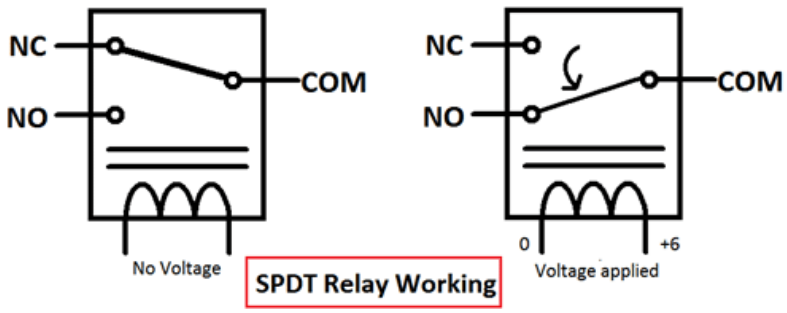
The following figure shows how a Relay looks internally and how it can be constructed,



An Electromagnetic Relay

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. The relay also has two pins namely normally closed and normally opened (NC and 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.

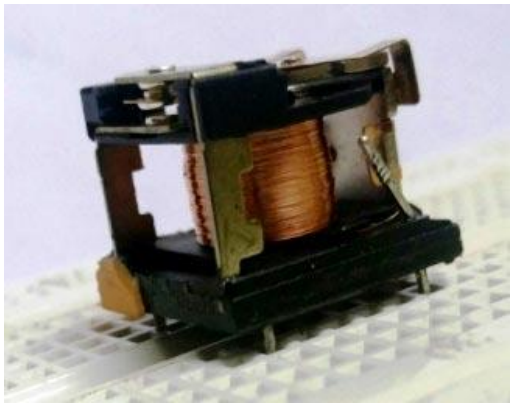
The general circuit representation of the relay is as shown in the figure below



SPDT Relay Working

What is inside a Relay -

An electromechanical relay is basically designed using few mechanical parts like Electromagnet, a movable armature, contacts, yoke, and a spring/frame/stand, these parts are showing in the internal pictures of Relay below. All these are arranged logically to form into a relay.



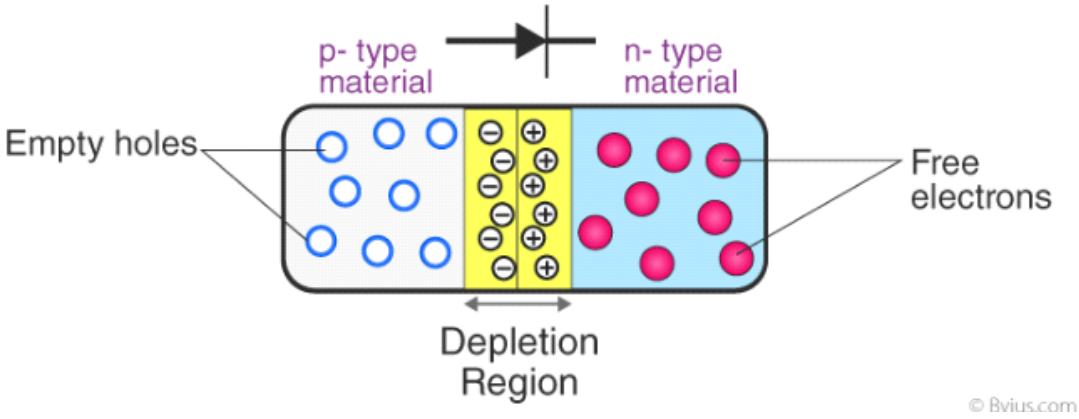
Here we have explained the internal mechanical parts of a Relay:

Electromagnet: 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 and can attract the metals within its range.

P-N Junction

Definition: ***A p-n junction is an interface or a boundary between two semiconductor material types, namely the p-type and the n-type, inside a semiconductor.***

The p-side or the positive side of the semiconductor has an excess of holes and the n-side or the negative side has an excess of electrons. In a semiconductor, the p-n junction is created by the method of doping. The process of doping is explained in further detail in the next section.



Formation of P-N Junction :

As we know if we use different semiconductor materials to make a p-n junction, there will be a grain boundary that would inhibit the movement of electrons from one side to the other by scattering the electrons and holes and thus we use the process of doping. We will understand the process of doping with the help of this example. Let us consider a thin p-type silicon semiconductor sheet. If we add a small amount of pentavalent impurity to this, a part of the p-type Si will get converted to n-type silicon. This sheet will now contain both p-type region and n-type region and a junction between these two regions. The processes that follow after the formation of a p-n junction are of two types – diffusion and drift. As we know, there is a difference in the concentration of holes and electrons at the two sides of a junction, the holes from the p-side diffuse to the n-side and the electrons from the n-side diffuse to the p-side. These give rise to a diffusion current across the junction.

Also, when an electron diffuses from the n-side to the p-side, an ionized donor is left behind on the n-side, which is immobile. As the process goes on, a layer of positive charge is developed on the n-side of the junction. Similarly, when a hole goes from the p-side to the n-side, and ionized acceptor is left behind in the p-side, resulting in the formation of a layer of negative charges in the p-side of the junction. This region of positive charge and negative charge on either side of the junction is termed as the depletion region. Due to this positive space charge region on either side of the junction, an electric field direction from a positive charge towards the negative charge is

developed. Due to this electric field, an electron on the p-side of the junction moves to the n-side of the junction. This motion is termed as the drift. Here, we see that the direction of drift current is opposite to that of the diffusion current.

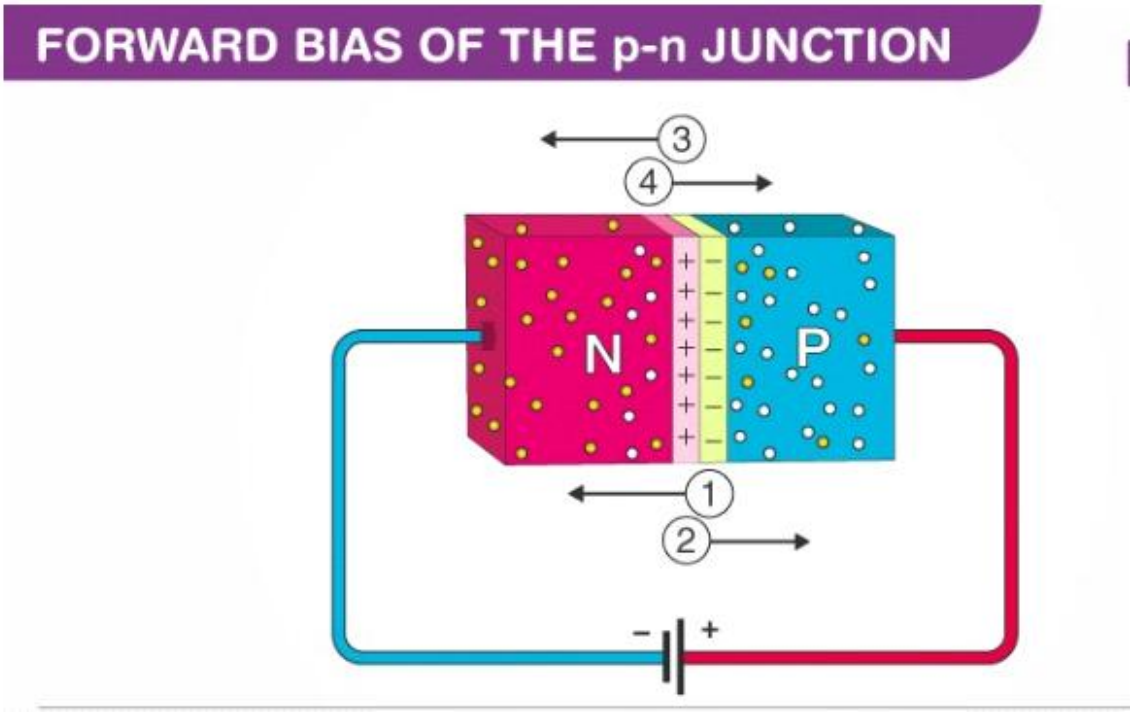
Biasing conditions for the p-n Junction Diode :

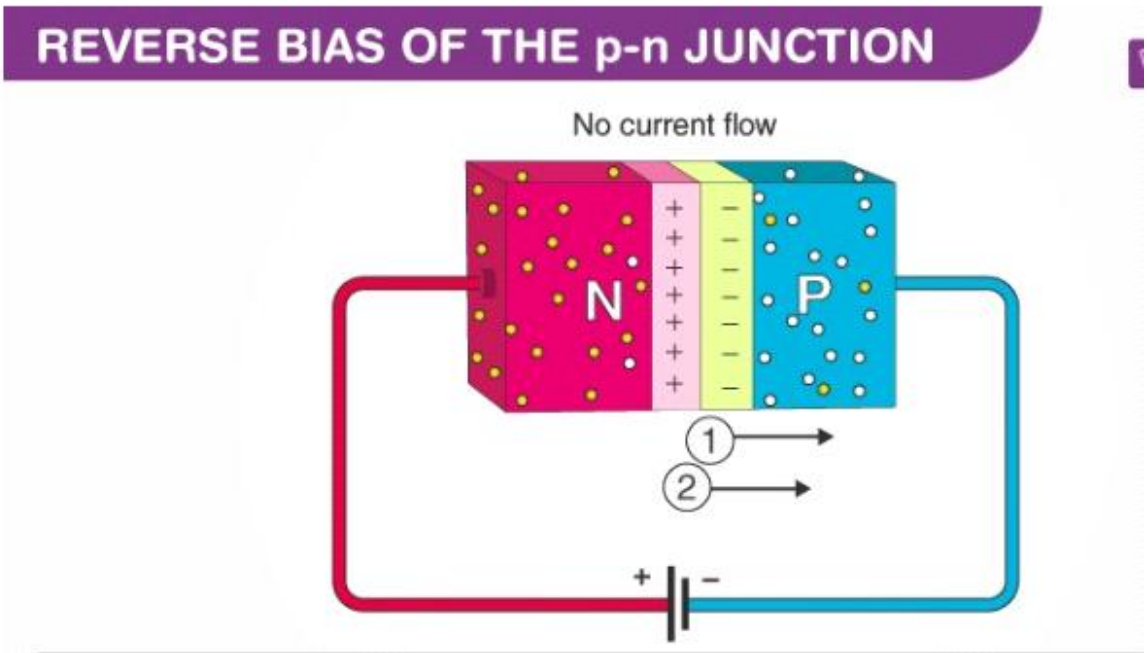
There are two operating regions in the p-n junction diode:

- P-type
- N-type

There are three biasing conditions for p-n junction diode and this is based on the voltage applied:

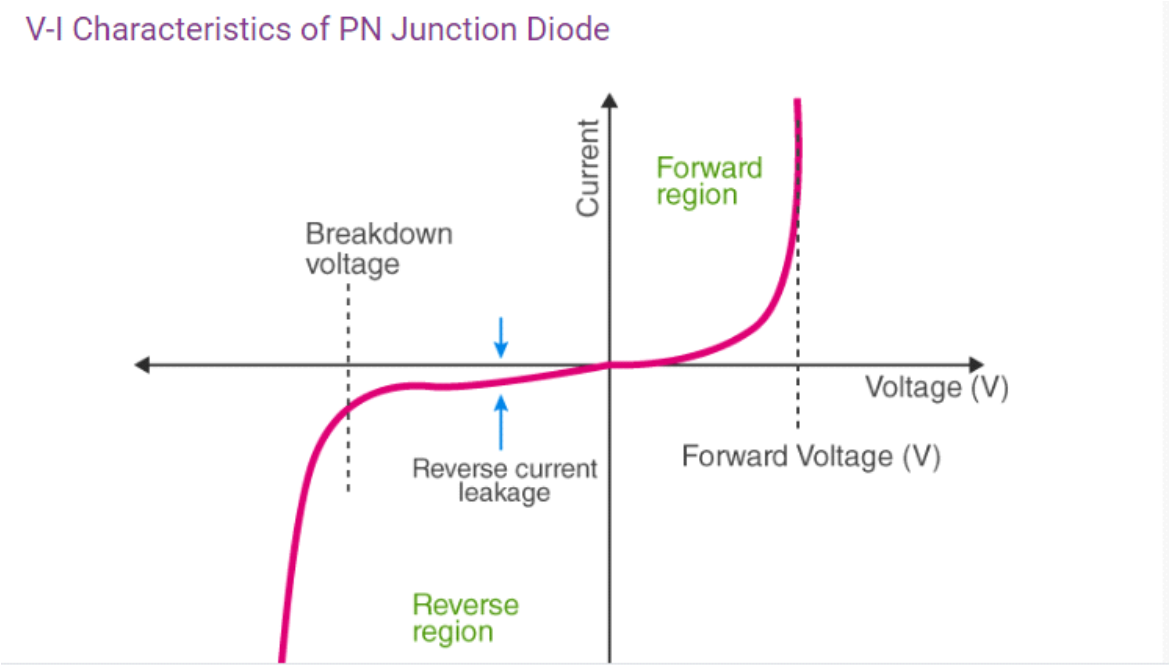
- Zero bias: There is no external voltage applied to the p-n junction diode.
- Forward bias: The positive terminal of the voltage potential is connected to the p-type while the negative terminal is connected to the n-type.
- Reverse bias: The negative terminal of the voltage potential is connected to the p-type and the positive is connected to the n-type.





Current flow in PN junction diode :

The flow of electrons from the n-side towards the p-side of the junction takes place when there is an increase in the voltage. Similarly, the flow of holes from the p-side towards the n-side of the junction takes place along with the increase in the voltage. This results in the concentration gradient between both sides of the terminals. Because of the formation of the concentration gradient, there will be a flow of charge carriers from higher concentration regions to lower concentration regions. The movement of charge carriers inside the pn junction is the reason behind the current flow in the circuit.



VI characteristics of PN junction diode is a curve between the voltage and current through the circuit. Voltage is taken along the x-axis while the current is taken along the y-axis. The above graph is the VI characteristics curve of the PN junction diode. With the help of the curve we can understand that there are three regions in which the diode works, and they are:

- Zero bias
- Forward bias
- Reverse bias

When the PN junction diode is under zero bias condition, there is no external voltage applied and this means that the potential barrier at the junction does not allow the flow of current.

When the PN junction diode is under forward bias condition, the p-type is connected to the positive terminal while the n-type is connected to the negative terminal of the external voltage. When the diode is arranged in this manner, there is a reduction in the potential barrier. For silicone diodes, when the voltage is 0.7 V and for germanium diodes, when the voltage is 0.3 V, the potential barriers decreases and there is a flow of current.

When the diode is in forward bias, the current increases slowly and the curve obtained is non-linear as the voltage applied to the diode is overcoming the potential barrier. Once the potential barrier is overcome by the diode, the diode behaves normal and the curve rises sharply as the external voltage increases and the curve so obtained is linear.

When the PN junction diode is under negative bias condition, the p-type is connected to the negative terminal while the n-type is connected to the positive terminal of the external voltage. This results in an increase in the potential barrier. Reverse saturation current flows in the beginning as minority carriers are present in the junction.

When the applied voltage is increased, the minority charges will have increased kinetic energy which affects the majority charges. This is the stage when the diode breaks down. This may also destroy the diode.

Applications of PN Junction Diode

- p-n junction diode can be used as a photodiode as the diode is sensitive to the light when the configuration of the diode is reverse-biased.
- It can be used as a solar cell.

When the diode is forward-biased, it can be used in LED lighting applications**Biploar transistor:**

A transistor is basically a Si on Ge crystal containing three separate regions. It can be either NPN or PNP type. The middle region is called the base and the outer two regions are called

emitter and the collector. The outer layers although they are of same type but their functions cannot be changed. They have different physical and electrical properties.

In most transistors, emitter is heavily doped. Its job is to emit or inject electrons into the base. These bases are lightly doped and very thin, it passes most of the emitter-injected electrons on to the collector. The doping level of collector is intermediate between the heavy doping of emitter and the light doping of the base.

The collector is so named because it collects electrons from base. The collector is the largest of the three regions; it must dissipate more heat than the emitter or base. The transistor has two junctions. One between emitter and the base and other between the base and the collector. Because of this the transistor is similar to two diodes, one emitter diode and other collector base diode.

When transistor is made, the diffusion of free electrons across the junction produces two depletion layers. For each of these depletion layers, the barrier potential is 0.7 V for Si transistor and 0.3 V for Ge transistor.

The depletion layers do not have the same width, because different regions have different doping levels. The more heavily doped a region is, the greater the concentration of ions near the junction. This means the depletion layer penetrates more deeply into the base and slightly into emitter. Similarly, it penetration more into collector. The thickness of collector depletion layer is large while the base depletion layer is small as shown in

both the junctions are forward biased using two d.c sources, as shown in free electrons (majority carriers) enter the emitter and collector of the transistor, joins at the base and come out of the base. Because both the diodes are forward biased, the emitter and collector currents are large**.** If both the junction are reverse biased as shown in then small currents flows through both junctions only due to thermally produced minority carriers and surface leakage. Thermally produced carriers are temperature dependent it approximately doubles for every 10 degree celsius rise in ambient temperature. The surface leakage current increases with voltage**.**

When the emitter diode is forward biased and collector diode is reverse biased as shown in **fig.** 3.1 then one expect large emitter current and small collector current but collector current is almost as large as emitter current.

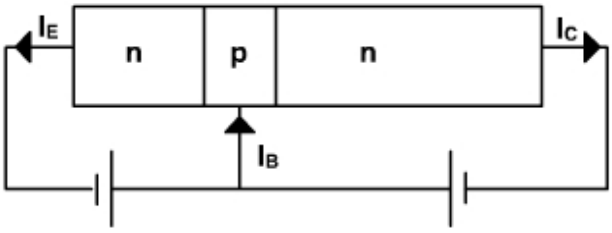


Fig.3.1

When emitter diodes forward biased and the applied voltage is more than 0.7 V (barrier potential) then larger number of majority carriers (electrons in n-type) diffuse across the junction.

Once the electrons are injected by the emitter enter into the base, they become minority carriers. These electrons do not have separate identities from those, which are thermally generated, in the base region itself. The base is made very thin and is very lightly doped. Because of this only few electrons traveling from the emitter to base region recombine with holes. This gives rise to recombination current. The rest of the electrons exist for more time. Since the collector diode is reverse biased, (n is connected to positive supply) therefore most of the electrons are pushed into collector layer. These collector elections can then flow into the external collector lead.

Thus, there is a steady stream of electrons leaving the negative source terminal and entering the emitter region. The V_{EB} forward bias forces these emitter electrons to enter the base region. The thin and lightly doped base gives almost all those electrons enough lifetime to diffuse into the depletion layer. The depletion layer field pushes a steady stream of electron into the collector region. These electrons leave the collector and flow into the positive terminal of the voltage source. In most transistor, more than 95% of the emitter injected electrons flow to the collector, less than 5% fall into base holes and flow out the external base lead. But the collector current is less than emitter current.

3.2 Relation between different currents in a transistor:

The total current flowing into the transistor must be equal to the total current flowing out of it. Hence, the emitter current I_E is equal to the sum of the collector (I_C) and base current (I_B). That is,

$$I_E = I_C + I_B \qquad \text{(E-1)}$$

The currents directions are positive directions. The total collector current I_C is made up of two components.

1. The fraction of emitter (electron) current which reaches the collector ($\alpha_{dc} I_E$)

2. The normal reverse leakage current I_{CO}

$$\therefore I_C = \alpha_{dc} I_E + I_{co}$$

$$\text{or} \quad \alpha_{dc} = \frac{I_C - I_{co}}{I_E} \qquad \text{(E-2)}$$

a_{dc} is known as large signal current gain or dc alpha. It is always positive. Since collector current is almost equal to the I_E therefore $\alpha dc\ I_E$ varies from 0.9 to 0.98. Usually, the reverse leakage current is very small compared to the total collector current.

$$\text{Neglecting } I_{co}, \quad \alpha_{dc} = \frac{I_C}{I_E} \qquad \text{(E-3)}$$

NOTE:

The forward bias on the emitter diode controls the number of free electrons infected into the base. The larger (V_{BE}) forward voltage, the greater the number of injected electrons. The reverse bias on the collector diode has little influence on the number of electrons that enter the collector. Increasing V_{CB} does not change the number of free electrons arriving at the collector junction layer.

The symbol of npn and pnp transistors are shown in **fig.** 3.2.

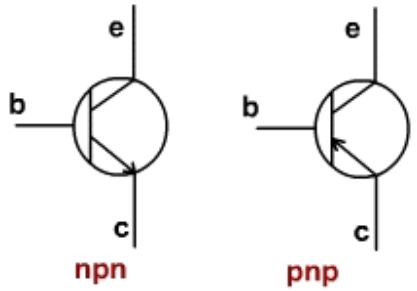


Fig. 3.2

3.3 Breakdown Voltages:

Since the two halves of a transistor are diodes, two much reverse voltage on either diode can cause breakdown. The breakdown voltage depends on the width of the depletion layer and the doping levels. Because of the heavy doping level, the emitter diode has a low breakdown voltage approximately 5 to 30 V. The collector diode is less heavily doped so its breakdown voltage is higher around 20 to 300 V.

3.4 The Common Base Configuration :

If the base is common to the input and output circuits, it is known as common base configuration as shown in **fig.** 3.3.

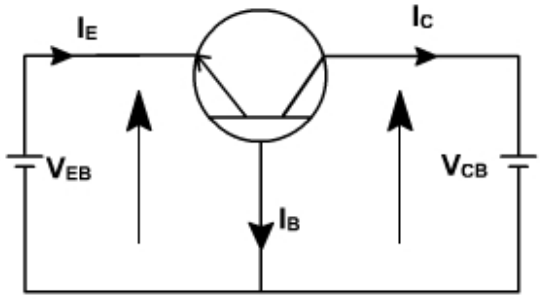


Fig. 3.3

For a pnp transistor the largest current components are due to holes. Holes flow from emitter to collector and few holes flow down towards ground out of the base terminal. The current directions are shown in **fig.** 3.3.

($I_E = I_C + I_B$). **(E-4)**

For a forward biased junction, V_{EB} is positive and for a reverse biased junction V_{CB} is negative. The complete transistor can be described by the following two relations, which give the input voltage V_{EB} and output current I_C in terms of the output voltage (V_{CB}) and input current I_E.

$V_{EB} = f_1(V_{CB}, I_E)$ **(E-5)**

$I_C = f_2(V_{CB}, I_E)$ **(E-6)**

3.4.1The output characteristic:

The collector current I_C is completely determined by the input current I_E and the V_{CB} voltage. The relationship is given in **fig.**3.4**.** It is a plot of I_C versus V_{CB}, with emitter current I_E as parameter. The curves are known as the output or collector or static characteristics. The transistor consists of two diodes placed in series back to back (with two cathodes connected together). The complete characteristic can be divided in three regions.

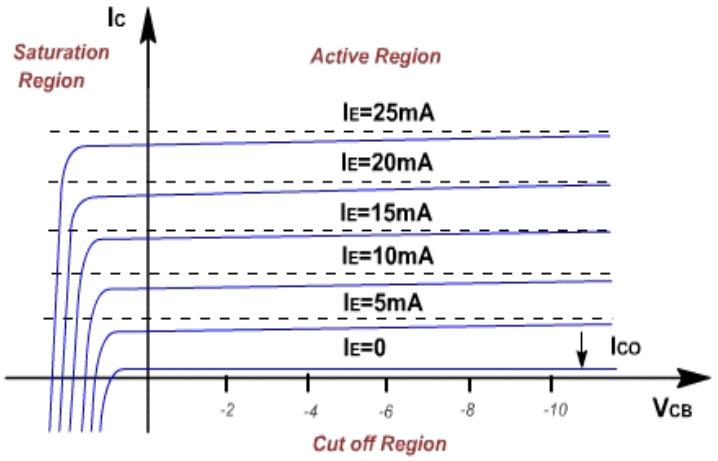


Figure 3.4

(1). Active region:

In this region the collector diode is reverse biased and the emitter diode is forward biased. Consider first that the emitter current is zero. Then the collector current is small and equals the reverse saturation current I_{CO} of the collector junction considered as a diode.

If the forward current I_B is increased, then a fraction of I_E ie. $\alpha_{dc}I_E$ will reach the collector. In

the active region, the collector current is essentially independent of collector voltage and depends only upon the emitter current. Because α_{dc} is, less than one but almost equal to unity, the magnitude of the collector current is slightly less that of emitter current. The collector current is almost constant and work as a current source.

The collector current slightly increases with voltage. This is due to early effect. At higher voltage collector gathers in a few more electrons. This reduces the base current. The difference is so small, that it is usually neglected. If the collector voltage is increased, then space charge width increases; this decreased the effective base width. Then there is less chance for recombination within the base region.

(2). Saturation region:

The region to the left of the ordinate $V_{CB} = 0$, and above the $I_E = 0$, characteristic in which both emitter and collector junction are forward biased, is called saturation region.

When collector diode is forward biased, there is large change in collector current with small changes in collector voltage. A forward bias means, that p is made positive with respect to n, there is a flow of holes from p to n. This changes the collector current direction. If diode is sufficiently forward biased the current changes rapidly. It does not depend upon emitter current.

(3). Cut off region:

The region below $I_E = 0$ and to the right of V_{CB} for which emitter and collector junctions are both reversed biased is referred to cutoff region. The characteristics $I_E = 0$, is similar to other characteristics but not coincident with horizontal axis. The collector current is same as I_{CO}. I_{CBO} is frequently used for I_{CO}. It means collector to base current with emitter open. This is also temperature dependent.

3.4.2 The Input Characteristic:

In the active region the input diode is forward biased, therefore, input characteristic is simply the forward biased characteristic of the emitter to base diode for various collector voltages. **fig. 3.5.** Below cut in voltage (0.7 or 0.3) the emitter current is very small. The curve with the collector open represents the forward biased emitter diode. Because of the early effect the emitter current increases for same V_{EB}. (The diode becomes better diode).

When the collector is shorted to the base, the emitter current increases for a given V_{EB} since the collector now removes minority carriers from the base, and hence base can attract more holes from the emitter. This mean that the curve V_{CB}= 0, is shifted from the character when V_{CB} = open.

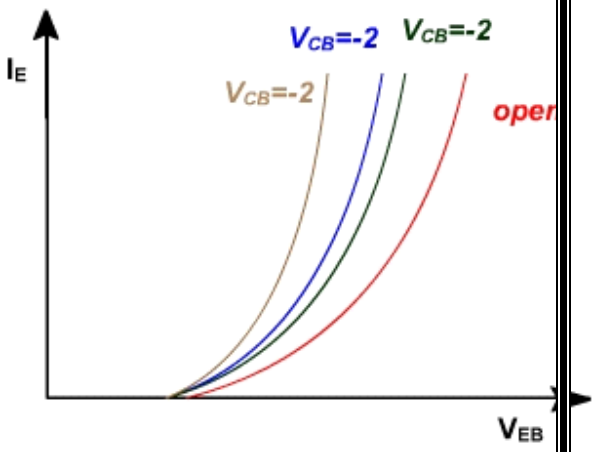


Fig. 3.5

3.5 Common Base Amplifier:

The common base amplifier circuit is shown in **Fig.**3.6**.** The V_{EE} source forward biases the emitter diode and V_{CC} source reverse biased collector diode. The ac source v_{in} is connected to emitter through a coupling capacitor so that it blocks dc. This ac voltage produces small fluctuation in currents and voltages. The load resistance R_L is also connected to collector through coupling capacitor so the fluctuation in collector base voltage will be observed across R_L.

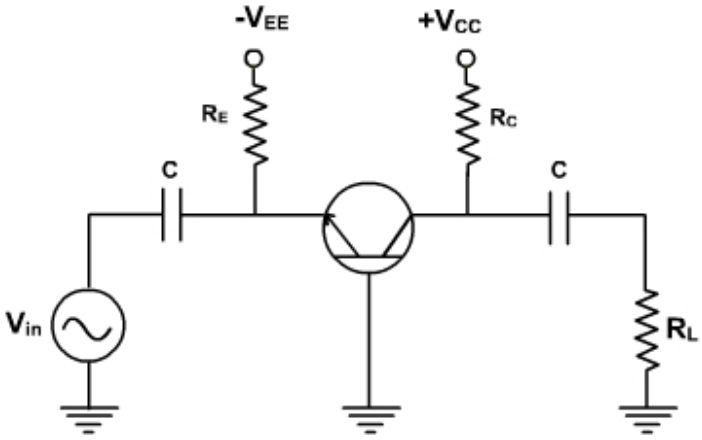


Fig. 3.6

The dc equivalent circuit is obtained by reducing all ac sources to zero and opening all capacitors. The dc collector current is same as I_E and V_{CB} is given by

$V_{CB} = V_{CC} - I_C R_C$. (E-7)

Field Effect Transistor:

The field effect transistor is a semiconductor device, which depends for its operation on the control of current by an electric field. There are two of field effect transistors:

- JFET (Junction Field Effect Transistor)
- MOSFET (Metal Oxide Semiconductor Field Effect Transistor)

The FET has several advantages over conventional transistor.

- In a conventional transistor, the operation depends upon the flow of majority and minority carriers. That is why it is called bipolar transistor. In FET the operation depends upon the flow of majority carriers only. It is called unipolar device.
- The input to conventional transistor amplifier involves a forward biased PN junction with its inherently low dynamic impedance. The input to FET involves a reverse biased PN junction hence the high input impedance of the order of M-ohm.
- It is less noisy than a bipolar transistor.
- It exhibits no offset voltage at zero drain current.

- It has thermal stability.
- It is relatively immune to radiation.

The main disadvantage is its relatively small gain bandwidth product in comparison with conventional transistor.

4.2 Operation of FET:

Consider a sample bar of N-type semiconductor. This is called N-channel and it is electrically equivalent to a resistance as shown in **fig. 4.1**.

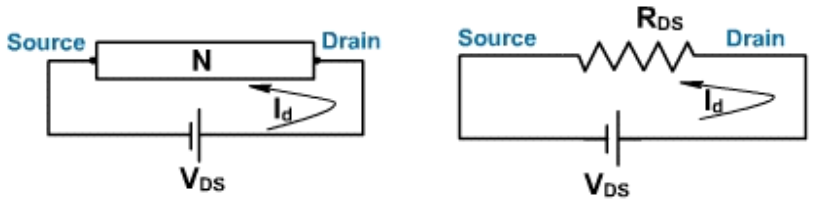


Fig. 4.1

Ohmic contacts are then added on each side of the channel to bring the external connection. Thus if a voltage is applied across the bar, the current flows through the channel.

The terminal from where the majority carriers (electrons) enter the channel is called source designated by S. The terminal through which majority carriers leaves the channel is called drain and designated by D. For an N-channel device, electrons are the majority carriers. Hence the circuit behaves like a dc voltage V_{DS} applied across a resistance R_{DS}. The resulting current is the drain current I_D. If V_{DS} increases, I_D increases proportionally.

Now on both sides of the n-type bar heavily doped regions of p-type impurity have been formed by any method for creating pn junction. These impurity regions are called gates (gate1 and gate2) as shown in **fig.4.**1.

Both the gates are internally connected and they are grounded yielding zero gate source voltage (V_{GS} =0).

The word gate is used because the potential applied between gate and source controls the channel

width and hence the current.

As with all PN junctions, a depletion region is formed on the two sides of the reverse biased PN junction. The current carriers have diffused across the junction, leaving only uncovered positive ions on the n side and negative ions on the p side. The depletion region width increases with the magnitude of reverse bias. The conductivity of this channel is normally zero because of the unavailability of current carriers.

The potential at any point along the channel depends on the distance of that point from the drain, points close to the drain are at a higher positive potential, relative to ground, then points close to the source. Both depletion regions are therefore subject to greater reverse voltage near the drain. Therefore the depletion region width increases as we move towards drain. The flow of electrons from source to drain is now restricted to the narrow channel between the no conducting depletion regions. The width of this channel determines the resistance between drain and source.

Consider now the behavior of drain current I_D vs drain source voltage V_{DS}. The gate source voltage is zero therefore $V_{GS}= 0$. Suppose that V_{DS} is gradually linearly increased linearly from 0V. I_D also increases.

Since the channel behaves as a semiconductor resistance, therefore it follows ohm's law. The region is called ohmic region, with increasing current, the ohmic voltage drop between the source and the channel region reverse biased the junction, the conducting portion of the channel begins to constrict and I_D begins to level off until a specific value of V_{DS} is reached, called the **pinch of voltage V_P**.

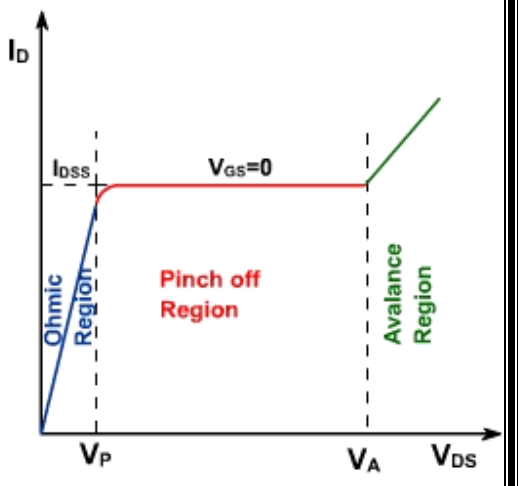


Fig. 4.2

At this point further increase in V_{DS} do not produce corresponding increase in I_D. Instead, as V_{DS} increases, both depletion regions extend further into the channel, resulting in a no more cross section, and hence a higher channel resistance. Thus even though, there is more voltage, the

resistance is also greater and the current remains relatively constant. This is called pinch off or saturation region. The current in this region is maximum current that FET can produce and designated by I_{DSS}. (Drain to source current with gate shorted).

As with all pn junctions, when the reverse voltage exceeds a certain level, avalanche breakdown of pn junction occurs and I_D rises very rapidly as shown in **fig.4.2**.

Consider now an N-channel JFET with a reverse gate source voltage as shown in **fig.4.3**.

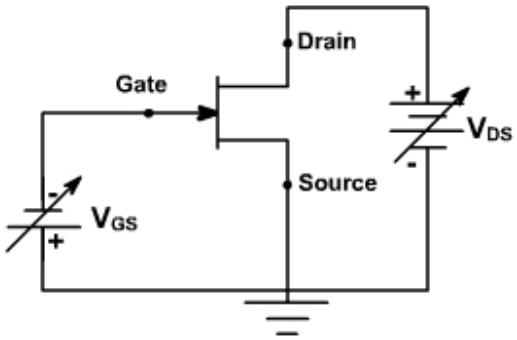


Fig. 4.3

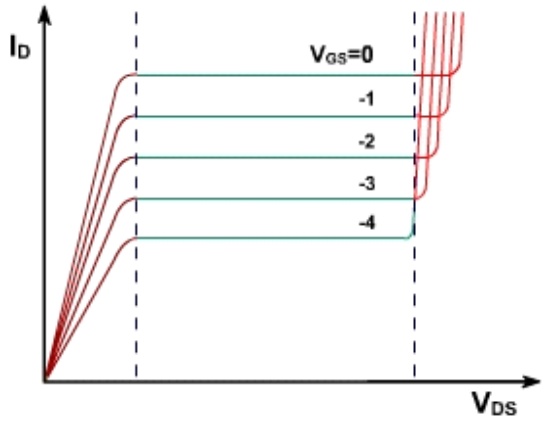


Fig. 4.5

The additional reverse bias, pinch off will occur for smaller values of $|V_{DS}|$, and the maximum drain current will be smaller. A family of curves for different values of V_{GS}(negative) is shown in **fig. 4.5**.

Suppose that V_{GS}= 0 and that due of V_{DS} at a specific point along the channel is +5V with respect to ground. Therefore reverse voltage across either p-n junction is now 5V. If V_{GS} is decreased from 0 to –1V the net reverse bias near the point is 5 - (-1) = 6V. Thus for any fixed value of V_{DS}, the channel width decreases as V_{GS} is made more negative.

Thus I_D value changes correspondingly. When the gate voltage is negative enough, the depletion layers touch each other and the conducting channel pinches off (disappears). In this

case the drain current is cut off. The gate voltage that produces cut off is symbolized V_{GS}(off) . It is same as pinch off voltage.

Since the gate source junction is a reverse biased silicon diode, only a very small reverse current flows through it. Ideally gate current is zero. As a result, all the free electrons from the source go to the drain i.e. $I_D = I_S$. Because the gate draws almost negligible reverse current the input resistance is very high 10's or 100's of M ohm. Therefore where high input impedance is required, JFET is preferred over BJT. The disadvantage is less control over output current i.e. FET takes larger changes in input voltage to produce changes in output current. For this reason, JFET has less voltage gain than a bipolar amplifier.

4.3 Biasing the Field Effect Transistor

Transductance Curves: The transductance curve of a JFET is a graph of output current (I_D) vs input voltage (V_{GS}) as shown in **fig.4.6**.

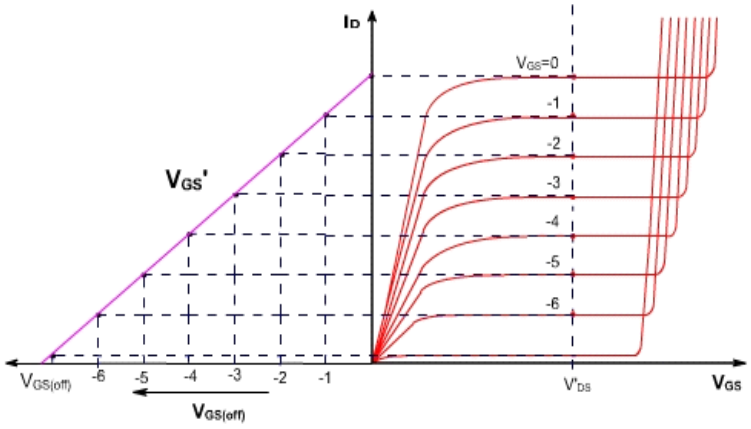


Fig. 4.6

By reading the value of I_D and V_{GS} for a particular value of V_{DS}, the transductance curve can be plotted. The transductance curve is a part of parabola. It has an equation of

$$I_D = I_{DSS}\left(1 - \frac{V_{GS}}{V_{GS(Off)}}\right)^2 \qquad \text{(E-4.1)}$$

Data sheet provides only I_{DSS} and V_{GS}(off) value. Using these values the transductance curve

can be plotted.

Biasing the FET:

The FET can be biased as an amplifier. Consider the common source drain characteristic of a JFET. For linear amplification, Q point must be selected somewhere in the saturation region. Q point is selected on the basis of ac performance i.e. gain, frequency response, noise, power, current and voltage ratings.

4.3.1 Gate Bias:

Fig.4.7, shows a simple gate bias circuit.

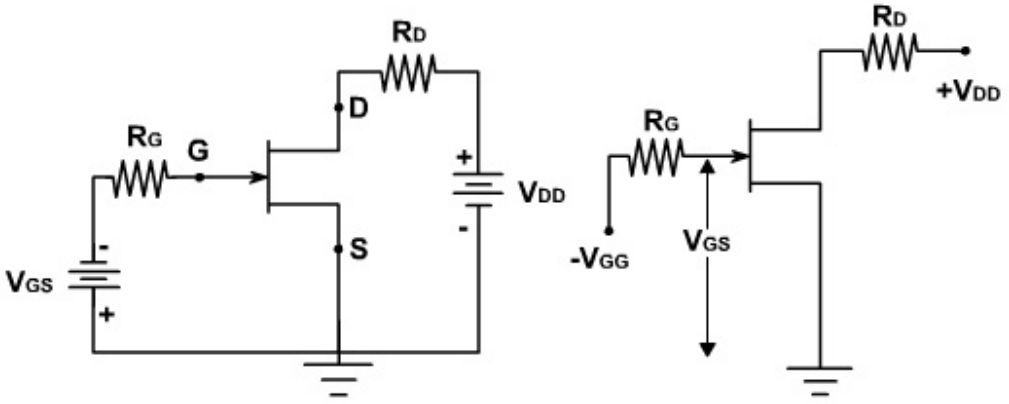


Fig. 4.7

Separate V_{GS} supply is used to set up Q point. This is the worst way to select Q point. The reason is that there is considerable variation between the maximum and minimum values of FET parameters e.g.

	I_{DSS}	V_{GS}(off)
Minimum	4mA	-2V
Maximum	13mA	-8V

This implies that the minimum and maximum transductance curves are displaced as shown in **fig. 4.8**.

Gate bias applies a fixed voltage to the gate. This fixed voltage results in a Q point that is highly sensitive to the particular JFET used. For instance, if V_{GS}= -1V the Q point may very from Q_1 to Q_2 depending upon the JFET parameter is use.

At Q_1, I_D= 0.016 (1 - (1/8))^2 = 12.3 mA

At Q_2, I_D= 0.004 (1-(1/2))^2 = 1 mA.

The variation in drain current is very large.

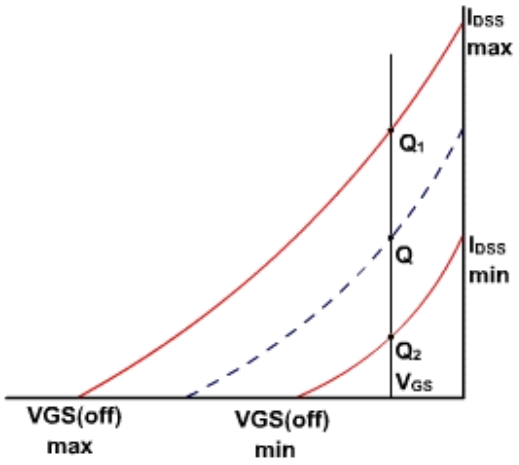


Fig. 4.8

4.3.2 Self Bias:

Fig.4.9, shows a self bias circuit another way to bias a FET. Only a drain supply is used and no gate supply. The idea is to use the voltage across R_S to produce the gate source reverse voltage.

This is a form of a local feedback similar to that used with bipolar transistors. If drain current increases, the voltage drop across R_S increases because the I_D R_S increases. This increases the gate source reverse voltage which makes the channel narrow and reduces the drain current. The overall effect is to partially offset the original increase in drain current. Similarly, if I_D decreases, drop across R_S decreases, hence reverse bias decreases and I_D increases.

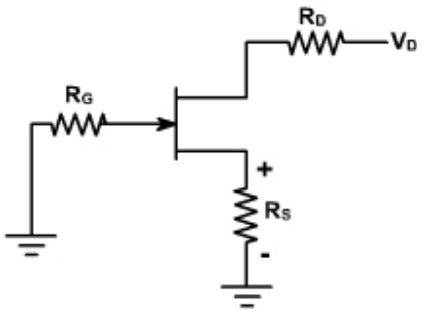


Fig. 4.9

Since the gate source junction is reverse biased, negligible gate current flows through R_G and so the gate voltage with respect to ground is zero.

V_G= 0;

The source to ground voltage equals the product of the drain current and the source resistance.

$\ V_S = I_D R_S$. (E-4.2)

The gate source voltage is the difference between the gate voltage and the source voltage.

$V_{GS} = V_G - V_S = 0 - I_D R_S$

$V_{GS} = -I_D R_S$. (E-4.3)

This means that the gate source voltage equals the negative of the voltage across the source resistor. The greater the drain current, the more negative the gate source voltage becomes.

Rearranging the equation:

$I_D = -V_{GS} / R_S$ (E-4.4)

The graph of this equation is called self base line a shown in **Fig.4.10**.

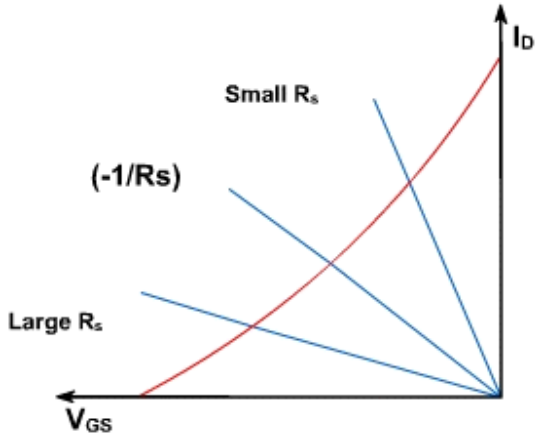


Fig:4.10

The operating point on transductance curve is the intersection of self bias line and transductance curve. The slope of the line is (-1 / R_S). If the source resistance is very large (-1 / R_S is small) then Q-point is far down the transductance curve and the drain current is small. When R_S is small, the Q point is far up the transductance curve and the drain current is large. In between there is an optimum value of R_S that sets up a Q point near the middle of the transductance curve.

The transductance curve varies widely for FET (because of variation in I_{DSS} and V_{GS}(off)) as shown in **fig. 4.11**. The actual curve may be in between there extremes. A and B are the optimum points for the two extreme curves. To find the optimum resistance R_S, so that Q-point is correct for all the curves, A and B points are joined such that it passes through origin.

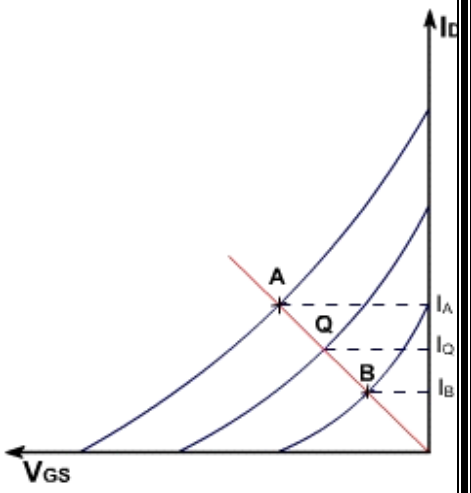


Fig. 4.11

The slope of this line gives the resistance value R_S($V_{GS} = -I_D R_S$). The current I_Q is such that $I_A > I_Q > I_B$. Here A, Q and B all points are in straight line.

Consider the case where a line drawn to pass between points A and B does not pass through the origin. The equation $V_{GS} = - I_D R_S$ is not valid. The equation of this line is $V_{GS} = V_{GG} – I_D R_S$.

Such a bias relationship may be obtained by adding a fixed bias to the gate in addition to the source self bias as shown in **fig.4.12**.

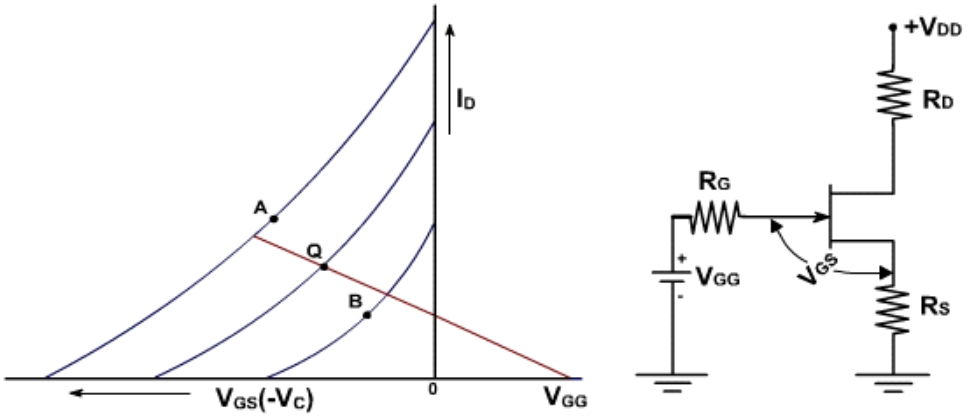


Fig.4.12

In this circuit.

$V_{GG} = R_S I_G + V_{GS} + I_D R_S$ (E-4.5)

Since $R_S I_G = 0$;

$V_{GG} = V_{GS} + I_D R_S$

or $V_{GS} = V_{GG} - I_D R_S$ (E-4.6)

4.3.3 Voltage Divider Bias :

The biasing circuit based on single power supply is shown in **fig. 4.13**. This is similar to the voltage divider bias used with a bipolar transistor.

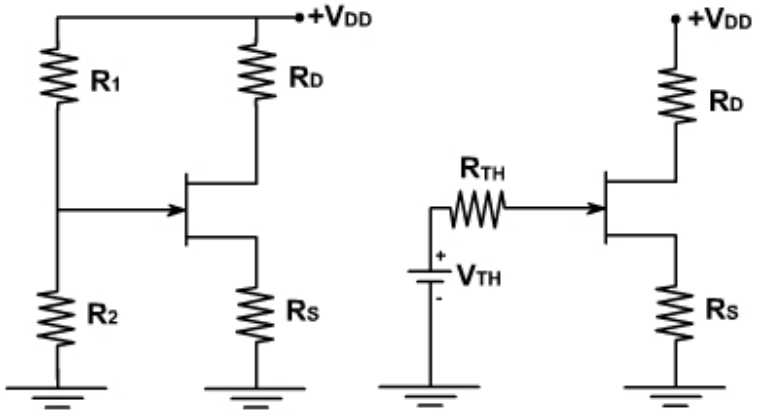


Fig. 4.13

The Thevenin voltage V_{TH} applied to the gate is

$$V_{TH} = \frac{R_2}{R_1 + R_2} V_{DD} \qquad \text{(E-4.7)}$$

The Thevenin resistance is given as

$$R_{TH} = \frac{R_2 R_1}{R_1 + R_2} \qquad \text{(E-4.8)}$$

The gate current is assumed to be negligible. V_{TH} is the dc voltage from gate to ground.

$$V_{TH} = V_{GS} + V_S \text{ (neglecting } I_G)$$
$$\therefore V_S = V_{TH} = V_{GS} \qquad \text{(E-4.9)}$$

The drain current ID is given by

$$I_D = \frac{V_{TH} - V_{GS}}{R_S} \qquad \text{(E-4.10)}$$

and the dc voltage from the drain to ground is $V_D = V_{DD} – I_D R_D$.

If V_{TH} is large enough to swamp out V_{GS} the drain current is approximately constant for any JFET as shown in **fig.4.14**.

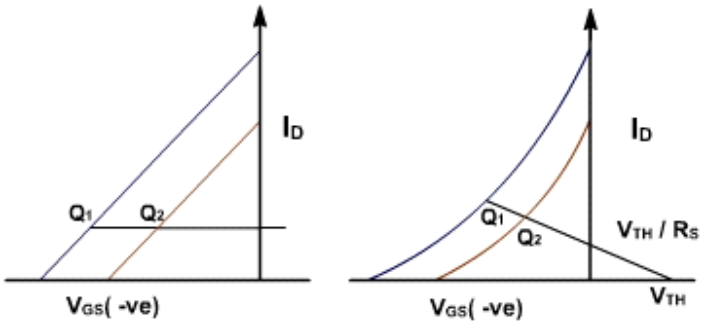


Fig. 4.14

There is a problem in JFET. In a BJT, V_{BE} is approximately 0.7V, with only minor variations from one transistor to other. In a FET, V_{GS} can vary several volts from one JFET to another. It is therefore, difficult to make V_{TH} large enough to swamp out V_{GS}. For this reason, voltage divider bias is less effective with, FET than BJT. Therefore, V_{GS} is not negligible. The current increases slightly from

Q2 to Q1. However, voltage divider bias maintains I_D nearly constant.

Consider a voltage divider bias circuit shown in **fig.4.15**.

$V_{GS(min)} = -1, \quad V_{GS(max)} = -5V$

$V_{TH} = 15V$

$I_{D(min)} = \frac{15-(-1)}{7.5K} = 2.13 \text{ mA}$

$I_{D(max)} = \frac{15-(-5)}{7.5K} = 2.67 \text{ mA}$

Difference in $I_{D\,(min)}$ and $I_{D\,(max)}$ is less

$V_{D\,(max)} = 30 - 2.13 * 4.7 = 20 \text{ V}$

$V_{D\,(min)} = 30 - 2.67 * 4.7 = 17.5 \text{ V}$

4.3.4 Current Source Bias:

This is another way to produce solid Q point. The aim is to produce a drain current that is independent of V_{GS}. Voltage divider bias and self bias attempt to do this by swamping out of variations in V_{GS}.

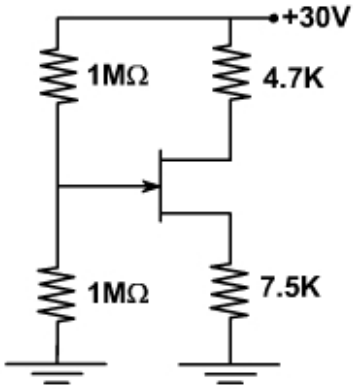


Fig.4.15

Using two power supplies:

The current source bias can be used to make I_D constant **fig.4.16**.

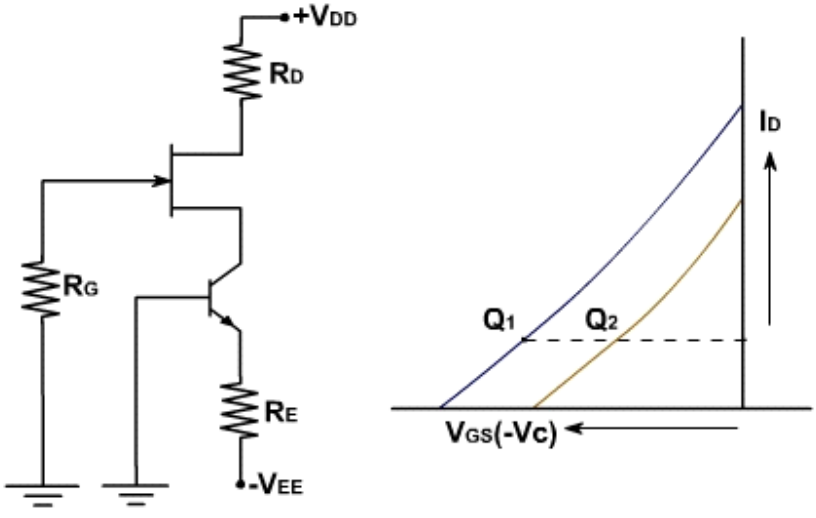


Fig. 4.16

The bipolar transistor is emitter biased; its collector current is given by

$I_C = (V_{EE} – V_{BE}) / R_E$. (E-4.11)

Because the bipolar transistor acts like a current source, it forces the drain current to equal the bipolar collector current.

$I_D = I_C$

Since I_C is constant, both Q points have the same value of drain current. The current source effectively wipes out the influence of V_{GS}. Although V_{GS} is different for each Q point, it no longer influences the value of drain current.

Using One power supply:

When only a positive supply is available, the circuit shown in **fig. 5**, can be used to set up a constant drain current.

In this case, the bipolar transistor is voltage divider biased. Assuming a stiff voltage divider, the emitter and collector currents are constant for all bipolar transistors. This forces the FET drain current equal the bipolar collector current.

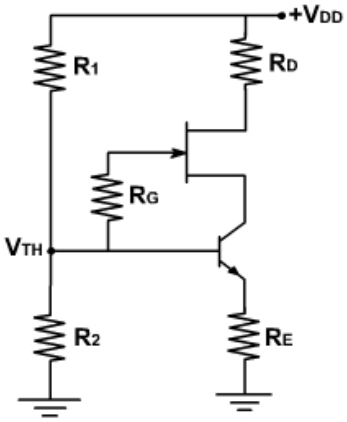


Fig. 4.17

$$V_{TH} = \frac{R_2 \, V_{DD}}{R_1 + R_2}$$

$$I_E = \frac{V_{TH} - V_{BE}}{R_E}$$

Since V_{TH} is constant, I_E is also constant

$$I_C = I_S = I_D = \text{constant}$$

Transductance:

The transductance of a FET is defined as

$$g_m = \left. \frac{\Delta I_D}{\Delta V_{gs}} \right|_{V_{DS}=0} \mu A / Volts \qquad \text{(E-4.12)}$$

Because the changes in I_D and V_{GS} are equivalent to ac current and voltage. This equation can be written as

$$g_m = \left. \frac{i_d}{V_{gs}} \right|_{V_{ds}=0} \qquad \text{(E-4.13)}$$

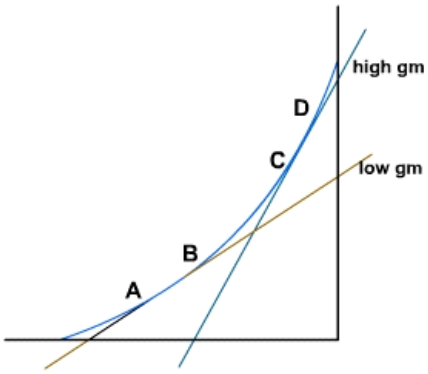


Fig. 4.18

The unit of g_m is mho or siemems.

Typical value of g_m is 2000 m A / V.

The value of g_m can be obtained from the transductance curve as shown in **fig.4.18**.

If A and B points are considered, than a change in V_{GS} produces a change in I_D. The ratio of I_D

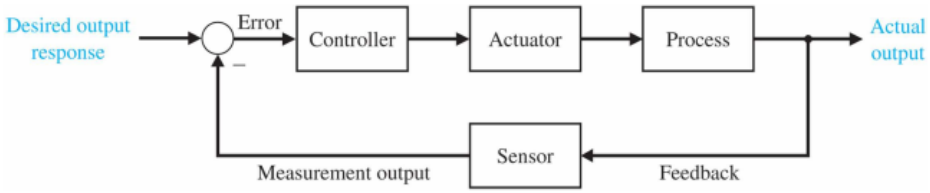
and V_{GS} is the value of g_m between A and B points. If C and D points are considered, then same change in V_{GS} produces more change in I_D. Therefore, g_m value is higher. In a nutshell, g_m tells us how much control gate voltage has over drain current. Higher the value of g_m, the more effective is gate voltage in controlling gate current. The second parameter r_d is the drain resistance.

$$r_d = \left.\frac{V_{ds}}{i_d}\right|_{V_{gs}=0} \quad (r_d \text{ is negligible})$$

(E-4.14)

Pneumatic & hydraulic actuation systems

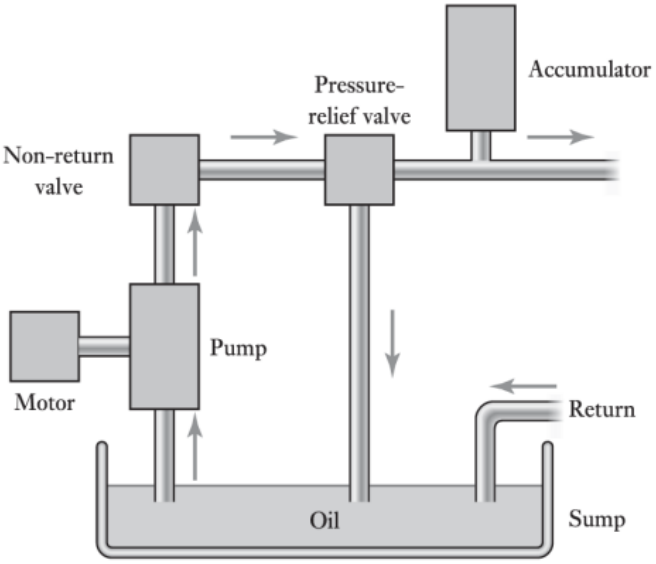
- **Pneumatic deals with air pressure**
- **Hydraulic deals with fluid motion and pressure**



Typical Hydraulic Power System

With a hydraulic system, pressurized oil (fluid) is provided by a pump driven by an electrical motor.

- **The pump pumps oil from a sump through a non return valve and an accumulator to the system, from which it return to the sump.**
- **The pressure relief valve is to release the pressure if it rises above a safe level,**
- **The accumulator is to smooth out any short term fluctuations in the output oil pressure**



How They Work

• Pneumatic linear actuators consist of a piston inside a hollow cylinder. Pressure from an external compressor or manual pump moves the piston inside the cylinder. As pressure increases, the cylinder moves along the axis of the piston, creating a linear force. The piston returns to its original position by either a spring-back force or fluid being supplied to the other side of the piston.

• Hydraulic linear actuators operate similarly to pneumatic actuators, but an incompressible liquid from a pump rather than pressurized air moves the cylinder.

• An electric linear actuator converts electrical energy into torque. An electric motor mechanically connected turns a lead screw. A threaded lead or ball nut with corresponding threads that match those of the screw is prevented from rotating with the screw. When the screw rotates, the nut gets driven along the threads. The direction the nut moves depends on which direction the screw rotates and also returns the actuator to its original position.

Pneumatic Actuators

Advantages

• The benefits of pneumatic actuators come from their simplicity. Most pneumatic aluminum actuators have a maximum pressure rating of 150 psi with bore sizes ranging from ½ to 8 in., which translate into approximately 30 to 7,500 lb. of force. Steel actuators have a maximum pressure rating of 250 psi with bore sizes ranging from ½ to 14 in., and they generate forces ranging from 50 to 38,465 lbf.

• Pneumatic actuators generate precise linear motion by providing accuracy, for example, within 0.1 inches and repeatability within .001 inches.

• Pneumatic actuators typical applications involve areas of extreme temperatures. A typical temperature range is -40°F to 250°F. In terms of safety and inspection, by using air, pneumatic actuators avoid using hazardous materials. They meet explosion protection and machine safety requirements because they create no magnetic interference due to their lack of motors.

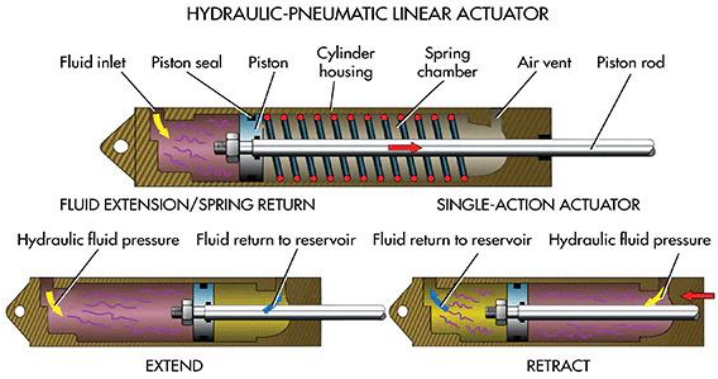
• In recent years, pneumatics has seen many advances in miniaturization, materials, and integration with electronics and condition monitoring. The cost of pneumatic actuators is low compared to other actuators. According to Bimba Manufacturing, for example, the average pneumatic actuator costs $50 to $150. Pneumatic actuators are also lightweight, require minimal maintenance, and have durable components that make pneumatics a cost-effective method of linear motion.

Disadvantages

• Pressure losses and air's compressibility make pneumatics less efficient than other linear-motion methods. Compressor and air delivery limitations mean that operations at lower pressures will have lower forces and slower speeds. A compressor must run continually operating pressure even if nothing is moving.

• To be truly efficient, pneumatic actuators must be sized for a specific job. Hence, they cannot be used for other applications. Accurate control and efficiency requires proportional regulators and valves, but this raises the costs and complexity.

• Even though the air is easily available, it can be contaminated by oil or lubrication, leading to downtime and maintenance. Companies still have to pay for compressed air, making it a consumable, and the compressor and lines are another maintenance issue.



The top image shows a spring return actuator. The maximum spring compression pushes back on the piston and the hydraulic fluid exits the cylinder and returns to its starting position. The bottom image is a double-acting cylinder where fluid enters either side of the piston depending on the desired motion.

Hydraulic Actuators

Advantages

• Hydraulic actuators are rugged and suited for high-force applications. They can produce forces 25 times greater than pneumatic cylinders of equal size. They also operate in pressures of up to 4,000 psi.

• Hydraulic motors have high horsepower-to-weight ratio by 1 to 2 hp/lb greater than a pneumatic motor.

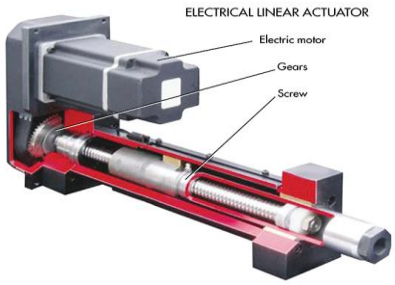
• A hydraulic actuator can hold force and torque constant without the pump supplying more fluid or pressure due to the incompressibility of fluids

• Hydraulic actuators can have their pumps and motors located a considerable distance away with minimal loss of power.

Disadvantages

• Hydraulics will leak fluid. Like pneumatic actuators, loss of fluid leads to less efficiency. However, hydraulic fluid leaks lead to cleanliness problems and potential damage to surrounding components and areas.

• Hydraulic actuators require many companion parts, including a fluid reservoir, motors, pumps, release valves, and heat exchangers, along with noise-reduction equipment. This makes for linear motions systems that are large and difficult to accommodate.



The electric motor is part of the actuator instead of being separate like a pneumatic or hydraulic system. While the electric linear actuator provides high precision, it does have large spacing requirements.

Electrical Actuators

Advantages

• Electrical actuators offer the highest precision-control positioning. An example of the range of accuracy is +/- 0.000315 in. and a repeatability of less than 0.0000394 in. Their setups are scalable for any purpose or force requirement, and are quiet, smooth, and repeatable.

• Electric actuators can be networked and reprogrammed quickly. They offer immediate feedback for diagnostics and maintenance.

• They provide complete control of motion profiles and can include encoders to control velocity, position, torque, and applied force.

• In terms of noise, they are quieter than pneumatic and hydraulic actuators

• Because there are no fluids leaks, environmental hazards are eliminated.

Disadvantages

• The initial unit cost of an electrical actuator is higher than that of pneumatic and hydraulic actuators. According to the example from Bimba Manufacturing, an electrical actuator can range from $150 to greater than $2,000 depending on its design and electronics.

• Electrical actuators are not suited for all environments, unlike pneumatic actuators, which are safe in hazardous and flammable areas

• A continuously running motor will overheat, increasing wear and tear on the reduction gear. The motor can also be large and create installation problems.

• The motor chosen locks in the actuator's force, thrust, and speed limits to a fixed setting. If a different set of values for force, thrust, and speed are desired, the motor must be changed.

UNIT 4

Microprocessors, Microcontrollers and Programmable Logic Controllers: Logic Concepts and Design, System Interfaces, Communication and Computer Networks, Fault Analysis in Mechatronic Systems, Synchronous and Asynchronous Sequential Systems, Architecture, **Microcontrollers**

INTRODUCTION

Introduction

Microprocessor is a Central Processing Unit (CPU) etched on a single chip. A single Integrated Circuit (IC) has all the functional components of a CPU namely Arithmetic Logic Unit (ALU), Control Unit and registers. The 8085 microprocessor is an 8-bit processor that includes on its chip most of the logic circuitry for performing computing tasks and for communicating with peripherals. The architecture of a microprocessor is to be learnt in terms of registers, memory addressing, addressing modes, instruction set, interfacing with memory and Input and Output

(I/O) devices and interrupt handling. It is necessary to learn about the above mentioned concepts to write efficient assembly language programs, and to design microprocessor based systems. This unit gives you an overall idea about the microprocessors, the detailed discussion about 8085 architecture and interfacing of 8085 with Programmable Peripheral Interface (PPI) devices.

Functional Components of a Microprocessor

A digital computer is a programmable machine specially designed for making computation. Its main components are: CPU (Central Processing Unit), memory, input device and output device as shown in figure 1.1.

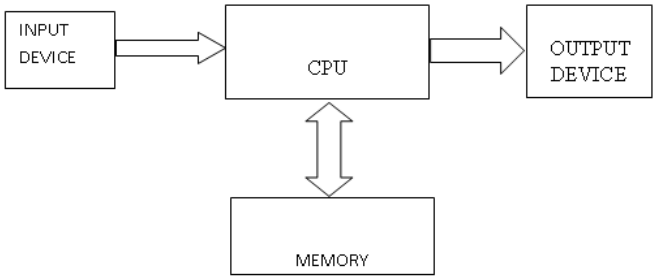


Fig 1 Schematic Diagram of a Digital Computer

A microcomputer is a small digital computer. The CPU of a microcomputer is a microprocessor. Other components are same as those of any other digital computer. In figure 1, if we change the label CPU as Microprocessor, we get the organization of a microcomputer.

The physical devices and circuitry of a computer are called hardware. A physical device may be electronic, magnetic, mechanical or an optical device etc. A sequence of instructions to perform a particular task is called a program. A set of programs written for a particular computer is known as software for that computer. The input and output devices are known as peripherals. Sometimes the term peripheral also includes memory. Programs are subroutines stored in ROM (Read Only Memory)s, Programmable ROM (PROM)s, Erasable PROM(EPROM)s and/or EEPROMs are known as firmware. The commonly available firmwares are: monitors, microprograms, subroutines for input and output devices.

The Central Processing Unit (CPU) fetches instructions from the memory and performs specified tasks. It stores results in the memory or sends results to the output device according to the instructions given in the program. The CPU controls and communicates with memory and input/output devices. Under the control of the CPU, programs and data are stored in the memory and displayed on Cathode Ray Tube (CRT). The schematic diagram of a CPU is shown in fig 2.

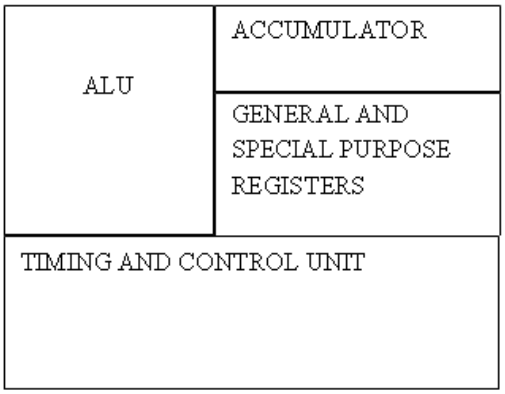


Fig 2 Schematic Diagram of a CPU or a Microprocessor

The CPU of a large computer is implemented on one or more circuit boards. ICs are used as its components. Recent practice is to use microprocessors to perform different functions within the CPU of a large computer. The major sections of a CPU are Arithmetic and Logic Unit (ALU), Accumulator, General and Special purpose registers and Timing and Control Unit. The function of an ALU is to perform arithmetic operations such as addition and subtraction; and logical operations such as AND, OR and EXCLUSIVE-OR. Timing and control unit controls the entire operations of a computer. It acts as a brain. It also controls all other devices connected to the CPU. It generates timing signals necessary for input and output devices. The accumulator is a register, which contains one of the operands and stores results of most arithmetic and logical operations. General purpose registers are used for temporary storage of data and intermediate results while computer is making execution of a program. Special purpose registers are used by the microprocessor itself. Some of them are not accessible to programmers. Examples of special purpose registers are program counter, stack pointer, instruction register and status register.

The memory is a storage device. It stores program, data, results etc.

The computer receives data and instructions through input devices. An input device converts instructions, input data and signals into proper binary form suitable for a digital computer. A key-board and simple switches are used as input devices. The user enters instructions and data through a key-board or simple switches. Computers are also used to measure and control physical quantities like temperature, pressure, speed, position etc. For these purposes transducers are used to convert physical quantities into proportional electrical signals. A/D converters are used to convert analog electrical signals into digital signals, which are sent to the computer. Transducers and sensors, data acquisition system etc. are also included in input devices. A/D converter forms a part of data acquisition system.

The computer sends results to output devices. An output device may store, print, display or send electrical signal to control/actuate certain equipment. The examples of simple output devices are printers, CRT, LEDs, D/A converter, controllers, actuators etc. Sometimes input and output devices may be combined in a single unit, which acts as both an input as well as an output device. A keyboard and CRT are combined to form a video terminal, which is a common I/O device for human interaction with a computer.

With the advent of LSI and VLSI technology it became possible to build the entire CPU on a single chip IC. A CPU built into a single LSI/VLSI chip is called a microprocessor. A digital computer using microprocessor as its CPU is called a microcomputer. The term micro initiates its physical size; not it's computing power. Today the computing power of a powerful microprocessor approaches that a CPU on earlier large computer. The main sections of a microprocessor are: ALU, timing and control unit, accumulator, general purpose and special purpose registers. In this subject we'll study about two microprocessors namely Intel 8085 (8-bit) and Intel 8086 (16-bit).

Evolution of Microprocessors

The first microprocessor was introduced in 1971 by Intel Corporation, U.S.A. It was a 4-bit

microprocessor, the Intel 4004. The 4004 originally ran at a clock speed of 108KHz (108,000 cycles per second, or just over one-tenth a megahertz). The 4004 contained 2,300 transistors and was built on a 10-micron process. This means that each line, trace, or transistor could be spaced about 10 microns (millionths of a meter) apart. Data was transferred 4 bits at a time, and the maximum addressable memory was only 640 bytes. The 4004 was designed for use in a calculator but proved to be useful for many other functions because of its inherent programmability. In 1972, Intel introduced the 1st 8-bit processor, the Intel 8008. The Intel 8004 and 8008 both used Positive Channel Metal Oxide Semiconductor (PMOS) technology. In 1973 a more powerful and faster 8-bit processor, the Intel 8080 was introduced. It employed Negative Channel metal Oxide semiconductor (NMOS) technology. The 8008 processor contained 3,500 transistors and was built on the same 10-micron process as the previous processor. The big change in the 8008 was that it had an 8-bit data bus, which meant it could move data 8 bits at a time twice as much as the previous chip. It could also address more memory, up to 16KB. This chip was primarily used in dumb terminals and general-purpose calculators.

The next chip in the lineup was the 8080, introduced in April 1974, running at a clock rate of 2MHz. Due to mostly the faster clock rate, the 8080 processor had 10 times the performance of the 8008. The 8080 chip contained 6,000 transistors and was built on a 6-micron process. Similar to the previous chip, the 8080 had an 8-bit data bus, so it could transfer 8 bits of data at a time. The 8080 could address up to 64KB of memory, significantly more than the previous chip. It was the 8080 that helped start the PC revolution because this was the processor chip used in what is generally regarded as the first personal computer, the Altair 8800. The CP/M operating system was written for the 8080 chip, and Microsoft was founded and delivered its first product: Microsoft BASIC for the Altair. These initial tools provided the foundation for a revolution in software because thousands of programs were written to run on this platform. In fact, the 8080 became so popular that it was cloned. A company called Zilog formed in late 1975, joined by several ex-Intel 8080 engineers. In July 1976, it released the Z-80 processor, which was a vastly improved version of the 8080. It was not pin compatible but instead combined functions such as the memory interface and RAM refresh circuitry, which enabled cheaper and simpler systems to

be designed. The Z-80 also incorporated a superset of 8080 instructions, meaning it could run all 8080 programs. It also included new instructions and new internal registers, so software designed for the Z-80 would not necessarily run on the older 8080. The Z-80 ran initially at 2.5MHz (later versions ran up to 10MHz) and contained 8,500 transistors. The Z-80 could access 64KB of memory.

Intel released the 8085, it's follow-up to the 8080, in March 1976. Even though it predated the Z-80 by several months, it never achieved the popularity of the Z-80 in personal computer systems. It was popular as an embedded controller, finding use in scales and other computerized equipment. The 8085 ran at 5MHz and contained 6,500 transistors. It was built on a 3-micron process and incorporated an 8-bit data bus. Along different architectural lines, MOS Technologies introduced the 6502 in 1976. This chip was designed by several ex-Motorola engineers who had worked on Motorola's first processor, the 6800. The 6502 was an 8-bit processor like the 8080, but it sold for around $25, whereas the 8080 cost about $300 when it was introduced. The price appealed to Steve Wozniak, who placed the chip in his Apple I and Apple II designs. The chip was also used in systems by Commodore and other system manufacturers. The 6502 and its successors were also used in game consoles, including the original Nintendo Entertainment System (NES) among others. Motorola went on to create the 68000 series, which became the basis for the Apple Macintosh line of computers. Today those systems use the PowerPC chip, also by Motorola and a successor to the 68000 series.

All these previous chips set the stage for the first PC processors. Intel introduced the 8086 in June 1978. The 8086 chip brought with it the original x 86 instructions set that is still present in current x86-compatible chips such as the Pentium 4 and AMD Athlon. A dramatic improvement over the previous chips, the 8086 was a full 16-bit design with 16-bit internal registers and a 16-bit data bus. This meant that it could work on 16-bit numbers and data internally and also transfer 16 bits at a time in and out of the chip. The 8086 contained 29,000 transistors and initially ran at up to 5MHz. The chip also used 20-bit addressing, so it could directly address up to 1MB of memory. Although not directly backward compatible with the 8080, the 8086 instructions and language were very similar and enabled older programs to quickly be ported

over to run. This later proved important to help jumpstart the PC software revolution with recycled CP/M (8080) software.

Although the 8086 was a great chip, it was expensive at the time and more importantly required expensive 16-bit board designs and infrastructure to support it. To help bring costs down, in 1979 Intel released what some called a crippled version of the 8086 called the 8088. The 8088 processor used the same internal core as the 8086, had the same 16-bit registers, and could address the same 1MB of memory, but the external data bus was reduced to 8 bits. This enabled support chips from the older 8-bit 8085 to be used, and far less expensive boards and systems could be made. These reasons are why IBM chose the 8088 instead of the 8086 for the first PC. This decision would affect history in several ways. The 8088 was fully software compatible with the 8086, so it could run 16-bit software. Also, because the instruction set was very similar to the previous 8085 and 8080, programs written for those older chips could be quickly and easily modified to run. This enabled a large library of programs to be quickly released for the IBM PC, thus helping it become a success. The overwhelming blockbuster success of the IBM PC left in its wake the legacy of requiring backward compatibility with it. To maintain the momentum, Intel has pretty much been forced to maintain backward compatibility with the 8088/8086 in most of the processors it has released since then.

To date, backward compatibility has been maintained, but innovating and adding new features has still been possible. One major change in processors was the move from the 16-bit internal architecture of the 286 and earlier processors to the 32-bit internal architecture of the 386 and later chips, which Intel calls IA-32 (Intel Architecture, 32-bit). Intel's 32-bit architecture dates to 1985, and it took a full 10 years for both a partial 32-bit mainstream OS (Windows 95) as well as a full 32-bit OS requiring 32-bit drivers (Windows NT) to surface, and another 6 years for the mainstream to shift to a fully 32-bit environment for the OS and drivers (Windows XP). That's a total of 16 years from the release of 32-bit computing hardware to the full adoption of 32-bit computing in the mainstream with supporting software.

Now we are in the midst of another major architectural jump, as Intel and AMD are in the

process of moving from 32-bit to 64-bit computing for servers, desktop PCs, and even portable PCs. Intel had introduced the IA-64 (Intel Architecture, 64-bit) in the form of the Itanium and Itanium 2 processors several years earlier, but this standard was something completely new and not an extension of the existing 32-bit technology. IA-64 was first announced in 1994 as a CPU development project with Intel and HP (codenamed Merced), and the first technical details were made available in October 1997. The result was the IA-64 architecture and Itanium chip, which was officially released in 2001. The fact that the IA-64 architecture is not an extension of IA-32 but is instead a whole new and completely different architecture is fine for non-PC environments such as servers, but the PC market has always hinged on backward compatibility. Even though emulating IA-32 within IA-64 is possible, such emulation and support is slow.

With the door now open, AMD seized this opportunity to develop 64-bit extensions to IA-32, which it calls AMD64 (originally known as x86-64). Intel eventually released its own set of 64-bit extensions, which it calls EM64T or IA-32e mode. As it turns out, the Intel extensions are almost identical to the AMD extensions, meaning they are software compatible. It seems for the first time that Intel has unarguably followed AMD's lead in the development of PC architecture. To make 64-bit computing a reality, 64-bit operating systems and 64-bit drivers are also needed. Microsoft began providing trial versions of Windows XP Professional x64 Edition (which supports AMD64 and EM64T) in April 2005, and major computer vendors now offer systems with Windows XP Professional x64 already installed. Major hardware vendors have also developed 64-bit drivers for current and recent hardware. Linux is also available in 64-bitcompatible versions, making the move to 64-bit computing possible.

The latest development is the introduction of dual-core processors from Intel, IBM, Sun and AMD. Dual-core processors have two full CPU cores operating off of one CPU package in essence enabling a single processor to perform the work of two processors. Although dual-core processors don't make games (which use single execution threads and are usually not run with other applications) play faster, dual-core processors, like multiple single-core processors, split up the workload caused by running multiple applications at the same time. If you've ever tried to scan for viruses while checking email or running another application, you've probably seen how

running multiple applications can bring even the fastest processor to its knees. With dual-core processors available from both Intel and AMD, your ability to get more work done in less time by multitasking is greatly enhanced. Current dual-core processors also support AMD64 or EM64T 64-bit extensions, enabling you to enjoy both dual-core and 64-bit computing's advantages.

PCs have certainly come a long way. The original 8088 processor used in the first PC contained 29,000 transistors and ran at 4.77MHz. The AMD Athlon 64FX has more than 105 million transistors, while the Pentium 4 670 (Prescott core) runs at 3.8GHz and has 169 million transistors thanks to its 2MB L2 cache. Dual-core processors, which include two processor cores and cache memory in a single physical chip, have even higher transistor counts: The Intel Pentium D processor has 230 million transistors, and the AMD Athlon 64 X2 includes over 233 million transistors. As dual-core processors and large L2 caches continue to be used in more and more designs, look for transistor counts and real-world performance to continue to increase. And the progress doesn't stop there because, according to Moore's Law, processing speed and transistor counts are doubling every 1.52 years.

INTEL 8085

Intel 8085 is an 8-bit, N-channel Metal Oxide semiconductor (NMOS) microprocessor. It is a 40 pin IC package fabricated on a single Large Scale Integration (LSI) chip. The Intel 8085 uses a single +5V DC supply for its operation. Its clock speed is about 3MHz. The clock cycle is of 320 ns. The time for the clock cycle of the Intel 8085 is 200 ns. It has 80 basic instructions and 246 opcodes. The 8085 is an enhanced version of its predecessor, the 8080A; its instruction set is upward compatible with that of the 8080A, meaning that 8085 instruction set includes all the 8080A instructions plus some additional ones. Programs written for 8080A will be executed by 8085, but the 8085 and 8080A are not pin compatible.

8085 System Bus

Typical system uses a number of busses, collection of wires, which transmit binary numbers, one bit per wire. A typical microprocessor communicates with memory and other devices (input and output) using three busses: Address Bus, Data Bus and Control Bus.

Fig 3 Bus Organization

Address Bus: One wire for each bit, therefore 16 bits = 16 wires. Binary number carried alerts memory to 'open' the designated box. Data (binary) can then be put in or taken out.The Address Bus consists of 16 wires, therefore 16 bits. Its "width" is 16 bits. A 16 bit binary number allows 216 different numbers, or 32000 different numbers, ie 0000000000000000 up to 1111111111111111. Because memory consists of boxes, each with a unique address, the size of the address bus determines the size of memory, which can be used. To communicate with memory the microprocessor sends an address on the address bus, eg 0000000000000011 (3 in decimal), to the memory. The memory the selects box number 3 for reading or writing data. Address bus is unidirectional, ie numbers only sent from microprocessor to memory, not other way.

Question?: If you have a memory chip of size 256 kilobytes (256 x 1024 x 8 bits), how many wires does the address bus need, in order to be able to specify an address in this memory?

Note: the memory is organized in groups of 8 bits per location, therefore, how many locations must you be able to specify?

Data Bus: Data Bus: carries 'data', in binary form, between μP and other external units, such as memory. Typical size is 8 or 16 bits. Size determined by size of boxes in memory and μP size helps determine performance of μP. The Data Bus typically consists of 8 wires. Therefore, 28 combinations of binary digits. Data bus used to transmit "data", ie information, results of arithmetic, etc, between memory and the microprocessor. Bus is bi-directional. Size of the data bus determines what arithmetic can be done. If only 8 bits wide then largest number is 11111111 (255 in decimal). Therefore, larger number have to be broken down into chunks of 255. This

slows microprocessor. Data Bus also carries instructions from memory to the microprocessor. Size of the bus therefore limits the number of possible instructions to 256, each specified by a separate number.

Control Bus: Control Bus are various lines which have specific functions for coordinating and controlling uP operations. Eg: Read/NotWrite line, single binary digit. Control whether memory is being 'written to' (data stored in mem) or 'read from' (data taken out of mem) 1 = Read, 0 = Write. May also include clock line(s) for timing/synchronising, 'interrupts', 'reset' etc. Typically μP has 10 control lines. Cannot function correctly without these vital control signals. The Control Bus carries control signals partly unidirectional, partly bi-directional. Control signals are things like "read or write". This tells memory that we are either **reading from** a location, specified on the address bus, or **writing to** a location specified. Various other signals to control and coordinate the operation of the system. Modern day microprocessors, like 80386, 80486 have much larger busses. Typically 16 or 32 bit busses, which allow larger number of instructions, more memory location, and faster arithmetic. Microcontrollers organized along same lines, except: because microcontrollers have memory etc inside the chip, the busses may all be internal. In the microprocessor the three busses are external to the chip (except for the internal data bus). In case of external busses, the chip connects to the busses via buffers, which are simply an electronic connection between external bus and the internal data bus.

Input/Output Devices

The input/output unit allows the microprocessor to **communicate** with the outside world, either to receive or to send data. Most of the time, the input/output unit will also act as an **interface** for the microprocessor, that is to convert the data into a suitable format for the microprocessor. Data can be in the form of parallel (8 bit) or serial format (single line).

Input devices are devices that input data or send data to the computer. Input devices are such as keyboard, punched card readers, sensors, switches, etc.

Output devices are devices that output data or perform various operations under the control of

the CPU. Output devices are LEDs, 7-segment display unit, speaker, CRT, printer, digital speedometer, fuel injectors, etc.

Memory

Memory is the term used to the various **storage devices** in which are used to store the programs and data for the microprocessor. These storage devices are made of semiconductor devices, and also known as Primary Storage Devices.

Types of memories:

• RAM

•ROM

RAM is a Random Access Memory or Read/Write Memory. One can read and write in RAM.

RAM is of two types:

• SRAM

• DRAM.

SRAM : It is also known as static RAM. In SRAM data will remain stored temporary as long as power is supplied. The basic cell in SRAM is a flip-flop.

DRAM : It is also known as dynamic RAM. In DRAM the basic cell is a capacitor.

ROM : ROM is the Read Only Memory. It is a permanent storage device. It is used in computers, microprocessors.

ROM is of four types

(a) Masked ROM

(b) PROM

(G) EPROM

(d) EEPROM.

(a) Masked ROM : It is stored permanently through photomasking during fabrication. This programming is done through masking and metalization process.

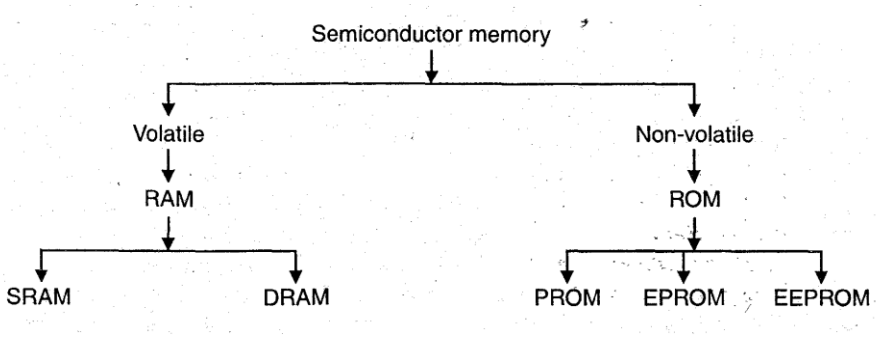
(b) PROM : It is the programmable read only memory.

(c) EPROM : It is the erasable programmable read only memory.

(d) EEPROM : It is the electrically erasable programmable read only memory

Semiconductor memory: Semiconductor memory is used to store data. The main advantages of semiconductor memories are

1. Small in size
2. Low cost
3. Better reliability
4. High speed.



Volatile Memory: If the information stored in a memory is lost when the electrical power is switched OFF, then the memory is called as volatile memory. These are called RAM (Random Access Memories).

Non-volatile Memory: If the information once stored in memory does not change unless altered deliberately are called as non-volatile memory. These are ROM, PROM, EPROM etc.

The semiconductor memory is of 2 types that is Read Only Memory (ROM) and Read Write Memory (RWM). RWM is popularly known as Random Access Memory (RAM).

1. Read Only Memory (ROM): It is used to store programs and data that need not to be altered,

i.e. permanent storage. Programs and data stored in ROMs can only be read by the CPU. Special equipment is used to write programs and data into the ROMs and is called EPROM Programmer. The **monitor program** is normally stored in the ROM. Monitor program is actually the 'resource manager' of the microcomputer system, similar to DOS or Windows in a personal computer. An example of a EPROM chip is the 2764 (8K X 8).

Fig 4 RWM and ROM

2. Read Write Memory (RWM): It is used to store user programs and data, and can be altered at any time, i.e. temporary storage. The information stored in RAM or RWM can be easily read and altered by the CPU. The contents (data or programs) stored is lost if power supply to this chip is turned off. An example of a RWM chip is the CMOS 6116 (2K X 8).

Encoder

An **encoder** is a device, circuit, transducer, software program, algorithm or person that converts information from one format or code to another, for the purposes of standardization, speed, secrecy, security, or saving space by shrinking size.

A **decoder** is a device which does the reverse of an encoder, undoing the encoding so that the original information can be retrieved. The same method used to encode is usually just reversed in order to decode.

In digital electronics, a decoder can take the form of a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different. e.g. n-to-2^n, binary-coded decimal decoders. Enable inputs must be on for the decoder to function, otherwise its outputs assume a single "disabled" output code word. Decoding is necessary in applications such as data multiplexing, 7 segment display and memory address decoding.

The example decoder circuit would be an AND gate because the output of an AND gate is

"High" (1) only when all its inputs are "High." Such output is called as "active High output". If instead of AND gate, the NAND gate is connected the output will be "Low" (0) only when all its inputs are "High". Such output is called as "active low output".

A slightly more complex decoder would be the n-to-2^n type binary decoders. These type of decoders are combinational circuits that convert binary information from 'n' coded inputs to a maximum of 2^n unique outputs. We say a *maximum* of 2^n outputs because in case the 'n' bit coded information has unused bit combinations, the decoder may have less than 2^n outputs. We can have 2-to-4 decoder, 3-to-8 decoder or 4-to-16 decoder. We can form a 3-to-8 decoder from two 2-to-4 decoders (with enable signals).

Similarly, we can also form a 4-to-16 decoder by combining two 3-to-8 decoders. In this type of circuit design, the enable inputs of both 3-to-8 decoders originate from a 4th input, which acts as a selector between the two 3-to-8 decoders. This allows the 4th input to enable either the top or bottom decoder, which produces outputs of D(0) through D(7) for the first decoder, and D(8) through D(15) for the second decoder.

A decoder that contains enable inputs is also known as a decoder-demultiplexer. Thus, we have a 4-to-16 decoder produced by adding a 4th input shared among both decoders, producing 16 outputs.

In electronics, a **multiplexer** or **mux** (occasionally the terms **muldex** or **muldem** are also found for a combination multiplexer-demultiplexer) is a device that performs multiplexing; it selects one of many analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2^n inputs has n select lines, which are used to select which input line to send to the output.

An electronic multiplexer makes it possible for several signals to share one device or resource, for example one A/D converter or one communication line, instead of having one device per input signal. On the other end, a demultiplexer (or **demux**) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input. A

multiplexer is often used with a complementary demultiplexer on the receiving end.

An electronic multiplexer can be considered as a multiple-input, single-output switch, and a demultiplexer as a single-input, multiple-output switch. The schematic symbol for a multiplexer is an isosceles trapezoid with the longer parallel side containing the input pins and the short parallel side containing the output pin. The schematic on the right shows a 2-to-1 multiplexer on the left and an equivalent switch on the right. The *sel* wire connects the desired input to the output.

In telecommunications, a multiplexer is a device that combines several input information signals into one output signal, which carries several communication channels, by means of some multiplex technique. A **demultiplexer** is in this context a device taking a single input signal that carries many channels and separates those over multiple output signals.

In telecommunications and signal processing, an analog time division multiplexer (TDM) may take several samples of separate analogue signals and combine them into one pulse amplitude modulated (PAM) wide-band analogue signal. Alternatively, a digital TDM multiplexer may combine a limited number of constant bit rate digital data streams into one data stream of a higher data rate, by forming data frames consisting of one timeslot per channel.

In telecommunications, computer networks and digital video, a statistical multiplexer may combine several variable bit rate data streams into one constant bandwidth signal, for example by means of packet mode communication. An inverse multiplexer may utilize several communication channels for transferring one signal.

Terminologies used in microprocessor (Beyond The Syllabus)

- Digital Computer: A programmable machine that processes binary data. It is traditionally represented by five components: CPU, ALU plus control unit, memory, input and output.
- Memory: A medium that stores binary information.
- Microprocessor: A semiconductor device manufactured by using the LSI technique. It includes the ALU, register arrays and control circuits on a single chip. The term MPU is

also synonymous with the microprocessor.

- Microcontroller: A device that includes microprocessor, memory and I/O signal lines on a single chip fabricated using VLSI technology.
- Bus: A group of lines used to transfer bits between the microprocessor and other components of the computer system.
- Bit: A binary digit 0 or 1.
- Byte: A group of 8 bits.
- Nibble: A group of 4 bits.
- Word: A group of bits the computer recognizes and processes at a time.
- Instruction: A command in binary that is recognized and executed by the computer to accomplish a task.
- Mnemonics: A combination of letters to suggest the operation of an instruction.
- Assembly Language: A medium of communication with a computer in which programs are written in mnemonics. It is specific to a given computer.

8085 MICROPROCESSOR ARCHITECTURE

8085 Microprocessor

The salient features of 8085 μp are:

• It is an 8 bit microprocessor.

• It is manufactured with N-MOS technology.

• It has 16-bit address bus and hence can address up to 216 = 65536 bytes (64KB) memory locations through A_0-A_{15}.

• The first 8 lines of address bus and 8 lines of data bus are multiplexed $AD_0 – AD_7$.

• Data bus is a group of 8 lines $D_0 – D_7$.

• It supports external interrupt request.

• A 16 bit program counter (PC)

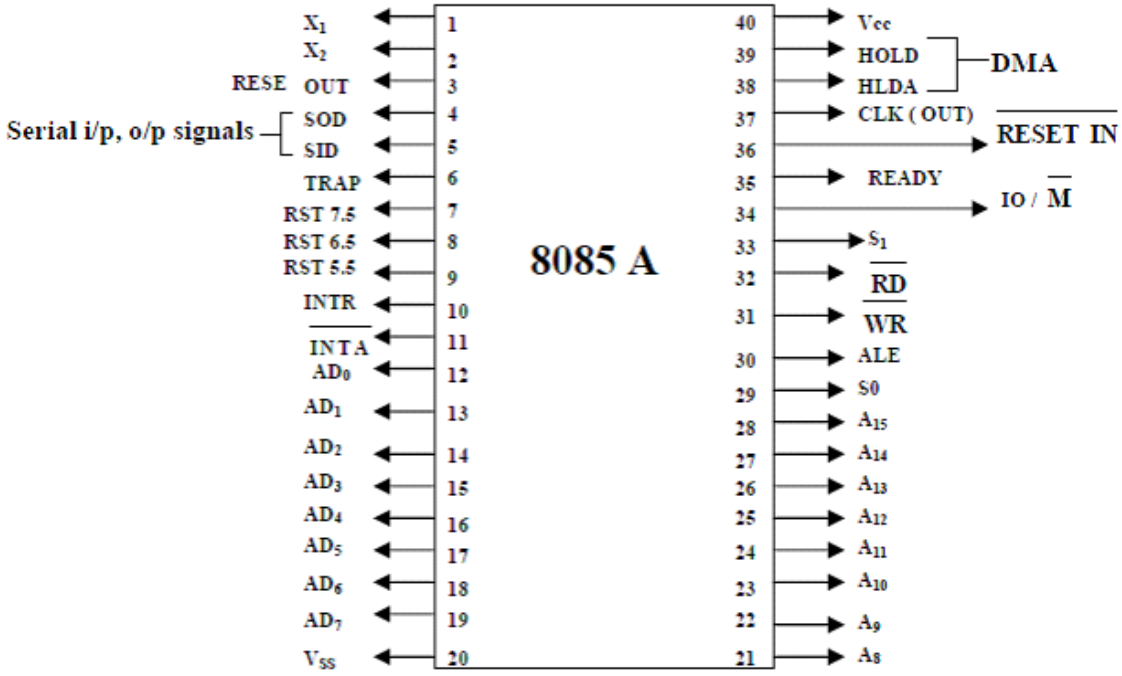
• A 16 bit stack pointer (SP)

• Six 8-bit general purpose register arranged in pairs: BC, DE, HL.

• It requires a signal +5V power supply and operates at 3.2 MHZ single phase clock.

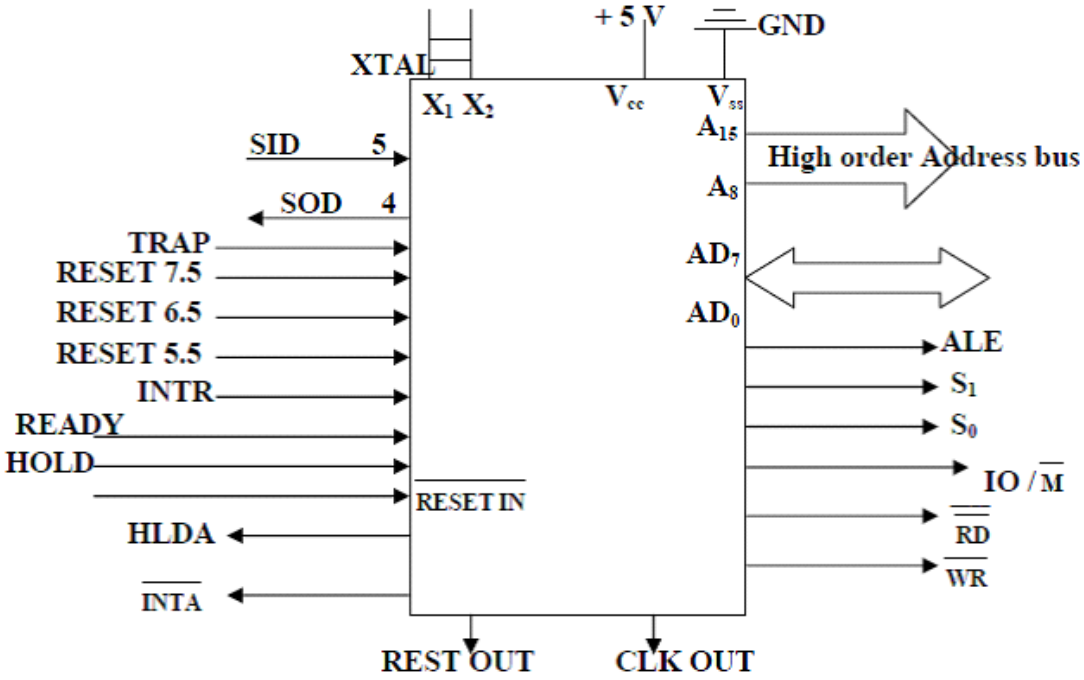
• It is enclosed with 40 pins DIP (Dual in line package).

Functional Block Diagram



Pin Diagram of 8085

Signal Group of 8085



The following describes the function of each pin:

A6 - A1s (Output 3 State): Address Bus; The most significant 8 bits of the memory address or the 8 bits of the I/0 address,3 stated during Hold and Halt modes.

AD0 - 7 (Input/Output 3state): Multiplexed Address/Data Bus; Lower 8 bits of the memory address (or I/0 address) appear on the bus during the first clock cycle of a machine state. It then becomes the data bus during the second and third clock cycles. 3 stated during Hold and Halt modes.

ALE (Output): Address Latch Enable: It occurs during the first clock cycle of a machine state and

enables the address to get latched into the on chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. ALE can also be used to strobe the status information. ALE is never 3stated.

SO, S1 (Output): Data Bus Status. Encoded status of the bus cycle:

S1	S0	
O	O	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

S1 can be used as an advanced R/W status.

RD (Output 3state): READ; indicates the selected memory or 1/0 device is to be read and that the Data Bus is available for the data transfer.

WR (Output 3state): WRITE; indicates the data on the Data Bus is to be written into the selected memory or 1/0 location. Data is set up at the trailing edge of WR. 3 stated during Hold and Halt modes.

READY (Input): If Ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

HOLD (Input): HOLD; indicates that another Master is requesting the use of the Address and Data Buses. The CPU, upon receiving the Hold request will relinquish the use of buses as soon as the completion of the current machine cycle. Internal processing can continue. The processor can regain the buses only after the Hold is removed. When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are 3stated.

HLDA (Output): HOLD ACKNOWLEDGE; indicates that the CPU has received the Hold request and that it will relinquish the buses in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycle after HLDA goes low.

INTR (Input): INTERRUPT REQUEST; is used as a general purpose interrupt. It is sampled only

during the next to the last clock cycle of the instruction. If it is active, the Program Counter (PC) will be inhibited from incrementing and an INTA will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

INTA (Output): INTERRUPT ACKNOWLEDGE; is used instead of (and has the same timing as) RD during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.

RST 5.5

RST 6.5 - (Inputs)

RST 7.5

RESTART INTERRUPTS; These three inputs have the same timing as I NTR except they cause an internal RESTART to be automatically inserted.

RST 7.5 Highest Priority

RST 6.5

RST 5.5 Lowest Priority

The priority of these interrupts is ordered as shown above. These interrupts have a higher priority than the INTR.

TRAP (Input): Trap interrupt is a nonmaskable restart interrupt. It is recognized at the same time as INTR. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt.

RESET IN (Input): Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flipflops. None of the other flags or registers (except the instruction register) are affected

The CPU is held in the reset condition as long as Reset is applied.

RESET OUT (Output): Indicates CPU is being reset. Can be used as a system RESET. The signal is synchronized to the processor clock.

X1, X2 (Input): Crystal or R/C network connections to set the internal clock generator X1 can also be an external clock input instead of a crystal. The input frequency is divided by 2 to give the internal operating frequency.

CLK (Output): Clock Output for use as a system clock when a crystal or R/ C network is used as an input to the CPU. The period of CLK is twice the X1, X2 input period.

IO/M (Output): IO/M indicates whether the Read/Write is to memory or l/O Tristated during Hold and Halt modes.

SID (Input): Serial input data line The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

SOD (output): Serial output data line. The output SOD is set or reset as specified by the SIM instruction.

Vcc: +5 volt supply.

Vss: Ground Reference.

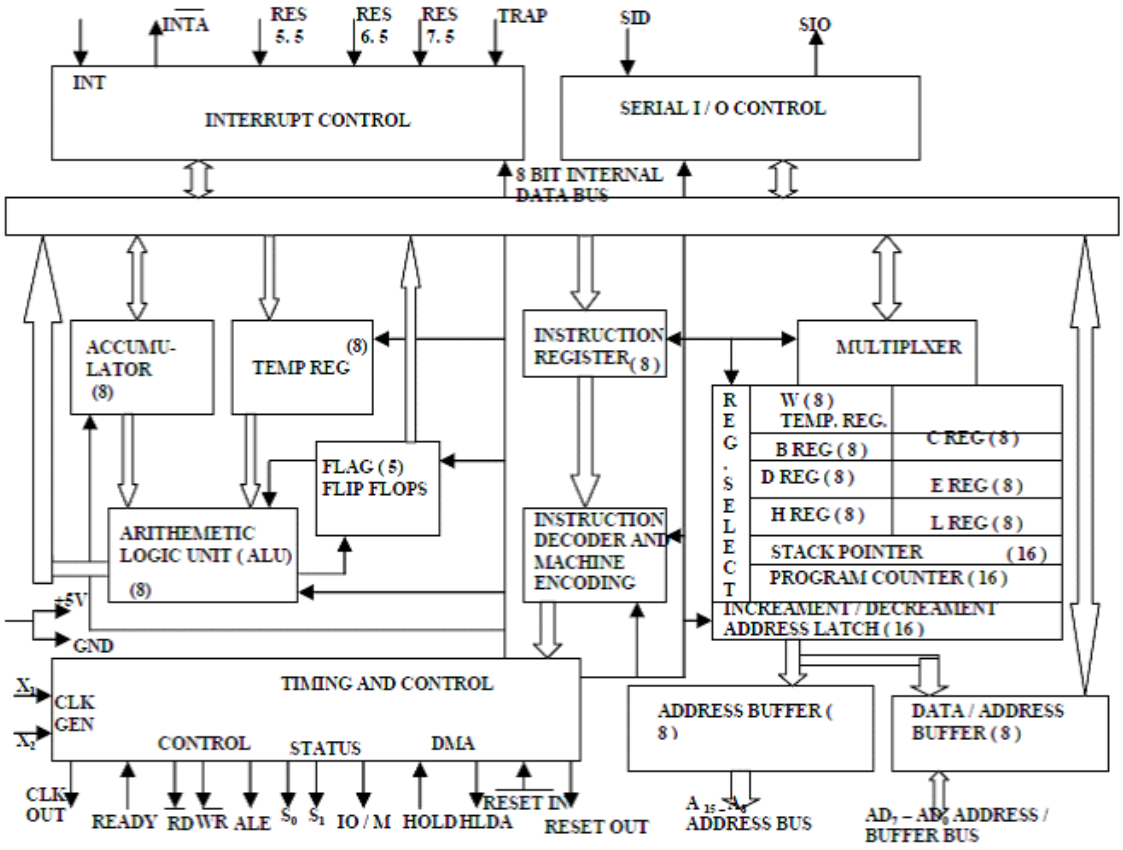


Fig Internal Block Diagram

Register Organization

The 8085 microprocessor has different types of registers. It includes six , 8 – bit registers (B, C, D. E, H and L), one 8-bit Accumulator and two 16-bit registers (SP and PC). Also there are two 8-bit temporary registers W and Z. Among these registers W and Z are not accessible to the user; they are used by the processor for internal, intermediate operations. The remaining registers are accessible to the user. The organization of 8085 registers is shown in Fig 1

The various registers of 8085 are classified into three types. They are

(i).Temporary registers.

(ii).General purpose registers

(iii).Special purpose registers.

(i) Temporary registers:

The Temporary registers are temporary data registers, W register and Z register. All are 8-bit registers. The temporary data register is associated with the ALU operations. One of the operand is stored in this register. This is not accessible to user.

Similarly W and Z are also temporary registers used to hold 8-bit data during execution of certain instructions.As these registers are internally used by the CPU, they are not accessible to the user.

The W and Z registers are used by the processor during CALL instruction. When a CALL instruction is encountered in any program, the current Program counter (PC) contents are pushed on to the stack and the given address is loaded on to PC. The given address is temporarily stored in W and Z registers and placed on the bus for the fetch cycle. Thus the program control is transferred to the address given in the instruction.

Another example is, during the execution of XCHG instruction, the contents of H-L pair are exchanged with D-E pair. At the time of exchange W and Z registers are used for temporary storage of data.

(ii) General purpose registers:

B, C, D, E, H and L are six; 8-bit general purpose registers to store data. These registers can be used as separate 8-bit registers and also can be paired as 16-bit registers to store the address of a memory location. But they must be paired as B-C; D-E and H-L register pairs only as shown below.

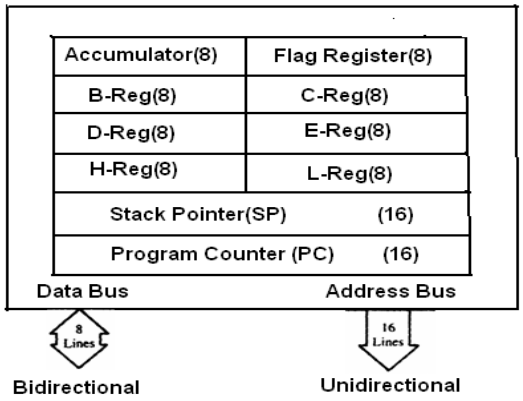


Fig 1 Register organization

When used as pair, for example B-C, the higher order byte moves to the first register (B) and the low order byte moves to the second register (C). The H-L pair also functions as a data pointer or memory pointer

For Ex: LXI H, 8500 H.

This will load immediately the address of memory location (8500H) in to H-L pair .Now the H-L pair points to the location 8500 H.

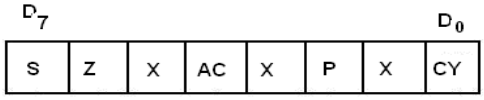
(iii) Special purpose registers:

The Special purpose registers, as their name indicates, are used for some specific purpose. The Special purpose registers are Accumulator (A), Flag Register, Instruction Register(IR), Program Counter (PC) and Stack Pointer (SP).

Accumulator (Register A): It is an 8-bit tri-state register. It is mainly used for arithmetic, logic, load and store operations. It is also used in I/O operations. In most of operations, the result is stored in Accumulator after execution.

Flag Register: It is an 8-bit register, Which consists of only five flags.Each flag bit is a flip flop that indicates either a set or reset state. The five flags are Sign, Zero, Auxiliary carry, Parity and

Carry as shown below



Here X means undefined.

Sign Flag: The sign flag is set to1 if the most significant bit of the result of an arithmetic or logic operations is 1. Else it is reset (0).

Zero Flag: The Zero status flag is set to 1 if the result of an arithmetic or logic operation is Zero For non-Zero result it is reset to 0.G

Auxiliary carry Flag: This flag is set if there is a carry from 3^{rd} bit to 4^{th} bit during BCD operations (carry from lower nibble to higher nibble). This flag is not accessible to the user.

Parity Flag: Parity is defined by the number of 1s present in a binary number stored in A register. After any arithmetic or logical operation, if the result has an even number of 1s it is called even parity and the Parity Flag is set to 1. Otherwise. i.e. If there is odd number of 1s in the result, it is called Odd Parity and the Parity flag is set 0.

Program Counter (PC):

It is a 16-bit special purpose register, which stores the address of the next instruction to be fetched or executed. The execution of a program is initiated by loading the PC by the address of the first instruction of the program. Once the first instruction is executed, the PC is automatically incremented to point to the next instruction unless a jump to some specific address occurs. This process is repeated till the last instruction of the program.

In case of JUMP or CALL instructions, current address is stored in the Program Counter. The processor then fetches the next instruction from the new address specified by the JUMP or CALL instruction. In conditional JUMP and conditional CALL instructions, if the condition is

not satisfied, the processor increments the Program Counter by three so that it points the instruction followed by the conditional JUMP or CALL instruction. Otherwise the processor fetches the next instruction from the new address specified by JUMP or CALL instruction.

Stack Pointer (SP):

It is a 16-bit special purpose register which always stores the address of top of the Stack. i.e. it always points to top of the Stack. Stack is a part of the memory location used to store the data temporarily. A stack works on Last in First out (LIFO) basis. As the Stack pointer always points to the top of the Stack, only top of the Stack of the memory can be accessed. When a Write operation (PUSH) takes place, the contents of the stack pointer is decremented by two so that the SP points to the new location. Similarly when the Read operation (POP) occurs, the Stack pointer is incremented by two to point to the next data on top of the Stack.

The Stack Pointer is initialized by load register pair immediate instruction.

Ex: LXI SP, 8530 H

Here 8530 H is the 16 bit address of the top of Stack location.

Instruction Register and Decoder: The instruction register and the decoder are also part of the ALU. When an instruction is fetched from memory, it is loaded in the instruction register. The Decoder decodes the instruction and develops the sequence of events to follow. The instruction register is a 8 – bit special register, but it is not a programmable and is not accessible to the user. The instruction decoder decodes the instruction at a binary level and sends the appropriate signals to the control unit.

Memory

Program, data and stack memories occupy the same memory space. The total addressable memory size is 64 KB.

Program memory - program can be located anywhere in memory. Jump, branch and call instructions use 16-bit addresses, i.e. they can be used to jump/branch anywhere within 64 KB. All jump/branch instructions use absolute addressing.

Data memory - the processor always uses 16-bit addresses so that data can be placed anywhere.
Stack memory is limited only by the size of memory. Stack grows downward.
First 64 bytes in a zero memory page should be reserved for vectors used by RS instructions.

Interrupts

The processor has 5 interrupts. They are presented below in the order of their priority (from lowest to highest):

INTR is maskable 8080A compatible interrupt. When the interrupt occurs the processor fetches from the bus one instruction, usually one of these instructions:

One of the 8 RST instructions (RST_0 - RST_7). The processor saves current program counter into stack and branches to memory location N * 8 (where N is a 3-bit number from 0 to 7 supplied with the RST instruction).

CALL instruction (3 byte instruction). The processor calls the subroutine, address of which is specified in the second and third bytes of the instruction.

RST5.5 is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 2CH (hexadecimal) address.

RST6.5 is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 34H (hexadecimal) address.

RST7.5 is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 3CH (hexadecimal) address.

TRAP is a non-maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 24H (hexadecimal) address.

All maskable interrupts can be enabled or disabled using EI and DI instructions.

RST 5.5, RST6.5 and RST7.5 interrupts can be enabled or disabled individually using SIM instruction.

CISC: Complex Instruction Set Computers

Earlier developments were based around the idea that making the CPU more complex and supporting a larger number of potential instructions would lead to increased performance. This idea is at the root of CISC processors, such as the Intel x 86 ranges, which have very large instruction sets reaching up to and above three hundred separate instructions. They also have increased complexity in other areas, with many more specialised addressing modes and registers also being implemented, and variable length of the instruction codes themselves.

Performance was improved here by allowing the simplification of program compilers, as the range of more advanced instructions available led to less refinement having to be made at the compilation process. However, the complexity of the processor hardware and architecture that resulted can cause such chips to be difficult to understand and program for, and also means they can be expensive to produce.

RISC: Reduced Instruction Set Computers

In opposition to CISC, the mid-1980s saw the beginnings of the RISC philosophy. The idea here was that the best way to improve performance would be to simplify the processor workings as much as possible. RISC processors, such as the IBM PowerPC processor, have a greatly simplified and reduced instruction set, numbering in the region of one hundred instructions or less. Addressing modes are simplified back to four or less, and the length of the codes is fixed in order to allow standardisation across the instruction set.

Changing the architecture to this extent means that less transistors are used to produce the processors. This means that RISC chips are much cheaper to produce than their CISC counterparts. Also the reduced instruction set means that the processor can execute the instructions more quickly, potentially allowing for greater speeds. However, only allowing such simple instructions means a greater burden is placed upon the software itself. Less instructions in the instruction set means a greater emphasis on the efficient writing of software with the

instructions that *are* available. Supporters of the CISC architecture will point out that their processors are of good enough performance and cost to make such efforts not worth the trouble.

CISC		**RISC**
Large (100 to 300)	*Instruction Set*	Small (100 or less)
Complex (8 to 20)	*Addressing Modes*	Simple (4 or less)
Specialised	*Instruction Format*	Simple
Variable	*Code Lengths*	Fixed
Variable	*Execution Cycles*	Standard for most
Higher	*Cost / CPU Complexity*	Lower

Looking at the most modern processors, it becomes evident that the whole rivalry between CISC and RISC is now not of great importance. This is because the two architectures are converging closer to each other, with CPUs from each side incorporating ideas from the other. CISC processors now use many of the same techniques as RISC ones, while the reduced instruction sets of RISC processors contain similar numbers of instructions to those found in certain CISC chips. However, it is still important that you understand the ideas behind these two differing architectures, and why each design path was chosen.

Concept of multiplexing and demultiplexing of buses (Beyond The Syllabus)

The 8085 microprocessor is used **IC 74LS373** to latch the address of 8085. Basically latch is consists of 8 flip flops. Generally we used D-flip flops (Delay). The clock of these flip flops are connected together and available as a output pin called enable.

Working : The address will appear on AD0 AD7 lines. The ALE will go high and forcing **Enable = 1**. This will make latch enable and ready to work. Before address disappears ALE = 0. This will make latch disable. AD0 — AD7 will now be used as data bus.

Hence, AD0 — AD7 (low order) address bus of the 8085 microprocessor is multiplexed (time-shared) with the data bus. The buses need to be demultiplexed.

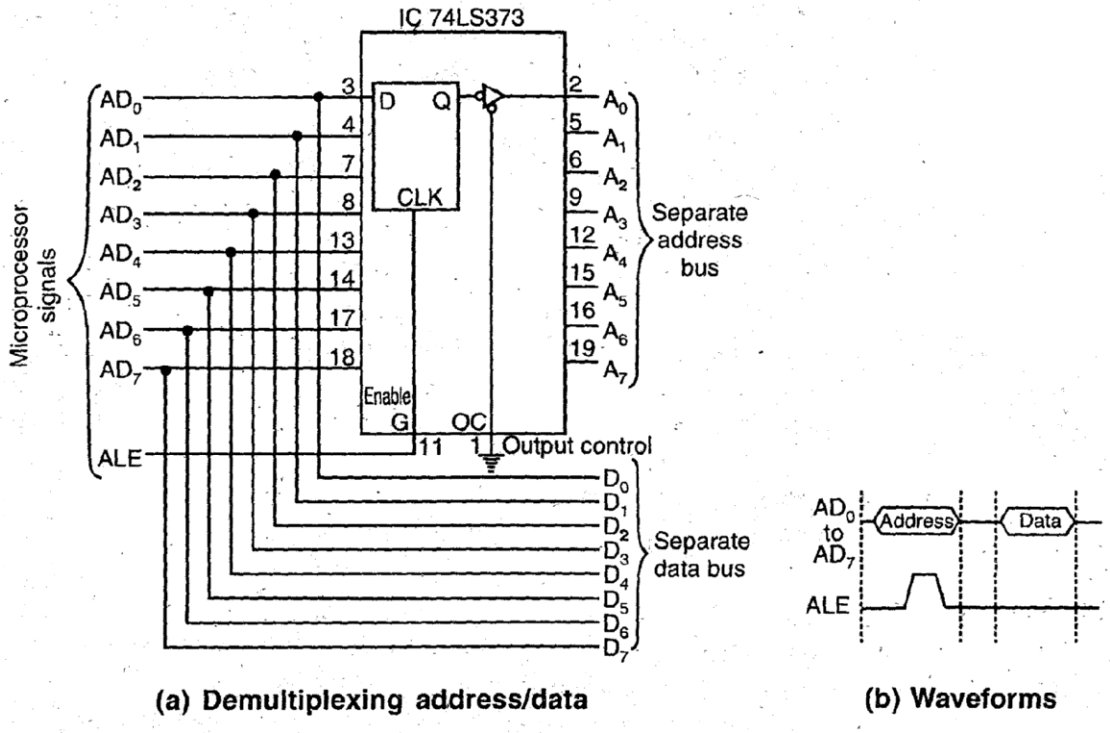


Fig 1.8 Demultiplexing of address and data bus

Basically, ALE signal give us the information that which of the bus data bus or address bus we are using

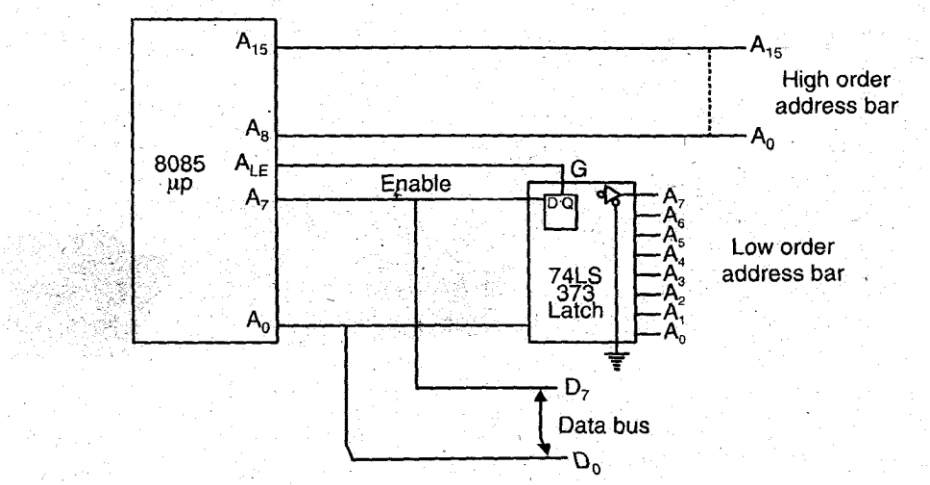


Figure 1.9 ALE Signal for demultiplexing

ALE signal is used to demultiplexing the bus

The bus AD7 — AD0 is connected as the input to latch 74LS373 ALE signal is attached to enable pin of the latch (4)

CC — output control ALE when high — latch is transparent output changes according to input data

To reduce the number of pins to make it compact one and easy to use data bus a" address bus is demultiplexed

Data bus — 8 bits

Address bus —* 16 bits

So 8 bits of address bus and data bus are multiplex and them ALE (Address I enable) is used to demultiplexed them.

INTRODUCTION

The microcontroller incorporates all the features that are found in microprocessor. The microcontroller has built in ROM, RAM, Input Output ports, Serial Port, timers, interrupts and clock circuit. A microcontroller is an entire computer manufactured on a single chip. Microcontrollers are usually dedicated devices embedded within an application. For example, microcontrollers are used as engine controllers in automobiles and as exposure and focus controllers in cameras. In order to serve these applications, they have a high concentration of on-chip facilities such as serial ports, parallel input output ports, timers, counters, interrupt control, analog-to-digital converters, random access memory, read only memory, etc. The I/O, memory, and on-chip peripherals of a microcontroller are selected depending on the specifics of the target application. Since microcontrollers are powerful digital processors, the degree of control and programmability they provide significantly enhances the effectiveness of the application. The 8051 is the first microcontroller of the MCS-51 family introduced by Intel Corporation at the end of the 1970s. The 8051 family with its many enhanced members enjoys the largest market share, estimated to be about 40%, among the various microcontroller architectures. The microcontroller has on chip peripheral devices. In this unit firstly we differentiate microcontroller from microprocessor then we will discuss about Hardware details of 8051 and then introduce the Assembly level language in brief.

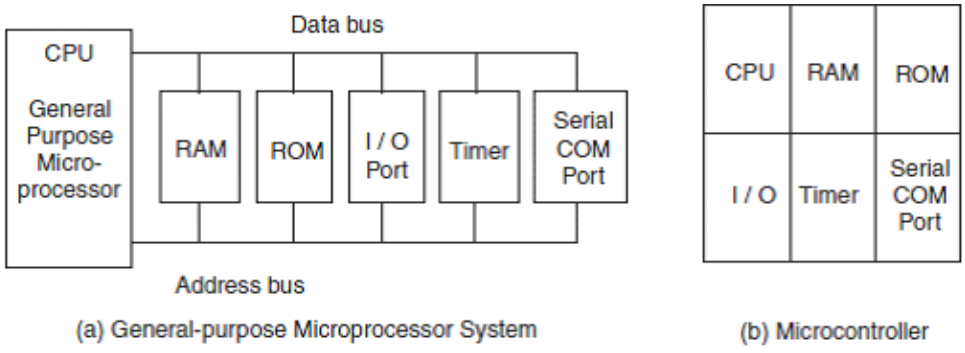
Microcontrollers

• Microcontroller (MC) may be called computer on chip since it has basic features of microprocessor with internal ROM, RAM, Parallel and serial ports within single chip. Or we can say microprocessor with memory and ports is called as microcontroller. This is widely used in washing machines, vcd player, microwave oven, robotics or in industries. • Microcontroller can be classified on the basis of their bits processed like 8bit MC, 16bit MC.

• 8 bit microcontroller, means it can read, write and process 8 bit data. Ex. 8051 microcontroller. Basically 8 bit specifies the size of data bus. 8 bit microcontroller means 8 bit data can travel on the data bus or we can read, write process 8 bit data.

DIFFERENCE BETWEEN MICROCONTROLLER AND MICROPROCESSOR

It is very clear from figure that in microprocessor we have to interface additional circuitry for providing the function of memory and ports, for example we have to interface external RAM for data storage, ROM for program storage, programmable peripheral interface (PPI) 8255 for the Input Output ports, 8253 for timers, USART for serial port. While in the microcontroller RAM, ROM, I/O ports, timers and serial communication ports are in built. Because of this it is called as "system on chip". So in micro-controller there is no necessity of additional circuitry which is interfaced in the microprocessor because memory and input output ports are inbuilt in the microcontroller. Microcontroller gives the satisfactory performance for small applications. But for large applications the memory requirement is limited because only 64 KB memory is available for program storage. So for large applications we prefer microprocessor than microcontroller due to its high processing speed.

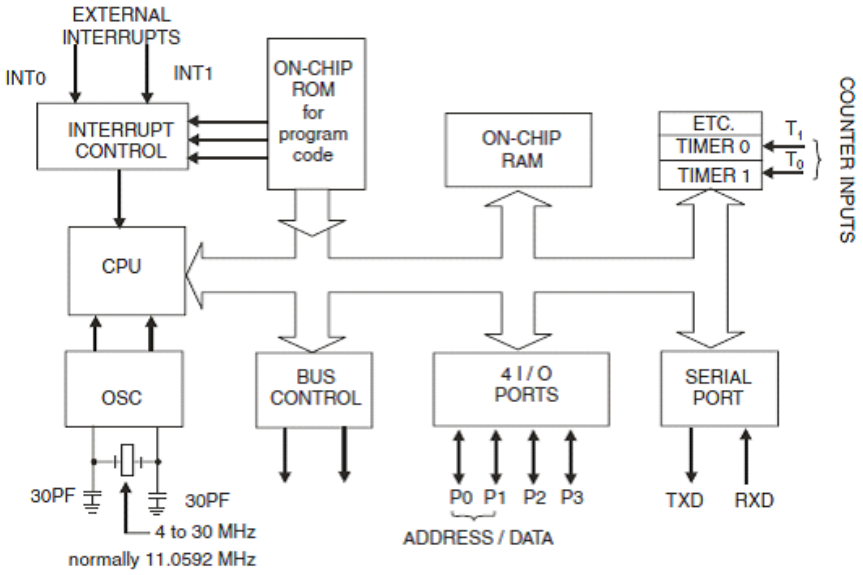


(a) General-purpose Microprocessor System

(b) Microcontroller

MICROCONTROLLER 8051 ARCHITECTURE

It is 8-bit microcontroller, means MC 8051 can Read, Write and Process 8 bit data. This is mostly used microcontroller in the robotics, home appliances like mp3 player, washing machines, electronic iron and industries. Mostly used blocks in the architecture of 8051 are as follows:



128 Byte RAM for Data Storage

MC 8051 has 128 byte Random Access memory for data storage. Random access memory is non volatile memory. During execution for storing the data the RAM is used. RAM consists of the register banks, stack for temporary data storage. It also consists of some special function register (SFR) which are used for some specific purpose like timer, input output ports etc. Normally microcontroller has 256 byte RAM in which 128 byte is used for user space which is normally Register banks and stack. But other 128 byte RAM which consists of SFRs. We will discuss the RAM in detail in next section. Now what is the meaning of 128 byte RAM. What are address range which is provided for data storage. We will discuss here.

We know that 128 byte = 2^7 byte

	7	6	5	4		3	2	1	0	
if initial address is	0	0	0	0		0	0	0	0	(00H)
then final or last address is	0	1	1	1		1	1	1	1	(7FH)
		7					F			

Since 2^7 bytes so last 7 bits can be changed so total locations are from 00H to 7F H. This procedure of calculating the memory address is called as **"*memory mapping*".** We can save data on memory locations from 00H to 7FH. Means total 128 byte space from 00H to 7FH is provided for data storage.

4KB ROM

• In 8051, 4KB read only memory (ROM) is available for program storage. This is used for permanent data storage. Or the data which is not changed during the processing like the program or algorithm for specific applications.

• This is volatile memory; the data saved in this memory does not disappear after power failure.

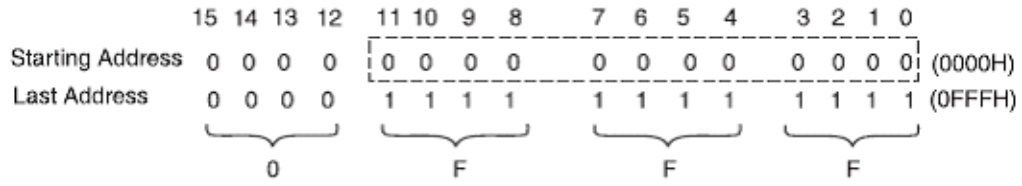
• We can interface up to 64KB ROM memory externally if the application is large. These sizes are specified different by their companies.

Address Range of PC: Address range of PC means program counter (which points the next instruction to be executing) can be moved between these locations or we can save the program from this location to this location.

The address range can be calculated in the same way just like the RAM which is discussed in previous section.

4KB = 22 ◊ 210 B(since 1KB = 210 B)

= 212 Byte



Address range of PC is 0000H to 0FFFH means total 4KB locations are available from 0000H to 0FFFH. At which we can save the program.

Difference between RAM and ROM

• RAM is used for data storage while ROM is used for program storage.

• Data of RAM can be changed during processing while data of ROM can't be changed during processing.

• We can take an example of calculator. If we want to perform addition of two numbers then we type the two numbers in calculator, this is saved in the RAM, but the Algorithms by which the calculation is performed is saved in the ROM. Data which is given by us to calculator can be changed but the

algorithm or program by which calculation is performed can't be changed.

Timers and Counters (Including Beyond The Syllabus)

Timer means which can give the delay of particular time between some events. For example on or off the lights after every 2 sec. This delay can be provided through some assembly program but in microcontroller two hardware pins are available for delay generation. These hardware pins can be also used for counting some external events. How much times a number is repeated in the given table is calculated by the counter.

• In MC8051, two timer pins are available T0 and T1, by these timers we can give the delay of particular time if we use these in timer mode.

• We can count external pulses at these pins if we use these pins in counter mode.

• 16 bits timers are available. Means we can generate delay between 0000H to FFFFH.

• Two special function registers are available.

• If we want to load T0 with 16 bit data then we can load separate lower 8 bit in TL0 and higher 8 bit in TH0.

• In the same way for T1.

• TMOD, TCON registers are used for controlling timer operation.

Serial Port

• There are two pins available for serial communication TXD and RXD.

• Normally TXD is used for transmitting serial data which is in SBUF register, RXD is used for receiving the serial data.

• SCON register is used for controlling the operation.

• There are four modes of serial communication

Input Output Ports

• There are four input output ports available P0, P1, P2, P3.

• Each port is 8 bit wide and has special function register P0, P1, P2, P3 which are bit addressable means each bit can be set or reset by the Bit instructions (SETB for high, CLR for low) independently.

• The data at any port which is transmitting or receiving is in these registers.

• The port 0 can perform dual works. It is also used as Lower order address bus (A0 to A7) multiplexed with 8 bit data bus P0.0 to P0.7 is AD0 to AD7 respectively the address bus and data bus is demultiplex by the ALE signal and latch which is further discussed in details.

• Port 2 can be used as I/O port as well as higher order address bus A8 to A15.

• Port 3 also have dual functions it can be worked as I/O as well as each pin of P3 has specific function.

P3.0 – RXD – {Serial I / P for Asynchronous communication Serial O / P for synchronous communication.

P3.1 – TXD – Serial data transmit.

P3.2 – INT0 – External Interrupt 0.

P3.3 – INT1 – External Interrupt 1.

P3.4 – T0 – Clock input for counter 0.

P3.5 – T1 – Clock input for counter 1.

P3.6 – WR – Signal for writing to external memory.

P3.7 – RD – Signal for reading from external memory.

When external memory is interfaced with 8051 then P0 and P2 can't be worked as I/O port they works as address bus and data bus, otherwise they can be accessed as I/O ports.

Oscillator

• It is used for providing the clock to MC8051 which decides the speed or baud rate of MC.

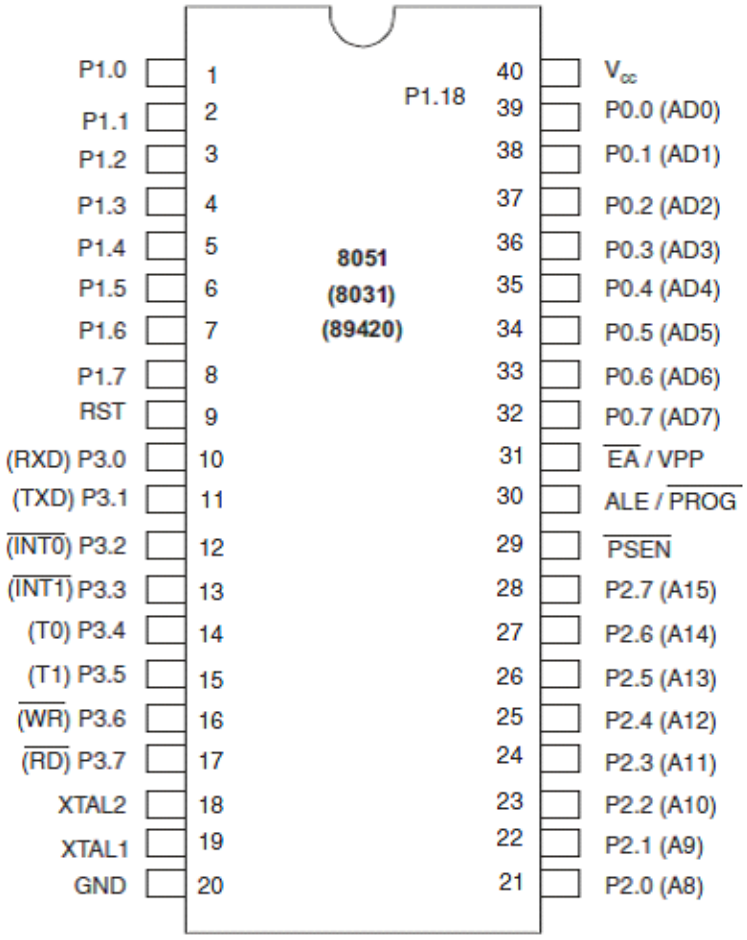
• We use crystal which frequency vary from 4MHz to 30 MHz, normally we use 11.0592 MHz frequency.

Interrupts

• Interrupts are defined as requests because they can be refused (masked) if they are not used, that is when an interrupt is acknowledged. A special set of events or routines are followed to handle the interrupts. These special routines are known as interrupt handler or interrupt service routines (ISR). These are located at a special location in memory.

• INT0 and INT1 are the pins for external interrupts.

PIN DIAGRAM OF 8051



Description of each pin is discussed here:

• VCC → 5V supply

• VSS → GND

• XTAL2/XTALI are for oscillator input

• Port 0 – 32 to 39 – AD0/AD7 and P0.0 to P0.7

• Port 1 – 1 to 8 – P1.0 to P1.7

• Port 2 – 21 to 28 – P2.0 to P2.7 and A 8 to A15

• Port 3 – 10 to 17 – P3.0 to P3.7

• P 3.0 – RXD – Serial data input – SBUF

• P 3.1 – TXD – Serial data output – SBUF

• P 3.2 – BAR INT0 – External interrupt 0 – TCON 0.1

• P 3.3 – BAR INT1 – External interrupt 1 – TCON 0.3

• P 3.4 – T0 – External timer 0 input – TMOD

• P 3.5 – T1 – External timer 1 input – TMOD

• P 3.6 –BAR WR – External memory write cycle – Active LOW

• P 3.7 – BAR RD – External memory read cycle – Active LOW

• RST – for Restarting 8051

• ALE – Address latch enable

1 – Address on AD 0 to AD 7

0 – Data on AD 0 to AD 7

• BAR PSEN – Program store enable

Interrupt

An interrupt is an external or internal event that interrupts the microcontroller to inform it that a device needs its service. A single microcontroller can serve several devices by two ways Interrupts whenever any device needs its service, the device notifies the microcontroller by sending it an interrupt signal. Upon receiving an interrupt signal, the microcontroller interrupts whatever it is doing and serves the device. The program which is associated with the interrupt is called the interrupt service routine (ISR) or interrupt handler.

Polling

- The microcontroller continuously monitors the status of a given device.
- When the conditions met, it performs the service.
- After that, it moves on to monitor the next device until every one is serviced.

Polling can monitor the status of several devices and serve each of them as certain conditions are

met. The polling method is not efficient, since it wastes much of the microcontroller's time by polling devices that do not need service.
ex. JNB TF,target

The advantage of interrupts is that the microcontroller can serve many devices (not all at the same time)

- Each device can get the attention of the microcontroller based on the assigned priority.
- For the polling method, it is not possible to assign priority since it checks all devices in a round-robin fashion.

The microcontroller can also ignore (mask) a device request for service. This is not possible for the polling method.
For every interrupt, there must be an interrupt service routine (ISR), or interrupt handler.

- When an interrupt is invoked, the microcontroller runs the interrupt service routine
- For every interrupt, there is a fixed location in memory that holds the address of its ISR
- The group of memory locations set aside to hold the addresses of ISRs is called interrupt vector table

Upon activation of an interrupt, the microcontroller goes through the following steps:
1. It finishes the instruction it is executing and saves the address of the next instruction (PC) on the stack.
2. It also saves the current status of all the interrupts internally (i.e: not on the stack).
3. It jumps to a fixed location in memory, called the interrupt vector table, that holds the address of the ISR.
4. The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it

- It starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine which is RETI (return from interrupt) .

5. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted.

- First, it gets the program counter (PC) address from the stack by popping the top two

bytes of the stack into the PC.

- Then it starts to execute from that address.

Six interrupts are allocated as follows:

- Reset – power-up reset
- Two interrupts are set aside for the timers:
  one for timer 0 and one for timer 1
- Two interrupts are set aside for hardware external interrupts
- P3.2 and P3.3 are for the external hardware interrupts INT0 (or EX1), and INT1 (or EX2)
- Serial communication has a single interrupt that belongs to both receive and transfer

Interrupt Vector Table:

Interrupt	ROM Location (hex)	Pin
Reset	0000	9
External HW (INT0)	0003	P3.2 (12)
Timer 0 (TF0)	000B	
External HW (INT1)	0013	P3.3 (13)
Timer 1 (TF1)	001B	
Serial COM (RI and TI)	0023	

Upon reset, all interrupts are disabled (masked), meaning that none will be responded to by the microcontroller if they are activated

- The interrupts must be enabled by software in order for the microcontroller to respond to them.
- There is a register called IE (interrupt enable) that is responsible for enabling (unmasking) and disabling (masking) the interrupts.

8085 Microprocessor Instructions

Instruction set of 8085

An Instruction is a command given to the microprocessor to perform a given task on specified data. Each instruction has two parts one is the task to be performed called the operation code (op-code) and the second is the data to be operated on, known as operand. The operand or data can be specified in various ways.

Instruction and data formats (Beyond The Syllabus)

The format of a typical instruction is composed of two parts: an operation code or op-code and an operand. Every instruction needs an op-code to specify what the operation of the instruction is and then an operand that gives the appropriate data needed for that particular operation code.
According to the word or byte size the 8085 instructions are classified into three types. They are
(a) One byte (single) instructions.
(b)Two byte instructions.
(c) Three byte instructions.

One–byte instructions: An instruction with only opcode and do not require any dat or address is called a one byte instruction.

Ex: 1. MOV C, A Hex code = 4FH (one byte)
2. ADD B Hex code = 80H (one byte)
3. CMA Hex code = 2FH (one byte)

Two–byte instructions: At wo byte instruction is one which contains an 8-bit op-code and 8-bit operand (Data).

Ex: 1. MVI A, 09 Hex code = 3E, 09 (two bytes)
2. ADD B, 07 Hex code = 80, 07 (two bytes)
3. SUB A, 05 Hex code = 97, 05 (two bytes)

Three–byte instructions: A three byte instruction contains an opcode plus a 16 – bit address.

Ex: 1.LXI H, 8509 Hex code = 21, 09, 85 (Three bytes)
2 .LDA 8509 Hex code = 3A, 09, 85 (Three bytes)

3. JMP 9567 Hex code = C3, 67, 95 (Three bytes)

4. STA 3525 Hex code = 32, 35, 25 (Three bytes)

DATA FORMATS: The 8085 is an 8-bit microprocessor which process only binary numbers. But it is very difficult to understand these binary numbers by a common user. So, we have to code these binary numbers into different data formats. The commonly known data formats are ASCII, BCD, signed integers and unsigned integers. The ASCII code is a 7-bit alpha-numeric code that represents decimal numbers, English alphabets and certain special characters. The ASCII stands for "American Standard code for Information Interchange" The term BCD stands for binary coded decimal, used for decimal numbers from 0-9.An 8-bit register can store two BCD numbers. A signed integer is either a positive or a negative number. In 8085 microprocessor the most significant bit D_7 is used for the sign. Here 0 denotes positive sign and 1 denotes the negative sign. An integer without a sign can be represented by all the 8-bits in a microprocessor register. So, the largest number that can be processed at one time is FFH. The numbers larger than 8-bits like 16, 24, 32 bits can be processed by dividing them in groups of 8-bits.

CLASSIFICATION OF INSTRUCTIONS

An instruction is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the instruction set, determines what functions the microprocessor can perform. The 8085 microprocessor instruction set has 74 operation codes that result in 246 instructions. This instruction set includes all the 8080A instructions plus two additional instructions namely SIM and RIM.

The instruction set of 8085 microprocessor is classified into five groups. They are:

- Data transfer (copy) group.
- Arithmetic group
- Logic group
- Branch control group
- Machine control and I/O group.

Data transfer (copy) instructions:

The data transfer instructions are used to transfer data from one register to another register, from memory to register or register to memory but not from one memory location to another memory

location. Actually this data transfer instruction copies the data from source to destination and the contents of the source are not altered. So, the data transfer instruction performs basically 'copy' operation.

Examples of data transfer instructions are MOV, MVI (Move Immediate), LXI (Load Immediate H-L Pair), LDA (Load Accumulator), STA (Store Accumulator), LHLD (Load H-L pair direct), SHLD (Store H-L pair direct), XCHG (Exchange the contents of H-L pair with D-E pair) etc…

Ex: MVI A, 55H ; Move the data 55H into Accumulator
MOV B, C ; Copies the contents of 'C' register into 'B' register
IN 00H ; Read the Input port(00H is the port address)
OUT 01H ; write data to an output port(01H is the port address)
LXIH 8570H ; Load H-L pair by address 8570H.

In the 8085 microprocessor, data transfer instructions do not affect any flags.

Arithmetic Instructions:

The arithmetic operations like addition, subtraction, increment and decrement are performed by the 8085 microprocessor using the following arithmetic instructions.

ADD, ADI (Add Immediate), SUB (Subtract), SUI (Subtract Immediate), INR (Increment), DCR (Decrement) etc. The arithmetic operations Add and subtract are performed in relation to the contents of the accumulator. But, the increment or the decrement operations can be performed in any register.

Ex:ADD B, C ; Add the contents of B register to the B register contents
ADI 08 ; Add the data 08 to the accumulator.
SUB A, B ; Subtract the contents of B register from accumulator.
SUI 05 ;Subtract immediate the 8-bit data from accumulator
INR B ; Increment the B register contents by one bit
DCR C ; Decrement the C register contents by one bit.

Arithmetic instructions modify all the flags according to the data conditions of the result. The INR and DCR instructions affect all flags except the carry flag.

Logical Group of Instructions:

Since the microprocessor is a programmable logic chip, it can be perform all the logic functions of the hard-wired logic through its instruction set. The 8085 processor can perform the logic instructions like, AND, OR, NOT (Complement) and X-OR (Exclusive OR) etc… The mnemonics of these instructions are given below.

ANA : Logically AND the contents of a register

ANI : Logically AND immediate the 8-bit data.
ORA : Logically OR the contents of a register.
OR : Logically OR immediate the 8-bit data.
XRA : Exclusive-OR the contents of a register.
XRI : Immediate Exclusive-OR the 8-bit data
CMA : Complement the accumulator

All the logic operations are performed in relation to the contents of the accumulator. The CMA instruction does not affect any flags. The executions of the logical instruction do not affect the contents of the operand register.

Branch Instructions:

These instructions are very important because they allow the microprocessor to change the sequence of a program either conditionally or unconditionally. The conditional branch instructions transfer the program to the specified label when certain condition is satisfied. The unconditional branch instructions transfer the program to the specified location unconditionally.

We know that the microprocessor is a sequential machine. So, it executes machine codes from one memory location to the next. Branch instructions instruct the microprocessor to go to a different memory location and the processor continues executing machine codes from the new location. The address of the new locations either specified explicitly or provided by the microprocessor or some times by additional hardware. The Branch instructions are classified into three categories. They are

(a). Jump instructions
(b). Call and return instructions
(c). Restart instructions.

Jump instructions specify memory locations explicitly and they are 3-byte instructions. These Jump instructions are of two types. They are , Unconditional Jump and Conditional Jump.

Unconditional Jump:

This is similar to Unconditional Go to statement in BASIC. When this instruction is executed the 16-bit address available immediately in the instruction is loaded into the program counter , so that the next sequence of instruction execution starts from this location. This Unconditional Jump instruction enables the programmer to create continuous loops.

JMP (16 bit address). So, this is a 3-byte instruction where the first byte is op-code and the second, third bytes specify memory address.

For example, the instruction JMP 8500H, instructs the microprocessor to go to the memory location8500H unconditionally. Sometimes, the jump location is specified using a label also.

Conditional Jump:

This instruction allows the microprocessor to make decision depending on certain conditions indicated by flags. The 8085 processor Jump instruction is associated with four flags. Namely Carry flag (CY), Zero flag (Z), Sign flag (S) and Parity flag (P). The following instructions shown in Table 3.3 transfer the program sequence to the memory location specified under the given conditions.

S. No	Instruction	Description
1	JC (16 bit Addr)	Jump on carry (if CY=1)
2	JNC (16 bit Addr)	Jump on no carry (if CY=0)
3	JZ (16 bit Addr)	Jump on Zero (if Z=1)
4	JNZ (16 bit Addr)	Jump on no Zero (if Z=0)
5	JP (16 bit Addr)	Jump on plus (if D_7=0; S=0)
6	JM (16 bit Addr)	Jump on minus (if D_7=1; S=1)
7	JPE (16 bit Addr)	Jump on Even Parity (if P=1)
8	JPO (16 bit Addr)	Jump on Odd Parity (if P=0)

Table 2.1 various conditional jump instructions

To understand the instructions, let us consider the instruction JC (16 bit address). The meaning of this instruction is, the microprocessor is instructed to jump the specified 16 bit memory location if there exists a carry after the arithmetic operation else it will execute the next instruction in the sequence.

CALL and RETURN Instructions

The microprocessor uses the two instructions CALL and RETURN to implement subroutines. Here CALL instruction calls a subroutine program which is not a part of the main program and the RET instruction at the end of the subroutine program to return the control to the main program.

Ex: CALL (16 bit memory address)
RET

RESET (RST) Instruction

The 8085 processor provides eight RST instructions to transfer the program control to a specific location on page 00H. These instructions are 1-byte instructions. The various RST instructions and their call locations are given in the following Table 3.4

S. No	Mnemonics	Hex code	Call location In Hex
1	RST 0	C7	0000
2	RST1	CF	0008
3	RST2	D7	0010
4	RST3	DF	0018
5	RST4	E7	0020
6	RST5	EF	0028
7	RST6	F7	0030
8	RST7	FF	0038

Table 2.2 Various RST instructions and their call locations

Machine control and I/O Instructions

There are six basic machine control instructions. They are

- EI (Enable Interrupt)
- DI (Disable Interrupt)
- NOP (No Operation)
- SIM (Set Interrupt Mask)
- RIM (Read Interrupt Mask)
- HLT (Halt)

EI (Enable Interrupt): This is a one byte instruction used to enable the interrupt. This instruction is used to enable the interrupts when the microprocessor is reset or the interrupt

enable flag is reset after interrupt acknowledge. This instruction takes one machine cycle with four states. The op-code is FBH.

DI (Disable Interrupt): This is a one byte instruction which resets the interrupt enable flag to disable all the interrupts except TRAP. It takes one machine cycle with four states. The op-code is F3H.

NOP (No Operation): when this instruction is executed, the microprocessor performs nothing. Microprocessor spends four states doing nothing. It is a one byte instruction whose op-code is 00H.This instruction is normally used to generate very small time delays of the order of few micro seconds. This NOP instruction is also very useful when we are required to insert a few instructions in the main program additionally .

SIM (Set Interrupt Mask): This instruction masks the interrupt as desired. This is a dual purpose instruction. The first purpose is to set or reset the mask of the maskable interrupt. The second purpose is to send the data out through the SOD pin at pin number 4 of the microprocessor.

RIM (Read Interrupt Mask): This instruction copies the status of the interrupts into the accumulator. It is also used to read the serial data through the SID pin

HLT (Halt): After execution of this instruction the microprocessor goes into the halt state. The processor can be restarted by a valid interrupt or by applying a RESET signal. The microprocessor takes 5T states to implement the halt instruction.

I/O instructions:
There are two important instructions to input the data into the microprocessor's accumulator through the input port and output the data from the accumulator to the output port. They are

- IN (port address)
- OUT (port address)

This port address is an 8-bit address. In both these instructions the default register is Accumulator.

Ex: (i) IN 01H. This instruction will copy the contents into the Accumulator through the port

whose address is 01H. It takes three machine cycles and takes 10 states. The op-code is DBH.

(ii)OUT 02H. This instruction sends the contents of Accumulator to the outport whose address is 02H. It is a two byte instruction which requires 10 states. The op-code for this instruction is D3H.

DETAILED INSTRUCTION SET

DATA TRANSFER INSTRUCTIONS

Opcode	Operand	Description
Move from source to destination		
MOV	Rd, Rs M, Rs Rd, M	This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers. Example: MOV B, C or MOV B, M
Move immediate 8-bit		
MVI	Rd, data M, data	The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers. Example: MVI B, 57H or MVI M, 57H
Load accumulator		
LDA	16-bit address	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. Example: LDA 2034H
Load accumulator indirect		
LDAX	B/D Reg. pair	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. Example: LDAX B
Load register pair immediate		
LXI	Reg. pair, 16-bit data	The instruction loads 16-bit data in the register pair designated in the operand. Example: LXI H, 2034H or LXI H, XYZ
Load H and L registers direct		
LHLD	16-bit address	The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered. Example: LHLD 2040H

Store accumulator direct

STA	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. Example: STA 4350H

Store accumulator indirect

STAX	Reg. pair	The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered. Example: STAX B

Store H and L registers direct

SHLD	16-bit address	The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. Example: SHLD 2470H

Exchange H and L with D and E

XCHG	none	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example: XCHG

Copy H and L registers to the stack pointer

SPHL	none	The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered. Example: SPHL

Exchange H and L with top of stack

XTHL	none	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered. Example: XTHL

Push register pair onto stack

PUSH Reg. pair	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high-order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location.
	Example: PUSH B or PUSH PSW

Pop off stack to register pair

POP Reg. pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1.
	Example: POP H or POP PSW

Output data from accumulator to a port with 8-bit address

OUT 8-bit port address	The contents of the accumulator are copied into the I/O port specified by the operand.
	Example: OUT F8H

Input data to accumulator from a port with 8-bit address

IN 8-bit port address	The contents of the input port designated in the operand are read and loaded into the accumulator.
	Example: IN 8CH

ARITHMATIC INSTRUCTIONS

Opcode	Operand	Description
Add register or memory to accumulator		
ADD	R M	The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition. Example: ADD B or ADD M
Add register to accumulator with carry		
ADC	R M	The contents of the operand (register or memory) and the carry flag are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition. Example: ADC B or ADC M
Subtract register or memory from accumulator		
SUB	R M	The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction. Example: SUB B or SUB M
Subtract source and borrow from accumulator		
SBB	R M	The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction. Example: SBB B or SBB M

Subtract immediate from accumulator

SUI	8-bit data	The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction. Example: SUI 45H

Subtract immediate from accumulator with borrow

SBI	8-bit data	The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction. Example: SBI 45H

Increment register or memory by 1

INR	R M	The contents of the designated register or memory are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers. Example: INR B or INR M

Increment register pair by 1

INX	R	The contents of the designated register pair are incremented by 1 and the result is stored in the same place. Example: INX H

Decrement register or memory by 1

DCR	R M	The contents of the designated register or memory are decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers. Example: DCR B or DCR M

Decrement register pair by 1

DCX	R	The contents of the designated register pair are decremented by 1 and the result is stored in the same place. Example: DCX H

Decimal adjust accumulator

DAA	none	The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits. Example: DAA

BRANCHING INSTRUCTIONS

Opcode	Operand	Description
Jump unconditionally		
JMP	16-bit address	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Example: JMP 2034H or JMP XYZ
Jump conditionally		
	Operand: 16-bit address	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below. Example: JZ 2034H or JZ XYZ

Opcode	Description	Flag Status
JC	Jump on Carry	CY = 1
JNC	Jump on no Carry	CY = 0
JP	Jump on positive	S = 0
JM	Jump on minus	S = 1
JZ	Jump on zero	Z = 1
JNZ	Jump on no zero	Z = 0
JPE	Jump on parity even	P = 1
JPO	Jump on parity odd	P = 0

Unconditional subroutine call

CALL 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.

Example: CALL 2034H or CALL XYZ

Call conditionally

Operand: 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below. Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.

Example: CZ 2034H or CZ XYZ

Opcode	Description	Flag Status
CC	Call on Carry	CY = 1
CNC	Call on no Carry	CY = 0
CP	Call on positive	S = 0
CM	Call on minus	S = 1
CZ	Call on zero	Z = 1
CNZ	Call on no zero	Z = 0
CPE	Call on parity even	P = 1
CPO	Call on parity odd	P = 0

Return from subroutine unconditionally

RET none — The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.

Example: RET

Return from subroutine conditionally

Operand: none

The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described below. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.

Example: RZ

Opcode	Description	Flag Status
RC	Return on Carry	CY = 1
RNC	Return on no Carry	CY = 0
RP	Return on positive	S = 0
RM	Return on minus	S = 1
RZ	Return on zero	Z = 1
RNZ	Return on no zero	Z = 0
RPE	Return on parity even	P = 1
RPO	Return on parity odd	P = 0

Load program counter with HL contents

PCHL none — The contents of registers H and L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte.

Example: PCHL

Restart

RST 0-7

The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. The instructions are generally used in conjunction with interrupts and inserted using external hardware. However these can be used as software instructions in a program to transfer program execution to one of the eight locations. The addresses are:

Instruction	Restart Address
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

The 8085 has four additional interrupts and these interrupts generate RST instructions internally and thus do not require any external hardware. These instructions and their Restart addresses are:

Interrupt	Restart Address
TRAP	0024H
RST 5.5	002CH
RST 6.5	0034H
RST 7.5	003CH

LOGICAL INSTRUCTIONS

Opcode	Operand	Description

Compare register or memory with accumulator

CMP R / M — The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved. The result of the comparison is shown by setting the flags of the PSW as follows:

if (A) < (reg/mem): carry flag is set
if (A) = (reg/mem): zero flag is set
if (A) > (reg/mem): carry and zero flags are reset

Example: CMP B or CMP M

Compare immediate with accumulator

CPI 8-bit data — The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:

if (A) < data: carry flag is set
if (A) = data: zero flag is set
if (A) > data: carry and zero flags are reset

Example: CPI 89H

Logical AND register or memory with accumulator

ANA	R M	The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.

Example: ANA B or ANA M

Logical AND immediate with accumulator

ANI	8-bit data	The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.

Example: ANI 86H

Exclusive OR register or memory with accumulator

XRA	R M	The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: XRA B or XRA M

Exclusive OR immediate with accumulator

XRI	8-bit data	The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: XRI 86H

Logical OR register or memory with accumulator

ORA	R M	The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: ORA B or ORA M

Logical OR immediate with accumulator

ORI	8-bit data	The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: ORI 86H

Rotate accumulator left

RLC none

Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7.S, Z, P, AC are not affected.

Example: RLC

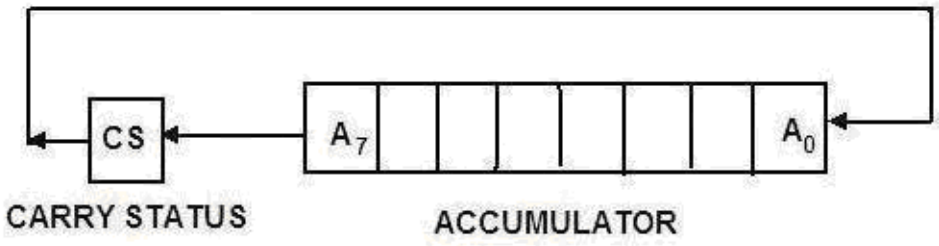


Figure 2.2. Schematic Diagram for RLC

Rotate accumulator right

RRC none

Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0.S, Z, P, AC are not affected.
Example: RRC

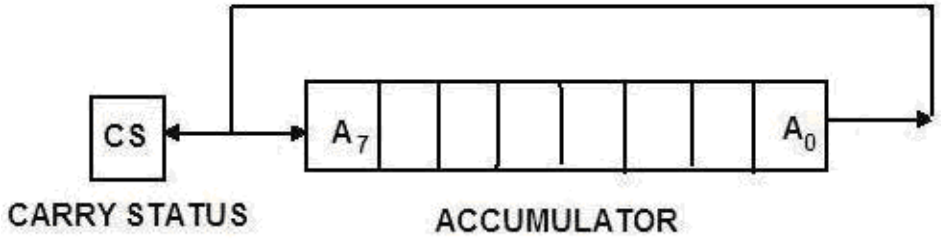


Figure 2.3 Schematic diagram for RRC

Rotate accumulator left through carry

RAL none

Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0.CY is modified according to bit D7. S, Z, P, AC are not affected.

Example: RAL

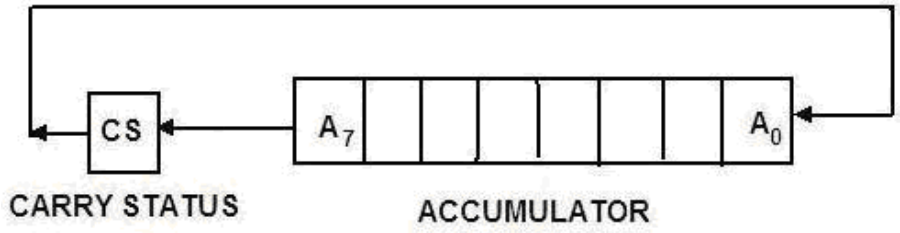


Figure 2.4 Schematic diagram for RAL

Rotate accumulator right through carry

RAR	none	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected. Example: RAR

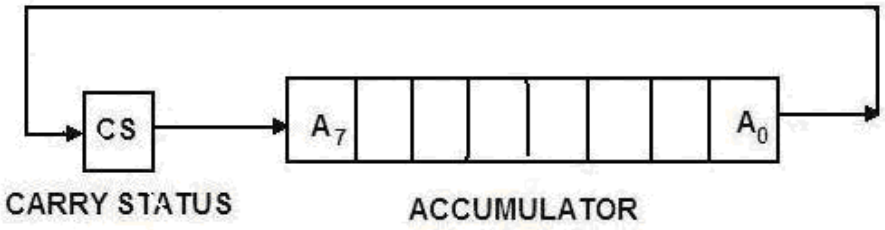


Figure 2.5 Schematic diagram for RAR

CONTROL INSTRUCTIONS

Opcode	Operand	Description
No operation		
NOP	none	No operation is performed. The instruction is fetched and decoded. However no operation is executed. Example: NOP
Halt and enter wait state		
HLT	none	The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state. Example: HLT
Disable interrupts		
DI	none	The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected. Example: DI
Enable interrupts		
EI	none	The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to reenable the interrupts. (except TRAP). Example: EI

Read interrupt mask

RIM none

This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.

Example: RIM

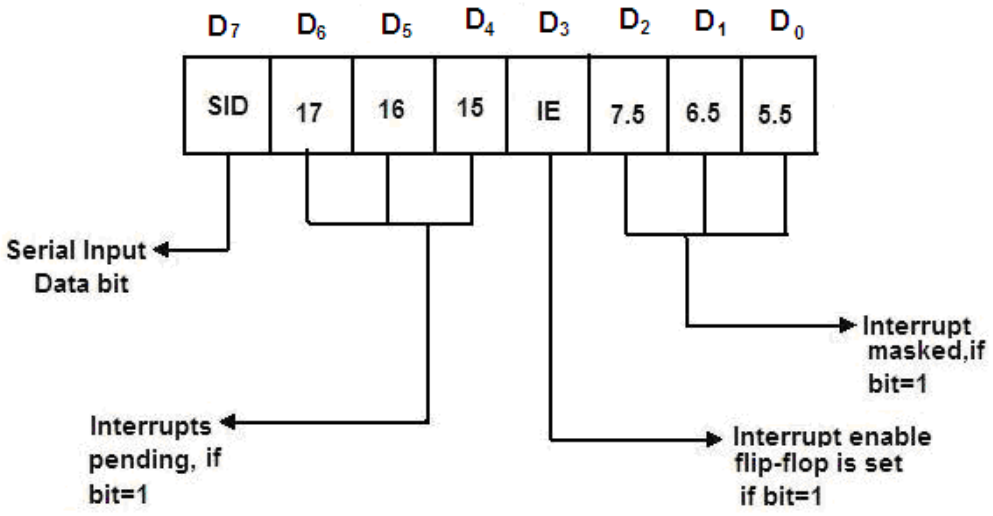


Figure 2.6 Read Interrupt mask Register

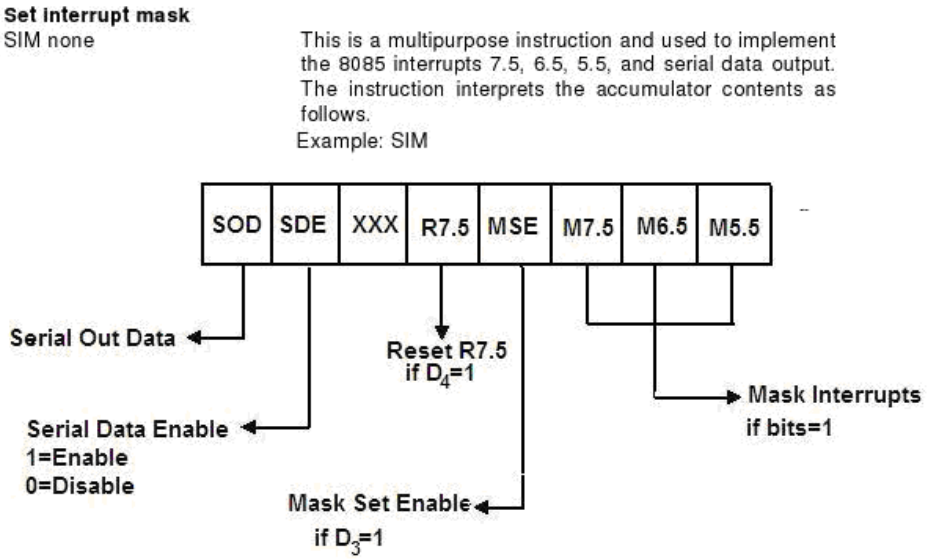


Figure 2.7 Set interrupt mask register

SOD – Serial output Data .Bit D_7 of accumulator is latched in to the SOD output line made available to serial peripheral if bit D_6 =1 .

SOE – Serial output enable.If this bit =1 ,it enables the serial output.

XXX – Don't care condition

R7.5 – Reset RST7.5.If this bit = 1 ,RST7.5 flip-flop is reset .This is an additional control to reset

RST7.5

MSE – Mask set Enable.If this bit is high ,it enables the function of bits D_2,D_1 and D_0.This is a master control over all the interrupt masking bits.

M7.5 – D_2=0 ,RST 7.5 is enabled

D_2=1 RST7.5 is masked or disabled

M6.5 – D_1=0 RST6.5 is enabled

D_1=1 RST 6.5 masked or disabled.

M5.5 – D_0=0 RST5.5 is enabled

D_0=1 RST 5.5 is disabled or masked.

ADDRESSING MODES (Beyond The Syllabus)

Majority of the instructions of 8085 microprocessor requires an operand (either data or address) on which the intended operation can be performed. Some instructions may require only one operand and some other instructions require two operands for its instruction execution. The speed of execution mainly depends on the position of the operand in the instruction. The scheme involved in identifying the position of operands in an instruction is known as addressing mode.

There are five addressing modes 8085 processor.They are
(i). Immediate addressing mode
(ii). Direct addressing mode
(iii). Register addressing mode.
(iv). Register indirect addressing mode.
(v). Implicit addressing mode.

(i). Immediate Addressing mode: The mode of addressing in which the operand is a part of the instruction itself is known as Immediate Addressing mode. If the immediate data is 8-bit, the instruction will be of two bytes. If the immediate data is 16 bit, the instruction is of 3 bytes.

Ex: (1). ADI DATA ; Add immediate the data to the contents of the accumulator.
(2).LXIH 8500H : Load immediate the H-L pair with the operand 8500H
(3). MVI 08H ; Move the data 08 H immediately to the accumulator
(4). SUI 05H ; Subtract immediately the data 05H from the accumulator

(ii) Direct Addressing mode: The mode of addressing in which the 16-bit address of the operand is directly available in the instruction itself is called Direct Addressing mode. i.e., the address of the operand (data) is available in the instruction itself. This is a 3-byte instruction.

Ex: (1). LDA 9525H ; Load the contents of memory location into Accumulator.
(2). STA 8000H ; Store the contents of the Accumulator in the location 8000H
(3). IN 01H ; Read the data from port whose address is 01H.

(iii). Register addressing modes: The mode, in which the operand is in one of the general purpose registers, is known as the register addressing mode.

Ex: (1). MOV A, B; Move the contents of B register to A register.
(2). SUB D; Subtract the contents of D register from Accumulator.
(3). ADD B, C; Add the contents of C register to the contents of B register.

(iv). Register indirect addressing modes: The 16-bit address location of the operand stored in a register pair (H-L) is given in the instruction. The address of the operand is given in an indirect way with the help of a register pair. Hence it is called Register indirect addressing mode

Data Communication & Computer Network

Data communications refers to the transmission of this digital data between two or more computers and a computer network or data network is a telecommunications network that allows computers to exchange data. The physical connection between networked computing devices is established using either cable media or wireless media. The best-known computer network is the Internet.

This tutorial should teach you basics of Data Communication and Computer Network (DCN) and will also take you through various advance concepts related to Data Communication and Computer Network.

Why to Learn Data Communication & Computer Network?

Network Basic Understanding

A system of interconnected computers and computerized peripherals such as printers is called computer network. This interconnection among computers facilitates information sharing among them. Computers may connect to each other by either wired or wireless media.

Network Engineering

Networking engineering is a complicated task, which involves software, firmware, chip level engineering, hardware, and electric pulses. To ease network engineering, the whole networking concept is divided into multiple layers. Each layer is involved in some particular task and is independent of all other layers. But as a whole, almost all networking tasks depend on all of these layers. Layers share data between them and they depend on each other only to take input and send output.

Internet

A network of networks is called an internetwork, or simply the internet. It is the largest network in existence on this planet.The internet hugely connects all WANs and it can have connection to LANs and Home networks. Internet uses TCP/IP protocol suite and uses IP as its addressing protocol. Present day, Internet is widely implemented using IPv4. Because of shortage of address spaces, it is gradually migrating from IPv4 to IPv6.

Internet enables its users to share and access enormous amount of information worldwide. It uses WWW, FTP, email services, audio and video streaming etc. At huge level, internet works on Client-Server model.

Internet uses very high speed backbone of fiber optics. To inter-connect various continents, fibers are laid under sea known to us as submarine communication cable.

Applications of Communication & Computer Network

Computer systems and peripherals are connected to form a network.They provide numerous advantages:

- Resource sharing such as printers and storage devices
- Exchange of information by means of e-Mails and FTP
- Information sharing by using Web or Internet
- Interaction with other users using dynamic web pages
- IP phones
- Video conferences
- Parallel computing
- Instant messaging

What is Networking and Communication?

Data communications refers to the transmission of this digital data between two or more computers and a computer network or data network is a telecommunications network that allows computers to exchange data. The physical connection between networked computing devices is established using either cable media or wireless media. The best-known computer network is the Internet.

What are the types of Computer Networks?

In computer networks, the data is passed in the form of packets. The devices that transmit or receive this data, such as a phone or a computer, are referred to as nodes. There are three main types of networks:

1. Local Area Network (LAN): It is usually a small network that is restricted to a small geographic area. A computer network available only to the residents of a building can be called a LAN.

2. Wide Area Network (WAN): As the name implies, these networks cover a broad range of geographic area. WANs are used to connect LANs and other types of networks together so that users and computers can communicate with computers in other regions. An example of a WAN is the much-used and loved, Internet.

3. Metropolitan Area Network (MAN): MAN is a network that connects the users with computer resources in a geographic area that is larger than LAN but not quite as large as WAN.

What are the basic components of Computer Networks?

1.Servers - Servers are computers that hold shared files, programs, and the network

operating system. Servers provide access to network resources to all the users of the network. There are many different kinds of servers, and one server can provide several functions. For example, there are file servers, print servers, mail servers, communication servers, database servers, print servers, fax servers and web servers, to name a few.

2.Clients - Clients are computers that access and use the network and shared network resources. Client computers are basically the customers(users) of the network, as they request and receive services from the servers.

3.Transmission Media - Transmission media are the facilities used to interconnect computers in a network, such as twisted-pair wire, coaxial cable, and optical fiber cable. Transmission media are sometimes called channels, links or lines.

4.Shared data - Shared data are data that file servers provide to clients such as data files, printer access programs and e-mail.

5.Shared printers and other peripherals - Shared printers and peripherals are hardware resources provided to the users of the network by servers. Resources provided include data files, printers, software, or any other items used by clients on the network.

6.Network Interface Card - Each computer in a network has a special expansion card called a network interface card (NIC). The NIC prepares(formats) and sends data, receives data, and controls data flow between the computer and the network. On the transmit side, the NIC passes frames of data on to the physical layer, which transmits the data to the physical link. On the receiver's side, the NIC processes bits received from the physical layer and processes the message based on its contents.

7.Local Operating System - A local operating system allows personal computers to access files, print to a local printer, and have and use one or more disk and CD drives that are located on the computer.

8.Network Operating System - The network operating system is a program that runs on computers and servers, and allows the computers to communicate over the network.

9.Hub - Hub is a device that splits a network connection into multiple computers. It is like a distribution center. When a computer request information from a network or a specific computer, it sends the request to the hub through a cable. The hub will receive the request and transmit it to the entire network. Each computer in the network should then figure out whether the broadcast data is for them or not.

10.Switch - Switch is a telecommunication device grouped as one of computer network components. It uses physical device addresses in each incoming messages so that it can deliver the message to the right destination or port.

Networking and Communicatio(Reviewed Version)

What is Networking and Communication?

Data communication refers to the transmission of the digital data between two or more computers The physical connection between networked computing devices is established using either cable media or wireless media. The best-known computer

network is the Internet.

What are the types of Computer Networks?

In computer networks, the data is passed in the form of packets . Everything you do on the Internet involves packets. For example, every Web page that you receive comes as a series of packets, and every e-mail you send leaves as a series of packets. The devices that transmit or receive this data, such as a phone or a computer, are referred to as nodes. There are three main types of networks:

1. Local Area Network (LAN): It is usually a small network that is restricted to a small geographic area. For instance, a computer network available only to the residents of a building can be called a LAN.

2. Wide Area Network (WAN): As the name implies, these networks cover a broad range of geographic area. WANs are used to connect LANs and other types of networks together so that users and computers can communicate with computers in other regions. An example of a WAN is the much-used and loved, Internet.

3. Metropolitan Area Network (MAN): MAN is a network that connects the users with computer resources in a geographic area that is larger than LAN but not quite as large as WAN.

What are the basic components of Computer Networks?

1.Servers - Servers are computers that hold shared files, programs, and the network operating system. Servers provide access to network resources to all the users of the network. There are many different kinds of servers, and one server can provide several functions. For example, there are file servers, print servers, mail servers, communication servers, database servers, print servers, fax servers and web servers, to name a few.

2.Clients - Clients are computers that access and use the network and shared network resources. Client computers are basically the customers(users) of the network, as they request and receive services from the servers.

3.Transmission Media - Transmission media are the facilities used to interconnect computers in a network. Transmission media are sometimes called channels, links or lines.

4.Shared data - Shared data are data that file servers provide to clients such as data files, printer access programs and e-mail.

5.Shared printers and other peripherals - Shared printers and peripherals are hardware resources provided to the users of the network by servers. Resources provided include data files, printers, software, or any other items used by clients on the network.

6.Network Interface Card - Each computer in a network has a special expansion card

called a network interface card (NIC). The NIC prepares(formats) and sends data, receives data, and controls data flow between the computer and the network. On the transmit side, the NIC passes frames of data on to the physical layer, which transmits the data to the physical link. On the receiver's side, the NIC processes bits received from the physical layer and processes the message based on its contents.

7.Local Operating System - A local operating system allows personal computers to access files, print to a local printer, and have and use one or more disk and CD drives that are located on the computer.

8.Network Operating System - The network operating system is a program that runs on computers and servers, and allows the computers to communicate over the network.

9.Hub - Hub is a device that splits a network connection into multiple computers. It is like a distribution center. When a computer request information from a network or a specific computer, it sends the request to the hub through a cable. The hub will receive the request and transmit it to the entire network. Each computer in the network should then figure out whether the broadcast data is for them or not.

10.Switch - Switch is a telecommunication device grouped as one of computer network components. It uses physical device addresses in each incoming messages so that it can deliver the message to the right destination or port.

Number System in Digital Electronics

Definition: In digital electronics, the number system is used for representing the information. The number system has different bases and the most common of them are the decimal, binary, octal, and hexadecimal. The **base or radix** of the number system is the total number of the digit used in the number system. Suppose if the number system representing the digit from 0 – 9 then the base of the system is the 10.

Types of Number Systems

Some of the important types of number system are

- Decimal Number System
- Binary Number System
- Octal Number System
- Hexadecimal Number System

These number systems are explained below in details.

1. Decimal Number Systems

The number system is having digit 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; this number system is known as a decimal number system because total ten digits are involved.The base of the decimal number system is 10.

2. Binary Number Systems

The modern computers do not process decimal number; they work with another number system known as a binary number system which uses only two digits 0 and1.The base of binary number system is 2 because it has only two digit 0 and 1.The digital electronic equipments are works on the binary number system and hence the decimal number system is converted into binary system.

The table is shown below the decimal, binary, octal, and hexadecimal numbers from 0 to 15 and their equivalent binary number.

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5

6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Showing 1 to 16 of 16 entries

3. Octal Numbers

The base of a number system is equal to the number of digits used, i.e., for decimal number system the base is ten while for the binary system the base is two. The octal system has the base of eight as it uses eight digits 0, 1, 2, 3, 4, 5, 6, 7.

All these digits from 0 to 7 have the same physical meaning as by decimal symbols, the next digit in the octal number is represented by 10, 11, 12, which are equivalent to decimal digits 8, 9, 10 respectively. In this way, the octal number 20 will represent the decimal digit and subsequently, 21, 22, 23.. Octal numbers will represent the decimal number digit 17, 18, 19… etc. and so on.

4. Hexadecimal Numbers

These numbers are used extensively in microprocessor work.The hexadecimal number system has a base of 16, and hence it consists of the following sixteen number of digits.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

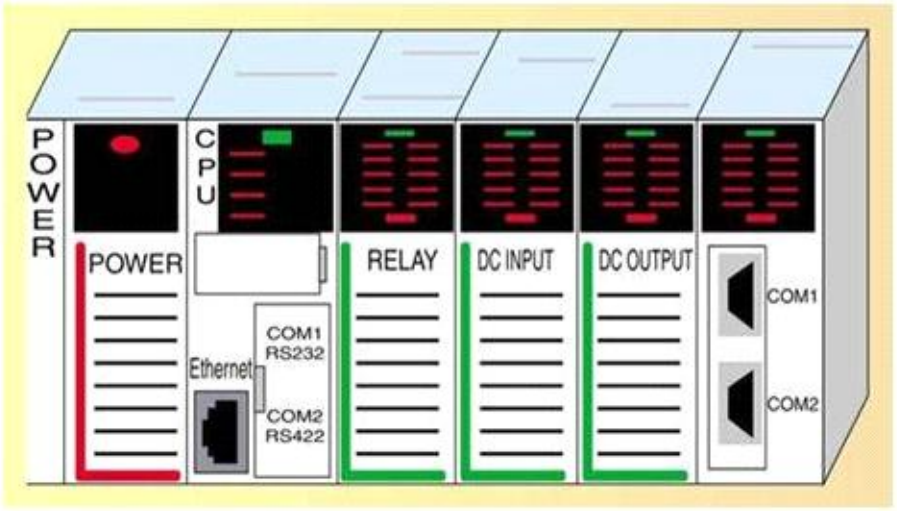
The size of the hexadecimal is much shorter than the binary number which makes them easy to write and remember. Let 0000 to 000F representing hexadecimal numbers from zero to fifteen, then 0010, 0011, 0012, …etc. Will represent sixteen, seventeen, eighteen… etc. till 001F which

represent thirty open and so on.

UNIT 5

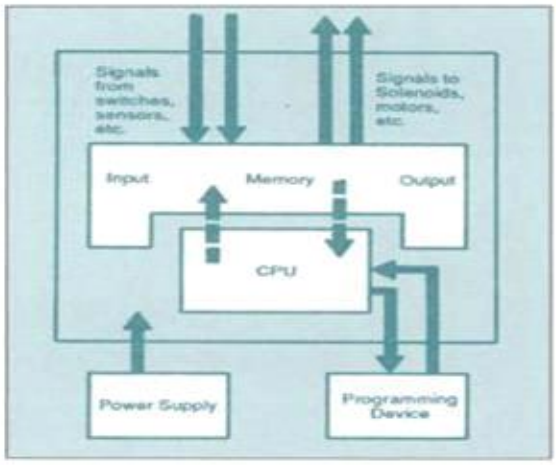
Programmable Logic Controllers (PLCs): Architecture, Number Systems Basics of PLC Programming, Logics, Timers and Counters, Application on real time industrial automation systems.

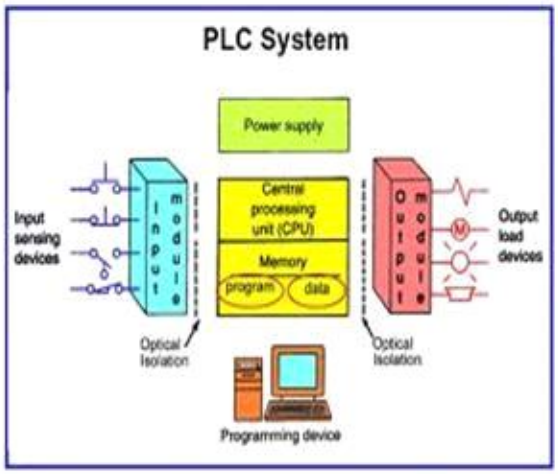
PLC stands for Programmable Logic Controllers. They are basically used to control automated systems in industries. They are one of the most advanced and simplest forms of control systems which are now replacing hard-wired logic relays at a large scale.



Programming Logic Controller (PLC)

PLC Architecture:



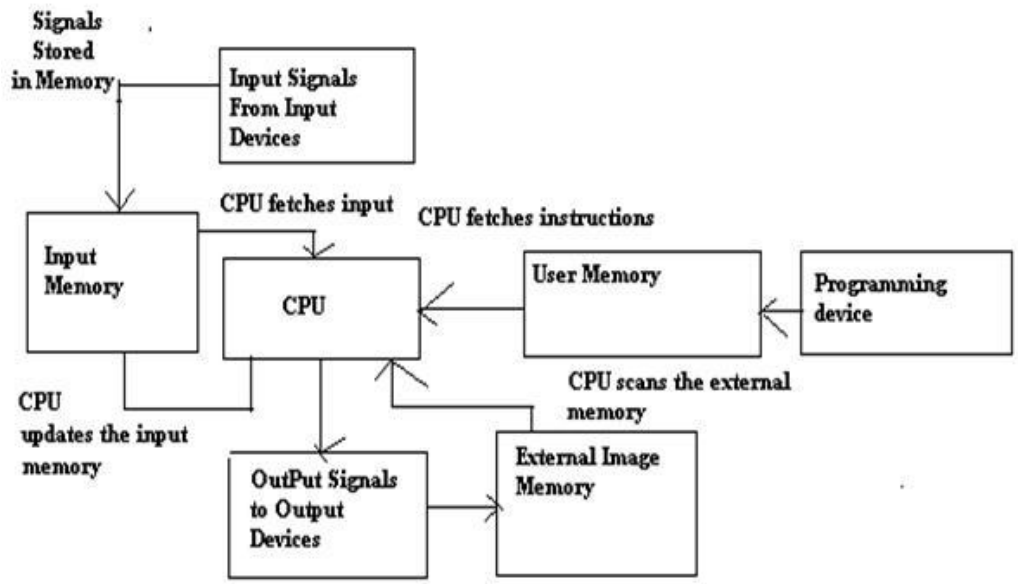


PLC Internal Architecture

A basic PLC system consists of the following sections:

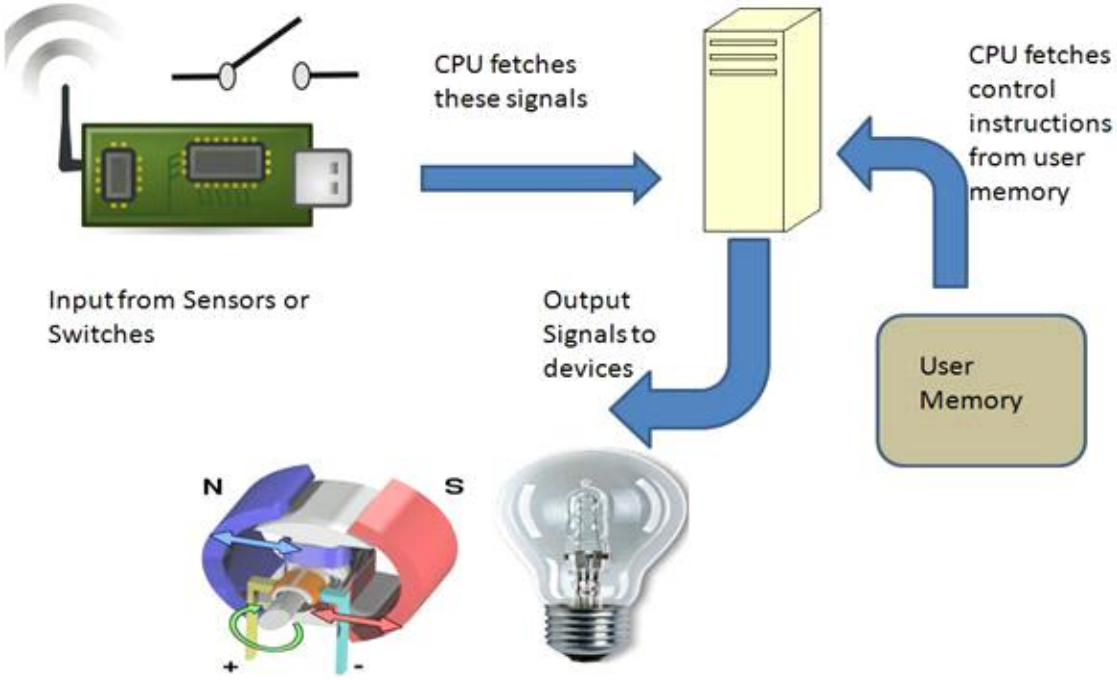
- **Input/ Output Section**: The input section or input module consists of devices like sensors, switches, and many other real-world input sources. The input from the sources is connected to the PLC through the input connector rails. The output section or output module can be a motor or a solenoid or a lamp or a heater, whose functioning is controlled by varying the input signals.
- **CPU or Central Processing Unit**: It is the brain of the PLC. It can be a hexagonal or an octal microprocessor. It carries out all the processing related to the input signals in order to control the output signals based on the control program.
- **Programming Device**: It is the platform where the program or the control logic is written. It can be a handheld device or a laptop or a computer itself.
- **Power Supply**: It generally works on a power supply of about 24 V, used to power input and output devices.
- **Memory**: The memory is divided into two parts- The data memory and the program memory. The program information or the control logic is stored in the user memory or the program memory from where the CPU fetches the program instructions. The input and output signals and the timer and counter signals are stored in the input and output external image memory respectively.

Working of a PLC



PLC

Working Schematic



Working of PLC

- The input sources convert the real-time analog electric signals to suitable digital electric signals and these signals are applied to the PLC through the connector rails.
- These input signals are stored in the PLC external image memory in locations known as bits. This is done by the CPU

- The control logic or the program instructions are written onto the programming device through symbols or through mnemonics and stored in the user memory.
- The CPU fetches these instructions from the user memory and executes the input signals by manipulating, computing, processing them to control the output devices.
- The execution results are then stored in the external image memory which controls the output drives.
- The CPU also keeps a check on the output signals and keeps updating the contents of the input image memory according to the changes in the output memory.
- The CPU also performs internal programming functions like setting and resetting of the timer, checking the user memory.

Programming in PLC

The basic functioning of the PLC relies on the control logic or the programming technique used. Programming can be done using flowcharts or using ladder logic or using statement logics or mnemonics.

Interlinking all these, let us see how we can actually write a program in PLC.

Programming PLCs

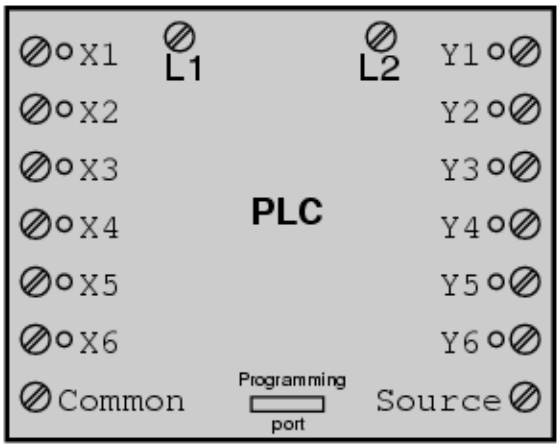
A PLC has many "input" terminals, through which it interprets "high" and "low" logical states from sensors and switches. It also has many output terminals, through which it outputs "high" and "low" signals to power lights, solenoids, contactors, small motors, and other devices lending themselves to on/off control.

In an effort to make PLCs easy to program, their programming language was designed to resemble ladder logic diagrams. Thus, an industrial electrician or electrical engineer accustomed to reading ladder logic schematics would feel comfortable programming a PLC to perform the same control functions.

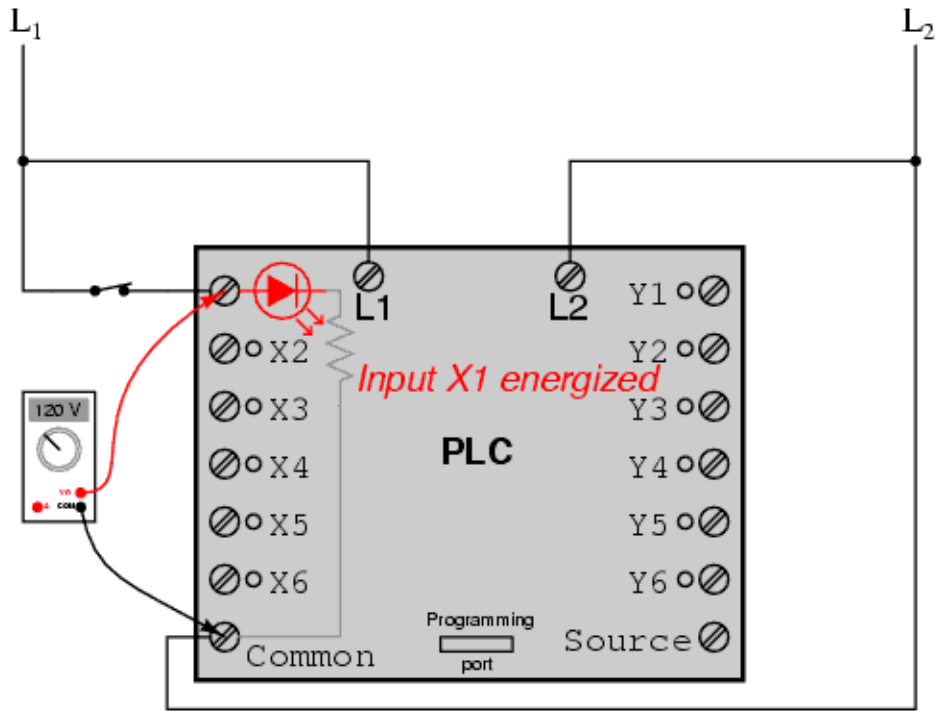
PLCs are industrial computers, and as such their input and output signals are typically 120 volts AC, just like the electromechanical control relays they were designed to replace. Although some PLCs have the ability to input and output low-level DC voltage signals of the magnitude used in logic gate circuits, this is the exception and not the rule. Signal connection and programming standards vary somewhat between different models of PLC, but they are similar enough to allow a "generic" introduction to PLC programming here.

The following illustration shows a simple PLC, as it might appear from a front view. Two screw terminals provide connection to 120 volts AC for powering the PLC's internal circuitry, labeled L1 and L2. Six screw terminals on the left-hand side provide connection to input devices, each terminal representing a different input "channel" with its own "X" label.

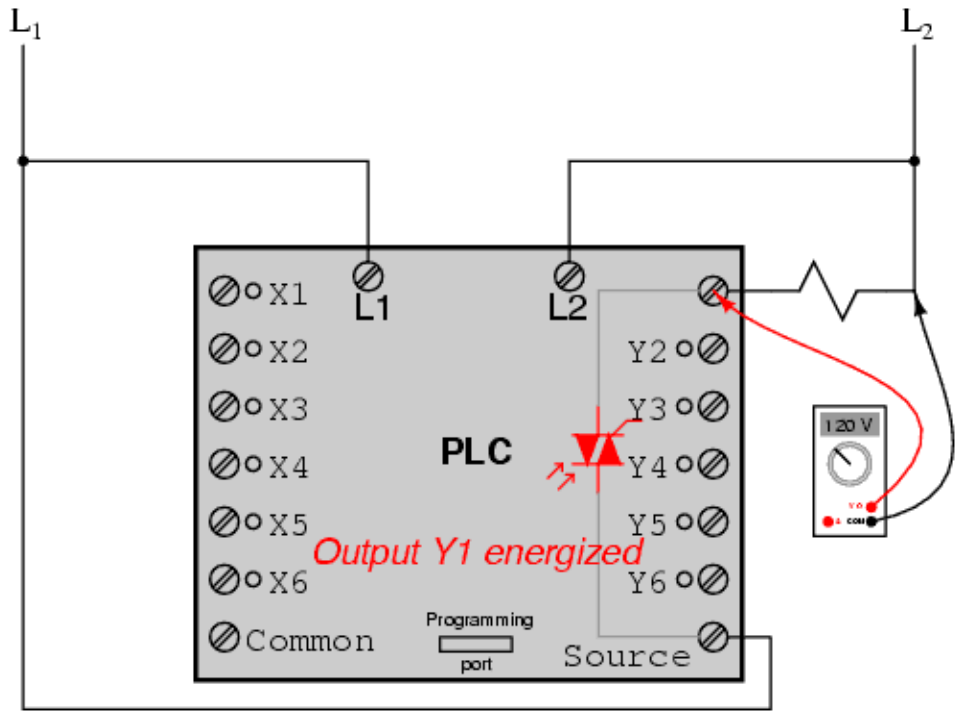
The lower-left screw terminal is a “Common” connection, which is generally connected to L2 (neutral) of the 120 VAC power source.



Inside the PLC housing, connected between each input terminal and the Common terminal, is an opto-isolator device (Light-Emitting Diode) that provides an electrically isolated “high” logic signal to the computer’s circuitry (a photo-transistor interprets the LED’s light) when there is 120 VAC power applied between the respective input terminal and the Common terminal. An indicating LED on the front panel of the PLC gives visual indication of an “energized” input:

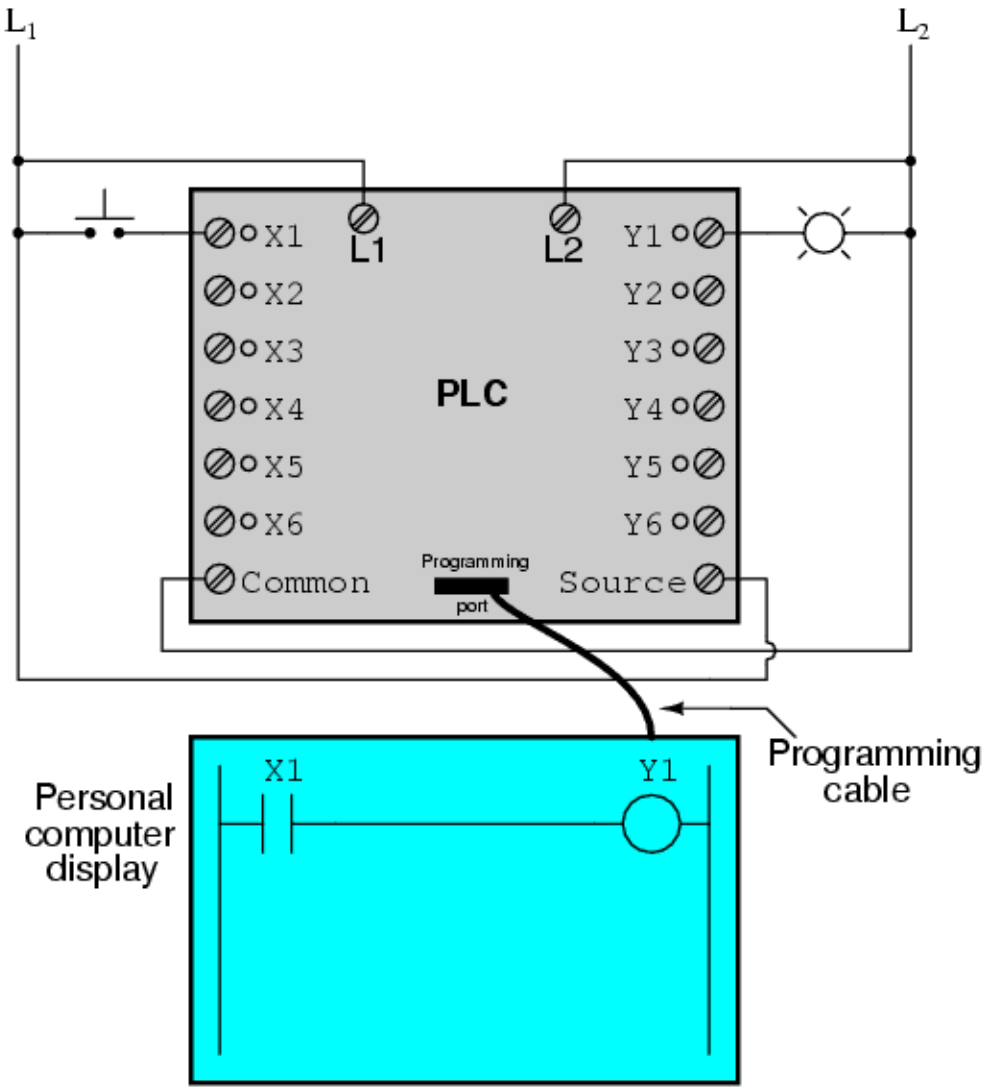


Output signals are generated by the PLC's computer circuitry activating a switching device (transistor, TRIAC, or even an electromechanical relay), connecting the "Source" terminal to any of the "Y-" labeled output terminals. The "Source" terminal, correspondingly, is usually connected to the L1 side of the 120 VAC power source. As with each input, an indicating LED on the front panel of the PLC gives visual indication of an "energized" output:



In this way, the PLC is able to interface with real-world devices such as switches and solenoids. The actual *logic* of the control system is established inside the PLC by means of a computer program. This program dictates which output gets energized under which input conditions.

Although the program itself appears to be a ladder logic diagram, with switch and relay symbols, there are no actual switch contacts or relay coils operating inside the PLC to create the logical relationships between input and output. These are *imaginary* contacts and coils, if you will. The program is entered and viewed via a personal computer connected to the PLC's programming port. Consider the following circuit and PLC program:

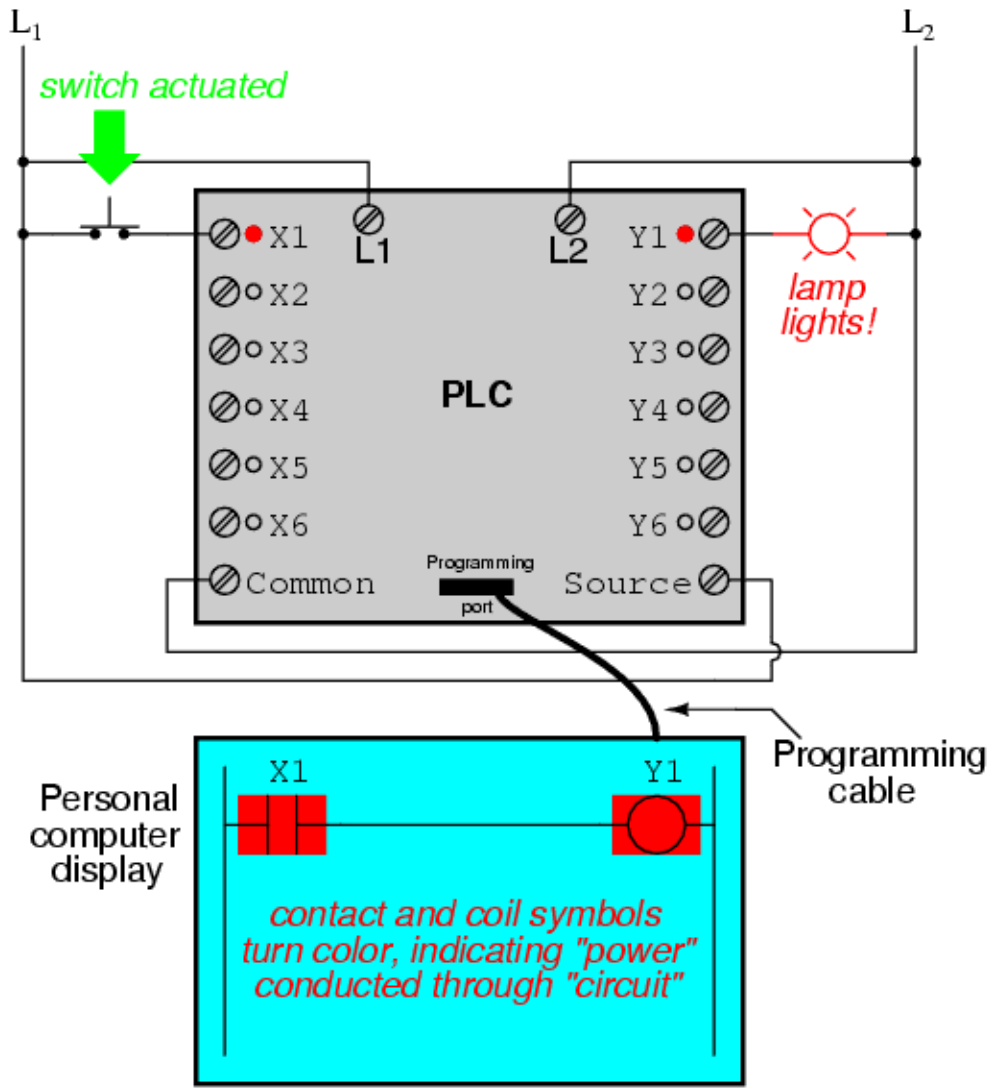


When the pushbutton switch is unactuated (unpressed), no power is sent to the X1 input of the PLC. Following the program, which shows a normally-open X1 contact in series with a Y1 coil, no "power" will be sent to the Y1 coil. Thus, the PLC's Y1 output remains de-energized, and the indicator lamp connected to it remains dark.

If the pushbutton switch is pressed, however, power will be sent to the PLC's X1 input. Any and all X1 contacts appearing in the program will assume the actuated (non-normal) state, as though they were relay contacts actuated by the energizing of a relay coil named "X1".

In this case, energizing the X1 input will cause the normally-open X1 contact will "close," sending "power" to the Y1 coil. When the Y1 coil of the program "energizes," the real Y1 output

will become energized, lighting up the lamp connected to it:



It must be understood that the X1 contact, Y1 coil, connecting wires, and "power" appearing in the personal computer's display are all *virtual*. They do not exist as real electrical components. They exist as commands in a computer program—a piece of software only—that just happens to resemble a real relay schematic diagram.

Equally important to understand is that the personal computer used to display and edit the PLC's program is not necessary for the PLC's continued operation. Once a program has been loaded to

the PLC from the personal computer, the personal computer may be unplugged from the PLC, and the PLC will continue to follow the programmed commands.

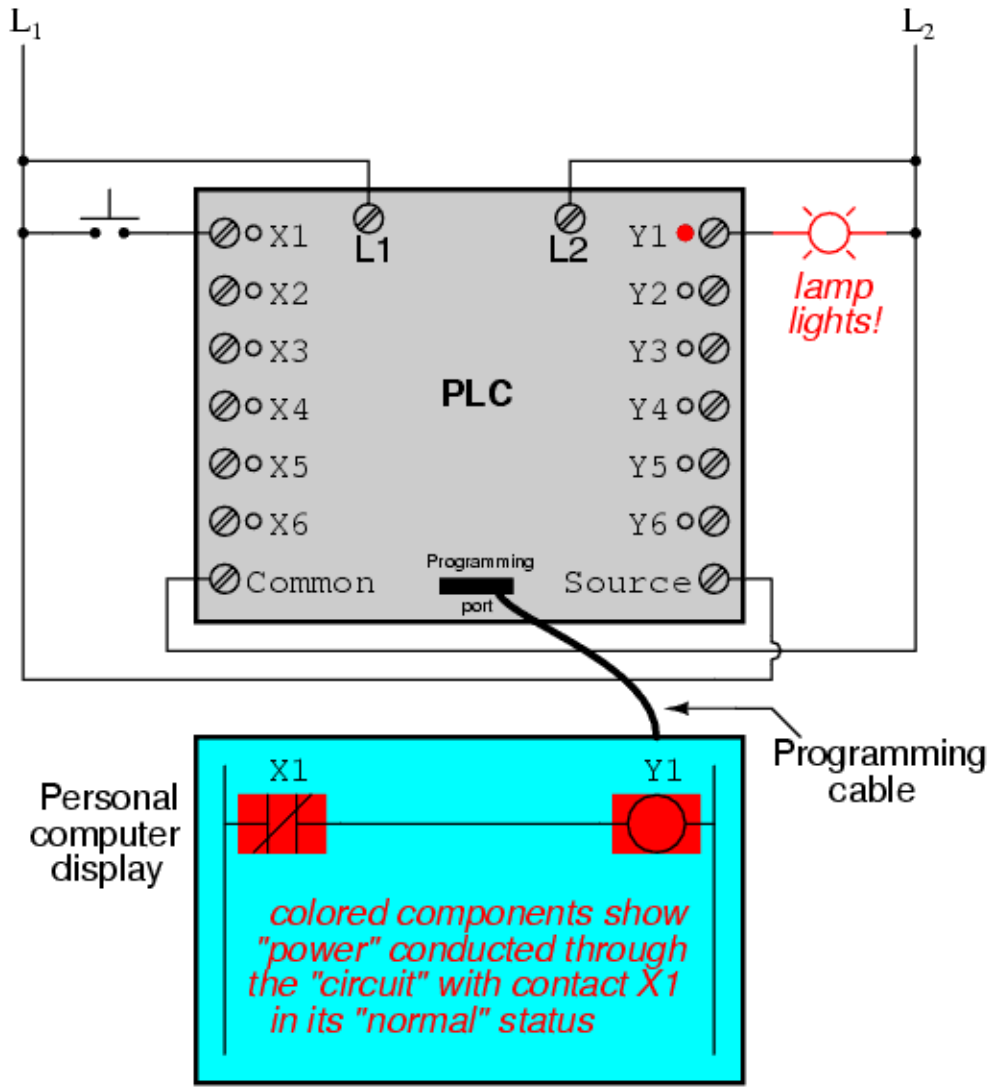
I include the personal computer display in these illustrations for your sake only, in aiding to understand the relationship between real-life conditions (switch closure and lamp status) and the program's status ("power" through virtual contacts and virtual coils).

Control System Behavior

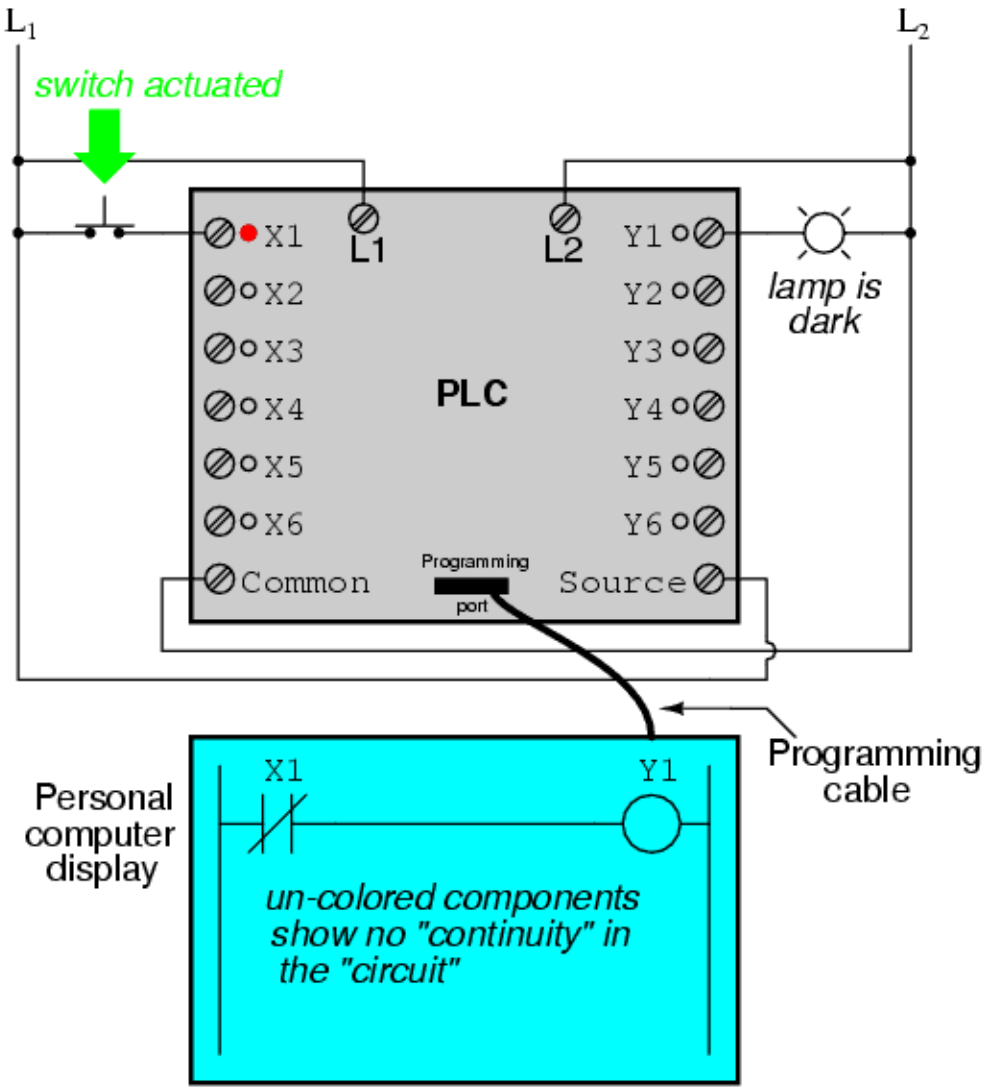
The true power and versatility of a PLC is revealed when we want to alter the behavior of a control system. Since the PLC is a programmable device, we can alter its behavior by changing the commands we give it, without having to reconfigure the electrical components connected to it.

For example, suppose we wanted to make this switch-and-lamp circuit function in an inverted fashion: push the button to make the lamp turn *off*, and release it to make it turn *on*. The "hardware" solution would require that a normally-closed pushbutton switch be substituted for the normally-open switch currently in place. The "software" solution is much easier: just alter the program so that contact X1 is normally-closed rather than normally-open.

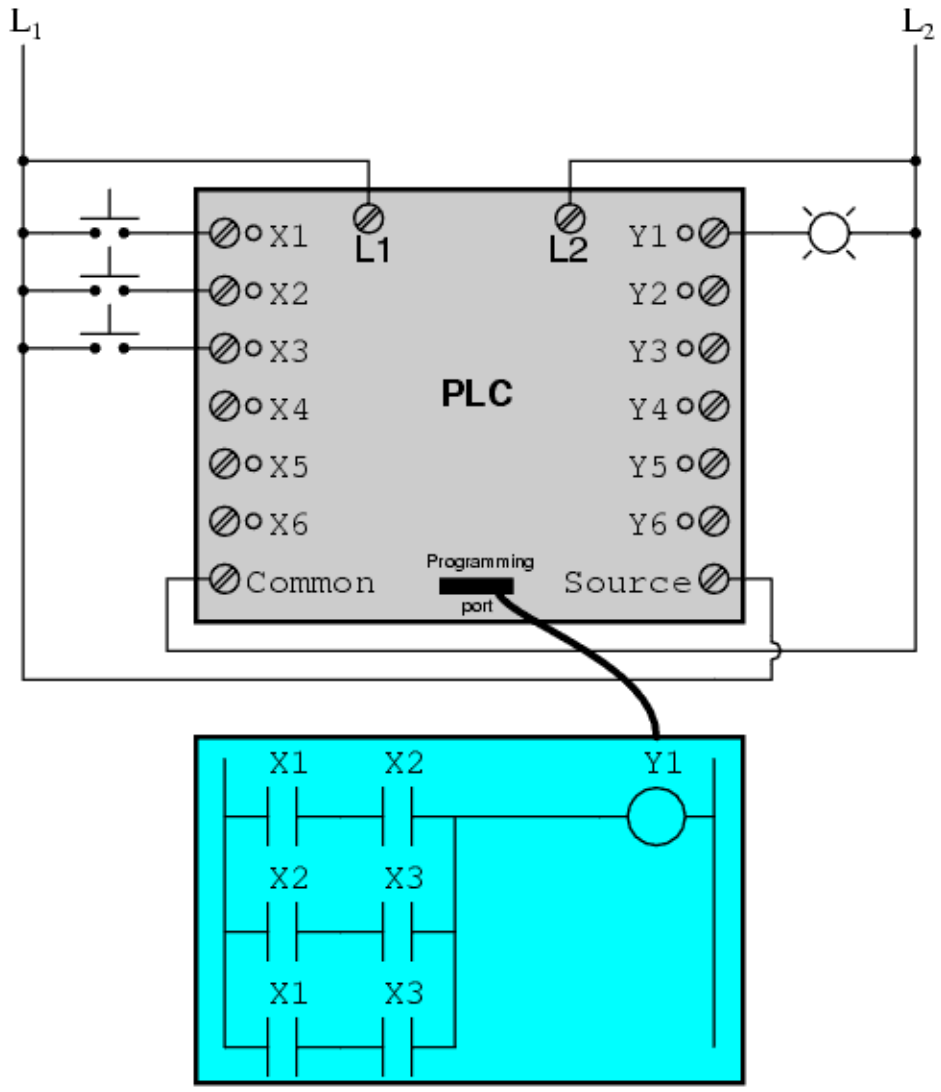
In the following illustration, we have the altered system shown in the state where the pushbutton is unactuated (*not* being pressed):



In this next illustration, the switch is shown actuated (pressed):

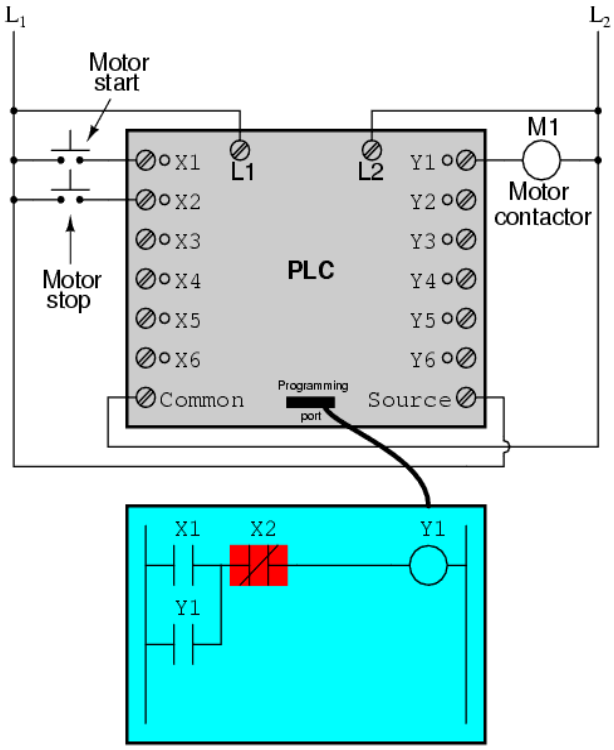


One of the advantages of implementing logical control in software rather than in hardware is that input signals can be re-used as many times in the program as is necessary. For example, take the following circuit and program, designed to energize the lamp if at least two of the three pushbutton switches are simultaneously actuated:



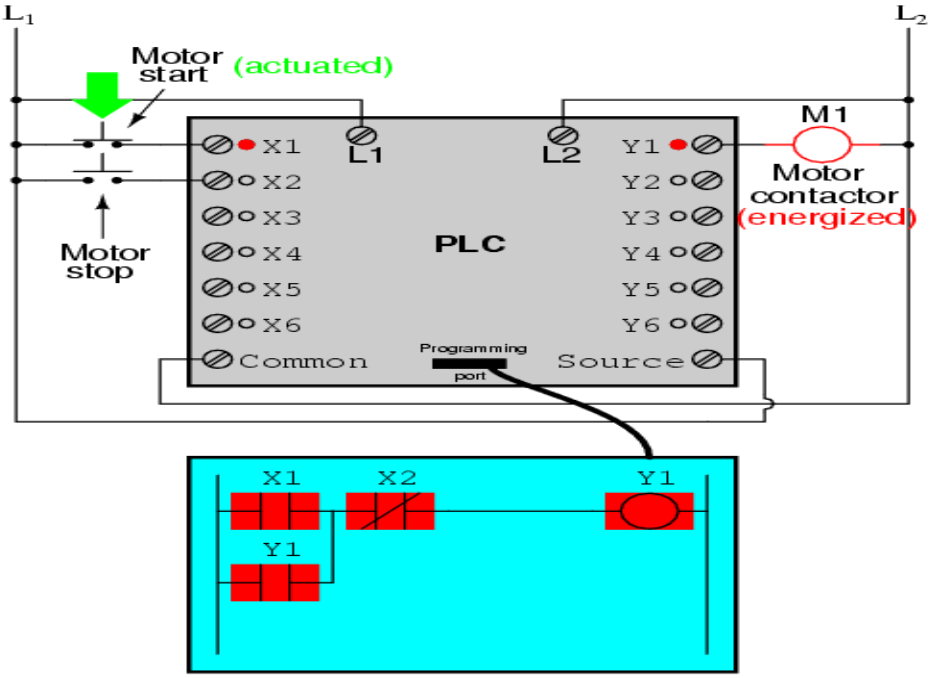
To build an equivalent circuit using electromechanical relays, three relays with two normally-open contacts each would have to be used, to provide two contacts per input switch. Using a PLC, however, we can program as many contacts as we wish for each "X" input without adding additional hardware, since each input and each output is nothing more than a single bit in the PLC's digital memory (either 0 or 1), and can be recalled as many times as necessary.

Furthermore, since each output in the PLC is nothing more than a bit in its memory as well, we can assign contacts in a PLC program "actuated" by an output (Y) status. Take for instance this next system, a motor start-stop control circuit:

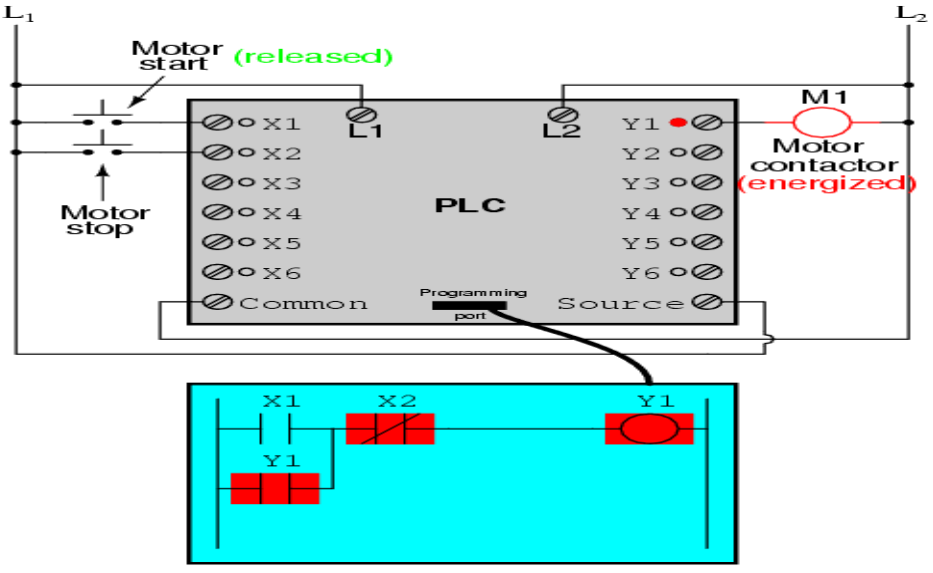


The pushbutton switch connected to input X1 serves as the "Start" switch, while the switch connected to input X2 serves as the "Stop." Another contact in the program, named Y1, uses the output coil status as a seal-in contact, directly, so that the motor contactor will continue to be energized after the "Start" pushbutton switch is released. You can see the normally-closed contact X2 appear in a colored block, showing that it is in a closed ("electrically conducting") state.

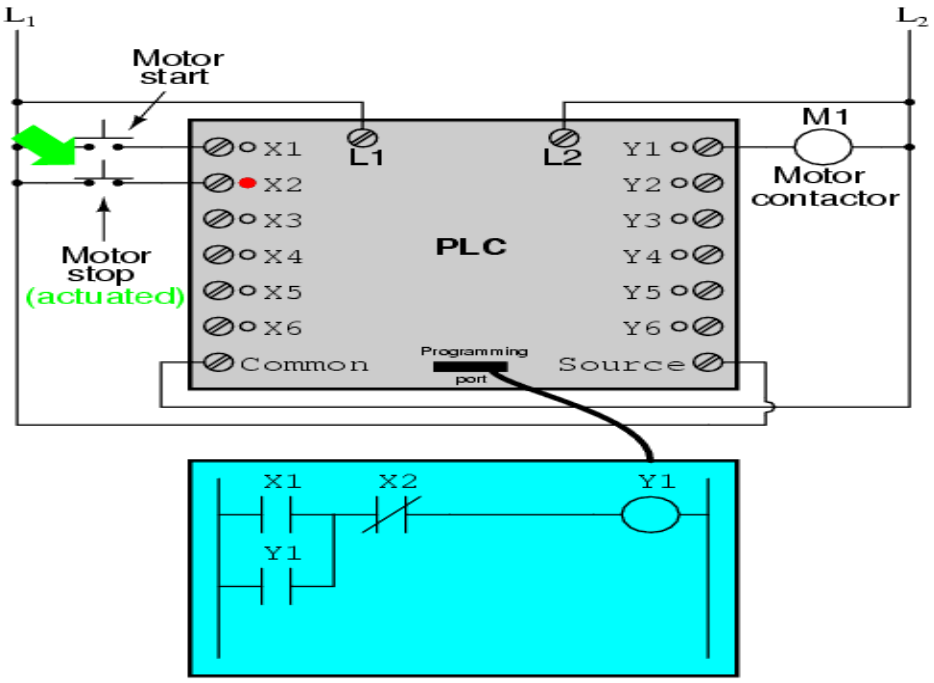
If we were to press the "Start" button, input X1 would energize, thus "closing" the X1 contact in the program, sending "power" to the Y1 "coil," energizing the Y1 output and applying 120 volt AC power to the real motor contactor coil. The parallel Y1 contact will also "close," thus latching the "circuit" in an energized state:



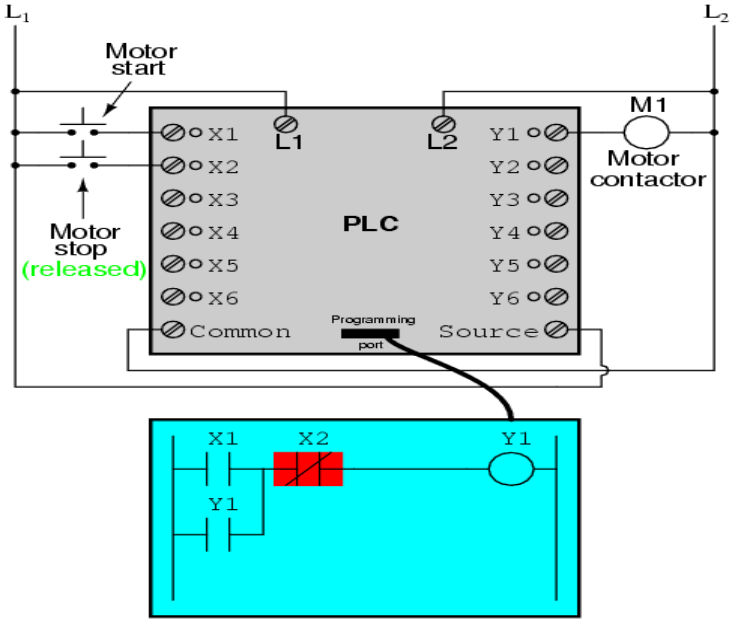
Now, if we release the "Start" pushbutton, the normally-open X1 "contact" will return to its "open" state, but the motor will continue to run because the Y1 seal-in "contact" continues to provide "continuity" to "power" coil Y1, thus keeping the Y1 output energized:



To stop the motor, we must momentarily press the “Stop” pushbutton, which will energize the X2 input and “open” the normally-closed “contact,” breaking continuity to the Y1 “coil:”



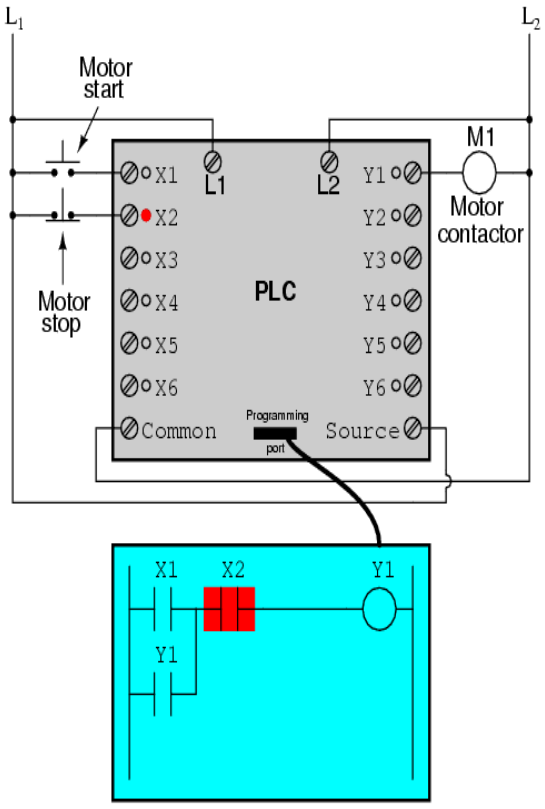
When the “Stop” pushbutton is released, input X2 will de-energize, returning “contact” X2 to its normal, “closed” state. The motor, however, will not start again until the “Start” pushbutton is actuated, because the “seal-in” of Y1 has been lost:



Fail-safe Design in PLC-Controlled Systems

An important point to make here is that *fail-safe* design is just as important in PLC-controlled systems as it is in electromechanical relay-controlled systems. One should always consider the effects of failed (open) wiring on the device or devices being controlled. In this motor control circuit example, we have a problem: if the input wiring for X2 (the "Stop" switch) were to fail open, there would be no way to stop the motor!

The solution to this problem is a reversal of logic between the X2 "contact" inside the PLC program and the actual "Stop" pushbutton switch:



When the normally-closed "Stop" pushbutton switch is unactuated (not pressed), the PLC's X2 input will be energized, thus "closing" the X2 "contact" inside the program. This allows the motor to be started when input X1 is energized, and allows it to continue to run when the "Start" pushbutton is no longer pressed. When the "Stop" pushbutton is actuated, input X2 will de-energize, thus "opening" the X2 "contact" inside the PLC program and shutting off the motor.

So, we see there is no operational difference between this new design and the previous design. However, if the input wiring on input X2 were to fail open, X2 input would de-energize in the same manner as when the "Stop" pushbutton is pressed. The result, then, for a wiring failure on the X2 input is that the motor will immediately shut off.

This is a safer design than the one previously shown, where a "Stop" switch wiring failure would have resulted in an *inability* to turn off the motor. In addition to input (X) and output (Y) program elements, PLCs provide "internal" coils and contacts with no intrinsic connection to the outside world. These are used much the same as "control relays" (CR1, CR2, etc.) are used in standard relay circuits: to provide logic signal inversion when necessary.

To demonstrate how one of these "internal" relays might be used, consider the following example circuit and program, designed to emulate the function of a three-input NAND gate.

Since PLC program elements are typically designed by single letters, I will call the internal control relay "C1" rather than "CR1" as would be customary in a relay control circuit:

In this circuit, the lamp will remain lit so long as *any* of the pushbuttons remain unactuated (unpressed). To make the lamp turn off, we will have to actuate (press) *all* three switches, like this:

Advanced PLC Functionality

This section on programmable logic controllers illustrates just a small sample of their capabilities. As computers, PLCs can perform timing functions (for the equivalent of time-delay relays), drum sequencing, and other advanced functions with far greater accuracy and reliability than what is possible using electromechanical logic devices. Most PLCs have the capacity for far more than six inputs and six outputs. The following photograph shows several input and output modules of a single Allen-Bradley PLC.

With each module having sixteen "points" of either input or output, this PLC has the ability to monitor and control dozens of devices. Fit into a control cabinet, a PLC takes up little room, especially considering the equivalent space that would be needed by electromechanical relays to perform the same functions:

Remote Monitoring and Control of PLCs Via Digital Computer Networks

One advantage of PLCs that simply *cannot* be duplicated by electromechanical relays is remote monitoring and control via digital computer networks. Because a PLC is nothing more than a special-purpose digital computer, it has the ability to communicate with other computers rather easily. The following photograph shows a personal computer displaying a graphic image of a real liquid-level process (a pumping, or "lift," station for a municipal wastewater treatment

system) controlled by a PLC.

The actual pumping station is located miles away from the personal computer display:

Timers and Counters

Timers and counters are internal instructions that provide the same functions as timing relays and counters. They are used to activate or de-activate a device after a preset interval of time.

Timers and Counters

The timer is assigned an address as well as being identified as a timer. Also included, as part of the timer instruction is the time base of the timer, the timer's preset value, and the accumulated value.

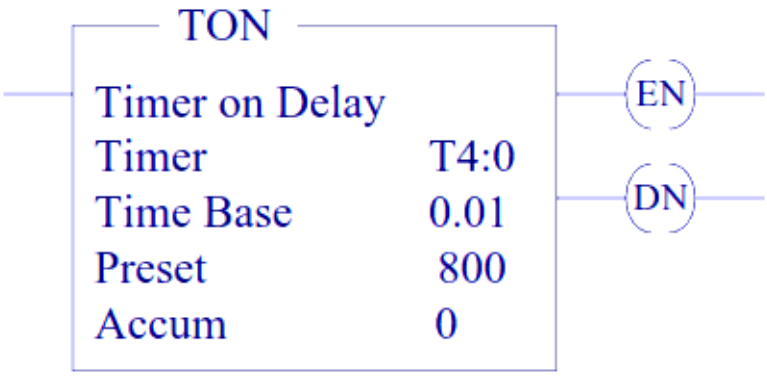
The timer instructions include:

Timer On-Delay (TON)

This instruction is programmed to provide time delay action. Once the rung has continuity, the timer begins counting time-based intervals and times until the accumulated value equals the preset value.

When the accumulated time equals the preset time, the output is energized, and the timed output contact associated with the output is closed. The timed contact can be used throughout the program as an NO or NC contact.

The accumulated value is reset when rung condition goes false.



For SLC-500 processor the time base can be selected as 0.01 sec or 1.0 sec.

The control word uses three control bits:

Enable (EN) bit

The enable bit (EN) is set when rung conditions are true; it is reset when the rung condition becomes false.

Done (DN) bit

The done bit (DN) is set when the accumulated value is equal to the preset value. It is reset when rung condition becomes false.

Timer-timing (TT) bit

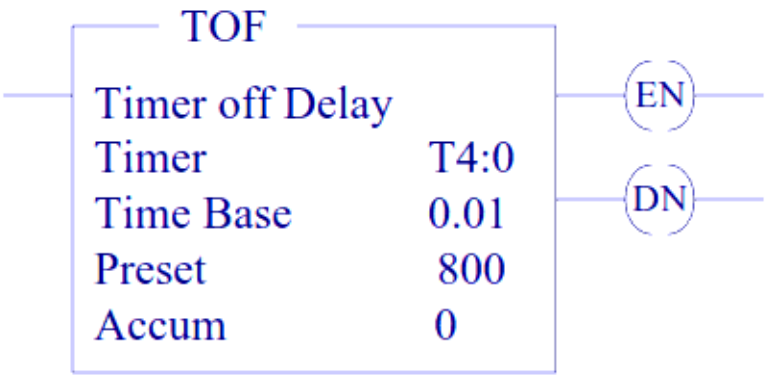
The Timer-timing bit (TT) is true when the timer is timing. When the timer is not timing the TT bit is false.

Timer Off-Delay (TOF)

This instruction is programmed to provide time delay action. If the rung does not have continuity, the timer begins counting time-based intervals and times until the accumulated value equals the preset value.

When the accumulated time equals the preset time, the output is energized, and the timed output contact associated with the output is closed. The timed contact can be used throughout the program as an NO or NC contact.

The accumulated value is reset when rung condition goes true.



The done bit (DN) is set when the accumulated value is equal to the preset value. It is reset when rung condition becomes true. The enable bit (EN) is set when rung conditions are true; it is reset when the rung condition becomes false.

Counters

Counters are similar to timers, except that they do not operate on an internal clock, but are dependent upon external or program sources for counting. The counter is assigned an address as well as being identified as a counter.

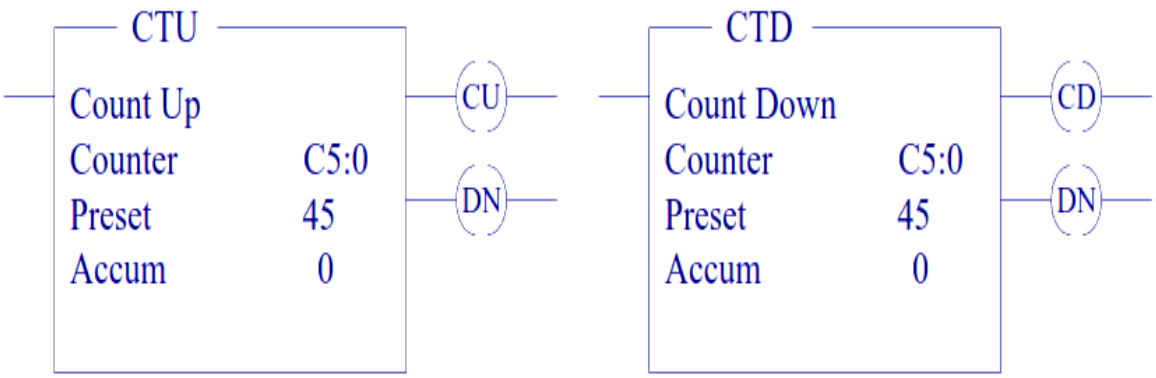
Also included, as part of the counter instruction is the counter's preset value as well as the current accumulated count for the counter. There are two basic types of counters, one that counts up and another one that can count down.

Count Up (CTU) and Count Down (CTD)

The up-counter output instruction will increment by one and the down-counter output instruction will decrement by one each time the counted event occurs.

These events could be caused by the number parts traveling past a detector or a limit switch. When the accumulated counts equal the preset count, the output is energized, and the counter output is closed.

The counter contact can be used throughout the program as an NO or NC contact



The done bit (DN) is set and remains set when the accumulated value is equal to the preset value. It is reset when rung condition becomes true.

The count up enable bit (CU) or the count down enable is set when rung conditions are true; it is reset when the rung condition becomes false or the appropriate reset instruction is enabled.

Counter and Timer Reset (RES)

The [RES] instruction is used to reset the timing and counting instruction accumulated values. A rung containing an NO or an NC contact together with the [RES] instruction is used.

The [RES] instruction must be given the same reference address as the related timer or counter. When the [RES] instruction is enabled, the counter accumulated value is reset.

Basics of Computers - Number System

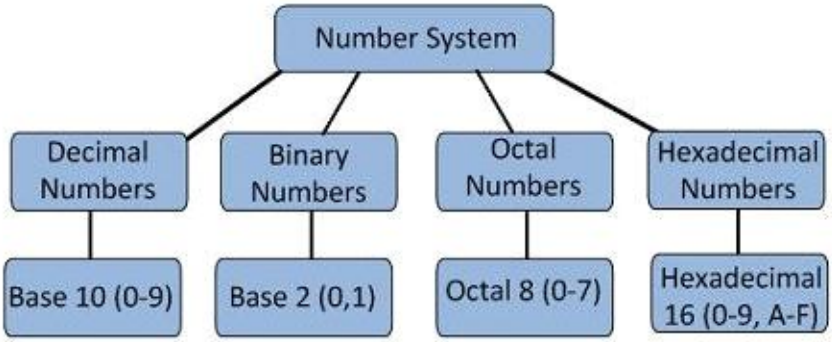
Types of Number Systems

Some of the important types of number system are

- Decimal Number System
- Binary Number System

- Octal Number System
- Hexadecimal Number System

These number systems are explained below in details.



1. Decimal Number Systems

The number system is having digit 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; this number system is known as a decimal number system because total ten digits are involved.The base of the decimal number system is 10.

2. Binary Number Systems

The modern computers do not process decimal number; they work with another number system known as a binary number system which uses only two digits 0 and1.The base of binary number system is 2 because it has only two digit 0 and 1.The digital electronic equipments are works on the binary number system and hence the decimal number system is converted into binary system.

The table is shown below the decimal, binary, octal, and hexadecimal numbers from 0 to 15 and their equivalent binary number.

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2

3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Showing 1 to 16 of 16 entries

3. Octal Numbers

The base of a number system is equal to the number of digits used, i.e., for decimal number system the base is ten while for the binary system the base is two. The octal system has the base of eight as it uses eight digits 0, 1, 2, 3, 4, 5, 6, 7.

All these digits from 0 to 7 have the same physical meaning as by decimal symbols, the next digit in the octal number is represented by 10, 11, 12, which are equivalent to decimal digits 8, 9, 10 respectively. In this way, the octal number 20 will represent the decimal digit and subsequently, 21, 22, 23.. Octal numbers will represent the decimal number digit 17, 18, 19… etc. and so on.

4. Hexadecimal Numbers

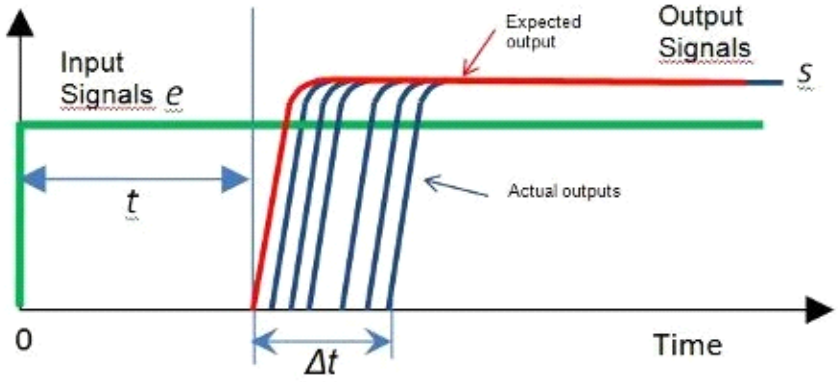
These numbers are used extensively in microprocessor work.The hexadecimal number system has a base of 16, and hence it consists of the following sixteen number of digits.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

The size of the hexadecimal is much shorter than the binary number which makes them easy to write and remember. Let 0000 to 000F representing hexadecimal numbers from zero to fifteen, then 0010, 0011, 0012, …etc. Will represent sixteen, seventeen, eighteen… etc. till 001F which represent thirty open and so on.

Fundamentals of real-time processing in automation and control

Almost all automation and control systems are designed, developed and built using data processors, microprocessors, digital signal processors (DSPs) or any other processing device. Part 1 of a series of articles on real-time control systems.



This is the first part of a series of articles on real-time systems, real-time processing and fundamentals of real-time software design, applied to automation and control systems.

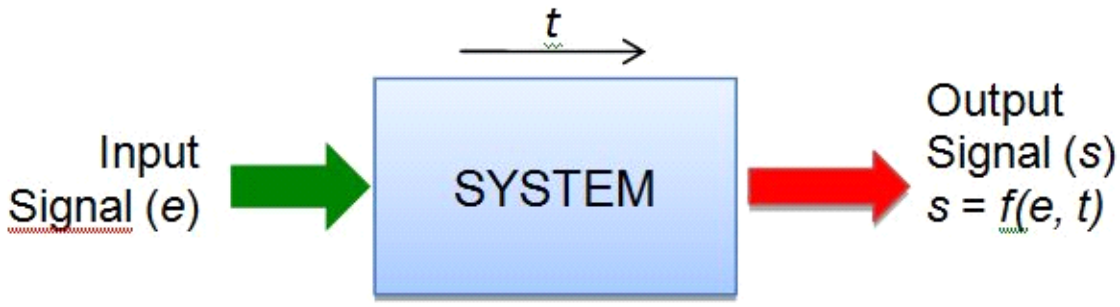


Figure 1: A simple system

Today, almost all automation and control systems are designed, developed and built using data processors, microprocessors, DSPs or any other processing device, which execute instructions derived or compiled from a software program. Just very few and specialized control devices still use plain hardware (or "programmable" hardware, as FPGA) to accomplish the job for which they were designed.

Nevertheless, to design and develop software applications for a controller or automation device, special skills and a good understanding of the automation and control problem are required. Many designers and programmers are around, building web servers, information systems, business processes management systems, and many other important and valuable infrastructures based on computing platforms. But when it comes to industrial process controllers, the designer recognizes that a new design and programming paradigm is required. This is well accepted throughout the automation and control industry.

Real-time processing

This special paradigm is closely related with many concepts involving signals acquisition, transducers, control set points and other aspects mostly related to process state variables, measurements and control. But mostly, such paradigm is closely connected with real-time processing. The concept of real-time processing is the main paramount that any engineer must take into account when a new automation and control system is designed, developed and deployed. This is, in other words, what differentiates automation software designers from designers of any other application software. It's not better or worse, easier or harder; it's just different.

Automation controllers are required to be real-time systems because they must control physical processes or plants that demand real-time control. At this point, the automation controller designer must have a very good idea on what a real-time system is. Hermann Kopetz, in his

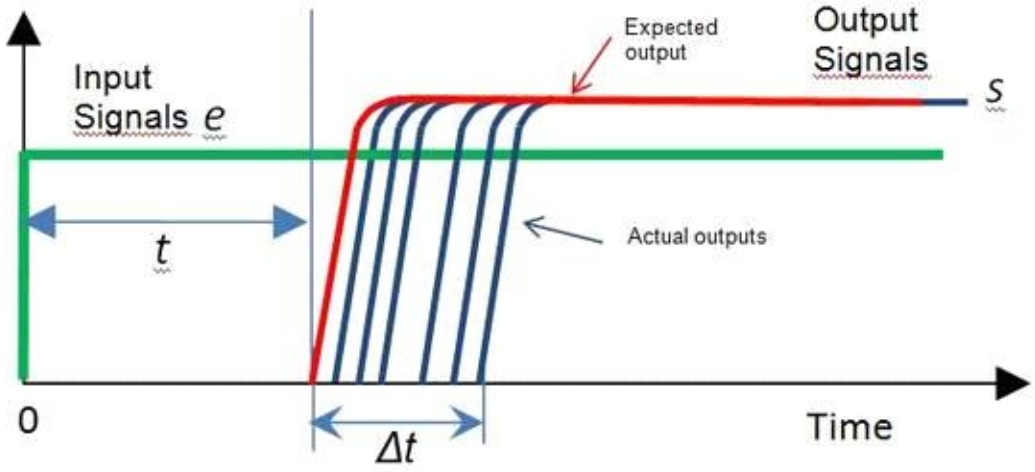
book, "Real Time Systems: Design Principles for Distributed Embedded Applications" (2011), stated that "A real-time computer system is a computer system where the correctness of the system behavior depends not only on the logical results of the computations, but also on the physical time when these results are produced. By system behavior we mean the sequence of outputs in time of a system." For any physical plant to be governed, it requires a controller capable of acquiring some input signals and producing some specific output signals within a very specific time frame. If the controller outputs occur outside such time frame, those outputs will no longer be valid, and will produce malfunctions or even a catastrophic failure in the physical plant.

Real-time explanation

How can we determine that a computer system is actually a real-time system? Figure 1 (above) can help answer this question. Suppose there is a very simple system, with just one input and one output. Such system generates an output signal every time it receives an input signal. Please note in Figure 1 that the output signal is produced t seconds after the input signal is introduced.

Now suppose that we repeatedly (but not periodically) apply the same input signal to this system. We may expect that this system will generate the same output signal every time, at exactly *t seconds* after the input signal is applied. Unfortunately, this is not what happens in actual controllers. The same output signal will be generated by the controller every time, but the time *t* that it takes the controller to produce each output may increase slightly (see Figure 2). If we repeat this experiment, we will find that the controller response times fall into a variation interval, say Δt. Some literature refers to Δt as "latency jitter."

The question is: Who determines this magnitude Δt? It strongly depends on how the controller hardware was designed, which components were used, and how the application software running inside it was designed and developed.



At this stage, something needs to be mentioned: the time magnitude *t* is not related to the time

variation value, Δt. The former depends on the function that the controller is supposed to perform, while the latter depends on how the designer implemented such function inside the controller.

It is important to point out that no controller system can reach $\Delta t = 0$. Therefore, the controller designer must know the maximum that the physical plant would allow without causing any failures or damages, and consequently he/she must build the right real-time controller which takes into account such constraint.

Hard real time, soft real time

Depending of the physical plant and the type of control to be performed, the controller may be classified as "hard real time" or "soft real time." If the specific characteristics of the plant or process to be controlled are such that a non-compliance of its Δt constraint will produce a malfunction or a failure, then the controller must be a "hard real-time" controller. If, on the other hand, the specific characteristics to be controlled are such that a non-compliance of its Δt constraint will generate a degradation of the plant functionality, but will not produce any malfunctions or failures, then a "soft real-time" controller may be used.

The value of Δt to be used as the main constraint for a process controller design depends heavily on the nature of the process that is to be controlled. Each physical process has its own "latency," that is, the mean time the process reacts from a change in one or several of the inputs. This latency is tied to the physical, chemical and electrical laws governing such process. For instance, the latency of an oil production process is very different from the latency of an electric power transmission system.

The table below shows latency times for some process types.

Table 1: Average Latencies

Process type	Avg. Latency
Oil Production & Transport	2-15 minutes
Water Transport	2-5 minutes
Gas Production & Transport	30-60 seconds
Electric Power Generation & Transmission	4 – 16 milliseconds
Telecommunications	2-5 milliseconds
Nuclear Power Generation	1-2 milliseconds
Military Weaponry	100-200 microseconds

Based on experience, a basic "rule of thumb" for automation controller design is that the controller's Δt must be at least 5 times smaller than the latency of the process such controller is intended to govern. Of course, this also depends of many other considerations, like power consumption, heat dissipation, available space and many other constraints, not discussed here.

Designing a hard real-time controller is a lot harder than building a soft real-time controller. Depending on the physical process and the specific application, the designer must chose to build

one or the other.

Most controllers for industrial applications available in the market are soft real-time controllers. The good news about this is that almost all industrial automation and control applications may be governed by soft real-time controllers. Just in a few, very specific situations, a hard real-time controller is justified and installed. But, as long as a good controller is designed or chosen, in which its latency jitter Δt is small enough compared to the process latency, a reliable automation solution may be provided.

Next up: Reliability criteria

What are the criteria to adequately design a reliable controller? This interesting subject will be the content of the next articles of this series.