DEPARTMENT OF APPLIED CIENCES

LASER DIVISION

DIGITAL ELECTRONICS LECTURE NOTES

4TH YEAR

WALID K. HAMOUDEI

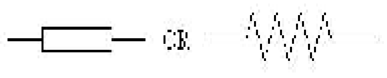
Summary of basic electronic parameters and components

Voltage: Voltage is the difference in charge between two points, measured in Volts. **Current:** Current is the flow of electrons through a conductor or semiconductor, measured in Amperes or Amps. Some materials conduct current better than others; these are known as conductors, semiconductors, and insulators. Current flow is from positive to negative. **Power:** Power determines how much work a circuit can do. It is measured in Watts (Watts = Volts * Amps).

Ground: Minimum voltage reference level. True ground connects to the earth but the circuits we work with may not actually be connected to the earth, especially if they are battery powered. Technically this is known as a floating ground.

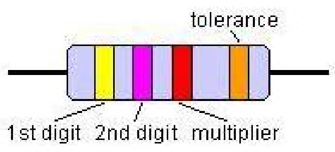
Resistance

Resistors are measured in Ohm and come between conductors, which conduct easily and insulators which don't conduct. The main function of resistors in a circuit is to control the flow of current and voltage drops to other components. For example; if too much current flows through an LED it is destroyed and will not light, so a resistor is used to limit the current but not so big as it will limit all the current. When a current flows through a resistor, energy is wasted and the resistor heats up. This will only be noticed if the resistor is working at its maximum power rating. The greater the current flowing through the resistor the hotter it gets. A battery or power supply has to do work to force electrons through the resistor and this work ends up as heat energy in the resistor. An important property to know about resistors is how much heat energy it can withstand before it's damaged or causes a fire. Resistors can dissipate different powers (Watts) depending on its power rating and the current passing through. It is difficult to make a resistor to an exact value, so resistances are given a tolerance. This is expressed as being plus or minus a percentage. A ±10% resistor with a stated value of 100 ohms could have a resistance anywhere between 90 ohms and 110 ohms. In circuit diagrams you will often see an 'R' instead of omega to represent ohms. The symbol and a few examples of this type are shown below:



Resistor Color Code

The resistor color code is a way of showing the value of a resistor. Instead of writing the resistance on its body, which would often be too small to read, a color code is used. Different colors represent the numbers 0 to 9. The first two colored bands on the body are the first two digits of the resistance, and the third band is the 'multiplier'. Multiplier just means the number of zeroes to add after the first two digits. Red represents the number 2, so a resistor with red, red, red bands has a resistance of 2 followed by 2 followed by 2 zeroes, which is 2200 ohms or 2.2 kilo Ohms. The final band is the tolerance (the accuracy ± x %). All resistors have a tolerance which is shown by the last band.

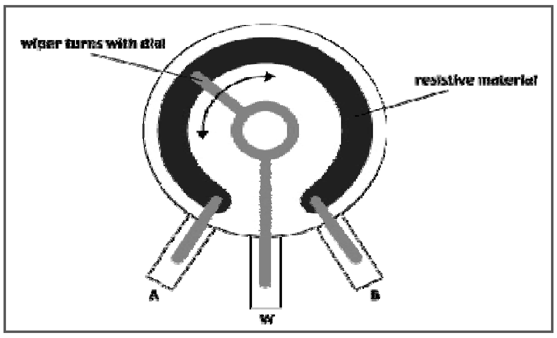


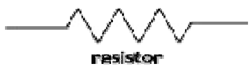
Color	1st Band	2nd Band	3rd Band	4th Band
Black	0	0	1	
Brown	1	1	10	
Red	2	2	100	
Orange	3	3	1000	
Yellow	4	4	10000	
Green	5	5	100000	
Blue	6	6	1000000	
Purple	7	7		
Grey	8	8		
White	9	9		
Red				1%
Gold				5%
Silver				10%

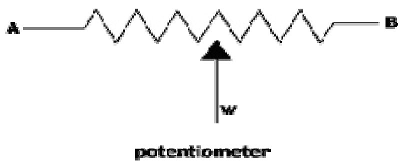
Examples:

*Yellow, Purple, Red, Gold = 47 x 100 = 4 700 Ω = 4.7 kΩ + 5%
*Brown, Black, Yellow, Gold = 10 x 10 000 = 100 kΩ + 5%
*Yellow, Purple, Black, Silver = 47 x 1 = 47 Ω + 10%
*Brown, Black, Red, Red = 10 x 100 = 1 000 Ω = 1 kΩ + 1%
*Brown, Black, Green, Gold = 10 x 100 000 = 1 000 kΩ = 1 MΩ + 5%

Potentiometer: It is a variable resistor, a wiper moves between two leads and the resistance between wiper and lead determines resistance. Resistance between leads is maximum resistance of potentiometer. With linear pots, resistance varies directly with the rotation of the knob while with logarithmic pots; resistance varies exponentially with the rotation of the knob.

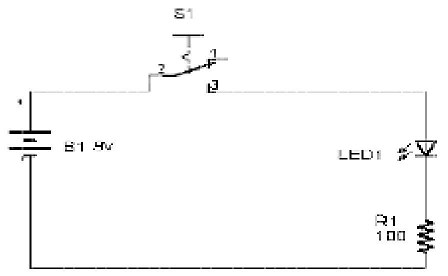




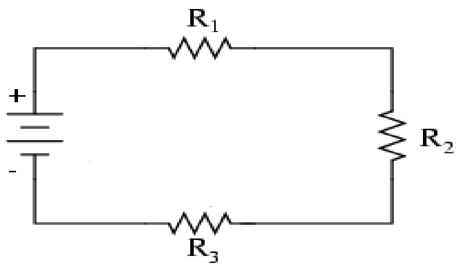


Ohms Law: Every circuit has Voltage, Current and Resistance.
V=IR. Voltage = Current * Resistance.
I=V/R. Current = Voltage/Resistance.
R=V/I. Resistance = Voltage/Current.

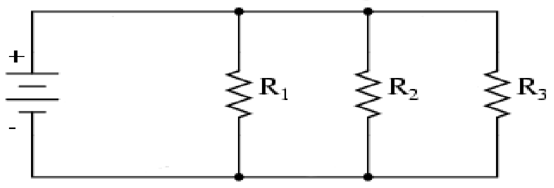
Circuits: A working circuit must have a closed loop of current flow through a load. The sum of the current entering a junction equals the sum of the current exiting a junction. Kirchoff's current law is: $I_{in} = I_{out}$. A circuit with a break in it is called an open circuit. A circuit without enough resistance in its load is called a short circuit. Switching an LED is shown at the example schematic below.



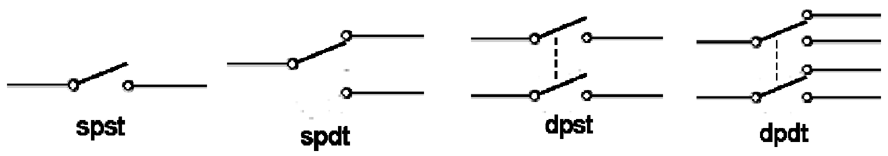
Series Circuits: All components are connected end to end. Single path for electrons to flow - all components share the same current. Total resistance of circuit is equal to sum of individual resistances. Total voltage in the circuit is equal to the sum of individual voltage drops.



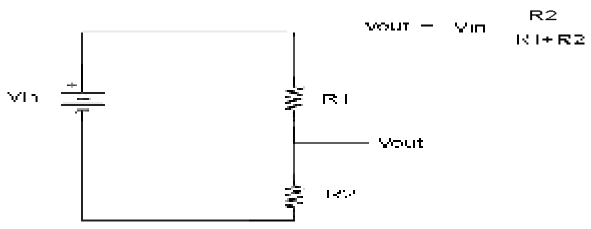
Parallel Circuits: All components are connected in parallel and share the same voltage. The total resistance of circuit is less than the value after adding individual resistances. Total current in circuit is equal to sum of individual branch currents.



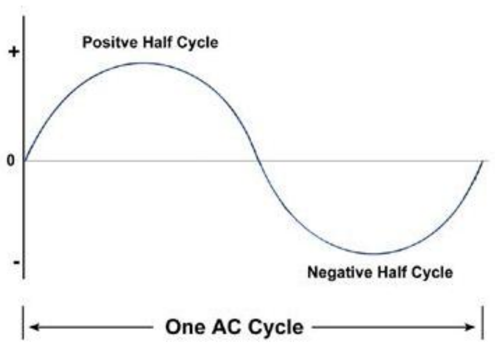
Switches: Mechanical devices that interrupt current flow. It is characterized by number of poles and number of throws.



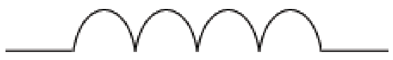
Voltage divider: With two resistors in series, the sum of the voltage drop across each equals the total voltage drop across both. To determine the voltage between the two resistors, we use the voltage divider formula; $V_{out} = V_{in}* (R2/R1+R2)$



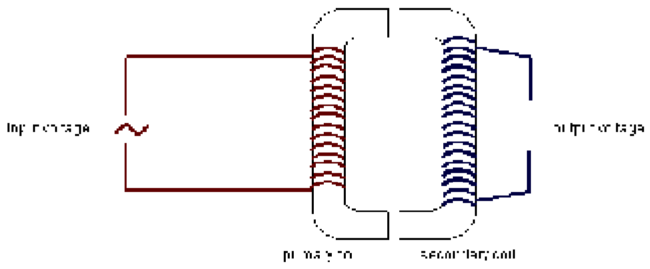
Alternating Current: AC Voltage alternates sinusoidal with time. AC Voltages are specified with a value equal to the DC voltage capable of doing the same amount of work. This value is 1/√2 times the peak voltage and is called the root means square or rms voltage. 1/√2 = 0.707. Household line voltages are specified at 240 Vac, meaning peak voltage is about 363v. An AC voltage can have a DC component and vice versa.



Inductors: Their values are measured in Henry and are commonly used as AC filters. By coiling wire we can increase strength of magnetic field created by current. This is called an inductor. A large inductor functions as an electromagnet. Strength of the magnetic field depends on number of coil turns, coil size, coil spacing, winding arrangement, core material, and shape of inductor.



Transformers: Four terminal device which turns ac input voltage into a higher or lower output voltage. Transformers consist of two coils called primary and secondary sharing a common iron core. Ratio of turns between primary and secondary coil determines step up/step down value. Power (V*I) is the same in the primary and secondary coil. Stepping down the voltage increases current while Stepping up the voltage decreases the current.

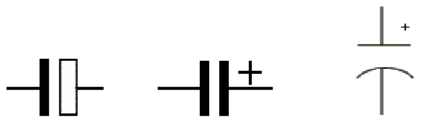


Relays: It is switch operated by an electromagnet and controlled by electrically isolated signal from switched current. It is slow, noisy and can pass AC or DC current. It generates unfriendly voltage spike when magnetic field in coil collapses.

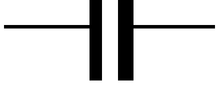
Inductive versus Resistive loads Inductive loads use magnetic fields as in motors, solenoids, and relays. If it moves, it's probably an inductive load. They can cause blowback voltage and circuits should be protected from this by diodes. Blowback is caused by a surge of voltage created by the collapsing magnetic field in an inductor. Resistive loads convert current into other forms of energy, such as heat.

Capacitors

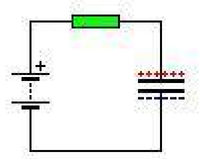
Capacitors are components that store an electrical charge. They can be charged up with energy from a battery, then return that energy back later. The capacitance of a capacitor is a measure of how much energy/charge it can store. In its simplest form a capacitor consists of two separated metal plates with air or another non-conductive material filling the gap, the bigger the plates the bigger the capacitance. To stop capacitors becoming impractically large they can be rolled up. Another way of increasing the capacitance is to put some non-conducting material between the plates. This is called a dielectric material. When a capacitor charges up, the protons and electrons in the dielectric separate out a little, this allows more charge to be stored on the plates than usual. Dielectrics are made of various materials Ceramic, paper, polyester, polystyrene, mica, etc. Capacitance is measured in Farads; one Farad is a very big unit and is usually found in the range of picot-to-micro farads. Capacitors come in two types, electrolytic and non-electrolytic. Electrolytic capacitors use special dielectrics sometimes a solid but the most common types are a liquid or paste which is formed into a very thin dielectric in the factory. Non-electrolytic capacitors have solid dielectrics. The symbol for electrolytic capacitors and a few examples of this type is shown below:



The symbol for non-electrolytic capacitors is shown below:

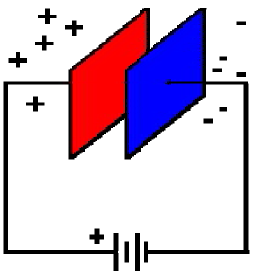


Electrolytic capacitors can store more charge but there are a couple of problems. They have a polarity and must be connected the right way around in a circuit. They also slowly leak their charge, and they have quite large tolerances i.e. 10% to 20%. Where as non-electrolytic capacitors still leak but not as fast as electrolytic and do not have a polarity problem but store less charge. When a capacitor is connected to a source it begins to charge. The current flows rapidly at first then more slowly as it gets to maximum its charge. Charge builds up on the two plates, negative charge on one plate and the same amount of positive charge on the other. The positive charge results from electrons leaving one of the plates and leaving positively-charged protons behind. As the capacitor fills with charge, it starts to oppose the current flowing in the circuit. It is as if another battery were working against the first. The current decreases and the capacitor charges more slowly. The plates become full of charge when no current flows, and the circuit acts like an open type.



charging a capacitor through a resistor

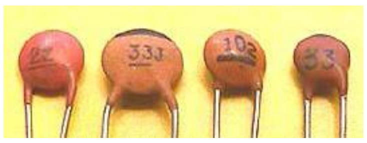
If a capacitor is shorted then it discharges. Charge flows out of the capacitor rapidly at first, then progressively more slowly. The speed at which the capacitor empties or charges depends on the resistance. If a simple wire shorts out a capacitor then it empties in a flash, often with a spark if it's a big capacitor. We've mentioned that when a capacitor is fully charged the current stops. In other words a continuous current cannot flow through a capacitor. A continuous current is called a direct current or D.C. An alternating current (A.C.) however can flow through a capacitor. An alternating current is one which is continually changing its direction. The Mains is A.C. and changes its direction 50 times a second. An alternating current continually charges and discharges a capacitor and hence is able to keep current flowing.



Capacitors hold charge when disconnected from power supply. Dielectric keeps charge from jumping from one plate to another. Lightening is a giant capacitive charge discharging. 1 Farad is equal to 1 amp of current at 1 volt for 1 second. Capacitors we work with are typically measured in Micro- Farads (µF) and Pico Farads (pF). Common uses of capacitors are camera flashes, lasers, decoupling noise, smoothing power supplies, timing etc.

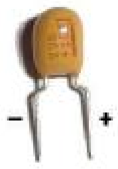
Capacitors - RC Time: Capacitors take time to charge and discharge, according to the amount of current. The charge/discharge time of capacitors is controlled using resistors. Charge time (to 63.2% of supply voltage) and discharge time (to 36.8% of supply voltage) is nicely equal to R*C (in seconds). RC Time allows us to control the rate that things happen in circuits, which turns out to be very useful.

Capacitor Types: Three major types of capacitors are ceramic, electrolytic, and tantalum. Ceramic capacitors are small in size and value, ranging from a few Pico Farads to 1 µF. Not polarized, so either end can go to ground. Value is given by a code somewhat like that of resistors.

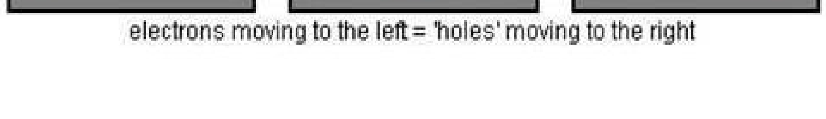


Electrolytic capacitors look like small cylinders and range in value from 1 µF to several Farads. Very inaccurate and change in value as the electrolytic ages. Polarized, cathode must go to ground. Cathode is marked with a minus sign on case. Value is usually written on case.

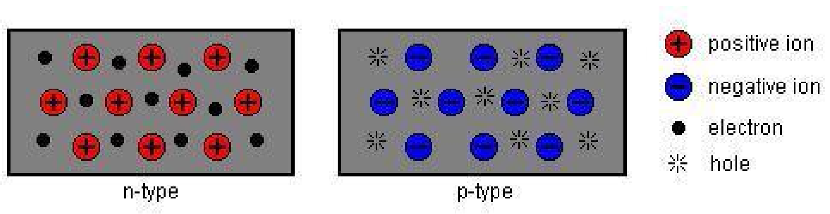
Tantalum capacitors are similar in size to ceramic but can hold more charge, up to several hundred µF. Accurate and stable, but relatively expensive. Usually polarized anode is marked with a plus sign.



Semiconductors: It is probably the most important discovery in electronics which happened last century. Without this discovery we wouldn't have televisions, computers, space rocket, CD players, etc. Unfortunately it's also one of the hardest areas to understand in electronics. The reason that makes metals such good conductors is that they have lots of electrons which are so loosely held that they're easily able to move when a voltage is applied. Insulators have fixed electrons and so are not able to conduct. Certain materials, called semiconductors, are insulators that have a few loose electrons. They are partly able to conduct a current. The free electrons in semiconductors leave behind a fixed positive charge when they move about (the protons in the atoms they come from). Charged atoms are called ions. The positive ions in semiconductors are able to capture electrons from nearby atoms. When an electron is captured another atom in the semiconductor becomes a positive ion. This behavior can be thought of as a 'hole' moving about the material, moving in just the same way that electrons move. So now there are two ways of conducting a current through a semiconductor, electrons moving in one direction and holes in the other. The holes don't really move of course. It is just fixed positive ions grabbing neighboring electrons, but it appears as if holes are moving.

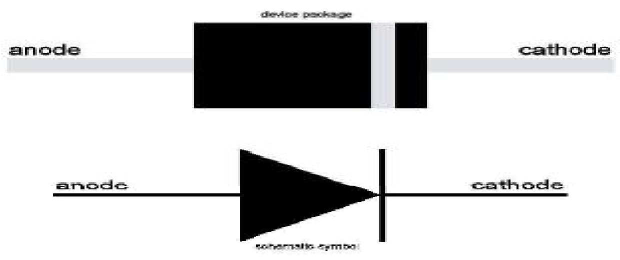

electrons moving to the left = 'holes' moving to the right

In a pure semiconductor there are not enough free electrons and holes to be of much use. Their number can be greatly increased however by adding an impurity, called a donor. If the donor gives up some extra free electrons we get an n-type semiconductor (n for negative). If the donor soaks up some of the free electrons we get a p-type semiconductor (p for positive). In both cases the impurity donates extra current carriers to the semiconductor. Adding impurities is called doping. In n-type semiconductors there are more electrons than holes and they are the main current carriers. In p-type semiconductors there are more holes than electrons and they are the main current carriers. The donor atoms become either positive ions (n-type) or negative ions (p-type).



The most common semiconductors are silicon (basically sand) and germanium. Common donors are arsenic and phosphorus. When we combine n-type and p-type semiconductors together we make useful devices, like transistors, diodes and chips.

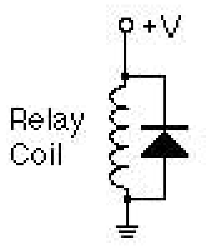
The Diode: Simplest useful semiconductor that allows current flow from anode to cathode but not in reverse. Cathode goes to ground.



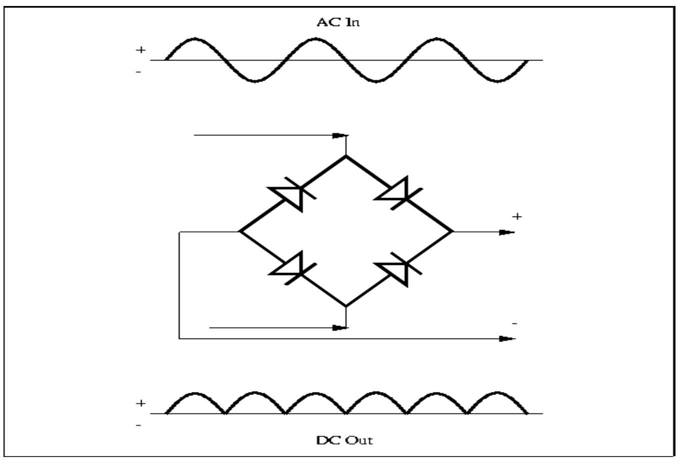
Diode applications examples

1) Reverse polarity protection.

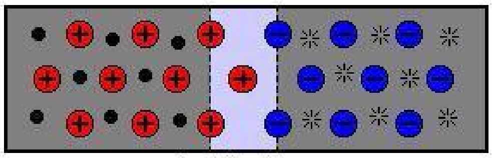
2) Reverse biased diode in parallel with an inductive load will snub the blowback current generated by the collapsing magnetic field.



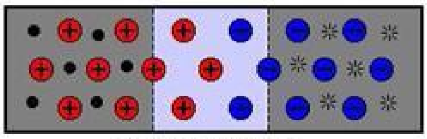
3) Rectifier converts AC into DC.

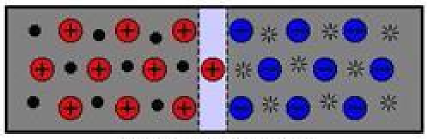


A diode consists of a piece of n-type and a piece of p-type semiconductor joined together to form a junction. Electrons in the n-type half of the diode are repelled away from the junction by the negative ions in the p-type region, and holes in the p-type half are repelled by the positive ions in the n-type region. A space on either side of the junction is left without either kind of current carriers. This is known as the depletion layer because there are no current carriers in this layer, so current can flow. The depletion layer is, in effect, an insulator.

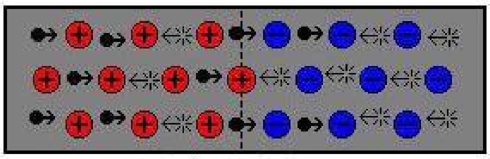

depletion layer

Consider what would happen if we connected a small voltage to the diode. Connected one way it would attract the current carriers away from the junction and make the depletion layer wider. Connected the other way it would repel the carriers and drive them towards the junction, so reducing the depletion layer. In neither case would any current flow because there would always be some of the depletion layer left.


depletion layer wider

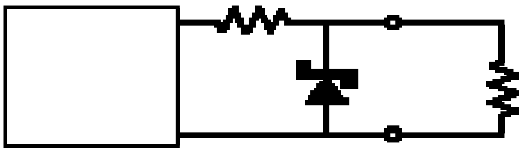

depletion layer narrower

Now consider increasing the voltage. In one direction there is still no current because the depletion layer is even wider (reverse biased), but in the other direction the layer disappears completely and current can flow (forward biased). Above a certain voltage the diode acts like a conductor. As electrons and holes meet each other at the junction they combine and disappear.


diode conducting

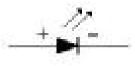
Thus a diode is a device which is an insulator in one direction and a conductor in the other. Diodes are extremely useful components. We can stop currents going where we don't want them to go. For example we can protect a circuit against the battery being connected backwards which might otherwise damage it.

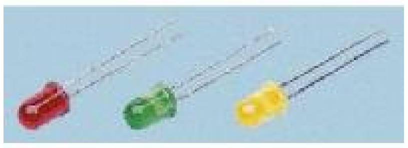
Zener Diodes: Conducts in reverse-bias direction at a specific breakdown voltage. It is used to provide reference voltage.


Zener diode used as a voltage regulator.

LED (Light Emitting Diode)

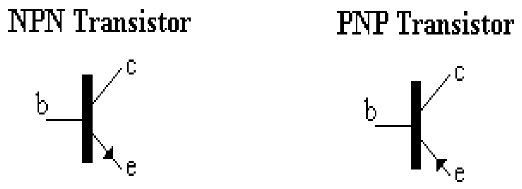
Light emitting diodes (LEDS) are special diodes that give out light when they conduct. The fact that they only conduct in one direction is often incidental to their use in a circuit. They are usually just being used as lights. They are small and cheap and they last practically forever, unlike traditional light bulbs which can burn out. The light comes from the energy given up when electrons combine with holes at the junction. The color of the light depends on the impurity in the semiconductor. It is easy to make bright red, green and yellow LEDS but technology can't make cheap LEDS of other colors like white or blue. The symbol and a few examples of this type is shown below (Note the cathode on the component is shown as a flat edge or the

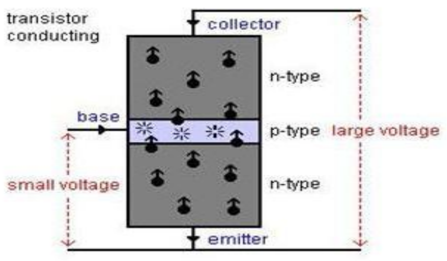




The Transistor

Transistors underpin the whole of modern day electronics. They are found in watches, calculators, microwaves, hi-fi's. A Pentium computer chip contains over a million transistors. Transistors work in two ways. They can work as switches (turning currents on and off) and as amplifiers (making currents bigger). When acting as an amplifier they operate in the linear mode and as a switch they are forced into saturation (on) or cut off (off). Transistors are sandwiches of three pieces of semiconductor material. A thin slice of n-type or p-type semiconductor is sandwiched between two layers of the opposite type. This gives two junctions rather than the one found in a diode. If the thin slice is n-type the transistor is called a p-n-p transistor, and if the thin slice is p-type it is called an n-p-n transistor. The middle layer is always called the base, and the outer two layers are called the collector and the emitter. In an n-p-n transistor (more common), electrons are the main current carriers (because n-type material predominates). When no voltage is connected to the base then the transistor is equivalent to two diodes connected back to back. Recall that current can only flow one way through a diode. A pair of back-to-back diodes can't conduct at all. If a small voltage is applied to the base (enough to remove the depletion layer in the lower junction), current flows from emitter to base like a normal diode. Once current is flowing however it is able to sweep straight through the very thin base region and into the collector, only a small part of the current flows out of the base. The transistor is now conducting through both junctions. A few of the electrons are consumed by the holes in the p-type region of the base, but most of them go straight through.



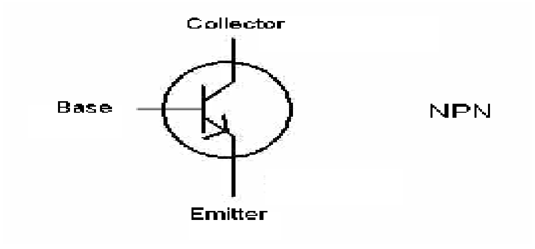


Electrons enter the emitter from the battery and come out of the collector. To see how a transistor acts as a switch, a small voltage applied to the base will switch the transistor on, allowing a current to flow in the rest of the transistor. NPN and PNP Transistor components look identical to each other the only way to tell the difference is by the component number. The symbol and a few examples of this type are shown below:

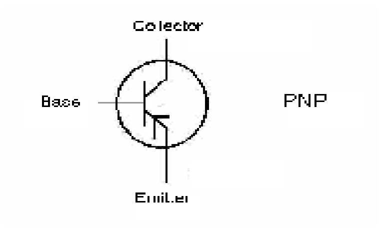
Transistor Basics: Use three layers of silicon and can be used as a switch or an amplifier. Processor chips are lots and lots of transistors in one package. Transistors have three leads - the base, collector and emitter.

Bipolar versus Field Effect Transistors: There are two main families of transistors, Bipolar and FET. FETs are more popular, waste less power (therefore run cooler), and are cheaper than bipolar. FETs can be easily damaged by static electricity, so this explains why bipolar types are used for teaching and training students. The basic operation of bipolar and FETs are the same.

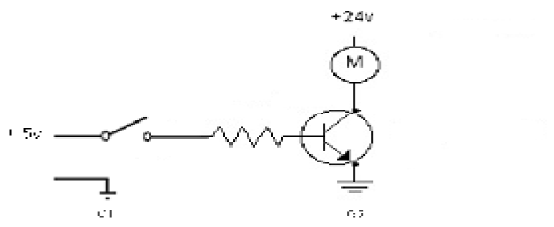
NPN versus PNP: In NPN, the base is at a higher voltage than the emitter, current flows from collector to emitter. A small amount of current also flows from base to emitter. NPN Voltage at base controls amount of current flow through transistor (collector to emitter).



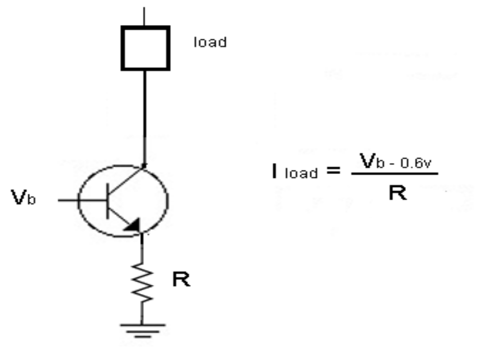
In PNP, the base is at a lower voltage than the emitter, current flows from emitter to collector. A small amount of current also flows from emitter to base. PNP. Voltage at base controls amount of current flow through transistor (emitter to collector). The arrow represents the direction of current flow.



Transistor as switch: Most sensors, processors, microcontrollers can't source enough power to make things happen in the real world. Transistors allow a large amount of current to be controlled by a small change in voltage. Grounds between control circuit and transistor must be common.



Transistor as current source: 1) Transistors have a fixed degree of amplification. (h_{FE}). When a transistor is at maximum amplification it is saturated. Saturation works when we want to pass the maximum amount of current though the transistor. By limiting the voltage at the base, we can limit the current through the circuit. Somewhat against intuition, when we limit current with a transistor, it's called amplification, because the small amount of voltage at the base is controlling the large current flow; $I_c = (V_b - 0.6v)/R$



Analog and Digital Signals

There are two different methods of sending an electronic signal from A to B. ANALOG signals are continuous, and can take any value. DIGITAL signals encode values into binary numbers. As a binary number is made up entirely from 0's and 1's, it may be transmitted in the form of electronic on/off pulses (on =1, off =0). When these pulses are received, they are processed. A digital signal is made up of discretely variable physical quantities.

Whilst these two types of signal both transmit information in electrical voltages, they each have their advantages and disadvantages. In recording audio signals, analog systems are useful, because they can give a faithful electronic representation of a complex waveform. However, because of the need for amplification of the electronic signal, 'noise' can be added along the signal path. This noise is due to unavoidable electron activity in the circuitry. Unfortunately, there is no easy way to get rid of noise from the original signal. Consequently, the noise (audible as a 'hiss') is added to the signal with each stage of transmission.

A digital equivalent to this system would sample the sound wave at selected intervals and transmit the values that correspond to the sound wave in binary code. The digital representation of the sound wave could then be moved around or processed within the system without picking up any additional noise. Although the electron (noise) activity is still taking place, whenever the digital signal is repeated, during each stage of the transmission, the noise can be omitted.

Analog Signal	Digital Signal
Accurate reproduction of signal needs extra work	Very immune from noise
Suffers from noise and distortion	Output is accurate but can have errors from the sampling process
Simple technique	Complicated but can operate at long distance

Table 1 outlines the basic characteristics of 3 modulation (encoding transmission signal) schemes: Amplitude modulation (AM), frequency modulation (FM) (both analog schemes) and digital modulation.

Table 1 **-** Comparison of AM, FM, and Digital Encoding Techniques

Parameter	**AM**	**FM**	**Digital**
Signal-to-Noise Ratio	Low-to-Moderate	Moderate-High	High
Performance vs. Attenuation	Sensitive	Tolerant	Invariant
Transmitter Cost	Moderate-High	Moderate	High
Receiver Cost	Moderate	Moderate-High	High
Receiver Gain Adjustment	Often Required	Not Required	Not Required
Installation	Adjustments Required	No Adjustments Required	No Adjustments Required
Multi-channel Capabilities	Require High Linearity Optics	Fewer Channels	Good
Performance Over Time	Moderate	Excellent	Excellent
Environmental Factors	Moderate	Excellent	Excellent

One difference between analog and digital transmission involves the bandwidth, or transmission capacity required for both schemes. Analog signals require much less bandwidth, only about 4.5 MHz with a 143.2 Mb/s data rate for the average video signal. By comparison, some digital video transmission standards require as much as 74.25 MHz with a data rate of 1485 Mb/s.

Another difference between analog and digital transmission deals with the hardware's **ability to recover the transmitted signal**. Analog modulation, which is continuously variable by nature, requires adjustment at the receiver end in order to reconstruct the transmitted signal. Digital transmission, however, because it uses only 1's and 0's to encode the signal, offers a simpler means of reconstructing the signal.

Both types of modulation can incorporate error detecting and error correcting information to the transmitted signal. However, the latest trend in signal transmission is forward error correcting (FEC). This scheme, which uses binary numbers, is suited to digital transmission. Extra bits of information are incorporated into the digital signal, allowing any transmission errors to be corrected at the receiver end.

Analog Signal Transmission

Analog transmission inserts signals of varying frequency or amplitude on carrier waves with a given frequency to produce a continuous wave. In a telephone system, an electric current or the reproduction of patterned sound waves are transmitted through a wire and into the telephone receiver. Once this is completed, they are then converted back into sound waves.

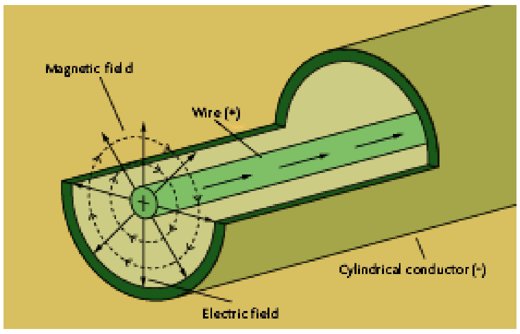
In digital transmission, the signals are converted into a binary code, which consists of two elements—positive (1) and non-positive (0). Every digit in a binary number is referred to as a bit and represents a power of two. As an example of digital transmission, in a type of digital telephone system, coded light signals travel through optical fibers and are then decoded by the receiver. When transmitting a telephone conversation, the light flashes on and off about 450 million times per second. This high rate enables two optical fibers to carry about 15,000 conversations simultaneously. Digital format is ideal for electronic communication as the string of 1s and 0s can be transmitted by a series of "on/off" signals represented by pulses of electricity or light. A pulse "on" can represent a 1, and the lack of a pulse "off" can represent a 0. Information in this form is very much easier to store electronically. Furthermore, digital transmission is usually faster and involves less noise and disturbances as compared to analog data transmission.

Analog signal transmission uses direct current (dc) variations in current or voltage to represent a data value used to communicate information. Most data acquisition signals can be described as analog, digital or pulse. While analog signals typically vary smoothly and continuously over time, digital signals are present at discrete points in time. Analog signals represent continuously variable entities such as temperatures, pressures, or flow rates. Because computer-based controllers and systems understand only discrete on/off information, conversion of analog signals to digital representations is necessary. In analog signal transmission the wiring system can effectively reduce noise interference. Analog signal transmission employs two-wire signal leads or three-wire signal leads for high precision and accuracy. The third signal lead, or shield, is grounded at the signal source to reduce noise. There are many different wiring options that are available to reduce unwanted noise pickup from entering the line. Four types of wires are fundamental in data acquisition-plain pair, shielded pair, twisted pair, and coaxial cable.

1. Plain wire is not very reliable in screening out noise and is not suggested. A shielded pair is a pair of wires surrounded by a conductor that does not carry current. The shield blocks the interfering current and directs it to the ground. When using shielded pair, it is very important to follow the rules in grounding. Again, the shield must only be grounded at one source, eliminating the possibility of ground-loop currents.

2. Twisted-pairs help in elimination of noise due to electromagnetic fields by twisting the two signal leads at regular intervals. Any induced disturbance in the wire will have the same magnitude and result in error cancellation.

3. A coaxial cable is another alternative for protecting data from noise. A coaxial cable consists of a central conducting wire separated from an outer conducting cylinder by an insulator. The central conductor is positive with respect to the outer conductor and carries a current. Coaxial cables do not produce external electric and magnetic fields and are not affected by them. This makes them ideally suited, although more expensive, for transmitting signals.



Coaxial Cable Construction

A sensor measures a variable by converting information about that variable into a dependent signal of either electrical or pneumatic nature. Cadmium sulfide resistance varies inversely and nonlinearly with light intensity and we can employ this device for light measurement. Analog signal conditioning provides the operations necessary to transform a sensor output into a form necessary to interface with other elements of the process-control loop.

We often describe the effect of the signal conditioning by the term transfer function. By this term we mean the effect of the signal conditioning on the input signal. Thus, a simple voltage amplifier has a transfer function of some constant that, when multiplied by the input voltage, gives the output voltage. **Signal conditioning can be categorized into the following types:**

1. Signal-Level Changes

The simplest method of signal conditioning is to change the level of a signal. The most common example is the necessity to either **amplify or attenuate a voltage level.** Generally, process-control applications result in slowly varying signals where dc or low-frequency response amplifiers can be employed. An important factor in the selection of an **amplifier is the input impedance that the amplifier offers to the sensor** (or any other element that serves as an input). In process control, the signals are always representative of a process variable. In accelerometers and optical detectors, the frequency response of the amplifier is very important.

2. Linearization

The process-control designer has little choice of the characteristics of a sensor output versus process variable. Often, the dependence that exists between input and output is nonlinear. Even those devices that are approximately linear may present problems when precise measurements of the variable are required. **Specialized analog circuits were devised to linearize signals.** For example, suppose a sensor output varied nonlinearly with a process variable, as shown in Figure-1a. A linearization circuit, indicated symbolically in Figure-1b, would condition the sensor output to produce voltage signal linear with the process variable, as shown in Figure -1c. The **modern approach to this problem is to provide the nonlinear signal as input to a computer and perform the linearization using software**.

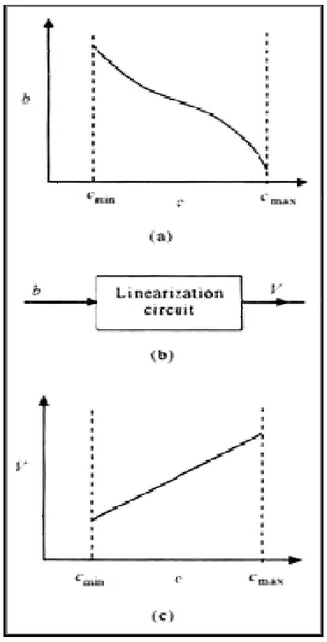


FIGURE- 1: The purpose of linearization is to provide an output that varies linearly with some variable even if the sensor output does not.

3. Conversions

Often, signal conditioning is used to convert one type of electrical variation into another. Thus, a **large class of sensors exhibit changes of resistance with changes in a dynamic variable**. In these cases, it is necessary to **convert this resistance change either to a voltage or a current signal.** This is generally accomplished by **bridges** when the fractional resistance change is small and/or by **amplifiers** whose gain varies with resistance.

An important type of conversion is associated with the process-control standard of transmitting signals as 4-20 mA current levels in wire. This gives rise to the need for converting resistance and voltage levels to an appropriate current level at the transmitting end and for converting the current back to voltage at the receiving end. Thus, voltage-to-current and current-to-voltage converters are often required.

The use of computers in process control requires conversion of analog data into a digital format using analog-to-digital converters (ADCs). Analog signal conversion is usually required to adjust the analog measurement signal to match the input requirements of the ADC.

4. Filtering and Impedance Matching

In the industrial environment, signals of considerable strength are present, especially those generated at 60-Hz line frequency. **Motor start transients also may cause pulses and other unwanted signals in the process-control loop**. In many cases, it is necessary to **use high-pass, low-pass, or notch filters to eliminate unwanted signals from the loop**. Such filtering can be accomplished by *passive* filters using only resistors, capacitors, and inductors; or active filters, using gain and feedback. **Impedance matching** is an important element of signal conditioning when transducer internal impedance or line impedance can cause errors in measurement of a dynamic variable. Both active and passive networks are employed to provide such matching.

5. Concept of Loading

Loading of one circuit by another introduces uncertainty in the amplitude of a voltage as it is passed through the measurement process. If this voltage represents some process variable, then we have uncertainty in the value of the variable.

Qualitatively, loading can be described as follows. Suppose the open circuit output of some element is a voltage, say V_x, when the element input is some variable of value x. *Open circuit* means that nothing is connected to the output. Loading occurs when we connect a load across the output and the output voltage of the element drops to some value, $V_y < V_x$. Different loads will result in different drops.

Quantitatively, loading is evaluated as follows. **Thevenin's theorem** tells us that the **output terminals of any element can be defined as a voltage source in series with output impedance** (output resistance). This is often called the Thevenin's equivalent circuit model of the element.

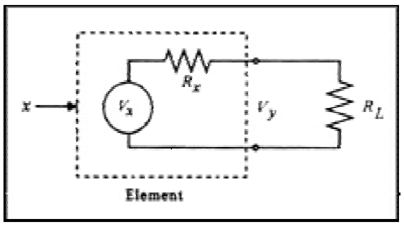


FIGURE -2 The Thevenin's equivalent circuit of a sensor allows easy visualization of how loading occurs

Figure -2 shows such an element modeled as a voltage V_x and a resistance R_x. Now suppose a load, R_L, is connected across the output of the element as shown in Figure -2. This could be the input resistance of an amplifier. A **current will flow and voltage will be dropped across R_x**. The loaded output voltage will thus be given by:

$$V_y = V_x\left(1 - \frac{R_x}{R_L + R_x}\right) \qquad \textbf{(2.1)}$$

The voltage that appears across the load is reduced by the voltage dropped across the internal resistance. This equation shows how the effects of loading can be reduced after making $R_L > R_X$.

EXAMPLE 2.1

An amplifier outputs a voltage that is ten times the voltage on its input terminals. It has an input resistance of 10 kW. A sensor outputs a voltage proportional to temperature with a transfer function of 20mV/°C. The sensor has an output resistance of 5kΩ. If the temperature is 50°C, find the amplifier output.

Solution

the unloaded output of the sensor is simply V_T = (20 mV/°C).50°C = 1.0 V. Since the amplifier has a gain of 10, the output of the amplifier appears to be $V_{out} = 10V_{in}$ = (10)1.0 V = 10 V. But this is wrong, because of loading since a voltage dropped will appear across the output resistance of the sensor. The actual amplifier input voltage will be given by Equation (2.1),

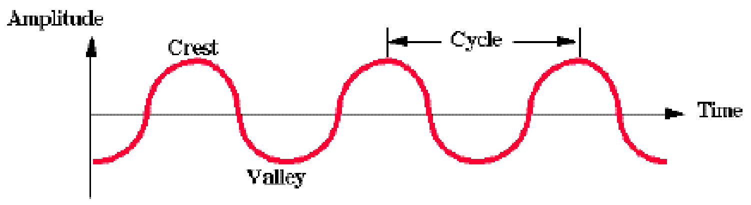
$$V_{in} = V_T\left(1 - \frac{5\text{ k}\Omega}{5\text{ k}\Omega + 10\text{ k}\Omega}\right)$$

Where V_T = 1.0 volts, so that V_{in} = 0.67 volts. Thus, the output of the amplifier is actually V_{out} = 10(0.67 V) = 6.7 V.

Analog and Digital Sound Representation:

Signals

When sound is *transmitted*, it may need to change *form*, without being *destroyed*.



Sound moves fast: in air, at 340 m/sec. Its two important characteristics are **Frequency** and **Amplitude**. Frequency is measured in Hz. Humans can hear frequencies between 20 Hz and 20,000 Hz. Amplitude is measured in decibels. Consider music:

1. Sound is pressure waves in air, caused by drums, guitar strings… etc or vocal cords
2. Converted to electrical signals by a microphone
3. Converted to magnetism when it's put on master tape and edited
4. Converted to spots on a CD when CD is manufactured
5. Converted to electricity when played by CD player
6. Converted back to sound by a speaker

A similar kind of story can be told about visual images (sequences of static images) stored on videotape or DVD and played on DVD player.

Degradation

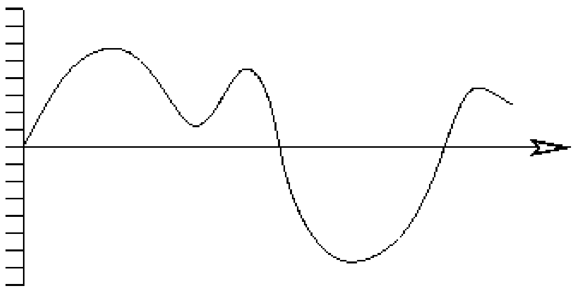
Any time signals are transmitted; there will be some degrading of quality:

1. signals may fade with time and distance
2. signals may get combined with interference from other sources (static)
3. signals may be chopped up or lost

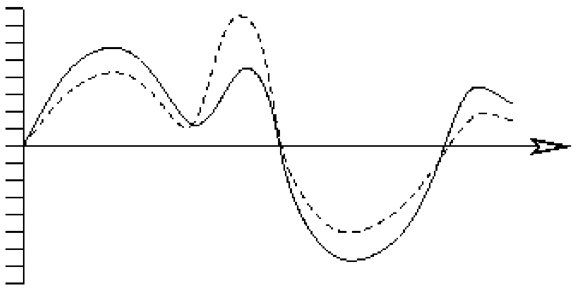
When we continue to transmit and transform signals, the effect is compounded. Think about photocopies of photocopies of photocopies...

Example

This is the transmitted signal:



And this is the received signal (dashed) compared to the transmitted signal:



The horizontal axis here is time. The vertical axis is some physical property of the signal, such as electrical voltage, pressure of a sound wave, or intensity of light.

The degradation may not be immediately obvious, but there is a general lessening of strength and there is some noise added near the second peak.

Analog Signals

The pictures above are examples of *analog* signals:

An analog signal varies some physical property, such as voltage, in proportion to the information that we are trying to transmit.

Examples of analog technology:

1. photocopiers
2. telephones
3. audio tapes
4. televisions (intensity and color info per scan line)

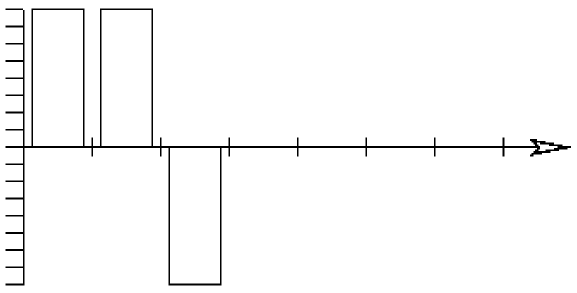
Analog signals always suffer from degradation.

Digital Signals

With a digital signal, we are using an analog signal to transmit numbers, which we convert into bits and then transmit the bits.

A digital signal uses some physical property, such as voltage, to transmit a single bit of information.

Suppose we want to transmit the number 6. In binary, that number is 110. We first decide that, say, "high" means1 and "low" means 0. Thus, 6 might look like:



The line is the signal, which rises to the maximum to indicate a 1 and falls to the minimum to indicate a 0.

Serial & Parallel

Serial and parallel transmission mechanisms are two ways of sending a digital signal from A to B.

A serial connection will send one line of data from transmitter to receiver. This method is electronically simpler, making it easier to determine what is going on, plus it's cheaper. It is, however, slower than parallel transmission.

Parallel transmission uses multiple lines of data to send more information more quickly. This method is more complex and more expensive than serial transmission. Also, it is usually only good for short distance transmissions of data.

Computers process data in binary form, or base 2. The following table gives the binary equivalent values for 0-15 (decimal):

Decimal	Binary				Decimal	Binary			
	2^3	2^2	2^1	2^0		2^3	2^2	2^1	2^0
0	0	0	0	0	8	1	0	0	0
1	0	0	0	1	9	1	0	0	1
2	0	0	1	0	10	1	0	1	0
3	0	0	1	1	11	1	0	1	1
4	0	1	0	0	12	1	1	0	0
5	0	1	0	1	13	1	1	0	1
6	0	1	1	0	14	1	1	1	0
7	0	1	1	1	15	1	1	1	0

In a 4 line parallel connection, the decimal number 13 is communicated by sending one/on pulse along line 1 (2^3), one/on pulse along line 2 (2^2), zero/off pulse along line 3 (2^1)and one/on pulse along line 4 (2^0). Computer memory is defined in terms of Bits and Bytes.

1 Bit = one on/off space in memory (0 or 1).

1 Byte = 8 bits, and can therefore hold any decimal value from 0 (00000000) to 255 (11111111).

1 Kilobyte ≈ 1000 bytes. In fact, 1KB = 1024 bytes = 2^{10} bytes.

1Megabyte = 1024 kilobytes, 1 Gigabyte = 1024 megabytes.

One way of converting a decimal (base 10) number to a binary (base 2) number is by divide the decimal number by 2:

```
 600 / 2 = 300 rem. 0          <-- LSB (least significant bit)
 300 / 2 = 150 rem. 0
 150 / 2 = 75 rem. 0
   75 / 2 = 37 rem. 1
   37 / 2 = 18 rem. 1
   18 / 2 =   9 rem. 0
     9 / 2 =   4 rem. 1
     4 / 2 =   2 rem. 0
     2 / 2 =   1 rem. 0
     1 / 2 =   0 rem. 1<-- MSB (most significant bit)
                    1001011000
```

All of the remainders from the division are then arranged in reverse order, from MSB to LSB to form the correct binary sequence. 1001011000 in binary = 600 in decimal.

Computers perform repetitive, complex calculations; they cannot actually subtract binary numbers. To get around this problem, assume that, in sequence of bits, the first bit signifies whether the number is positive (0) or negative (1). The decimal number '+78' = 0100 1110, (where the first 0 bit shows that the number is positive), that is said to be the number's normal form. To find the number's 1's compliment, bits 2-8 are inverted, giving 0011 0001. To find the number's 2's compliment, 1 is added to this 1's compliment value, here giving 0011 0010. So, if the number was minus 78, its 2's compliment would be 1011 0010, where the first bit 1 defines the number as being negative. To subtract 20 from 45, we would have to think of that equation as being (+45) + (-20). By changing -20 from its natural from to its 2's compliment, we go from 1001 0100 to 1110 1011 to 1110 1100, using the process described above. When the binary equivalent of +45 (0010 1101) is added to the 2's compliment of -25, the following result is arrived at:

```
              45 in normal form = 0010 1101
-20 in 2's compliment form = 1110 1100
                                        + ___________
                                            (1)0001 1001 = +25
```

By deleting the extra new bit at the start of the sequence, 0001 1001, or +25 is correctly found to be the result.

LOGIC GATES AND FUNCTIONS

Basic Logic Gates

While each logical element or condition must always have a logic value of either "0" or "1", we also need to have ways to combine different logical signals or conditions to provide a logical result. For example, consider the logical statement:

"If we move the switch on the wall up, the light will turn on."

At first glance, this seems to be a correct statement. However, if we look at a few other factors, we realize that there's more to it than this. In this example, a more complete statement would be:

"If we move the switch on the wall up *and* the light bulb is good *and* the power is on, the light will turn on." If we look at these two statements as logical expressions and use logical terminology, we can reduce the first statement to:

Light = Switch

This means nothing more than that the light will follow the action of the switch, so that when the switch is up/on/true/1 the light will also be on/true/1. Conversely, if the switch is down/off/false/0 the light will also be off/false/0. Looking at the second version of the statement, we have a slightly more complex expression:

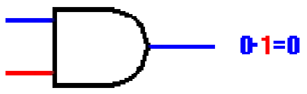
Light = Switch *and* Bulb *and* Power

Normally, we use symbols rather than words to designate the AND function that we're using to combine the separate variables of Switch, Bulb, and Power in this expression. The symbol normally used is a dot, which is the same symbol used for multiplication in some mathematical expressions. Using this symbol, our three-variable expression becomes:

Light = Switch •Bulb •Power

When we deal with logical circuits (as in computers), we not only need to deal with logical functions; we also need some special symbols to denote these functions in a logical diagram. There are three fundamental logical operations, from which all other functions, no matter how complex, can be derived. These functions are named ***and, or,* and *not***. Each of these has a specific symbol and a clearly-defined behavior, as follows:

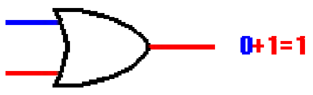
The AND Gate



The AND gate implements the AND function. With the gate shown above, both inputs must have logic 1 signals applied to them in order for the output to be logic 1. With either input at logic 0, the output will be held to logic 0. There is no limit to the number of inputs that may be applied to an AND function, however, for practical reasons, commercial AND gates are manufactured with 2, 3, or 4 inputs. A standard Integrated Circuit (IC) package contains 14 or 16 pins, for practical size and handling. A standard 14-pin package can contain four 2-input gates, three 3-input gates, or two 4-input gates, and still have room for two pins for power supply connections. The truth table for a two-input AND gate looks like

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

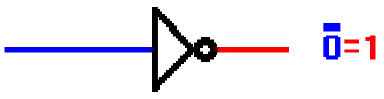
The OR Gate



The OR gate is sort of the reverse of the AND gate. The OR function, like its verbal counterpart, allows the output to be true (logic 1) if any one or more of its inputs are true. Verbally, we might say, "If it is raining OR if I turn on the sprinkler, the grass will be wet." Note that the grass will still be wet if the sprinkler is on and it is also raining. This is correctly reflected by the basic OR function. In symbols, the OR function is designated with a plus sign (+). In logical diagrams, the symbol above designates the OR gate. As with the AND function, the OR function can have any number of inputs, however, practical commercial OR gates are limited to 2, 3, and 4 inputs, as with AND gates. The truth table for a two-input OR gate looks like

A	**B**	**A+B**
0	0	0
0	1	1
1	0	1
1	1	1

The NOT Gate, or Inverter



The inverter is a little different from AND and OR gates in that it always has exactly one input as well as one output. Whatever logical state is applied to the input, the opposite state will appear at the output. The NOT function is necessary in many applications and highly useful in others. A practical verbal application might be: **The door is NOT locked = you may enter**

The NOT function is denoted by a horizontal bar over the value to be inverted, as shown in the figure below. In the inverter symbol, the triangle actually denotes only an amplifier, which in digital terms means that it "cleans up" the signal but does not change its logical sense. It is the circle at the output which denotes the logical inversion. The circle could have been placed at the input instead, and the logical meaning would still be the same. The truth table for the NOT gate is shown below

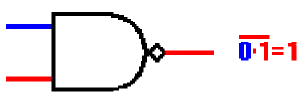
A	Ā
0	1
1	0

The logic gates shown above are used in various combinations to perform tasks of any level of complexity.

Derived Logic Functions and Gates

Some combinations of basic functions have been given names and logic symbols of their own. The first is called NAND, and consists of an AND function followed by a NOT function. The second, as you might expect, is called NOR. This is an OR function followed by NOT. The third is a variation of the OR function, called the Exclusive-OR, or XOR function. Each of these derived functions has a specific logic symbol and behavior, which we can summarize as follows:

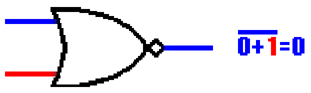
The NAND Gate



The NAND gate implements the NAND function, which is exactly inverted from the AND function. Both inputs must have logic 1 signals applied to them in order for the output to be logic 0. With either input at logic 0, the output will be held to logic 1. The circle at the output of the NAND gate denotes the logical inversion, just as it did at the output of the inverter. The over-bar over both inputs shows that the AND function itself that is inverted, rather than each separate input. There is no limit to the number of inputs that may be applied to a NAND function, however, for practical reasons, commercial NAND gates are manufactured with 2, 3, or 4 inputs, to fit in a 14-pin or 16-pin package. The truth table for a two-input NAND gate looks like (— over A.B)

A	**B**	**A.B**
0	0	1
0	1	1
1	0	1
1	1	0

The NOR Gate

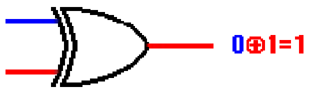


The NOR gate is an OR gate with the output inverted. Where the OR gate allows the output to be true (logic 1) if any one or more of its inputs are true, the NOR gate inverts this and forces the output to logic 0 when any input is true.

In symbols, the NOR function is designated with a plus sign (+), with an over-bar over the entire expression to indicate the inversion. This is an OR gate with a circle to designate the inversion. The NOR function can have any number of inputs, but practical commercial NOR gates are mostly limited to 2, 3, and 4 inputs, as with other gates in this class, to fit in standard IC packages. The truth table for a two-input NOR gate looks like (— over A+B):

A	B	A+B
0	0	1
0	1	0
1	0	0
1	1	0

The Exclusive-OR, or XOR Gate

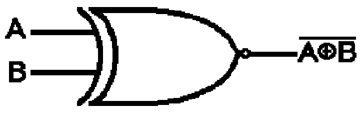


The Exclusive-OR, can be stated as, "Either A or B, but not both." The XOR gate produces logic 1 output only if its two inputs are *different*. If the inputs are the same, the output is logic 0. The XOR symbol is a variation on the standard OR symbol. It consists of a plus (+) sign with a circle around it. The logic symbol, as shown here, is a variation on the standard OR symbol. Unlike standard OR/NOR and AND/NAND functions, the XOR function always has exactly two inputs, and commercially manufactured XOR gates are the same. Four XOR gates fit in a standard 14-pin IC package. The truth table for a two-input XOR gate looks like: (+ inside the circle of AOB)

A	B	AOB
0	0	0
0	1	1
1	0	1
1	1	0

The eXclusive-NOR (XNOR) gate

The exclusive-NOR or XNOR gate has two or more inputs. The output is equivalent to inverting the output from the exclusive-OR gate described above. Therefore an equivalent circuit would comprise an XOR gate, the output of which feeds into the input of a NOT gate. In general, an XNOR gate gives an output value of 1 when there are an **even** number of 1's on the inputs to the gate. The truth table for a 3-input XNOR gate below illustrates this point. The XNOR gate is drawn using the same symbol as the XOR gate with an invert circle on the output line.



The output from the XNOR gate is written as AOB which reads "A XNOR B".

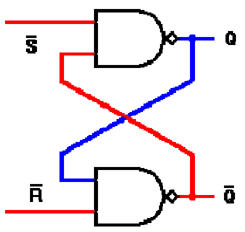
The truth table for a two-input XNOR gate looks like

A	B	AOB
0	0	1
0	1	0
1	0	0
1	1	1

SEQUENTIAL LOGIC

The Basic RS NAND Latch:

In order **for a logical circuit to "remember**" and retain its logical state even after the controlling input signal(s) have been removed, it is necessary for the circuit to include some form of **feedback**. We can use NAND or NOR gates, and using the extra input lines to control the circuit. The circuit shown below is a basic NAND latch with designated "S" and "R" for "Set" and "Reset" respectively. The **outputs** of any single-bit latch or memory are designated Q and Q'.

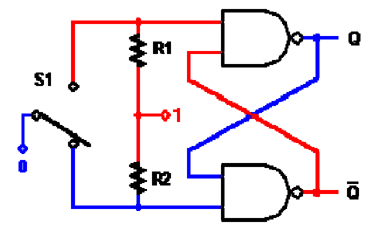


For the NAND latch circuit, both inputs should normally be at a logic1 level 1. Changing an input to a logic 0 level will force that output to logic 1. The same logic 1 will also be applied to the second input of the other NAND gate, allowing that output to fall to a logic 0 level. This in turn feeds back to the second input of the original gate, forcing its output to remain at logic1.

Applying another logic 0 input to the same gate will have no further effect on this circuit. However, applying logic 0 to the *other* gate will cause the same reaction in the other direction, thus changing the state of the latch circuit the other way.

It is forbidden to have both inputs at a logic 0 level at the same time. That state will force both outputs to a logic1, overriding the feedback latching action. In this condition, whichever input goes to logic 1 first will lose control, while the other input (still at logic 0) controls the resulting state of the latch. If both inputs go to logic 1 simultaneously, the result is a "race" condition, and the final state of the latch cannot be determined ahead of time.

In this circuit, any mechanical switch added will experience a phenomenon called "contact bounce." Whenever you press the button or change the switch position, the physical contacts will flex a little, causing them to make and break several times before settling down. This is not noticed when turning on a light at home, but digital circuits are fast enough that they **do** notice this behavior. The solution is to use a pushbutton or switch, as shown in the figure below:

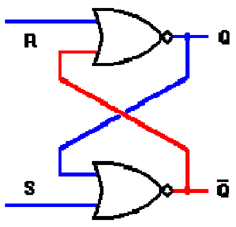


The unconnected input is held at a logic1 through its resistor, while the connected input is held at logic 0 by the direct connection through the switch.

When the button of the switch is pressed, the very first contact will cause the latch to change state, but additional bounces will have no further effect. This eliminates the contact bounce and sends a single, clean digital transition to the next circuit. **One problem** with the basic RS NAND latch is that the **input levels need to be inverted, sitting idle at logic 1**, in order for the circuit to work. It would be helpful if we had normal inputs which would idle at logic 0, and go to logic 1 only to control the latch. This can be done by placing inverters at the inputs. **Another problem** is how to control *when* the latch is allowed to change state, and when it is not. This is necessary if a group of latches and want to be sure they all change state (or not) at the same time. Both of these concerns can be easily addressed when using a Basic RS-NOR Latch.

The Basic RS-NOR Latch:

The circuit shown below is a basic NOR latch. The inputs are generally designated "S" and "R" for "Set" and "Reset" respectively. **Because the NOR inputs must normally be logic 0 to avoid overriding the latching action, the inputs are not inverted in this circuit**. For the NOR latch circuit, **both inputs should normally be at**

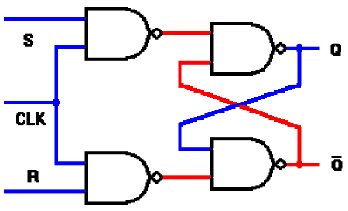


a logic 0 level. Changing an input to logic 1 level will force that output to logic 0. The same logic 0 will also be applied to the second input of the other NOR gate, allowing that output to raise to logic 1 level. This in turn feeds back to the second input of the original gate, forcing its output to remain at logic 0 even after removing external input.

Applying another logic1 input to the same gate will have no further effect on this circuit. However, applying logic 1 to the *other* gate will cause the same reaction in the other direction, thus changing the state of the latch circuit the other way. It is **forbidden to have both inputs at a logic1 level at the same time**. That state will force both outputs to logic 0, overriding the feedback latching action. In this condition, whichever input goes to logic 0 first will lose control, while the other input (still at logic 1) controls the resulting state of the latch. If both inputs go to logic 0 simultaneously, the result is a "race" condition, and the final state of the latch cannot be determined ahead of time. **One problem with the basic RS NOR latch is that the input signals actively drive their respective outputs to a logic 0, rather than to a logic 1.** Thus, the S input signal is applied to the gate that produces the Q' output, while the R input signal is applied to the gate that produces the Q output. The circuit works fine, but the input reversal is confusing.

The Clocked RS-NAND Latch:

By **adding a pair of NAND gates to the input circuits of the RS latch**, we accomplish two goals: normal rather than inverted inputs and a third input common to both gates which we can use to synchronize this circuit with others of its kind. The clocked RS NAND latch is shown below.



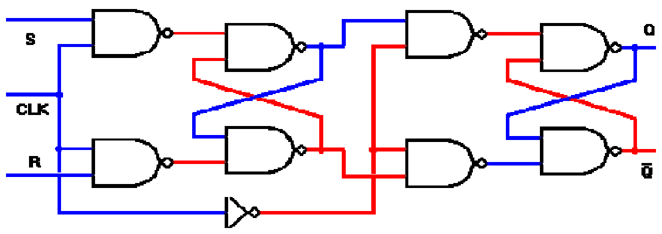
The clocked RS latch circuit is very similar in operation to the basic RS-NOR latch. The S and R inputs are normally at logic 0, and must be changed to logic 1 to change the state of the latch. However, with the third input, a new factor has been added. This input is typically designated *C* or *CLK*, because it is typically controlled by a clock circuit of some sort, which is used to synchronize several of these latch circuits with each other. The output can only change state while the CLK input is logic 1. When CLK is logic 0, the S and R inputs will have no effect.

The same rule about not activating both the S and R inputs simultaneously holds true: if both are logic 1 when the clock is also logic 1, the latching action is bypassed and both outputs will go to logic 1. The difference in this case is that if the CLK input drops to logic 0 first, there is no question or doubt -- a true race condition will exist, and we cannot tell which way the outputs will come to rest.

For correct operation, the selected R or S input should be brought to logic 1, and then the CLK input should be made logic 1 and then logic 0 again. Finally, the selected input should be returned to logic 0. The clocked RS latch solves some of the problems of basic RS latch circuit, and allows closer control of the latching action but does not offer a complete solution. A major problem remaining is that this latch circuit could easily experience a change in S and R input levels while the CLK input is still at logic 1 level. This allows the circuit to change state many times before the CLK input returns to logic 0. One way to solve this problem is by using The Edge-Triggered RS Flip-flop to make sure that the latch can only change its outputs at one instant of the clock cycle

The Edge-Triggered RS Flip-flop:

To adjust the clocked RS latch for edge triggering, we must actually combine two identical clocked latch circuits, but have them operate on opposite halves of the clock signal. The resulting circuit is commonly called a *flip-flop*, because its output can first flip one way and then flop back the other way. The clocked RS latch is also sometimes called a flip-flop, although it is more properly referred to as a latch circuit. The two-section flip-flop is also known as a *master-slave* flip-flop, because the input latch operates as the master section, while the output section is slaved to the master during half of each clock cycle. The edge-triggered RS NAND flip-flop is shown below.



The edge-triggered RS flip-flop consists of two identical RS latch circuits, as shown above. The inverter connected between the two CLK inputs ensures that the two sections will be enabled during opposite half-cycles of the clock signal. This is the key to the operation of this circuit.

If we start with the CLK input at logic 0, the S and R inputs are disconnected from the input (master) latch. Therefore, any changes in the input signals cannot affect the state of the final outputs. When the CLK signal goes to logic 1, the S and R inputs are able to control the state of the input latch, just as with the single RS latch circuit. At the same time, the inverted CLK signal applied to the output (slave) latch prevents the state of the input latch from having any effect here. Therefore, **any changes in the R and S input signals are tracked by the input latch while CLK is at logic 1, but are not reflected at the Q and Q' outputs.**

When CLK falls again to logic 0, the S and R inputs are again isolated from the input latch. At the same time, the inverted CLK signal now allows the current state of the input latch to reach the output latch. Therefore, the Q and Q' outputs can only change state when the CLK signal falls from a logic 1 to logic 0. This is known as the *falling edge* of the CLK signal; hence the designation *edge-triggered* flip-flop. There is still one problem left to solve: the possible race condition which may occur if both the S and

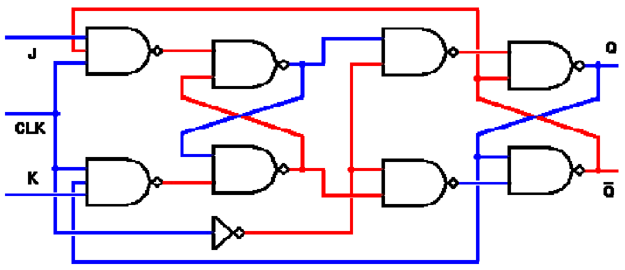
R inputs are at logic 1 when CLK falls from logic 1 to logic 0. The solution is to add some additional feedback from the slave latch to the master latch. The resulting circuit is called a *JK flip-flop*.

The JK Flip-flop:

To prevent any possibility of a "race" condition occurring when both the S and R inputs are at logic 1 when the CLK input falls from logic 1 to logic 0, we must prevent one of those inputs from having an effect on the master latch in the circuit. At the same time, we still want the flip-flop to be able to change state on each falling edge of the CLK input, if the input logic signals call for this. Therefore, the S or R input to be disabled depends on the current state of the slave latch outputs.

If the Q output is logic 1 (the flip-flop is in the "Set" state), the S input can't make it any more set than it already is. Therefore, we can disable the S input without disabling the flip-flop under these conditions. In the same way, if the Q output is logic 0 (the flip-flop is reset), the R input can be disabled without causing any harm. If we can accomplish this, then we will solve the problem of the "race" condition.

The circuit below shows the solution. To the RS flip-flop we have added two new connections from the Q and Q' outputs back to the original input gates. We change the designations of the logic inputs and of the flip-flop itself to J (instead of S) and K (instead of R). The entire circuit is known as a *JK flip-flop*.

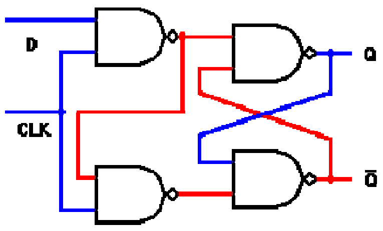


In most ways, the JK flip-flop behaves just like the RS flip-flop. The Q and Q' outputs will only change state on the falling edge of the CLK signal, and the J and K inputs will control the future output state pretty much as before. However, there are some important differences. Since one of the two logic inputs is always disabled according to the output state of the overall flip-flop, the master latch cannot change state back and forth while the CLK input is at logic 1. Instead, the enabled input can change the state of the master latch *once*, after which this latch will not change again. This was not true of the RS flip-flop.

If both the J and K inputs are held at logic 1 and the CLK signal continues to change, the Q and Q' outputs will simply change state with each falling edge of the CLK signal. (The master latch circuit will change state with each *rising* edge of CLK.). Because the behavior of the JK flip-flop is completely predictable under all conditions, this is the preferred type of flip-flop for most logic circuit designs. The RS flip-flop is only used in applications where it can be guaranteed that both R and S cannot be logic 1 at the same time.

The D Latch (Data Latch):

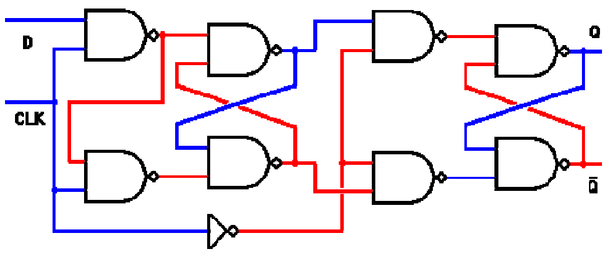
One very useful variation on the RS latch circuit is the Data latch which is constructed by using the inverted S input as the R input signal. The single remaining input is designated "D" to distinguish its operation from other types of latches. It makes no difference that the R input signal is effectively clocked twice, since the CLK signal will either allow the signals to pass both gates or it will not.



In the D latch, when the CLK input is logic 1, the Q output will always reflect the logic level present at the D input, no matter how that changes. When the CLK input falls to logic 0, the last state of the D input is trapped and held in the latch, for use by whatever other circuits may need this signal. Because the single D input is also inverted to provide the signal to reset the latch, this latch circuit cannot experience a "race" condition caused by all inputs being at logic 1 simultaneously. Therefore the D latch can be safely used in any circuit.

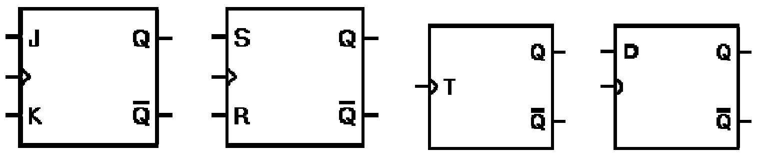
The D Flip-flop:

Edge-triggered D flip-flop is derived from its RS *ff* after replacing R input with an inverted version of S input, which thereby becomes D. This is needed in the master latch section; the slave remains unchanged. In D flip-flop, when the clock input falls to logic 0 and the outputs can change state, the Q output always takes on the state of the D input at the moment of the clock edge. This was not true of the RS and JK flip-flops. The RS master section would repeatedly change states to match the input signals while the clock line is logic 1, and the Q output would reflect whichever input most recently received an active signal. The JK master section would receive and hold an input to tell it to change state, and never change that state until the next cycle of the clock.



Flip-flop Symbols:

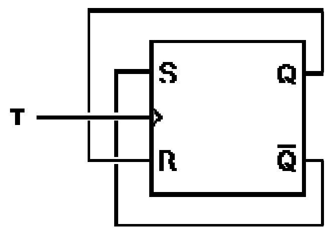
Placing all logic symbols in a diagram involving multiple flip-flops would rapidly generate so much clutter that the overall purpose of the diagram would be lost. To avoid this problem, we use the "black-box" approach. This is actually just one step further that the "black-box" approach we used in specifying logic gate symbols to represent specific clusters of electronic components — now we are using one symbol to represent a cluster of logic gates connected to perform a specific function. Some typical flip-flop symbols are shown below:



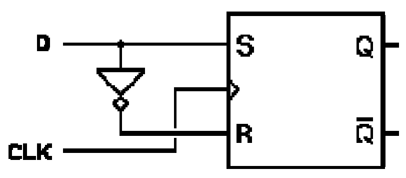
The symbols above are nearly identical — only the inputs vary. In each symbol the clock input is marked by small angle, rather than by letters CLK. That little angle marker actually provides two pieces of information, rather than one. First, of course, it marks the clocking input. Second, it specifies that these are edge-triggered flip-flops. The D latch shown uses a rounded marker for the clock input. This signifies that the circuit is controlled by the clock level, not the clock edge. If we change that rounded input to a sharp angle, it would indicate an edge-triggered master-slave D flip-flop.

Converting Flip-flop Input Types:

Sometimes it just happens that you need a particular type of flip-flop for a specific application, but all you have available is another type. This often happens with an application needing T flip-flops, since these are not generally available in commercial packages. Rather, it is necessary to re-wire an available type to perform as a T device. Converting an RS flip-flop involves simple feedback connections to ensure that the S and R inputs will always tell the flip-flop to change state at each clock pulse. Converting a D flip-flop to T operation is quite similar; the Q' output is connected back to the D input.



To convert an RS flip-flop to D operation, we need to add an inverter to supply the R input signal, as shown below:



Binary System

Unlike analog circuit which contains signals that are constantly changing from one value to another, such as amplitude or frequency, digital circuits process signals that contain just two voltage levels or states, labeled **logic "0"** and **logic "1"**. These discrete voltage levels are commonly known as **BI**nary digi**TS** and are normally referred to as **BITS**. Because there are only two valid Boolean values for representing either logic "1" or logic "0", the **Binary Numbering** system is ideal for use in digital or electronic circuits and systems. The Binary Numbers system is a Base-2 system which follows the same rules in mathematics as the common decimal system meaning instead of powers of ten, for example 1, 10, 100, 1000 etc, binary uses powers of two, 1, 2, 4, 8, 16, 32 etc.

Basic Concepts

To understand binary numbers, begin by recalling elementary mathematics. In the decimal system, things are organized into columns:

H | T | O

9 | 2 | 4

Such that "H" is the hundreds column, "T" is the tens column, and "O" is the ones column. So the number "934" is 9-hundreds plus 2-tens plus 4-ones.

The ones column meant 10^0, the tens column meant 10^1, the hundreds column 10^2 and so on, such that

$10^2|10^1|10^0$

9 | 2 | 4

The number 924 is really $\mathbf{\{(9*10^2)+(2*10^1)+(4*10^0)\}}$. The decimal system uses the digits 0-9 to represent numbers. If we wanted to put a larger number in column 10^n (e.g., 10), we would have to multiply $10*10^n$, which would give $10^{(n+1)}$, and be carried a column to the left. For example, twelve would be $\mathbf{12*10^0}$**, or** $\mathbf{10^0(10+2)}$**,** or $\mathbf{10^1+2*10^0}$. The binary system works under the exact same principles as the decimal system, only it operates in base 2 rather than base 10. In other words, instead of columns being $\mathbf{10^2|10^1|10^0}$**, they are** $\mathbf{2^2\ |2^1|2^0}$. Instead of using the digits 0-9, we only use 0-1 (again, if we used anything larger it would be like multiplying $2*2^n$ and getting 2^{n+1}, which would not fit in the 2^n column. Therefore, it would shift you one column to the left. For example, "3" in binary cannot be put into one column. The first column we fill is the right-most column, which is 2^0, or 1. Since 3>1, we need to use an extra column to the left, and indicate it as "11" in binary $(1*2^1) + (1*2^0)$.

Example: What would the binary number 10, 111, 10101 and 11110 be in decimal notation? Remember:

2^4	2^3	2^2	2^1	2^0
			1	0
		1	1	1
1	0	1	0	1
1	1	1	1	0

Example:

$$\begin{aligned} 110100 &= 1x2^5 + 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 0x2^0 \\ &= 32 + 16 + 0 + 4 + 0 + 0 \\ &= 52 \end{aligned}$$

The same approach applies to non-integral numbers so, for example

$$\begin{aligned} 110.101 &= 1x2^2 + 1x2^1 + 0x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} \\ &= 4 + 2 + 0 + 0.5 + 0 + 0.125 \\ &= 6.625 \end{aligned}$$

Binary Addition

Binary addition is completely straightforward and is done in the same way as standard decimal addition remembering that, in binary terms **"one plus one equals zero carry one"**. This is also true for fractional binary numbers as illustrated below.

Binary	**Decimal**	**Binary**	**Decimal**	**Binary**	**Decimal**
101	5	1001.1	9.5	110.1101	6.8125
+110	+6	1100.1	+12.5	+100.1010	+4.6250
____	__	_______	____	_________	_______
1011	11	10110.0	22.0	1011.0111	11.4375

Consider the addition of decimal numbers:

```
  23
 +48
 ‾‾‾
```

We begin by adding 3+8=11. Since 11 is greater than 10, a one is put into the 10's column (carried), and a 1 is recorded in the one's column of the sum. Next, add {(2+4) +1} (the one is from the carry)=7, which is put in the 10's column of the sum. Thus, the answer is 71. Binary addition works on the same principle, but the numerals are different. Begin with one-bit binary addition:

```
  0   0   1
 +0  +1  +0
___ ___ ___
  0   1   1
```

1+1 carries us into the next column. In decimal form, 1+1=2 while in binary, any digit higher than 1 puts us a column to the left (as would 10 in decimal notations). The decimal number "2" is written in binary notation as "10" $(1*2^1) + (0*2^0)$. Record the 0 in the ones column, and carry the 1 to the twos column to get an answer of "10." In our vertical notation,

```
  1
 +1
 ‾‾
 10
```

The process is the same for multiple-bit binary numbers:

```
  1010
 +1111
 ‾‾‾‾‾
```

Step one:

Column 2^0: 0+1=1.

Record the 1.

Temporary Result: 1; Carry: 0

Step two:

Column 2^1: 1+1=10.

Record the 0, carry the 1.

Temporary Result: 01; Carry: 1

Step three:

Column 2^2: 1+0=1 Add 1 from carry: 1+1=10.

Record the 0, carry the 1.

Temporary Result: 001; Carry: 1

Step four:

Column 2^3: 1+1=10. Add 1 from carry: 10+1=11.

Record the 11.

Final result: 11001

Alternately:

```
   11   (carry)
   1010
  +1111
 ______
  11001
```

Always remember

0+0=0

1+0=1

1+1=10

Try the following examples of binary addition:

```
     111      101     111
    +110     +111    +111
   ______   _____   _____
```

Rules of Binary Addition

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0, and carry 1 to the next more significant bit

For example,

00011010 + 00001100 = 00100110	*1 1*		carries
	0 0 0 1 1 0 1 0	=	$26_{(\text{base } 10)}$
	+ 0 0 0 0 1 1 0 0	=	$12_{(\text{base } 10)}$
	0 0 1 0 0 1 1 0	=	$38_{(\text{base } 10)}$

00010011 + 00111110 = 01010001	*1 1 1 1 1*		carries
	0 0 0 1 0 0 1 1	=	$19_{(\text{base } 10)}$
	+ 0 0 1 1 1 1 1 0	=	$62_{(\text{base } 10)}$
	0 1 0 1 0 0 0 1	=	$81_{(\text{base } 10)}$

Note: Rules of binary addition (without carries) are same as truths of the XOR gate.

Binary Subtraction

Binary subtraction usually takes place by *complementing* i.e. subtraction is via the addition of negative numbers. This technique requires the use of the so-called **ones (1's) complement** and **twos (2's) complement** of a binary number.

The 1's complement of a binary number is formed simply by complimenting each digit in turn. The 2's complement of a binary number is formed by adding 1 to the least significant bit of the 1's complement (**Note** in the case of fractional binary numbers this is not the same as adding 1 to the 1's complement number - see below).

Decimal	**Binary**	**1's Complement**	**2's Complement**
5	00000101	11111010	11111011
27	00011011	11100100	11100101
76	01001100	10110011	10110100
4.625	0100.1010	1011.0101	1011.0110

Note that in order to correctly express the 1's complement and 2's complement binary numbers a fixed length format must be chosen (8-bit in the case above) and leading zeroes <u>must</u> be included when writing the original pure binary format number.

In order to perform binary subtraction the rules are as follows:

- When the sum to be performed is A-B then the number to be subtracted (B) is converted to its 2's complement form and then added to (A) using standard binary addition.
- If, after the addition, the sign bit = 1 then a further 2 steps must be performed:
    - first take the 2's complement of the result;
    - then make the sign bit of the new number equal to 1;
    - interpret the result in true magnitude format.
- For sums of the form -A-B then take the 2's complement of A, add it to the 2's complement of B and then proceed as above;
- Sums of the form -A-(-B) can be converted to B-A before proceeding as above.

It is simple as long as we remember how subtraction and the base 2 number system. Let's first look at an easy example.

```
 111
- 10
 101
```

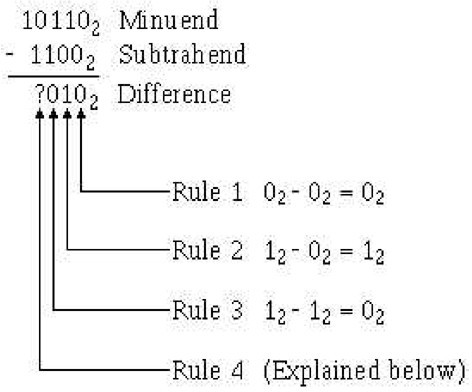
Note that the difference is the same if this was decimal subtraction. Also similar to decimal subtraction is the concept of "borrowing". Watch as "borrowing" occurs when a larger digit, say 8, is subtracted from a smaller digit, say 5, as shown below in decimal subtraction.

```
  35
-  8
  27
```

For 10 minus 1, 1 is borrowed from the "tens" column for use in the "ones" column, leaving the "tens" column with only 2. The following examples show "borrowing" in binary subtraction.

10	100	1010
- 1	- 10	- 110
1	10	100

The example (10110_2 - 1100_2) demonstrates the four rules of binary subtraction:



Rules of Binary Subtraction

0 - 0 = 0

0 - 1 = 1, and borrow 1 from the next more significant bit

1 - 0 = 1

1 - 1 = 0

Example

00100101 - 00010001 = 00010100

0 *Borrows*

0 0 ~~1~~ ^{1}0 0 1 0 1 = $37_{(\text{base }10)}$

- 0 0 0 1 0 0 0 1 = $17_{(\text{base }10)}$

0 0 0 1 0 1 0 0 = $20_{(\text{base }10)}$

00110011 - 00010110 = 00011101

0 ^{1}0 1 *Borrows*

0 0 ~~1~~ ~~1~~ ~~0~~ ^{1}0 1 1 = $51_{(\text{base }10)}$

- 0 0 0 1 0 1 1 0 = $22_{(\text{base }10)}$

0 0 0 1 1 1 0 1 = $29_{(\text{base }10)}$

Binary Multiplication and Binary Division

Binary multiplication and binary division are both most easily done by long multiplication and division methods as often taught for standard decimal numbers. In both cases all numbers must be in true magnitude format but with sign bits removed. For multiplication each partial product is calculated and then all partial products are summed using standard binary addition. For division it proceeds like decimal division. Finally the sign of the product or quotient is determined by summing all sign bits and retaining the LSB only of the resultant sum.

Binary Multiplication works the same way as in the decimal system:

- 1*1=1
- 1*0=0
- 0*1=0

```
  101
 * 11
 ____
  101
 1010
 _____
 1111
```

Multiplying by two is extremely easy. To multiply by two, just add a 0 on the end.

Rules of Binary Multiplication

0 x 0 = 0

0 x 1 = 0

1 x 0 = 0

1 x 1 = 1, and no carry or borrow bits

Example,

00101001 × 00000110 = 11110110

$$\begin{array}{rcl}
0\ 0\ 1\ 0\ 1\ 0\ 0\ 1 & = & 41_{(\text{base }10)} \\
\times\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0 & = & 6_{(\text{base }10)} \\
\hline
0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & & \\
0\ 0\ 1\ 0\ 1\ 0\ 0\ 1 & & \\
0\ 0\ 1\ 0\ 1\ 0\ 0\ 1 & & \\
\hline
0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0 & = & 246_{(\text{base }10)}
\end{array}$$

00010111 × 00000011 = 01000101

$$\begin{array}{rcl}
0\ 0\ 0\ 1\ 0\ 1\ 1\ 1 & = & 23_{(\text{base }10)} \\
\times\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1 & = & 3_{(\text{base }10)} \\
\hline
\mathit{1\ 1\ 1\ 1\ 1} & & \textit{carries} \\
0\ 0\ 0\ 1\ 0\ 1\ 1\ 1 & & \\
0\ 0\ 0\ 1\ 0\ 1\ 1\ 1 & & \\
\hline
0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1 & = & 69_{(\text{base }10)}
\end{array}$$

Binary Division Follows the same rules as in decimal division. For the sake of simplicity, throw away the remainder. For Example: 111011/11 gives 10011 r 10.

```
10011 r 10
  _______
 11)111011
   -11
    ______
     101
     -11
    ______
      101
       11
    ______
       10
```

Binary division is the repeated process of subtraction, just as in decimal division.

Example

00101010 ÷ 00000110 = 00000111

$$\begin{array}{r|cccccccc|cl}
 & & & & & & 1 & 1 & 1 & = & 7_{(\text{base }10)} \\
\hline
1\,1\,0\,) & 0 & 0 & \cancel{1} & {}^{1}0 & 1 & 0 & 1 & 0 & = & 42_{(\text{base }10)} \\
 & & & - & 1 & 1 & 0 & & & = & 6_{(\text{base }10)} \\
\hline
 & & & & & \mathit{1} & & & & & \textit{borrows} \\
 & & & & \cancel{1} & \cancel{0} & {}^{1}0 & 1 & & & \\
 & & & & - & 1 & 1 & 0 & & & \\
\hline
 & & & & & & 1 & 1 & 0 & & \\
 & & & & & - & 1 & 1 & 0 & & \\
\hline
 & & & & & & & & 0 & &
\end{array}$$

10000111 ÷ 00000101 = 00011011

$$\begin{array}{r|cccccccc|cl}
 & & & & 1 & 1 & 0 & 1 & 1 & = & 27_{(\text{base }10)} \\
\hline
1\,0\,1\,) & \cancel{1} & \cancel{0} & \cancel{0} & {}^{1}0 & 0 & 1 & 1 & 1 & = & 135_{(\text{base }10)} \\
 & - & 1 & 0 & 1 & & & & & = & 5_{(\text{base }10)} \\
\hline
 & & & 1 & \cancel{1} & {}^{1}0 & & & & & \\
 & & - & 1 & 0 & 1 & & & & & \\
\hline
 & & & & & 1 & 1 & & & & \\
 & & & & - & & 0 & & & & \\
\hline
 & & & & & 1 & 1 & 1 & & & \\
 & & & & - & 1 & 0 & 1 & & & \\
\hline
\end{array}$$

$$\begin{array}{r} 1\ 0\ 1 \\ -\ 1\ 0\ 1 \\ \hline 0 \end{array}$$

Binary-Decimal Conversion

Binary means "1's and 0's". The computer is made up of switches, each triggering and action in the computer when flipped. A 1 in binary symbolizes an "on" in the computer, whereas a 0 symbolizes an "off". The binary number system has a base number of two, meaning that each place is equal to the one before it times two. So that being said, if the first digit (the far right) is 1, then they go in this order from right to left: 1, 2, 4, 8, 16, 32, 64, 128, 256, and so on.

$\boxed{256, 128, 64, 32, 16, 8, 4, 2, 1}$

Look at the binary number 010011110 and convert it into decimal. This number is equal to 158. How did I get that? Well, if you follow the table that I showed you in the beginning, the first digit is 1, followed by 2, and so on. Each place where there is a one, you add the number in that place to your total. So, since there's a 0 in the "1's" place, we don't add 1, but since there's a 1 in the "2's" place, we will add 2. There's a 1 in each of these places: 2, 4, 8, 16, and 128, so, 2+4+8+16+128=158. That's so easy isn't it? Here, try this one: 101010101. By the way, when doing this while at the computer, the scientific mode of Windows Calculator has a feature to convert between hexadecimal, decimal, octal, and binary. You should have gotten 341 if you did it right. That was easy wasn't it? To convert from decimal is much harder. You take the next lowest place value from your number (unless your number is the same as a place value, such as 32), and you divide your number by that place, take the remainder, rinse and repeat until you get to 1. So, if our number was 100, we would divide it by 64, and get 36. We would then divide that by 32, and get 4. We divide 4 by 4, and get 1. Then we make our number and put a 1 in every place that we divided by. So, 100 in binary is 1100100.

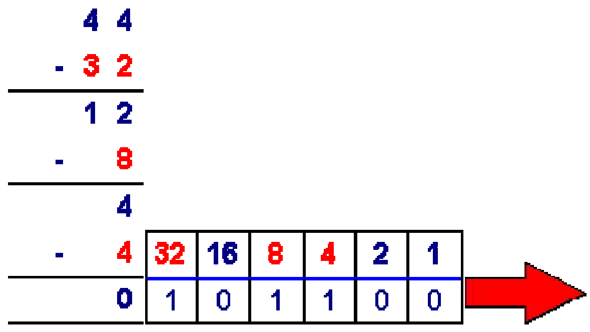
Binary to Decimal Conversion

Binary Evaluate	2^4	2^3	2^2	2^1	2^0			Decimal
Decimal Value	16	8	4	2	1		Value	Number
					0	>	0	0
					1	>	1	1
			1	1	0	>	4+2+0	6
		1	0	1	0	>	8+0+2+0	10
	1	0	1	1	0	>	16+0+4+2+0	22
	1	1	0	0	1	>	16+8+0+0+1	25
	1	1	1	1	1	>	16+8+4+2+1	31

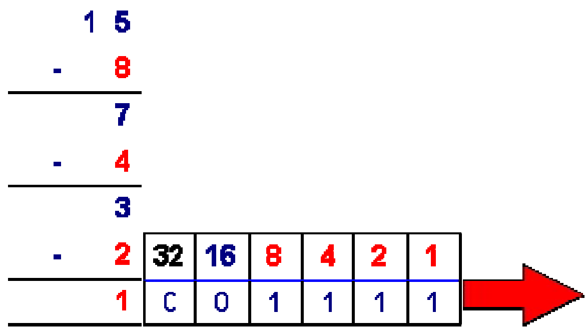
Decimal to Binary Conversion

To convert a decimal number to binary, first subtract the largest possible power of two, and keep subtracting the next largest possible power form the remainder, marking 1s in each column where this is possible and 0s where it is not.

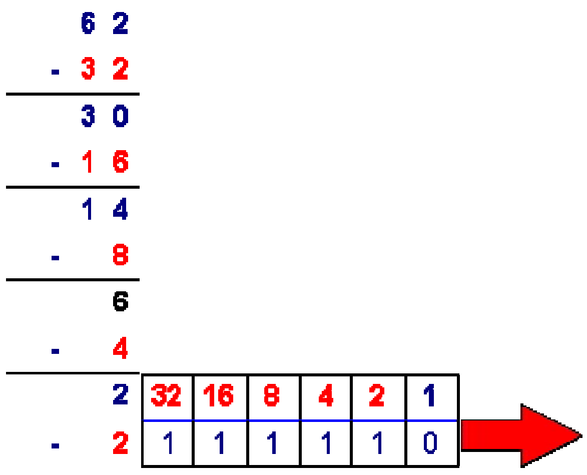
Example 1 - (Convert Decimal 44 to Binary)



Example 2 - (Convert Decimal 15 to Binary)



Example 3 - (Convert Decimal 62 to Binary)



<u>Question</u>

Perform the following sums using binary numbers:

1. 75-42
2. -75-42
3. -75-(-42)

<u>Answer</u>

First we choose to use 8-bit true magnitude format. In this format:

+75 = 01001011; +42 = 00101010

1.	01001011	+75 in true magnitude format
	+11010110	2's complement of +42
	________	
	00100001	this is +33 in true magnitude format

2. 10110101 2's complement of +75

+11010110 2's complement of +42

____

10001011 sign bit is 1 so perform 2's complement

01110101 2's complement of last line

11110101 replace sign bit

____

11110101 -117 in true magnitude format

3. **-75-(-42) = -75+42 = 42-75 and so**

00101010 +42 in true magnitude format

+10110101 2's complement of +75

____

11011111 sign bit is 1 so perform 2's complement

00100001 2's complement of last line

10100001 replace sign bit

____

10100001 -33 in true magnitude format

Question

The executive board of a small company consists of

- A managing director A
- A chief executive B
- Two non-executive directors C and D

In order for a decision to be made A requires the support of one other board member, similarly B requires the support of two other board members. Construct a truth table for the decision-making strategies and indicate clearly where voting results in the decision going

i. against A
ii. against B

Answer

In the truth table below, A=1 means that A voted in favor of a decision, etc. etc.

- Y=0 means that the decision is **not** carried
- Y=1 means that the decision is carried
- Y=A means that the decision is carried against A's wishes
- Y=B means that the decision is carried against B's wishes

A	B	C	D	Y		A	B	C	D	Y
0	0	0	0	0		1	0	0	0	0A
0	0	0	1	0		1	0	0	1	1B
0	0	1	0	0		1	0	1	0	1B
0	0	1	1	0		1	0	1	1	1B
0	1	0	0	0B		1	1	0	0	1
0	1	0	1	0B		1	1	0	1	1
0	1	1	0	0B		1	1	1	0	1
0	1	1	1	1A		1	1	1	1	1

To convert a fractional decimal number to binary then the procedure to follow is

- First divide the number at the decimal point and treat the two parts separately.

- For the integer part then repeatedly divide it by 2 and store the remainder until nothing is left.
- The remainders, when reverse-ordered, give the first part of the binary number. The reverse-ordering comes about since the first division by 2 gives the *least significant bit* (lsb) and so on until the last division which gives the *most significant bit* (msb).
- For the fractional part repeatedly multiply by 2 and record the carries i.e. when the resulting number is greater than 1. Repeat this process until the desired precision is achieved.

Convert **57.4801$_{\mathbf{dec}}$** to its binary equivalent to 14-bit accuracy.

Answer

As the decimal number has both an integer and a fractional part the problem has to be done in two steps

First take the integer part i.e. 57 and repeatedly divide by 2 noting the remainders of each division.

57	/	2	=	28	remainder	**1**	**lsb**
28	/	2	=	14	remainder	**0**	
14	/	2	=	7	remainder	**0**	
7	/	2	=	3	remainder	**1**	
3	/	2	=	1	remainder	**1**	
1	/	2	=	0	remainder	**1**	**msb**

The binary equivalent of **57$_{\mathbf{dec}}$** is therefore given by the remainders ordered from *most significant bit* (**msb**) to *least significant bit* (**lsb**) and is hence **1110001$_{\mathbf{bin}}$.**

The fractional part is given by repeatedly multiplying by 2 and storing the carries (when the result of the multiplication exceeds 1) until the required bit accuracy is reached.

$.4801 \times 2 = .9602 + \mathbf{0}$

$.9602 \times 2 = .9204 + \mathbf{1}$

$.9204 \times 2 = .8408 + \mathbf{1}$

$.8408 \times 2 = .6816 + \mathbf{1}$

$.6816 \times 2 = .3632 + \mathbf{1}$

$.3632 \times 2 = .7264 + \mathbf{0}$

$.7264 \times 2 = .4528 + \mathbf{1}$

and so

$\mathbf{57.4801_{dec} = 111001.0111101_{bin}}$

Example:

An analogue to digital converter (ADC) measures voltages in the range 0 to 25 V and has 12-bit accuracy. What is the smallest voltage step that the ADC can resolve?

Answer

12 bits = 2^{12} = 4096 therefore the ADC can measure 4096 different values of voltage (from 0 to 4095 inclusive), the number of voltage steps is thus 4095 (one fewer than the number of different values available). Assuming that we set digital 0 to be equivalent to 0V and digital 4095 to be equivalent to 25V then each voltage step is simply given by:

25V / 4095 = 0.006105V = 6.105mV

DIGITAL ELECTRONICS, PART-II

COMPUTER MEMORY AND STORAGE

- Digital memory is used to provide means to store and access binary data: sequences of 1's and 0's which have resistance to corruption.
- Many magnetic materials retain their strength of magnetization over time; so storing a voltage signal (data) by magnetizing a magnetic material
- In audio and video systems, a thin plastic tape is impregnated with iron-oxide particles, which can be magnetized or demagnetized via the application of a magnetic field from an electromagnet coil. The data is retrieved from the tape by moving the magnetized tape past another coil of wire. The magnetized spots on the tape induce a voltage in that coil and this will reproduce the voltage waveform initially used to magnetize the tape.
- As the tape ages and the magnetization fades, the analog signal on the tape will be reduced (signal is corrupted).
- Since analog signals have infinite resolution, the smallest degree of change will have an impact on the integrity of the data storage.
- In storing the data in binary digital form, the strength of magnetization on the tape would fall into two discrete levels: "high" and "low,"
- As the tape aged those same locations on the tape would experience slight alteration of magnetic field strength and unless the alterations were *extreme*, no data corruption would occur upon re-play of the tape.
- By reducing the resolution of the signal on the magnetic tape, we've gained significant immunity of the signal.
- When we store data in a device, we also require locating precisely *where* in the device that it is.
- Most memory devices can be thought of as a series of mail boxes, folders in a file cabinet. When we refer to the actual information being stored in the memory device, we usually refer to it as the *data*. The location of this data within the storage device is typically called the *address*.

- With some types of memory devices, the address in which certain data is stored can be called up by means of parallel data lines in a digital circuit.
- With other types of devices, data is addressed in terms of an actual physical location on the surface of some type of media (the *tracks* and *sectors* of circular computer disks, for instance).
- The process of storing a piece of data to a memory device is called *writing*, and the process of retrieving data is called *reading*.
- Some devices do not allow for the writing of new data, and are purchased "pre-written" from the manufacturer; this is referred to in as *read-only memory*, or ROM.
- Cassette audio and video tape, on the other hand, can be re-recorded (re-written) or purchased blank and recorded fresh by the user. This is often called *read-write memory*.
- The CPU has direct access to data in primary storage but not to data in secondary storage.
- To access data in secondary storage, the CPU must communicate with the storage device's storage controllers and request specified data.
- The CPU has a Control Unit which interprets and controls the execution of computer instructions it has loaded into RAM/ROM from the hard drive. It also has one or more Arithmetic Logic Units (ALUs) which carry out operations like addition/subtraction and an Input/output Unit.

The CPU's Registers

- Parts of a computer's primary storage system are the registers (less than 100) inside the CPU.
- Data the CPU is processing is stored in these registers. They may be 8 bit registers; 16 bit registers; 32 bit registers; or 64 bit registers.
- CPU may be classified according to the bit capacity of its registers: an 8 bit CPU will have 8 bit registers; a 16-bit CPU 16-bit registers; and so forth.
- An 8 bit register can transfer 8 bits of data at one clock stroke of the CPU; 16 bit registers, 16 bits; 32 bit registers, 32 bits; and 64 bit registers, 64 bits.
- The larger the bit capacity of a CPU's registers, the more powerful the CPU will be.

Computers Receive and Send Data by *Bus*

A computer has a number of buses. A computer bus is a computer's data transmission roadway. Computer bus system allows data in the form of info, commands, instructions, addresses, and data to move back and forth across its own inner road system. A computer needs a capable bus system so data traffic can move effectively back and forth throughout the system without delay. The CPU has a *CPU Bus* which consists of lanes like lanes on an interstate highway to allow data transmission between its Control Unit; its Arithmetic Logic Units; and its registers.
CPU Bus has:

**CPU Memory Bus* which consists of lanes that allow data transmission between it and its Primary Storage locations.

* *System Bus* which allows for data transmission between the CPU and all the devices installed on the computer system.

The wider these lanes are, the more adequately they can handle the flow of data traffic: if the CPU has 16 bit registers but the bus only has 8 bit lanes, the CPU can only send half the data (8 bits) in one operation. So, it will have to transmit the 16 bits of data in its register in two operations. This will slow down its performance.

The CPU's Cache Memory

Instructions that are repeatedly required to run programs are stored in cache memory. The L1 cache memory is usually built into the CPU, installed on the die. L1 cache typically has storage capability from about 312 KB to 1024 KB. L2 cache which may be installed inside or outside the CPU housing typically has storage capability of 512 KB to 2 MB. If the computer has L3 cache, it will be located on a chip outside close to the CPU or on the Motherboard. Cache is designated L1, L2, L3 based on its nearness to the CPU; L1 being closest.

Computer Storage

Computers are designed to allow data to be input, processed, stored, and output. There are two types of storage a computer uses when it boots, loads an operating system like Windows and begins to perform useful functions: 1) **primary storage**, which is called *memory*; and 2) **secondary storage**, which is referred to as *storage*. Primary storage is where a computer stores data on a temporary basis so it can process the data. It is a short term memory that is directly accessible to a computer processor and it is **volatile** because it is erased when the power is turned off. Data, the computer is currently processing or data which the computer knows it is about to need for processing is stored in primary storage. Memory in primary storage can be accessed quickly by the CPU. Its storage capacity is much smaller than what can be stored in secondary or tertiary storage. Computers need enough primary storage to function and temporarily hold anticipated amounts of data for processing. Secondary storage is where a computer stores data it is not currently processing but which it may need at some later time. Secondary storage can be thought of as "long term memory", or storage, and it is **non-volatile** in nature because data remains intact even when power to a computer is turned off. Operating systems, documents, music files and so on are typically stored in a secondary storage device such as a hard drive. Tertiary storage is storage which is not connected to the computer but is physically "near" the computer so that its media can be retrieved by mechanical means and brought to the computer and loaded into it. Offline storage is storage media which can be inserted into the computer or connected to the computer and used but which can then be removed from the computer and stored elsewhere.

Primary Storage

RAM Memory

Data in a computer's primary storage is stored for very fast retrieval. It is called Random Access Memory because any of the data in RAM can be accessed just as fast as any of the other data. Primary Storage (RAM) is designed to be large enough and accessible enough to cooperate with and complement the speed of the computer's Central Processing Unit (CPU) and its cache memory (L1 cache, L2 cache, etc.) as the CPU processes commands and data. Data stored as RAM is stored in two types: dynamic RAM and static RAM. Primary memory is stored largely as dynamic RAM, or DRAM. It is cheaper than static RAM, SRAM, and it cannot be accessed quickly. The computer uses SRAM in its L1, L2, and L3 cache memory. It is much more costly than DRAM but it can be accessed faster by the CPU. The computer needs data stored in SRAM right at the point where it will be loaded in the CPU's registers for processing. RAM temporary storage ranges from 256 MB to 4 GB.

ROM Memory

A CPU will also make use of some ROM, or Read Only Memory, in primary storage. This memory is used as the computer begins to boot up. Small programs called firmware are often stored in ROM chips on hardware devices and they contain instructions the computer can use in performing some of the most basic operations required to operate hardware devices. ROM memory cannot be easily or quickly overwritten or modified. The Central Processing Unit is the computer's brain and each CPU has an internal clock which determines how fast the CPU can process data. At each tick of its internal clock, the computer can perform an action of some type. CPU clock speed ranges from about 700 MHz to about 3 GHz. For example, if the CPU is a 2 GHz CPU, it can perform an action 2 billion times a second. The CPU is a small chip inside a box which plugs into the Motherboard of the computer.

Secondary Storage

The shelves of books in a library are similar to a computer's secondary storage. They are filled with data which is available to be retrieved. When someone removes a book from the shelves and sits down at a table in the library to read it, we can say the data has been removed from secondary storage and moved to primary storage. The table is the primary storage location. There are a few things we can take notice of here which are similar to computer storage.

First, primary storage is closer to where the data is being processed than secondary storage.

Second, primary storage can store far less data than secondary storage.

Third, data in primary storage can be accessed far easier and far faster than data in secondary storage.

Fourth, data in primary storage can be accessed as quickly as any of the other data in primary storage.

Hard Disk Drive (HDD)

The computer's largest secondary storage location is its hard disk drive. Hard drives are platters like dishes which are stacked top, middle, and bottom to make one unit. Hard drives are mechanical devices which store data magnetically. They are considered permanent storage. The capacity of hard drives typically ranges from about 40 GB to 400 GB, or higher up to 2 TB. Hard disk drives are read/write. They can be read over and over and they can be modified, or written to, over and over.

Tertiary Storage

Tertiary storage is very large storage which is separate from the computer. The most obvious example of tertiary storage is an automated storage facility where mechanical arms retrieve media and load it into large computers. Other tertiary storage may simply be off-grounds locations which allow vital data in various mediums to be safe-guarded for security purposes- fire, theft, etc.

Offline Storage

Offline storage is storage media which can be inserted into the computer and used but which can then be removed from the computer and stored elsewhere. It can also be external sources (ex. a remote hard drive) which are connected to the computer and then disconnected when the user is finished with them. CD drives, and DVD drives might also alternately be considered secondary storage because their drives are usually installed in the computer but the key here is the media the data is stored on. The media- the CD and the DVD is not a part of the computer and is usually removed from the computer after use. It is, therefore, "offline" storage.

USB Flash Drive

It is a compromise between RAM-type memories and ROM memories. Flash memory possesses the non-volatility of ROM memories while providing both read and write access. However, the access times of flash memories are longer than the access times of RAM.

- The flash drive is a small removable stick that plugs into a USB port on the computer. Flash drives are non-volatile. Data can be read from and written to them over and over again.
- They are a convenient, temporary storage medium. but considered a poor storage medium for long term storage because they are easily misplaced;

somewhat fragile; and they can be corrupted through repeated use, unintended damage and ensuing malfunction.

- They are, however, good for storing data and carrying it from home to office where the data will be used.
- Flash drives today typically store somewhere from about 128 MBs to 16 GBs, though a 256 GB drive. There are two basic standards of USB connectivity: USB 1.0 and 1.1 (an improved version of 1.0); 2.0; and 3.0 coming next year. USB 1.0 and 1.1 supports theoretical data transfer speeds of 1.5 Mbits/sec (low speed) to 12 Mbits/sec (full speed). It is used in low bandwidth devices such as keyboards, mice, and joy sticks.
- USB 2.0 supports theoretical data transfer speeds of 480 Mbits/sec (high speed).
- USB 3.0 supports theoretical data transfer speeds of 5 Gbits/sec, or 5000Mbits/sec (super speed).

CD Drive

CD drives store data on shiny 5" discs, 1.2 mm thick. There are a variety of disks the CD drive can "read". CDs have a storage capacity of 700 MBs. This storage capability is equivalent to about 80 minutes of sound recording.

CD-ROM

Can be written to once and this occurs in the manufacturing process. The surface of the CD-ROM is actually etched or embedded with the data. The result is that the surface of the CD now has "lands", or raised areas, and "pits" or recessed areas, each representing either a 1 or a 0, which is read by the laser beam passing over it. Whereas most of the other drives mentioned store data magnetically, the CD drive is an optical device. Most software the user can purchase comes on a CD-ROM which is used to load, or install, the software onto the user's hard drive. Books on CD and musical albums on CD also come on CD-ROM.

CD-R

R for "Recordable" can be written to once by the user but only once. After that, they can be read over and over again.

DVD Drive

A recordable CD has the capacity to store about 80 minutes of sounding a full-length movie on CD could take up to 7 CDs. That's where DVD disks come in. Single layer digital video disks, or DVDs have a storage capacity of 4.7 GB; double layer DVDs a storage capacity of 8.5 GBs. DVDs use both the top surface of the disc and the bottom surface of the disc for storage so the data on the disc is more densely

Blu-Ray Drive

Blu-Ray is the next jump in technology over the traditional DVD formats to play and record high definition titles and TV programs. Blu-Ray is also used in the Play Station 3, PS3, which can play Blu-Ray titles. The popular DVD formats use a red laser to access data on the DVD. Blu-Ray uses a blue-violet laser. The Blu-Ray laser has a shorter wavelength, 405 nanometers, versus the red laser, 650 nanometers. The shorter wavelength means the blue laser can focus on a smaller "data spot" on the disk. This allows the disk to be more densely packed with data. This translates into the Blu-Ray disk holding about 6x more data. Though the Blu-Ray disks are the same size as standard DVD disks (12 cm), a single layer Blu-ray disk holds 25 GBs of data; a double layer disk 50 GBs. The current maximum recoding speed is 12x, which is about 54 MBs per second. Also available, Mini Blu-Ray disks (8 cm) are single layer 7.8 GBs; double layer 15.7 GBs.

Memory Cells

Primary computer memory is made up of memory cells, where each memory cell contains exactly one number. Thus, **a memory cell can be thought of as box into which a single number can be placed**. The number contained in a memory cell can be changed over time. When a new number is stored into a memory cell, the old number contained in the memory cell is lost forever. At any time, the computer may peer into a memory cell to read the current contents of the memory cell. The computer may read the contents of a memory cell many times without disturbing it.

A typical computer has millions of memory cells, and **every memory cell has a name**. While a small number of memory cells are given specialized names, such as `Program Counter' or `Processor Status Register', most memory cells are named using a unique number **called its `address**.' No two memory cells have the same address. The address of a memory cell never changes over time. The address of a memory cell can be thought of as a number permanently imprinted on the side of the box. Thus, every memory cell has two numbers -- its address (which never changes) and its current contents (which changes over time). Since a memory cell has both a contents and an address, both of which are numbers, it is useful to adopt a notation to keep them separate. A fairly common notation is to enclose the memory cell address in square brackets, followed by a colon, followed by the memory cell contents. For example, `[223]: 17' is the notation used to specify that memory cell [223] currently has the number 17 as its contents. As an example, consider the two arbitrary memory cells named [223] and [873]. When power is first applied to these two memory cells, both cells will have initial contents of zero; this is shown below:

[223]: 0   [873]: 0

Eventually, the computer comes along and writes the number 104 into memory cell [223]; this is shown below:

[223]: 104   [873]: 0

Later on, the computer writes number 105 into memory cell [873] as shown below:

[223]: 104   [873]: 105

Even later on, the computer overwrites memory cell [223] with 72. The previous content of memory cell [223] is lost forever; this is shown below:

[223]: 72   [873]: 105

At later point in time, the number 104 is stored back into memory cell [223]. Again, the previous content of memory cell [223] is lost forever; this is shown below:

[223]: 104   [873]: 105

The computer reads and overwrites the contents of memory cells [223] and [873] as many times as needed to accomplish its current task.

Memory Arrays

Since every memory cell named with a unique address number, they are named using sequential numbers starting from the number zero. Thus, the first memory cell has an address number of [0], the second memory cell has an address number of [1], and so forth until all memory cells have been given an address number.

The memory cells can be listed vertically down the page as follows:

```
[0]: 0        [7]: 0
[1]: 0        [8]: 0
[2]: 0        [9]: 0
[3]: 0        [10]:0
[4]: 0        [11]: 0
[5]: 0        [12]: 0
[6]: 0
```

The 13 memory cells above have addresses from [0] to [12]. All 13 memory cells contain the number 0. While memory can be listed vertically as shown above, it is not very compact. A more compact representation is to list the memory cells as an array. An example memory array is shown below and the paragraph describing it follows:

```
 +----+-----------------------------------------------------------+
|1st |                    Last Digit                              |
|Dig.|  0    1    2    3    4    5    6    7    8    9  |
|----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 0 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|    +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|    +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 2 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|    +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 3 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|    +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 4 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|   +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 5 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|   +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 6 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|   +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 7 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|   +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 8 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|   +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 9 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|   +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 10 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|   +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 11 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|   +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 12 |   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
|   +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|... |                    ...                                  |
```

Each row lists the contents of ten consecutive memory cells. Thus, the first row lists the contents of the ten memory cells whose addresses are [0] through [9], the second row lists the contents of the ten memory cells whose addresses are [10] through [19], etc. For the memory array above, every memory cell has a content of zero. An even more compact representation is possible by dropping all of the lines between rows and columns and dropping all of the column headings. An example of the more compact representation is shown below:

```
[0x]:  0  0  0  0  0  0  0  0  0  0
[1x]:  0  0  0  0  0  0  0  0  0  0
[2x]:  0  0  0  0  0  0  0  0  0  0
[3x]:  0  0  0  0  0  0  0  0  0  0
```

The small `x' in [0x], [1x], [2x], and [3x] is a place holder for the last digit of the memory cell address. Thus, memory cell [20] corresponds to the first number after the colon on the row labeled [2x], memory cell [21] corresponds to the second number, and so forth. The last number on the row labeled [2x] corresponds to the [29] memory cell. As an example of memory use, let's store each character of the word `Hello!' into consecutive memory cells starting at memory cell [22]. The conversion for each character can be looked up in the ASCII character set table. For convenience, the conversions are listed below:

- `H' - 74
- `e' - 101
- `l' - 108
- `l' - 108
- `o' - 111
- `!' - 33

After the computer has stored the word `Hello!' into memory, the memory looks as follows:

```
[0x]:  0   0   0   0   0   0   0   0   0   0
[1x]:  0   0   0   0   0   0   0   0   0   0
[2x]:  0   0   74  101 108 108 111 33  0   0
[3x]:  0   0   0   0   0   0   0   0   0   0
```

The word `Hello!' can be changed into `Help!!' by overwriting memory cells [25] and [26] with the numbers corresponding to the letters `p' (112) and `!' (33). The resulting memory looks as follows:

```
[0x]:  0   0   0   0   0   0   0   0   0   0
[1x]:  0   0   0   0   0   0   0   0   0   0
[2x]:  0   0   74  101 108 112 33  33  0   0
[3x]:  0   0   0   0   0   0   0   0   0   0
```

To finish off the example, the word `Help!!' can be changed into `Bye!!!' by overwriting memory cells [22] through [25] with the numbers corresponding to the letters `B' (68), `y' (121), `e' (101), and `!' (33). The resulting memory looks as follows:

```
[0x]:  0   0   0   0   0   0   0   0   0   0
[1x]:  0   0   0   0   0   0   0   0   0   0
[2x]:  0   0   68  121 101 33  33  33  0   0
[3x]:  0   0   0   0   0   0   0   0   0   0
```

Memory Limitations

Memory cells have a restriction on how big of a number they can store. Most current computers use a memory cell whose content is limited to a number between 0 and 255. It is not important what the exact memory cell limit is; what is important is that large numbers do not fit in a single cell. Too large number is stored in a number of consecutive memory cells. **I**f a number is larger than five digits, it is broken into a sequence of smaller five digit numbers and stored in consecutive memory cells. For example, the nine digit number 123456789 is stored as the number 1234 in one cell and as 56789 in the next cell. This is shown as follows:

[22]: 1234
[23]: 56789

The most significant digits are stored in the first memory cell, followed by the least significant digits. It turns out that this is just a convention. The computer can just as easily store the least significant digits first, as shown below:

[22]: 56789
[23]: 1234

Some computers store big numbers with the most significant digits first and others store big numbers with the least significant digits first; the former computers are called `big-endian' and the latter computers are called `little-endian'.

The Role of Memory

The term "**memory**" applies to any electronic component capable of temporarily storing data. There are two main categories of memories:

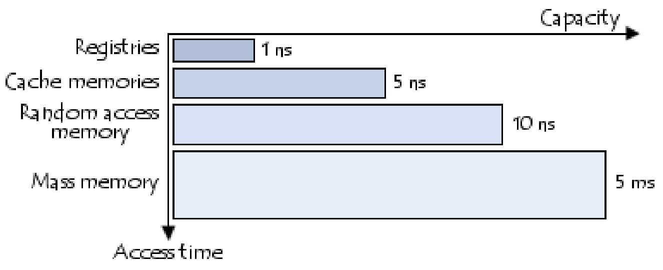
- **Internal memory** that temporarily memorizes data while programs are running. Internal memory uses micro-conductors, i.e. fast specialized electronic circuits. Internal memory corresponds to what we call random access memory (RAM).
- **Auxiliary memory** (also called *physical memory* or *external memory*) that stores information over the long term, including after the computer is turned off. Auxiliary memory corresponds to magnetic storage devices such as the

hard drive, optical storage devices such as CD-ROMs and DVD-ROMs, as well as read-only memories.

Technical Characteristics of memory

- **Capacity**, representing the global volume of information (in bits) that the memory can store
- **Access time**, corresponding to the time interval between the read/write request and the availability of the data
- **Cycle time**, representing the minimum time interval between two successive accesses
- **Throughput**, which defines the volume of information exchanged per unit of time, expressed in bits per second
- **Non-volatility**, which characterizes the ability of a memory to store data when it is not being supplied with electricity

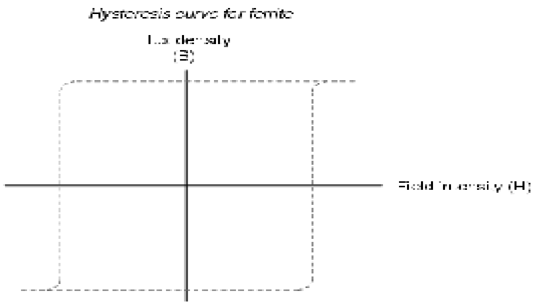
The ideal memory has a large capacity with restricted access time and cycle time, a high throughput and is non-volatile. However, fast memories are also the most expensive. This is why memories that use different technologies are used in a computer, interfaced with each other and organized hierarchically.



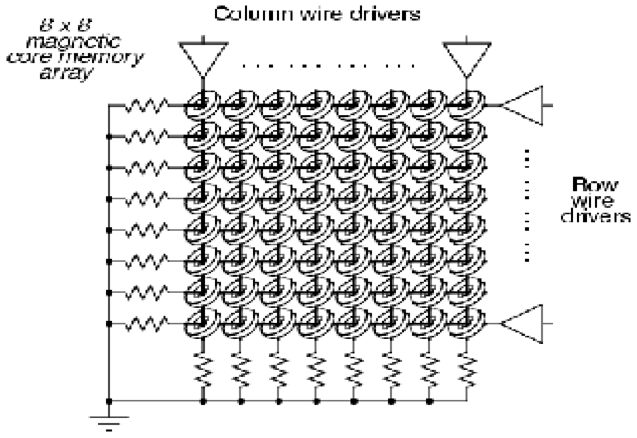
The fastest memories are located in small numbers close to the processor. Auxiliary memories, which are not as fast, are used to store information permanently.

Memory Material

Some magnetic materials are means of storing binary data where "ferrite," possessed hysteresis curves that were almost square:



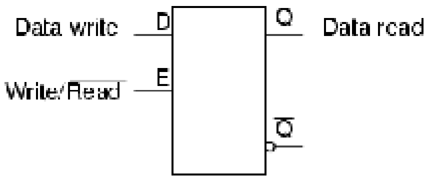
Shown on a graph is the strength of the applied magnetic field on the horizontal axis (*field intensity*), and the actual magnetization (orientation of electron spins in the ferrite material) on the vertical axis (*flux density*). Ferrite won't become magnetized in one direction until the applied field exceeds a critical threshold value, after which the electrons in the ferrite "snap" into magnetic alignment and the ferrite becomes magnetized. If the applied field is then turned off, the ferrite maintains full magnetism. To magnetize the ferrite in the other direction (polarity), the applied magnetic field must exceed the critical value in the opposite direction. Once that critical value is exceeded, the electrons in the ferrite "snap" into magnetic alignment in the opposite direction. Once again, if the applied field is then turned off, the ferrite maintains full magnetism. To put it simply, the magnetization of a piece of ferrite is "bistable." Exploiting this strange property of ferrite, we can use this natural magnetic "latch" to store a binary bit of data. To set or reset this "latch," we can use electric current through a wire or coil to generate the necessary magnetic field, which will then be applied to the ferrite.



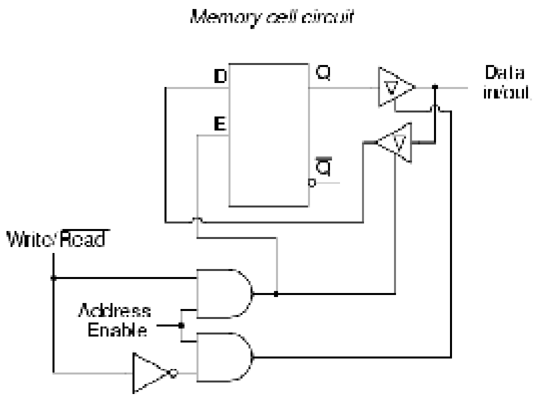
A grid of wires, electrically insulated from one another, crossed through the center of many ferrite rings, each of which being called a "core." As DC current moved through any wire from the power supply to ground, a circular magnetic field is generated around that energized wire. The resistor values were set so that the amount of current at the regulated power supply voltage would produce slightly more than 1/2 the critical magnetic field strength needed to magnetize any one of the ferrite rings. Therefore, if column #4 wire was energized, all the cores on that column would be subjected to the magnetic field from that one wire, but it would not be strong enough to change the magnetization of any of those cores. However, if column #4 wire and row #5 wire were both energized, the core at that intersection of column #4 and row #5 would be subjected to a sum of those two magnetic fields: a magnitude strong enough to "set" or "reset" the magnetization of that core. In other words, each core is addressed by the intersection of row and column. A core memory board from a Data General brand, "Nova" model computer, circa late 1960's or early 1970's had a total storage capacity of 4 Kbytes (that's *kilo*bytes, not *mega*bytes!). A core memory board of later design (circa 1971) is much smaller and more densely packed, giving more memory storage capacity than the former board (8 Kbytes instead of 4 Kbytes):

In a mineral called *garnet*, when arranged in a thin film and exposed to a constant magnetic field perpendicular to the film, tiny oppositely-magnetized regions called "magnetic bubbles" are generated. Bubbles could be created or destroyed with a tiny coil of wire placed in the bubbles' path. The presence of a bubble represents a binary "1" and the absence of a bubble represents a binary "0." Data could be read and written in this chain of moving magnetic bubbles as they passed by the tiny coil of wire. Like core memory, bubble memory is nonvolatile: a permanent magnet supplied the necessary background field needed to support the bubbles when the power is turned off. Bubble memory has a storage density of millions of bits that could be stored on a chip of garnet only a couple of square inches in size. The exclusion of bubble memory as a viable alternative to static and dynamic RAM is its slow speed.

A very simple modern memory is the bistable multivibrator. Capable of storing a single bit of data, it is volatile (requiring power to maintain its memory) and very fast. The D-latch is probably the simplest implementation of a bistable multivibrator for memory usage, the D input serving as the data "write" input, the Q output serving as the "read" output, and the enable input serving as the read/write control line:



If we desire more than one bit's worth of storage (and we probably do), we'll have to have many latches arranged in some kind of an array where we can selectively address which one (or which set) we're reading from or writing to. Using a pair of tri-state buffers, we can connect both the data write input and the data read output to a common data bus line, and enable those buffers to either connect the Q output to the data line (READ), connect the D input to the data line (WRITE), or keep both buffers in the High-Z state to disconnect D and Q from the data line (unaddressed mode). One memory "cell" would look like this, internally:



When the address enable input is 0, both tri-state buffers will be placed in high-Z mode, and the latch will be disconnected from the data input/output (bus) line. Only when the address enable input is active (1), the latch is connected to the data bus.

SHIFT REGISTER

A register that is capable of shifting data one bit at a time is called a shift register. One of the uses of a shift register is to convert between serial and parallel interfaces. This is useful as many circuits work on groups of bits in parallel, but serial interfaces are simpler to construct. Shift registers can be used as simple delay circuits and also as pulse extenders. A serial shift register consists of a chain of flip-flops connected in cascade, with the output of one flip-flop being connected to the input of its neighbor. The operation of the shift register is synchronous; thus each flip-flop is connected to a common clock. Using D flip-flops forms the simplest type of shift-registers. The basic data movements possible within a four-bit shift register is shown in Figure (1).

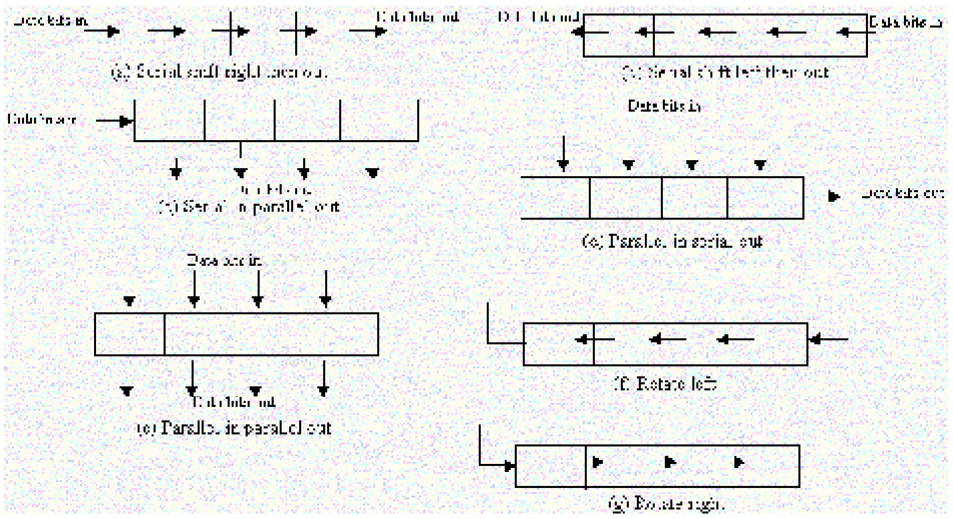


Figure (1): Data movement diagrams

Figure (2) shows the circuit diagram for a four-bit serial in-serial out shift register implemented using D flip-flops. Assuming that the register is initially clear, Figure (3) shows the waveform diagram when ‘1000’ is shifted through the register.

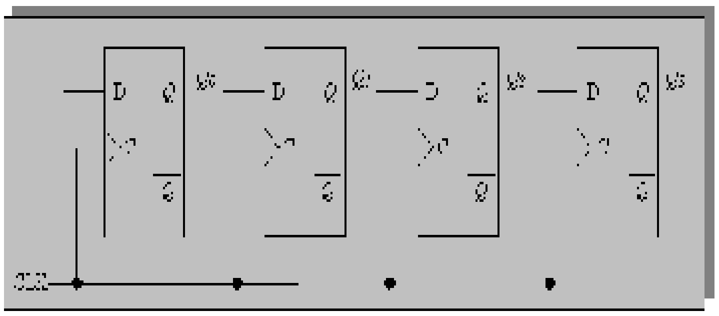


Figure (2): Four-bit serial in-serial out shift register

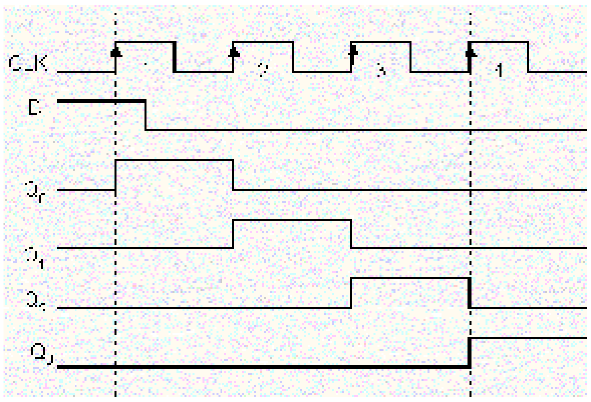


Figure: (3) Waveform Diagram

The effect of shifting the binary digit '1' through the register over time is shown below. We assume that the register is initially reset and the bit enters the left-most stage.

At time t=0

0	0	0	0	$(0)_{10}$

At time t=1

1	0	0	0	$(8)_{10}$

At time t=2

0	1	0	0	$(4)_{10}$

At time t=3

0	0	1	0	$(2)_{10}$

At time t=4

0	0	0	1	$(1)_{10}$

At each time step from t=1 onwards, the right shift results in a state change which indicates a division by two operation. Similarly, if we operate in reverse order (i.e. from t=4 to t=1), the left shift will result in a multiplication be two.

A bidirectional shift register, which is capable of shifting bits to the left (L) or right (R), will then be capable of performing modulo-2 division and multiplication operations. A HIGH on the R/L control input allows data bits inside the register to be shifted to the right, and a LOW allows for the data to be shifted to the left.

Types of Shift Register:

These are: serial-in, parallel-out (SIPO) and parallel-in, serial-out (PISO) types. There are also types that have both serial and parallel input and types with serial and parallel output. There are also bi-directional shift registers which allow varying the direction of the shift register.

Serial-In, Serial-Out

Destructive Readouts

In destructive readout - each datum is lost once it has been shifted out of the right-most bit. These are the simplest kind of shift register. The data string is presented at 'Data In', and is shifted right one stage each time 'Data Advance' is brought high. At each advance, the bit on the far left (i.e. 'Data In') is shifted into the first flip-flop's output. The bit on the far right (i.e. 'Data Out') is shifted out and lost.

Non-destructive readout

Non-destructive readout can be achieved if another input line is added - the Read/Write Control. When this is high (i.e. write) then the shift register behaves as normal, advancing the input data one place for every clock cycle, and data can be lost from the end of the register. However, when the R/W control is set low (i.e. read), any data shifted out of the register at the right becomes the next input at the left, and is kept in the system. Therefore, as long as the R/W control is set low, no data can be lost from the system.

Serial-In, Parallel-Out

This configuration allows conversion from serial to parallel format. Data are input serially, and once the data has been input, it may be either read off at each output simultaneously, or it can be shifted out and replaced.

Parallel-In, Serial-Out

This configuration has the data input in parallel format. To write the data to the register, the Write/Shift control line must be held LOW. To shift the data, the W/S control line is brought HIGH and the registers are clocked. As long as the number of clock cycles is not more than the length of the data-string, the Data Output, Q, will be the parallel data read off in order.

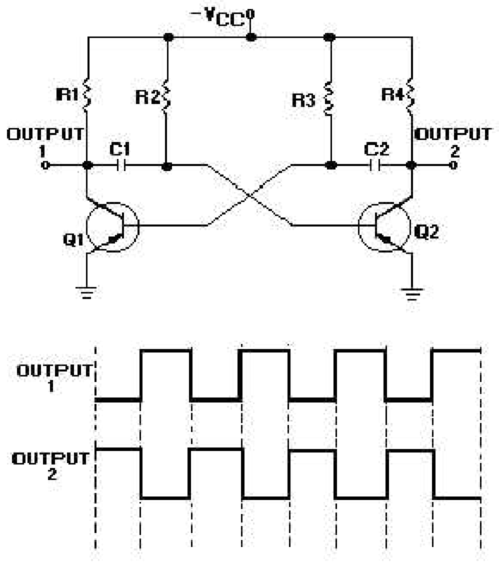
Parallel-In, Parallel-Out

This kind of shift register takes the data from the parallel inputs (D0-D3) and shifts it to the corresponding output (Q0-Q3) when the registers are clocked. It can be used as a kind of 'history', retaining old information as the input in another part of the system, until ready for new information, whereupon, the registers are clocked, and the new data are 'let through'.

CLOCKS AND COUNTERS

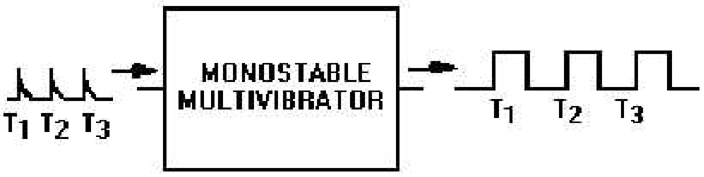
CLOCKS

Clock is a timing signal generated by the equipment to control operations. This control feature is demonstrated in both the D and J-K flip-flops. The clock output had to be in a certain condition for the flip-flops to perform their functions. The simplest form of a clock is the astable multivibrator shown below along with its output waveforms. This multivibrator circuit is called free running because it alternates between two different output voltages during the time it is active. Outputs 1 and 2 will be equal and opposite since Q_1 and Q_2 conduct alternately. The frequency of the outputs may be altered within certain limits by varying the values of R_2C_1 and R_3C_2.



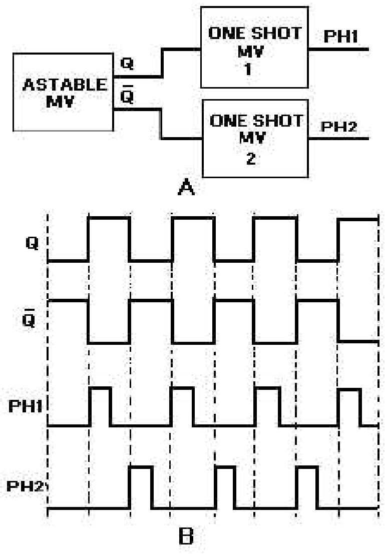
Free-running multivibrator.

Frequency stability of the astable multivibrator can be increased by applying a trigger pulse to the circuit. The frequency of the trigger must be higher than the free-running frequency of the multivibrator. The output frequency will match the trigger frequency and produce a more stable output. Another method of producing a stable clock pulse is to use a triggered monostable or one-shot multivibrator which has one stable state and will only change states when acted on by an outside source (the trigger). A block diagram of a monostable multivibrator with input and output signals is shown below. The duration of the output pulse is dependent on the charge time of an RC network in the multivibrator. Each trigger input results in a complete cycle in the output, as shown in the figure. Trigger pulses are supplied by an oscillator.



Monostable multivibrator block diagram

The circuits described previously are very simple clocks, however, as the complexity of the system increases, so do the timing requirements. Complex systems have multiphase clocks to control a variety of operations. A block diagram of a two-phase clock system is shown below, (A). The astable multivibrator provides the basic timing for the circuit, while the one-shot multivibrators are used to shape the pulses. Outputs Q and Q⁻ are input to one-shot multivibrators 1 and 2, respectively. The resulting outputs are in phase with the inputs, but the duration of the pulse is greatly reduced as shown in (B).



Two-phase clock: A. Block diagram; B. Timing diagram.

Clocks are designed to provide the most efficient operation of the equipment. During the design phase, the frequency, pulse width, and the number of phases required is determined; and the clock circuit is built to meet those requirements. Most modern high-speed equipment uses crystal-controlled oscillators as the basis for their timing networks. Crystals are stable even at extremely high frequencies.

COUNTERS

Introduction

A counter is a device used to count operations, quantities, or periods of time. They may also be used for dividing frequencies, for addressing information in storage, or for temporary storage. Counters are a series of flip-flops wired together to perform the type of counting desired. They will count up or down by ones, twos, or more. The total number of counts or stable states a counter can indicate is called MODULUS. For instance, the modulus of a four-stage counter would be 16_{10}, since it is capable of indicating 0000_2 to 1111_2. The term *modulo* is used to describe the count capability of counters; that is, modulo-16 for a four-stage binary counter, modulo-8 for a three-stage binary counter, and so forth. Hardware counters are limited to given number of columns of digits, i.e. there is a maximum number that a counter can represent. A 3-digit decimal counter can represent numbers from 000 through 999. We define such a counter as a modulus (mod) 1000 counter. One more count will cause it to cycle back to 000. There are counters that count up called up-counters and others that count down called down-counters. There may be counters that can count both ways and in this case they are up-down-counters. Counters are found in:

* Digital Multi-Meter, (DMM) in which a voltage is measured and displayed digitally. Inside the DMM, the A/D (Analog-to-Digital) converter probably uses a counter in the process of converting the analog voltage signal to a digital equivalent.

* A real estate agent uses a gadget placed on a wall to measure interior dimensions (distance between walls). The gadget uses a counter to time how long it takes an acoustical signal to travel across the room and return after reflecting off the opposite wall.

* You drive across two rubber tubes stretched across the pavement. A counter starts running when you hit the first one, and stops when you hit the second one. That way the police know how fast you were going and tell you that when the ticket comes in the mail.

Counting with JK flip-flop

* JK flip-flop operating in its toggle mode goes through the following sequence:

Clock pulse number:	0	1	2	3	4	5	6	…..
Flip-flop output Q :	0	1	0	1	0	1	0	…..

* Flip-flop behaves as a modulo-2 binary counter. Counters of higher modulo can be formed by joining other binary counters. A modulo-4 counter is made by joining two modulo-2 counters as follows:

Clock pulse number:	0	1	2	3	4	5	6	7	8
Counter outputs Q_1,Q_0:	00	01	10	11	00	01	10	11	00

The Toggle (T) Flip-flop - A One Stage Counter

*A toggle flip-flop is a single bit counter that uses two T flip-flops: One is driven by a clock and the other by the output of the first one. The clock generates a short pulse input to the flip-flop (ON) for a short while, then goes to OFF. When the input pulse goes OFF, the output of the first flip-flop changes state and when the output of the first flip-flop goes OFF, the second flip-flop changes state.

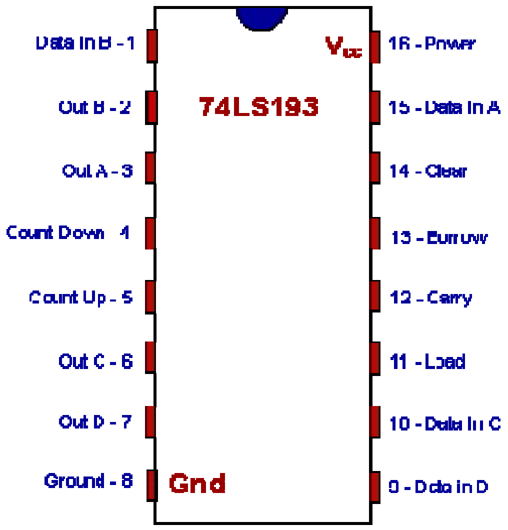
Counters - The 74193

* There are many different integrated circuit counters; one popular chip is the 74193, which may also be the 74LS193, etc. The 74193 has the following characteristics.

- The 74193 is an Up-Down counter. It can count in both directions, but only one direction at time.
- You can preload a count into the 74193.

- You can reset the 74193.
- There are 16 pins on the package, and as usual, power is applied to the corners. Pin 16 gets +5v, and pin 8 is grounded.

Here's the pin-out for the 74193 counter.



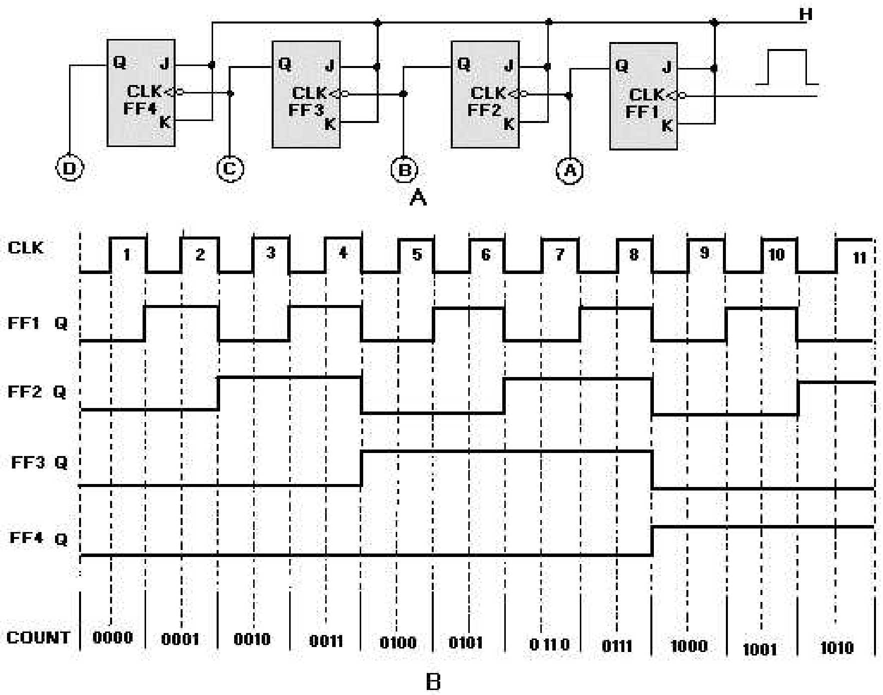
There are several items to note on this chip.

- The power is applied at the corners (pins 8 and 16 here) just as in many other logic chips.
- There are four count outputs, A, B, C and D. These are the bits in the count, and A is the least significant bit (LSB) and D is the most significant bit (MSB).
- There are two inputs, pins 4 & 5.
    - To count up, hold the down pin, #4, high (5v) and put the pulse in pin #5.
    - To count down, hold the up pin, #5, high (5v) and put the pulse in pin #4.
- There are lots of other pins that can be used use to preload a count into the counter, and there are borrow and carry pins that can be used to connect this chip to other 74193 chips when we need to count more than 4 bits. Those other pins can present problems. Pin 14 is a clear input. When you put a 1 on this

pin, the counter is cleared, i.e. reset to zero. Make this pin zero! (Connect it to ground). Pin 11 is for a load signal. When this pin goes high (1), the data inputs are loaded into the counter. Make this pin zero! (Connect it to ground). Pin 12 is a carry output. Useful for 8, 12 or 16 bit counters, etc., using more than one chip. Pin 13 is a borrow output. Useful when counting down in counters with more than four bits.

Basic Ripple Counters

Ripple counters work like a chain reaction that ripples through the counter because of the time involved. (A) shows a basic four-stage, or modulo-16, ripple counter. The inputs and outputs are shown in (B). The four J-K flip-flops are connected to perform a toggle function; which divides the input by 2. The HIGHs on the J and K inputs enable the flip-flops to toggle. The inverters on the clock inputs indicate that the flip-flops change state on the negative-going pulse.

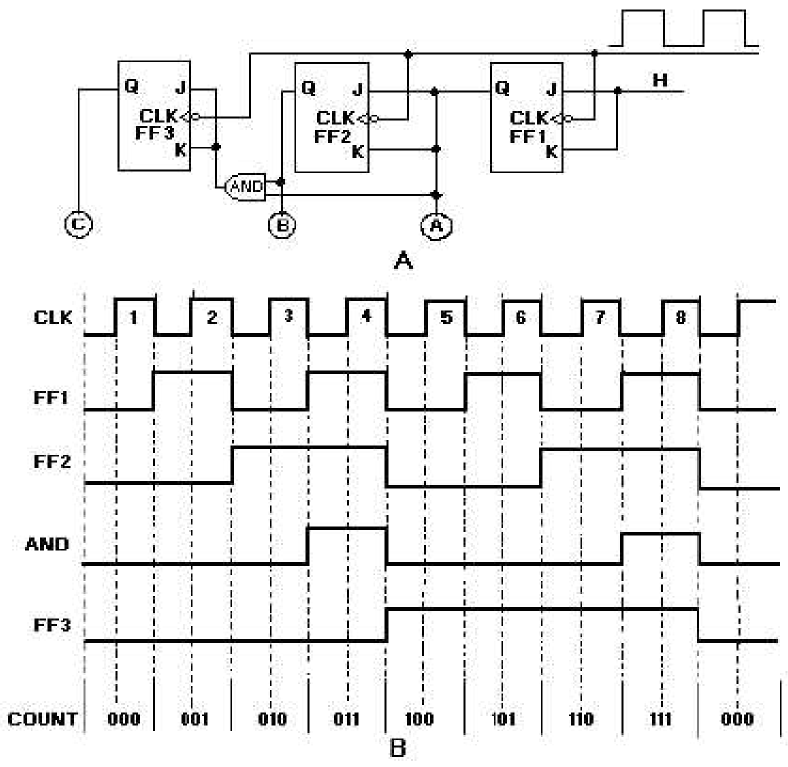


Four-stage ripple counter: A. Logic diagram; B. Timing diagram.

Assume that A, B, C, and D are lamps and that all the flip-flops are reset. The lamps will all be out, and the count indicated will be 0000_2. The negative-going pulse of clock pulse 1 causes FF1 to set. This lights lamp A and we have a count of 0001_2. The negative-going pulse of clock pulse 2 toggles FF1, causing it to reset. This negative-going input to FF2 causes it to set and causes B to light. The count after two clock pulses is 0010_2, or 2_{10}. Clock pulse 3 causes FF1 to set and lights lamp A. The setting of FF1 does not affect FF2, and lamp B stays lit. After three clock pulses, the indicated count is 0011_2. Clock pulse 4 causes FF1 to reset, which causes FF2 to reset, which causes FF3 to set, giving us a count of 0100_2. This step shows the ripple effect. This setting and resetting of the flip-flops will continue until all the flip-flops are set and all the lamps are lit. At that time the count will be 1111_2 or 15_{10}. Clock pulse 16 will cause FF1 to reset and lamp A to go out. This will cause FF2 through FF4 to reset, in order, and will extinguish lamps B, C, and D. The counter would then start at 0001_2 on clock pulse 17. To display a count of 16_{10} or 10000_2, we would need to add another FF**. The ripple counter is also called an ASYNCHRONOUS counter**. In this counter setting and resetting of flip-flops occur one after the other rather than all at once. Because the ripple count is asynchronous, it can produce erroneous indications when the clock speed is high. A high-speed clock can cause the lower stage flip-flops to change state before the upper stages have reacted to the previous clock pulse. The errors are produced by the flip-flops' inability to keep up with the clock.

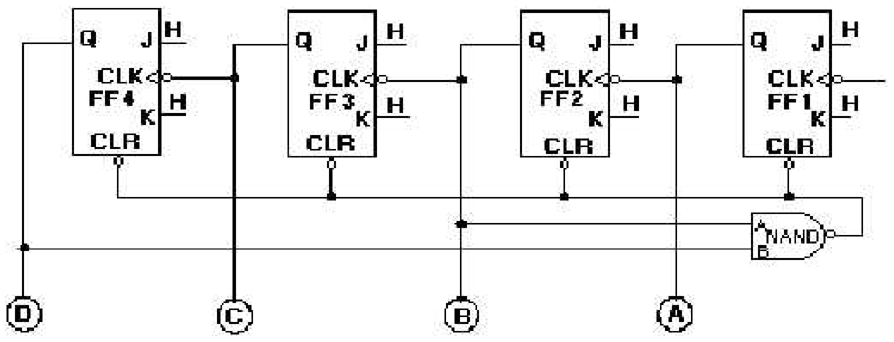
Synchronous Counter

High-frequency operations require that all the flip-flops of a counter be triggered at the same time to prevent errors. We use a SYNCHRONOUS counter for this type of operation**. The synchronous counter is similar to a ripple counter with two exceptions: The clock pulses are applied to each FF, and additional gates are added to ensure that the flip-flops toggle in the proper sequence.** A logic diagram of a three-state (modulo-8) synchronous counter is shown below, (A). The clock input is wired to each of the flip-flops to prevent possible errors in the count. A HIGH is wired to the J and K inputs of FF1 to make the FF toggle. The output of FF1 is wired to the J and K inputs of FF2, one input of the AND gate, and indicator A. The output of FF2 is wired to the other input of the AND gate and indicator B. The AND output is connected to the J and K inputs of FF3. The C indicator is the only output.



Three-stage synchronous counter: A. Logic diagram; B. Timing Diagram.

To explain the circuit, assume the following initial conditions: The outputs of all flip-flops, the clock, and the AND gate are 0; the J and K inputs to FF1 are HIGH. The negative-going portion of the clock pulse will be used throughout the explanation. Clock pulse 1 causes FF1 to set. This HIGH lights lamp A, indicating a binary count of 001. The HIGH is also applied to the J and K inputs of FF2 and one input of the AND gate. Notice that FF2 and FF3 are unaffected by the first clock pulse because the J and K inputs were LOW when the clock pulse was applied. As clock pulse 2 goes LOW, FF1 resets, turning off lamp A. In turn, FF2 will set, lighting lamp B and showing a count of 010_2. The HIGH from FF2 is also felt by the AND gate. The AND gate is not activated at this time because the signal from FF1 is now a LOW. A LOW is present on the J and K inputs of FF3, so it is not toggled by the clock. Clock pulse 3 toggles FF1 again and lights lamp A. Since the J and K inputs to FF2 were LOW when pulse 3 occurred, FF2 does not toggle but remains set. Lamps A and B are lit, indicating a count of 011_2. With both FF1 and FF2 set, HIGHs are input to both inputs of the AND gate, resulting in HIGHs to J and K of FF3. No change occurred in the output of FF3 on clock pulse 3 because the J and K inputs were LOW at the time. Just before clock pulse 4 occurs, we have the following conditions: FF1 and FF2 are set, and the AND gate is outputting a HIGH to the J and K inputs of FF3. With these conditions all of the flip-flops will toggle with the next clock pulse. At clock pulse 4, FF1 and FF2 are reset, and FF3 sets. The output of the AND gate goes to 0, and we have a count of 100_2. It appears that the clock pulse and the AND output both go to 0 at the same time, but the clock pulse arrives at FF3 before the AND gate goes LOW because of the transit time of the signal through FF1, FF2, and the AND gate. Between pulses 4 and 8, FF3 remains set because the J and K inputs are LOW. FF1 and FF2 toggle in the same sequence as they did on clock pulses 1, 2, and 3. Clock pulse 7 results in all of the flip-flops being set and the AND gate output being HIGH. Clock pulse 8 causes all the flip-flops to reset and all the lamps to turn off, indicating a count of 000_2. The next clock pulse (9) will restart the count sequence.



Decade counter

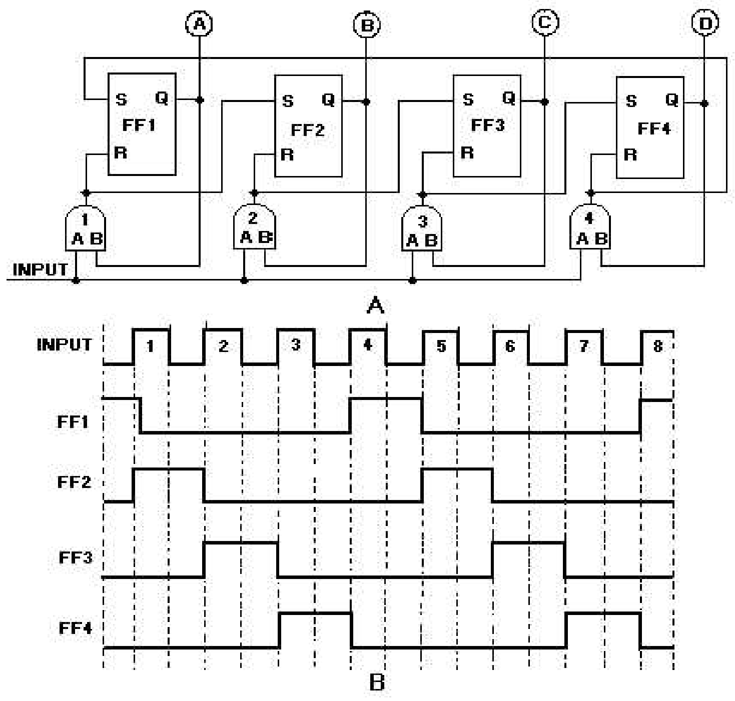
It is a counter that operates as a normal counter until it reaches a count of 1010_2, or 10_{10}. At that time, both inputs to the NAND gate are HIGH, and the output goes LOW. This LOW applied to the CLR input of the flip-flops causes them to reset to 0 (CLR overrides any existing condition of the flip-flop). Once the flip-flops are reset, the count may begin again. The following table shows the binary count and the inputs and outputs of the NAND gate for each count of the decade counter:

Binary count		NAND Gate Inputs	NAND Gate Outputs
*******	A	B	*******
0000	0	0	1
0001	0	0	1
0010	1	0	1
0011	1	0	1
0100	0	0	1
0101	0	0	1
0110	1	0	1
0111	1	0	1
1000	0	1	1
1001	0	1	1
1010	1	1	0

Changing the inputs to the NAND gate can cause the maximum count to be changed. For instance, if FF4 and FF3 were wired to the NAND gate, the counter would count to 1100_2 (12_{10}), and then reset**.**

Ring Counter

A ring counter is defined as a loop of bistable devices (flip-flops) interconnected in such a manner that only one of the devices may be in a specified state at one time. If the specified condition is HIGH, then only one device may be HIGH at one time. As the clock, or input, signal is received, the specified state will shift to the next device at a rate of 1 shift per clock, or input, pulse. Figure (A) below shows a typical four-stage ring counter. This particular counter is composed of R-S flip-flops. J-K flip-flops may be used as well. Notice that the output of each AND gate is input to the R, or reset side, of the nearest FF and to the S, or set side, of the next FF. The Q output of each FF is applied to the B input of the AND gate that is connected to its own R input.

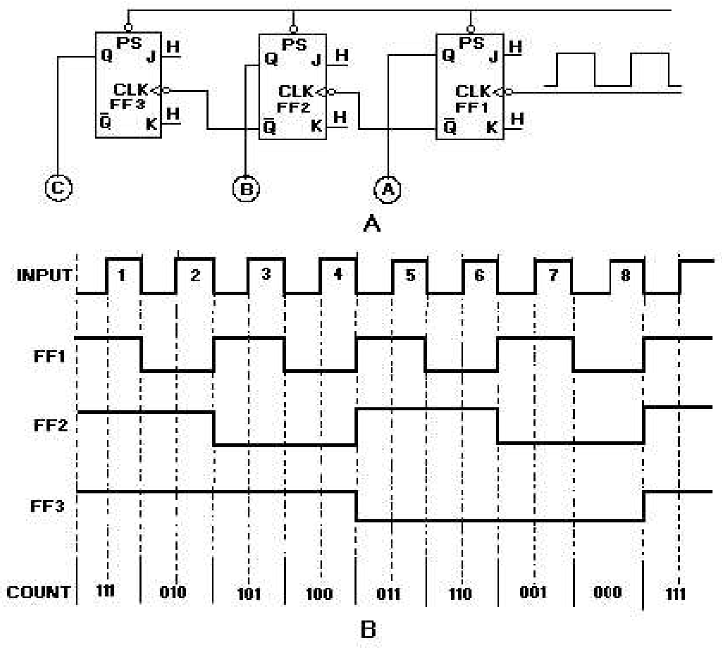


Ring counter: A. Logic diagram; B. Timing diagram.

The circuit input may be normal CLK pulses or pulses from elsewhere in the equipment that would indicate some operation has been completed. Now, let’s look at the circuit operation and observe the signal flow as shown in the figure (B). For an initial condition, let’s assume that the output of FF1 is HIGH and that the input and FF2, FF3, and FF4 are LOW. Under these conditions, lamp A will be lit; and lamps B, C, and D will be extinguished. The HIGH from FF1 is also applied to the B input of AND gate 1. The first input pulse is applied to the A input of each of the AND gates. The B inputs to AND gates 2, 3, and 4 are LOW since the outputs of FF2, FF3, and FF4 are LOW. AND gate 1 now has HIGHs on both inputs and produces a HIGH output. This HIGH simultaneously resets FF1 and sets FF2. Lamp (A) then goes out, and lamp B goes on. We now have a HIGH on AND gate 2 at the B input. We also have a LOW on AND gate 1 at input B. Input pulse 2 will produce a HIGH output from AND gate 2 since AND gate 2 is the only one with HIGHs on both inputs. The HIGH from AND gate 2 causes FF2 to reset and FF3 to set. Indicator B goes out and C goes on. Pulse 3 will cause AND gate 3 to go HIGH. This result in FF3 being reset and FF4 being set. Pulse 4 causes FF4 to reset and FF1 to set, bringing the counter full circle to the initial conditions. As long as the counter is operational, it will continue to light the lamps in sequence — 1, 2, 3, 4; 1, 2, 3, 4, etc. As we stated at the beginning of this section, only one FF may be in the specified condition at one time. The specified condition shifts one position with each input pulse.

Down Counters (counting in reverse)

An up counter starts at 0 and counts to a given number. DOWN counters start at a given number and count down to 0. Up counters are sometimes called INCREMENT counters. Increment means to increase. Down counters are called DECREMENT counters. Decrement means to decrease. A three-stage, ripple down counter is shown in figure (A) below. Notice that the PS (preset) input of the J-K flip-flops is used in this circuit. HIGHs are applied to all the J and K inputs. This enables the flip-flops to toggle on the input pulses.



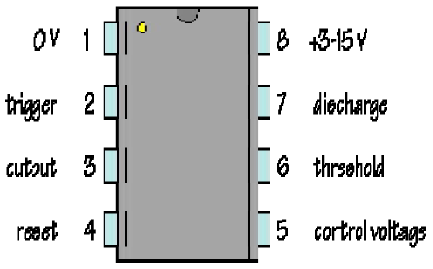
Down counter: A. Logic diagram; B. Timing diagram.

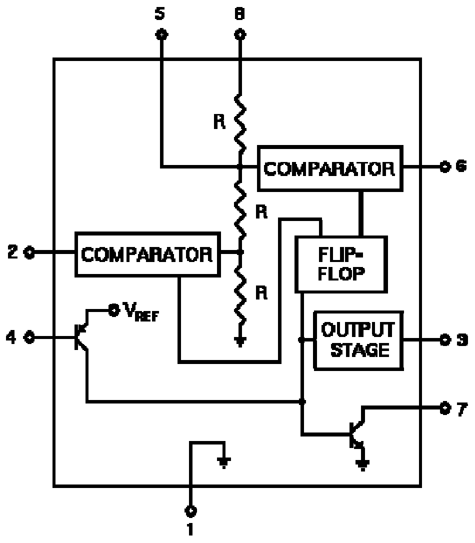
A negative-going pulse is applied to all PS terminals to start the countdown. This causes all the flip-flops to set and also lights indicators A, B, and C. The beginning count is 111_2 (7_{10}). At the same time, LOWs are applied to the CLK inputs of FF2 and FF3, but they do not toggle because the PS overrides any change. All actions in the counter will take place on the negative-going portion of the input pulse. Let’s go through the pulse sequences in figure (B). CP1 causes FF1 to toggle and output Q to go LOW. Lamp A is turned off. Notice that Q goes HIGH but no change occurs in FF2 or FF3. Lamps B and C are now on, A is off, and the indicated count is 110_2 (6_{10}). CP2 toggles FF1 again and lights lamp A. When Q goes HIGH, Q goes LOW. This negative-going signal causes FF2 to toggle and reset. Lamp B is turned off, and a HIGH is felt at the CLK input of FF3. The indicated count is 101_2 (5_{10}); lamps A and C are on, and B is off. At CP3, FF1 toggles and resets. Lamp A is turned off. A positive-going signal is applied to the CLK input of FF2. Lamp B remains off and C remains on. The count at this point is 100_2 (4_{10}). CP4 toggles FF1 and causes it to set, lighting lamp A. Now FF1, output Q, goes LOW causing FF2 to toggle. This causes FF2 to set and lights lamp B. Output of FF2, Q, then goes LOW, which causes FF3 to reset and turn off lamp C. The indicated count is now 011_2 (3_{10}). The next pulse, CP5, turns off lamp (A) but leaves B on. The count is now 010_2. CP6 turns on lamp A and turns off lamp B, for a count of 001_2. CP7 turns off lamp A. Now all the lamps are off, and the counter indicates 000.

On the negative-going signal of CP8, all flip-flops are set, and all the lamps are lighted. The CLK pulse toggles FF1, making output Q go HIGH. As output Q goes LOW, the negative-going signal causes FF2 to toggle. As FF2, output Q, goes HIGH, output Q goes LOW, causing FF3 to toggle and set. As each FF sets its indicator lamp lights. The counter is now ready to again start counting down from 111_2 with the next CLK pulse.

9. The 555 timer IC

The 555 timer IC is a simple versatile device that has been around now for many years and has been used in a number of different technologies. The two primary versions are the **original bipolar design and the more recent CMOS** equivalent. **These differences affect the amount of power they require and their maximum frequency of operation**; they are pin-compatible and functionally interchangeable.





The figure shows the functional block diagram of the 555 timer IC. The IC is available in either an 8-pin round TO3-style can or an 8-pin mini-DIP package. In either case, the pin connections are as follows:

1. Ground, 2. Trigger input, 3. Output, 4. Reset input, 5. Control voltage, 6. Threshold input, 7. Discharge, 8. +V_{CC}. +5 to +15 volts in normal use Ground.

The operation of the 555 timer revolves around the three resistors that form a voltage divider across the power supply, and the two comparators connected to this voltage divider. The IC is **quiescent** so long as the trigger input **(pin 2) remains at +V_{CC}** and the threshold input **(pin 6) is at ground**. Assume the reset input **(pin 4) is also at +V_{CC}** and therefore inactive, and that the control voltage input **(pin 5) is unconnected**. Under these conditions, the output (pin 3) is at ground and the discharge transistor (pin 7) is turned on, thus grounding whatever is connected to this pin.

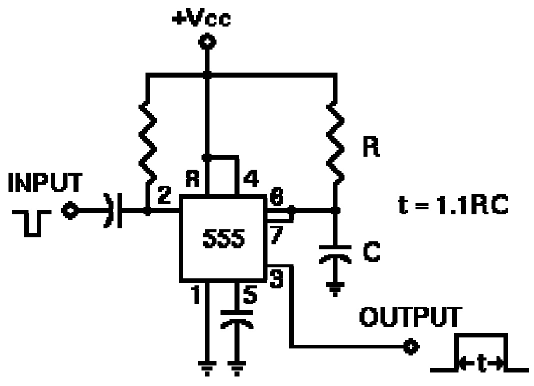
The three resistors in the voltage divider all have the same value (5K in the bipolar version of this IC), so the comparator reference voltages are 1/3 and 2/3 of the supply voltage, whatever that may be. The control voltage input at pin 5 can directly affect this relationship, although most of the time this pin is unused.

The internal flip-flop changes state when the trigger input at pin 2 is pulled down below +V_{CC}/3. When this occurs, the output (pin 3) changes state to +V_{CC} and the discharge transistor (pin 7) is turned off. The trigger input can now return to +V_{CC}; it will not affect the state of the IC.

However, if the threshold input **(pin 6)** is now raised **above (2/3) +V_{CC}**, the output will return to ground and the **discharge transistor will be turned on again**. When the threshold input returns to ground, the IC will remain in this state, which was the original state when we started this analysis.

The easiest way to allow the threshold voltage (pin 6) to gradually rise to (2/3) +V_{CC} is to connect it to a capacitor being allowed to charge through a resistor. In this way we can adjust the R and C values for almost any time interval we might want.

The 555 can operate in either monostable or astable mode, depending on the connections to and the arrangement of the external components. Thus, it can either produce a single pulse when triggered, or it can produce a continuous pulse train as long as it remains powered.



In monostable mode, the timing interval, **t**, is set by a single resistor and capacitor, as shown to above. Both the threshold input and the discharge transistor (pins 6 & 7) are connected directly to the capacitor, while the trigger input is held at +V_{CC} through a resistor. **In the absence of any input, the output at pin 3 remains low** and the discharge transistor prevents capacitor C from charging.

When an input pulse arrives, it is capacitively coupled to pin 2, the trigger input. The pulse can be either polarity; its falling edge will trigger the 555. At this point, the output rises to +V_{CC} and the discharge transistor turns off. Capacitor C charges through R towards +V_{CC}. During this interval, additional pulses received at pin 2 will have no effect on circuit operation. The standard equation for a charging capacitor applies here:

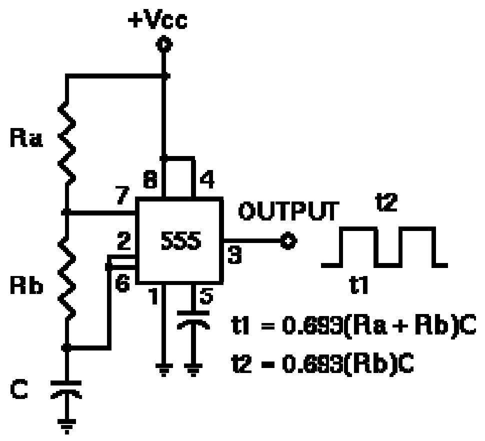
$$e = E\,(1 - \varepsilon^{(-t/RC)}).$$

Here, "e" is the capacitor voltage at some instant in time, "E" is the supply voltage, V_{CC}, and "ε" is the base for natural logarithms, approximately 2.718. The value "t" denotes the time that has passed, in seconds, since the capacitor started charging. The capacitor will charge until its voltage reaches (2/3) +V_{CC}, whatever that voltage may be. This doesn't give us absolute values for "e" or "E," but it does give us the ratio e/E = 2/3. We can use this to compute the time, t, required to charge capacitor C to the voltage that will activate the threshold comparator:

$$2/3 = 1 - \varepsilon^{(-t/RC)}$$
$$-1/3 = -\varepsilon^{(-t/RC)}$$
$$1/3 = \varepsilon^{(-t/RC)}$$
$$\mathrm{Ln}\,(1/3) = -t/RC$$
$$-1.0986123 = -t/RC$$
$$t = 1.0986123RC$$
$$t = 1.1RC$$

The values of R and C must be given in Ohms and Farads, respectively, and the time will be in seconds. You can scale the values as needed and appropriate for your application, provided you keep proper track of your powers of 10. For example, if you specify R in mega-ohms and C in microfarads, t will still be in seconds. But if you specify R in kilo-ohms and C in microfarads, t will be in milliseconds. It's not difficult to keep track of this, but you must be sure to do it accurately in order to correctly calculate the component values you need for any given time interval.

The timing interval is completed when the capacitor voltage reaches the (2/3) $+V_{CC}$ upper threshold as monitored at pin 6. When this threshold voltage is reached, the output at pin 3 goes low again, the discharge capacitor (pin 7) is turned on, and the capacitor rapidly discharges back to ground once more. The circuit is now ready to be triggered once again.



If we rearrange the circuit slightly so that both the trigger and threshold inputs are controlled by the capacitor voltage, we can cause the 555 to self trigger repeatedly. In this case, we need two resistors in the capacitor charging path so that one of them can also be in the capacitor discharge path. This gives us the circuit shown above (astable). In this mode, the initial pulse when power is first applied is a bit longer than the others, having a duration of 1.1(Ra + Rb) C. However, from then on, the capacitor alternately charges and discharges between the two comparator threshold voltages. When charging, C starts at (1/3)V_{CC} and charges towards V_{CC}. However, it is interrupted exactly halfway there, at (2/3)V_{CC}. Therefore, the charging time, t_1, is

-ln (1/2)(Ra + Rb)C = 0.693(Ra + Rb)C.

When the capacitor voltage reaches (2/3)V_{CC}, the discharge transistor is enabled (pin 7), and this point in the circuit becomes grounded. Capacitor C now discharges through Rb alone. Starting at (2/3)V_{CC}, it discharges towards ground, but again is interrupted halfway there, at (1/3)V_{CC}. The discharge time,

t_2, then, is –ln(1/2)(Rb)C = 0.693(Rb)C.

The total period of the pulse train is t1 + t2, or 0.693(Ra + 2Rb)C. The output frequency of this circuit is the inverse of the period, or:

1.44/(Ra + 2Rb)C.

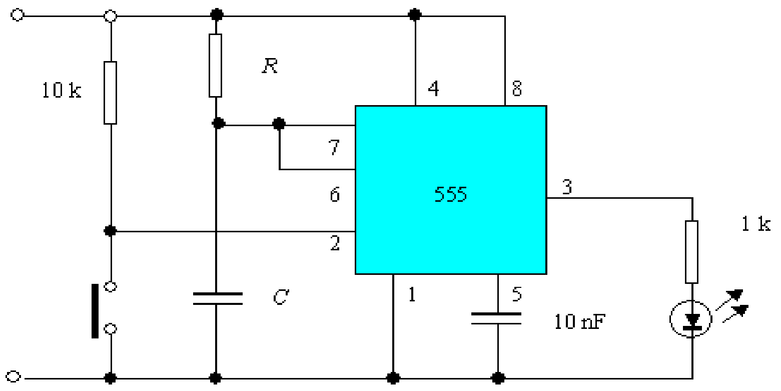
Note that the **duty cycle of the 555 timer circuit in astable mode cannot reach 50%.** On time must always be longer than off time, because Ra must have a resistance value greater than zero to prevent the discharge transistor from directly shorting V_{CC} to ground. Such an action would immediately destroy the 555 IC.

One interesting and very useful feature of the 555 timer in either mode is that the timing interval for either charge or discharge is independent of the supply voltage, V_{CC}. This is because the same V_{CC} is used both as the charging voltage and as the basis of the reference voltages for the two comparators inside the 555. Thus, the timing equations above depend only on the values for R and C in either operating mode.

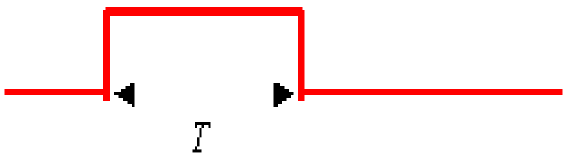
In addition, since all three of the internal resistors used to make up the reference voltage divider are manufactured next to each other on the same chip at the same time, they are as nearly identical as can be. Therefore, changes in temperature will also have very little effect on the timing intervals, provided the external components are temperature stable. A typical commercial 555 timer will show a drift of 50 parts per million per Centigrade degree of temperature change (50 ppm/°C) and 0.01%/Volt change in V_{CC}. This is negligible in most practical applications.

555 timer Applications:

Monostable Mode



The circuit will give a single output pulse like this:



The components *R* and *C* determine the time period *T* of the output pulse.

- When the push switch *S* is closed and released, the voltage at pin 2 goes from high to low to high again.
- This triggers the output to go to high. Pin 7 is also disconnected from zero.
- When the voltage across *C* gets to about 2/3 of the supply voltage, the output goes low.

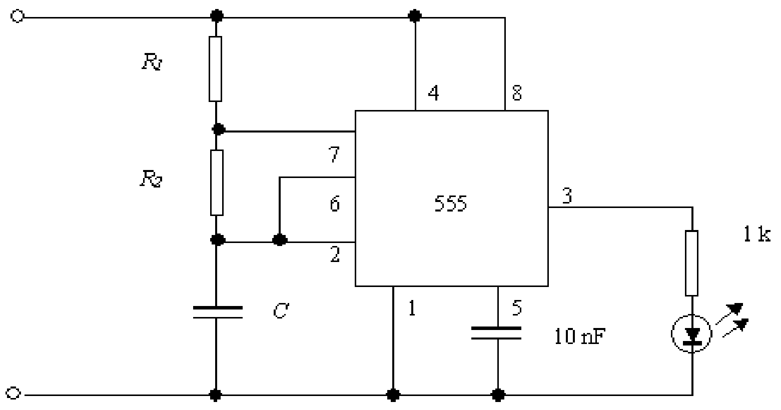
- The period of the pulse is given by simple relationship T » 1.1 RC.
- Once triggered, the circuit cannot be re-triggered to extend the period *T*.

In using a 555-timer we need to be aware of the following points:

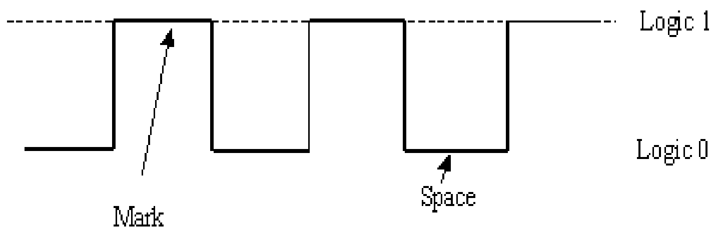
- The trigger period must be less than the output pulse.
- The 5 nF capacitor connected to pin 5 is needed to prevent false triggering.
- The circuit can produce brief dips in the voltage of the supply. This can be countered by placing a large value capacitor across the supply rails. This eliminates the voltage change (called **decoupling**).
- If electrolytic capacitors are used in the RC circuit, leakage currents and poor tolerances can result in the output pulse being greatly at variance.

Astable Mode

The 555 timer can be wired up to produce a train of pulses by ensuring that the circuit is **astable**, which means that it is not in a stable state. We can make astable circuits from other components, but the 555 timer gives a train of digital pulses. The diagram shows the circuit.



The output of the circuit is a square wave, as shown.



We need to consider some definitions:

- The **mark time [**t(H] is the time at which the output is a 1. t(H)= *0.7(*R_1 *+* R_2*)C*
- The **space time** [t(L)] is the time at which the output is a 0. t(L) = *0.7* R_2C
- The **mark to space ratio** = mark time ÷ space time.
- The **astable period** *T* is the time taken for one complete cycle, the mark and the space times added together. *T* = mark + space = t(L) + t(H).
- The **frequency** = 1 ÷ period.

1.4

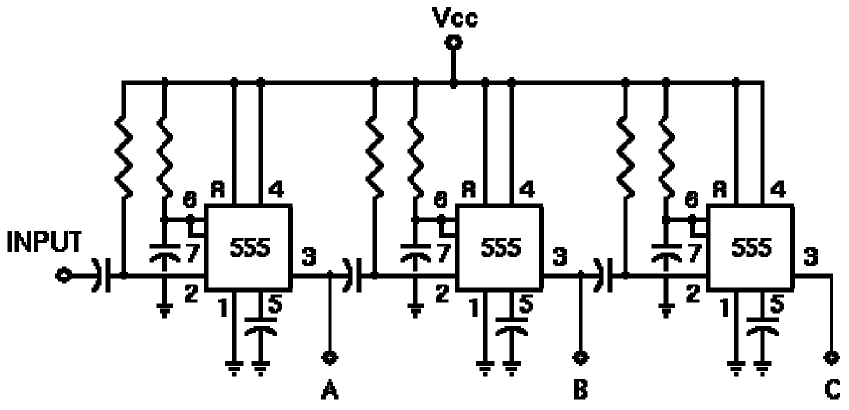
f = --------------

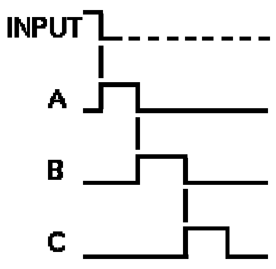
(R_1 *+ 2*R_2*)C*

The time t(H) will be longer than t(L), unless R_1 is very small compared to R_2. If this is the case, then t(H) will be approximately equal to t(L), but not quite equal. We can say to a first approximation that the mark to space ratio is 1. This will result in a square wave output.One requirement in certain digital circuits is for the generation of a series of pulses in time sequence, but on different lines. The pulse widths may or may not be the same, but they must occur one after the other, and therefore cannot come from the same source.

Pulse Sequencer

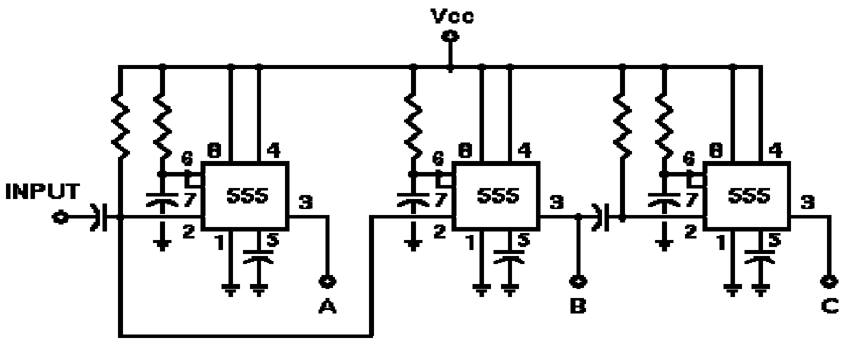
Sometimes pulses are required to overlap, or there must be a set delay following the end of one pulse before another pulse begins. Or, one pulse must begin a set time after another begins. The possible variations in timing requirements are almost endless, and many different approaches have been used to provide pulses with the necessary timing relationships. One inexpensive method that is perfectly satisfactory in many applications is to use interconnected 555 timers to generate the necessary timing intervals.

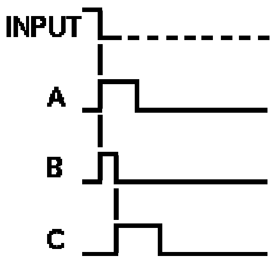




The circuit above; shows three 555 timers, all configured in monostable mode. Each one, from left to right, triggers the next at the end of its timing interval. The resulting pulse timing is shown in the timing diagram underneath it.

The sequence starts with the falling edge of the incoming trigger pulse. That edge triggers timer A, causing output A to go high. At this point, the incoming trigger pulse can either stay low or go high; it is no longer important. Output A will remain high for the duration of its timing interval, and then fall back to its low state. At this time, it triggers timer B. Output B therefore goes high as output A falls. The same thing happens again at the end of the B timing interval; output B falls and triggers timer C. At the end of the C timing interval, the sequence is over and all timers are quiescent, awaiting the arrival of the next triggering signal.





This time, the circuit is wired so that the incoming trigger signal is applied to both timers A and B. However, B's timing interval is very short, so it triggers timer C while the A output is still high. Depending on the designed timing intervals, C can easily be triggered and finish its timing interval while A is still active. Or, C can remain active after A falls back to its quiescent state.

Any number of 555 timers can be sequentially or jointly triggered with this kind of arrangement, and each timer still has its own individually-controlled timing interval. The possible combinations are endless, and independent pulses can overlap or not according to the needs of the application. The initial triggering signal can come from any source. It can even be from another 555 timer, operating in astable mode. In that case, this kind of circuit is self-starting and will operate continuously.

It is also possible to trigger this sort of circuit manually, using a momentary-contact pushbutton to provide the triggering pulse. Or it could be some sort of sensor device, triggering the circuit upon recognition of some external condition. Thus, the circuit can be triggered by any kind of event, just as it can be used to control any kind of circuit. There is one drawback to using daisy-chained 555 timers in this fashion. Once a 555 timer in monostable mode has been triggered, it cannot be re-triggered and the timing interval cannot be changed (at least without the addition of external circuitry to accomplish that). Therefore, if a second trigger pulse arrives while the input timer(s) are still active, it will be ignored. A worse possibility exists for the second example above: If the B interval has been completed but the A interval has not been completed when a new trigger pulse is received, timer B will be triggered but timer A will ignore it. If the timing relationship between the A and C pulses is critical, this could cause problems.

Example

What is the frequency of the square wave output from the circuit?

Use the formula:

$$f = \frac{1.4}{(R_1 + 2R_2)C}$$

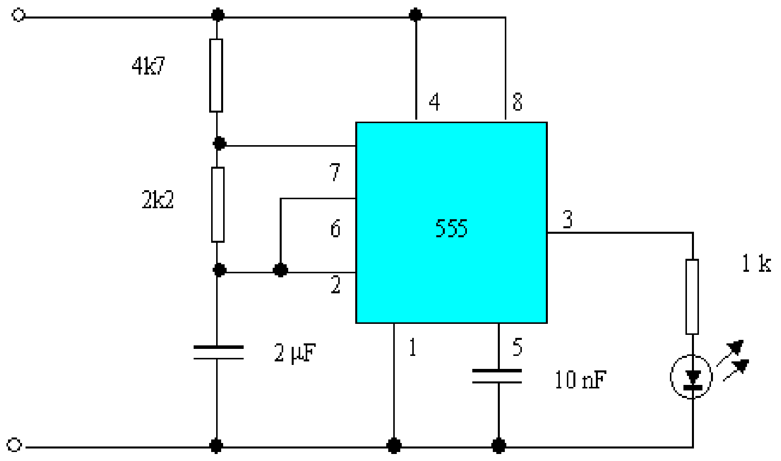


Figure 135 555-timer in astable mode with values

Answer:

From the diagram we can see that:

- $R_1 = 4700\ \Omega$
- $R_2 = 2200\ \Omega$
- $C = 2.0 \cdot 10^{-6}$ F

We need to substitute these values into the formula:

$$f = \frac{1.4}{(R_1 + 2R_2)C} = \frac{1.4}{[4700 + (2 \cdot 2200)] \cdot 2.0 \cdot 10^{-6}}$$

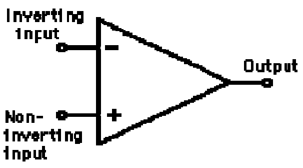
$$= \frac{1.4}{9100 \cdot 2.0 \cdot 10^{-6}} = \mathbf{77\ Hz}$$

Operational Amplifiers

Operational amplifier or "op-amp" refers to a class of high-gain DC coupled amplifiers with two inputs and a single output. The modern integrated circuit version is the famous 741 op-amp. Some of the general characteristics of the IC version are:

- High gain, on the order of a million
- High input impedance, low output impedance
- Used with split supply, usually +/- 15V

Used with feedback, with gain determined by the feedback network



The most common and most famous op-amp is the mA741C or just 741, which is packaged in an 8-pin mini-DIP. The integrated circuit contains 20 transistors and 11 resistors. Introduced by Fairchild in 1968, the 741 and subsequent IC op-amps including FET-input op-amps have become the standard tool for achieving amplification and a host of other tasks. Though it has some practical limitations, the 741 is an electronic bargain at less than a dollar.

The Ideal Op-amp

The IC Op-amp comes so close to ideal performance that it is useful to state the characteristics of an ideal amplifier without regard to what is inside the package.

1. Infinite voltage gain
2. Infinite input impedance
3. Zero output impedance
4. Infinite bandwidth
5. Zero input offset voltage (i.e., exactly zero out if zero in).

Positive Feedback

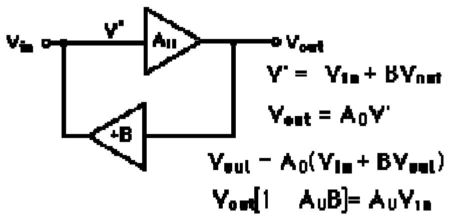
The use of positive feedback is useful for producing <u>oscillators</u>. The condition for positive feedback is that a portion of the output is combined in phase with the input. For an amplifier with positive feedback the gain is given by the expression below. The large <u>open loop gain</u> of an op-amp makes it inevitable that the condition:

$$A_0B = 1$$

will be reached, and the gain expression

$$A_f = \frac{V_{out}}{V_{in}} = \frac{A_0}{1 - A_0B}$$

becomes infinite



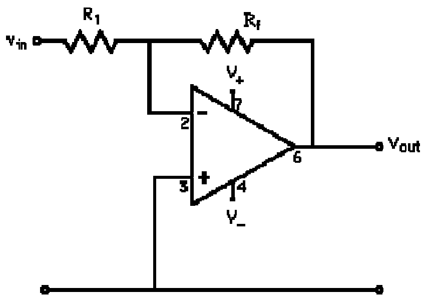
Practically speaking, the gain which applies at low signal amplitudes will be reduced until the output amplitude reaches some constant value. However, that limiting value will be independent of input, allowing the circuit to produce a designed output.

Inverting Amplifier

For an ideal op-amp, the inverting amplifier gain is given simply by

$$\frac{V_{out}}{V_{in}} = -\frac{R_f}{R_1}$$

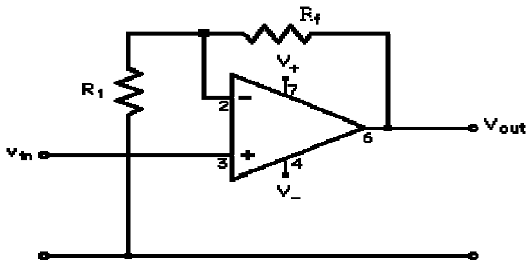
For equal resistors, gain = -1, and is used in digital circuits as an <u>inverting buffer</u>



Non-inverting Amplifier

For an ideal op-amp, the non-inverting amplifier gain is given by

$$\frac{V_{out}}{V_{in}} = 1 + \frac{R_f}{R_1}$$



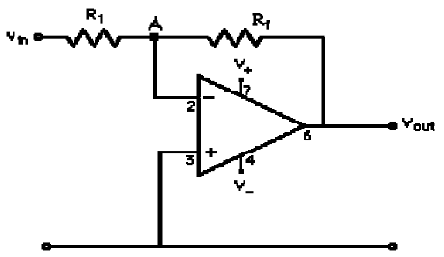
Rules: Inverting Amplifier

The behavior of most configurations of op-amps can be determined by applying the "golden rules". For an inverting amplifier, the current rule tries to drive the current to zero at point A. This requires:

$$\frac{V_{in}}{R_1} = -\frac{V_{out}}{R_f}$$

This gives the voltage amplification

$$\frac{V_{out}}{V_{in}} = -\frac{R_f}{R_1}$$



Rules: Non-inverting Amplifier

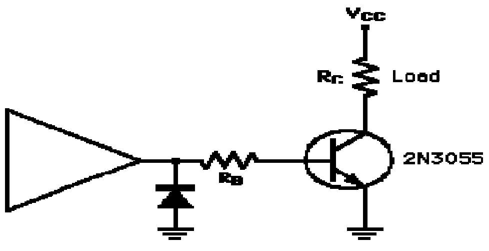
The behavior of most configurations of op-amps can be determined by applying the "golden rules". For an non-inverting amplifier, the current rule tries to drive the current to zero at point A and the voltage rule makes the voltage at A equal to the input voltage. This leads to

$$\frac{V_{in}}{R_1} = \frac{V_{out} - V_{in}}{R_f}$$

and amplification

$$\frac{V_{out}}{V_{in}} = 1 + \frac{R_f}{R_1}$$

An op-amp may be used instead of a mechanical switch to operate a transistor switch. The diode is used for protection of the base-emitter junction in case the op-amp swings to its negative supply voltage.

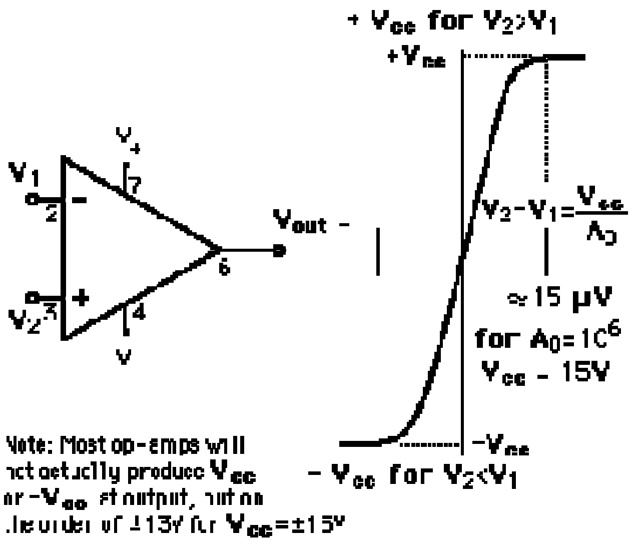


Digital Applications of Operational Amplifiers

The op-amp has so many functional applications that it shows up as part of the supporting cast in many types of circuits. Some of the applications in digital circuits are:

- An op-amp comparator is used in most analog-to-digital converters .
- An op-amp voltage follower can be used as a buffer in logic circuits.
- An op-amp inverting amplifier can be used as an inverting buffer in logic circuits.

Comparator



+ Vcc for V2>V1
+Vcc
V+
V1
7
2
−
+
3
V2
4
V
6
Vout
V2-V1=Vcc/A0
≈15 μV
for A0=10^6
Vcc - 15V
-Vcc
- Vcc for V2<V1
Note: Most op-amps will not actually produce Vcc or -Vcc at output, but on the order of ±13V for Vcc=±15V

The extremely large open-loop gain of an op-amp makes it an extremely sensitive device for comparing its input with zero. For practical purposes, if

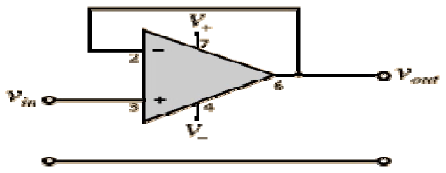
$V_2 > V_1$

the output is driven to the positive supply voltage and if

$V_2 < V_1$

it is driven to the negative supply voltage

Voltage Follower



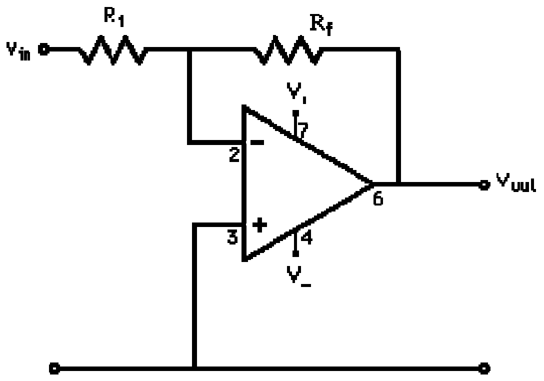
The voltage follower with an ideal op amp gives simply

$$v_{out} = v_{in}$$

but this turns out to be a very useful service, because the input impedance of the op amp is very high, giving effective isolation of the output from the signal source. You draw very little power from the signal source, avoiding "loading" effects. This circuit is a useful first stage.

The voltage follower is often used for the construction of buffers for logic circuits.

Inverting Amplifier



For an ideal op-amp, the inverting amplifier gain is given simply by

$$\frac{V_{out}}{V_{in}} = -\frac{R_f}{R_1}$$

For equal resistors, it has a gain of -1, and is used in digital circuits as an inverting buffer