UNIT - I

Computer Based Medical Instrumentaion – SBMA7006

Introduction

Contents - Generalized Instrumentation system - Feature of personal computer - PC based Instrumentation system – Principle of signal conditioning – Operational Amplifier – instrumentation amplifier – Bridge circuits – Half and Full Bridge Circuit, Filters – Noise reduction techniques.

1.1 GENERALIZED INSTRUMENTATION SYSTEM

The basic need of instrumentation in a process is to get the best and most amount of information so as to successfully complete the process. When referring to the completion of the project with reference to instrumentation, it basically means maximum efficiency with minimum production expense and desired output quality.

The information that is achieved from these processes may be very simple and may mostly involve a direct measurement method. But as the process becomes more complex, direct measurement may seem to be impracticable and so indirect methods must be used for measurements. These methods involve a derived relationship between the measured quantity and the result that is needed.

Most of the indirect methods involve electrical techniques as they have high speed and also simple processing methods. The output from such methods is easier to link to computers.

The obtained information may not necessarily be the direct value of a measured quantity. That is, the value obtained may be a variation of the value with respect to other parameters. It may also be a signal corresponding to the end limit. It could also be a specific value with an indicating hand over a suitable scale. Thus, one instrument may be needed to perform the required operations individually or a number of them at a time.

Basically it can be classified into two types

1. Analog instrumentation system
2. Digital instrumentation system

Analog instrumentation system

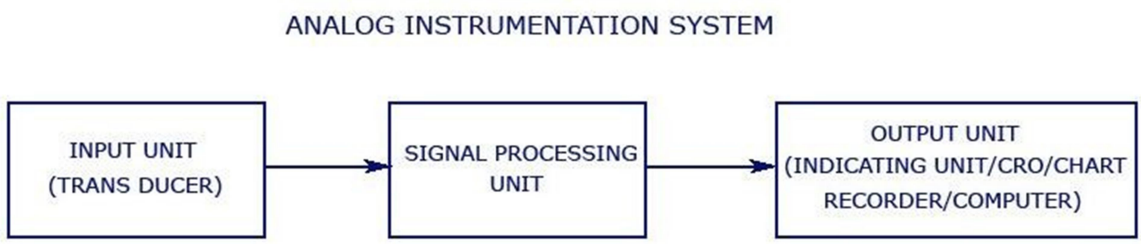


Figure 1.1 Analog Instrumentation Systems

The Primary Element/Transducer

The input receives the quantity whose value is to be measured and is converted into its proportional incremental electrical signal such as voltage, current, resis tance change, inductance or even capacitance. Thus, the changed variable contains the information of the measured variable. Such a functional element or device is called a transducer.

The Secondary Element/Signal Processing Unit

The output of the transducer is provided to the input of the signal processing unit. This unit amplifies the weak transducer output and is filtered and modified to a form that is acceptable by the output unit. Thus this unit may have devices like: amplifiers, filters, analog to digital converters, and soon.

The Final Element/Output Unit

The output from the signal processing unit is fed to the input of the output unit. The output unit measures the signal and indicates the value to the reader. The indication may be either through: an indicating instrument, a CRO, digital computer, and so on.

Digital instrumentation system

All the functional units that were used in an analog system will also be used here. He basic operation in a digital system includes the handling of analog signals, making the measurements, converting and handling digital data, programming and a lso control.
The block diagram and functional units are given below

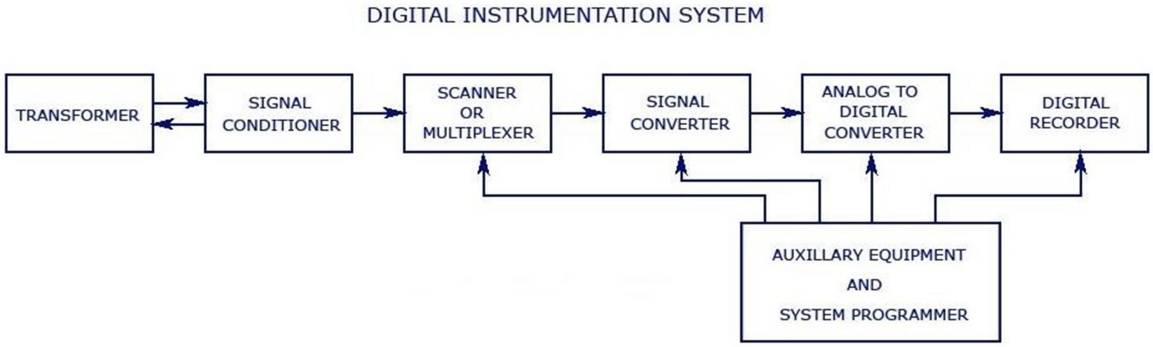


Figure 1.2 Digital Instrumentation system

Transducer

All the physical input parameters like temperature, pressure, displacement, velocity, acceleration and so on will be converted into its proportionate electrical signal.

Signal Conditioning Unit

This working of this unit is exactly the same as that of a signal processing unit in an analog instrumentation system. It includes all the balancing circuits ad calibrating elements along with it.

Scanner/Multiplexer

Multiple analog signals are received by this device and are sequentially provided on to a measuring instrument.

Signal Converter
It is used to convert an analog signal to a form that is acceptable by the analog to digital converter.

Analog to (A-D) Digital Converter
The analog signal is converted into its proportional digital signal. The output of an A- D converter is given to a digital display.

Auxiliary Equipment
All the system programming and digital data processing functions are carried out by this unit. The auxiliary equipment may be a single computer or may be a collection of individual instruments. Some of its basic functions include linearizing and limit comparison.

Digital Recorder
It is mostly a CRO or a computer.

1.2 FEATURES OF A PERSONAL COMPUTER

The basic components of IBM PC are system units, monitor, keyboard mouse, hard disk drive, floppy disk drive and printer. The visual part of a PC is the mother board which is housed in the system unit. It embodies a microprocessor and a few more supporting chips such as the RAM, EPROM, floppy disk controller, DMA controller,Priority interrupt controller, timer, etc.

Basically, a PC is a powerful data processing tool. Its data processing capabilities tend to double or triple with new PCs introduced every year.

Expansion slots:
For adding additional fearture to the PC the motherboard have various types of empty expansion slots and bring out the system bus signals through the slots for interfacing additional devices directly to the system.

Ports:
The basic configuration of a latest desktop PC provides a parallel port, serial port, upto six USB ports, mouse ports for connecting peripherals and additional devices for the PC . Usually the printer is connected to the parallel port, A modem is connected to a serial port, the keyboard is connected to PS/2 keyboard port.

Monitor:
One of the important feature of a pc is its monitor, A high resolution graphic display system. Computer monitors are the most convenient forms of display and are operated in texts or graphics mode.

Storage devices:
The PC have mass storage devices and can store various programs and huge amount of data on a hard disk or a floppy disk. The PC execute the programs by downloading them from a hard disk or a floppy disk into the memory.

Software:
Only the operating system makes the PC functional. Many operating system are available today for the PC. Windows 98/2000/XP/NT, UNIX, LINUX. An enormous amount of software development tools are available for designing applications for the PC. TurboC, Turbo C++, BASICS.

1.3 PC BASED INSTRUMENTATION SYSTEM

The instrumentation system have evolved over a period of time**.** they are in standalone and in modular types. Traditionally, the measurement and control are done using standalone instruments. The measurment and control are automated by exchanging the data b/w instruments & computers. The generalized block diagram is shown in below

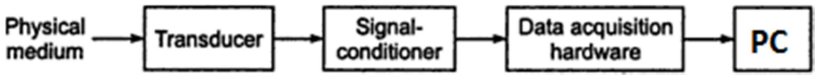


Figure 1.3 PC based instrumentation system

The design of PC based measuring system involves

(i) **Design of a data acquisition hardware, and**
(ii) **Design of appropriate application software.**

Data acquisition includes

1. A/D
2. D/A
3. DIGITAL I/O
4. TIMING I/O

Data acquisition is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer. Data acquisition systems (abbreviated with the acronym **DAS** or **DAQ**) typically convert analog waveforms into digital values for processing.

PC Interfaces:
Hardware interfaces exist in many of the components such as the various buses, storage devices, other I/O devices, etc. A hardware interface is described by the mechanical, electrical and logical signals at the interface and the protocol for sequencing them (sometimes called signalling). A standard interface, such as SCSI, decouples the design and introduction of computing hardware, such as I/O devices, from the design and introduction of other components of a computing system, thereby allowing users and manufacturers great flexibility in the implementation of computing systems. Hardware interfaces can be parallel with
several electrical connections carrying parts of the data simultaneously, or serial where data is sent one bit at a time.

A key principle of design is to prohibit access to all resources by default, allowing access only through well-defined entry points, i.e. interfaces. Software interfaces provide access to computer resources (such as memory, CPU, storage, etc.) of the underlying computer system; direct access (i.e. not through well designed interfaces) to such resources by software can

have major ramifications—sometimes disastrous ones—for functionality and stability.

Interfaces between software components can provide: constants, data types, types of procedures, exception specifications and method signatures. Sometimes, public variables are also defined as part of an interface.

An interface is hence a type definition; anywhere an object can be exchanged (for example, in a function or method call) the *type* of the object to be exchanged can be defined in terms of its interface rather than specifying a particular class. This means that any class that implements that interface can be used. For example, a dummy implementation may be used to allow development to progress before the final implementation is available. In another case, a fake or mock implementation may be substituted during testing. Such stub implementations are replaced by real code later in the development process.

1.4 SIGNAL CONDITIONING

Principle:

In control engineering applications, it is common to have a sensing stage (which consists of a sensor), a signal conditioning stage (where usually amplification of the signal is done) and a processing stage (normally carried out by an ADC and a micro- controller). Operational amplifiers (op-amps) are commonly employed to carry out the amplification of the signal in the signal conditioning stage.

Signal conditioning can include amplification, filtering, converting, range matching, isolation and any other processes required to make sensor output suitable for processing after conditioning.

Filtering

Filtering is the most common signal conditioning function, as usually not all the signal frequency spectrum contains valid data. The common example is 60 Hz AC power lines, present in most environments, which will produce noise if amplified.

Amplifying

Signal amplification performs two important functions: increases the resolution of the input signal, and increases its signal-to- noise ratio. For example, the output of an electronic temperature sensor, which is probably in the millivolts range is probably too low for an analog-to-digital converter (ADC) to process directly. In this case it is necessary to bring the voltage level up to that required by the ADC.
Commonly used amplifiers on signal on conditioning include sample and hold amplifiers, peak detectors, log amplifiers, antilog amplifiers, instrumentation amplifiers and programmable gain amplifiers.

Isolation

Signal isolation must be used in order to pass the signal from the source to the measurement device without a physical connection: it is often used to isolate possible sources of signal perturbations. Also notable is that it is important to isolate the potentially expensive equipment used to process the signal after conditioning from the sensor.

Magnetic or optic isolation can be used. Magnetic isolation transforms the signal from voltage to a magnetic field, allowing the signal to be transmitted without a physical

connection (for example, using a transformer). Optic isolation takes an electronic signal and modulates it to a signal coded by light transmission (optical encoding), which is then used for input for the next stage of processing.

1.5 OPERATIONAL AMPLIFIERS:

An **operational amplifier** ("op-amp") is a DC-coupled high-gain electronic voltage amplifier with a differential input and, usually, a single-ended output. In this configuration, an op-amp produces an output potential (relative to circuit ground) that is typically hundreds of thousands of times larger than the potential difference between its input terminals.

Operational amplifiers had their origins in analog computers, where they were used to do mathematical operations in many linear, non- linear and frequency-dependent circuits. The popularity of the op-amp as a building block in analog circuits is due to its versatility. Due to negative feedback, the characteristics of an op-amp circuit, its gain, input and output impedance, bandwidth etc. are determined by external components and have little dependence on temperature coefficients or manufacturing variations in the op-amp itself.

Op-amps are among the most widely used electronic devices today, being used in a vast array of consumer, industrial, and scientific devices. Many standard IC op-amps cost only a few cents in moderate production volume; however some integrated or hybrid operational amplifiers with special performance specifications may cost over
$100 US in small quantities. Op-amps may be packaged as components, or used as elements of more complex integrated circuits.

Symbol

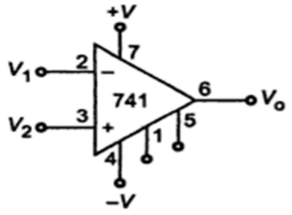


Figure 1.4 Symbol

When an op-amp operates in linear (i.e., not saturated) mode, the difference in voltage between the non- inverting (+) pin and the inverting (−) pin is negligibly small.The input impedance between (+) and (−) pins is much larger than other resistances in the circuit.

The input signal V_{in} appears at both (+) and (−) pins, resulting in a current i through R_g equal to V_{in}/R_g.

$$i = \frac{V_{in}}{R_g}$$

Since K irchhoff's current law states that the same current must leave a node as enter it, and since the impedance into the (−) pin is near infinity, we can assume practically all of the same current i flows through R_f, creating an output voltage

$$V_{\text{out}} = V_{\text{in}} + i \times R_f = V_{\text{in}} + \left(\frac{V_{\text{in}}}{R_g} \times R_f\right) = V_{\text{in}} + \frac{V_{\text{in}} \times R_f}{R_g} = V_{\text{in}}\left(1 + \frac{R_f}{R_g}\right)$$

By combining terms, we determine the closed- loop gain A_{CL}:

$$A_{\text{CL}} = \frac{V_{\text{out}}}{V_{\text{in}}} = 1 + \frac{R_f}{R_g}$$

Ideal op-amps

An equivalent circuit of an operational amplifier that models some resistive non-ideal parameters. An ideal op-amp is usually considered to have the following properties:

- Infinite open- loop gain G = v_{out} / 'v_{in}
- Infinite input impedance R_{in}, and so zero input current
- Zero input offset voltage
- Infinite voltage range available at the output
- Infinite bandwidth with zero phase shift and infinite slew rate
- Zero output impedance R_{out}
- Zero noise
- Infinite Common-mode rejection ratio (CMRR)
- Infinite Power supply rejection ratio.

AMPLIFIERS

Differential amplifier (difference amplifier)

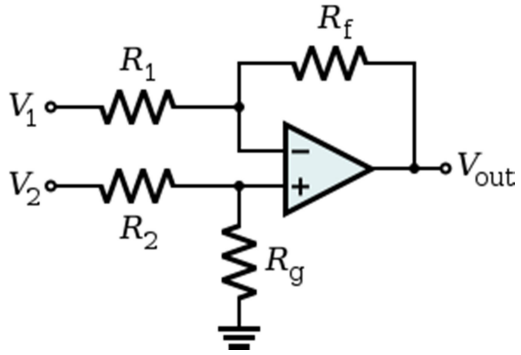


Figure 1.5 differential amplifier

Amplifies the difference in voltage between its inputs.

The name "differential amplifier" must not be confused with the "differentiator," which is also shown on this page.

The "instrumentation amplifier," which is also shown on this page, is a modification of the differential amplifier that also provides high input impedance.

The circuit shown computes the difference of two voltages, multiplied by some gain factor. The output voltage:

$$V_{\text{out}} = \frac{(R_{\text{f}} + R_1)\,R_{\text{g}}}{(R_{\text{g}} + R_2)\,R_1} V_2 - \frac{R_{\text{f}}}{R_1} V_1 = \left(\frac{R_1 + R_{\text{f}}}{R_1}\right) \cdot \left(\frac{R_{\text{g}}}{R_{\text{g}} + R_2}\right) V_2 - \frac{R_{\text{f}}}{R_1} V_1 .$$

Or, expressed as a function of the common mode input V_{com} and difference input V_{dif}

$$V_{com} = (V_1 + V_2)/2; V_{dif} = V_2 - V_1 \,,$$

the output voltage is

$$V_{out}\frac{R_1}{R_f} = V_{com}\frac{R_1/R_f - R_2/R_g}{1 + R_2/R_g} + V_{dif}\frac{1 + (R_2/R_g + R_1/R_f)/2}{1 + R_2/R_g}.$$

In order for this circuit to produce a signal proportional to the voltage difference of the input terminals, the coefficient of the V_{com} term (the common- mode gain) must be zero, or

$$R_1/R_f = R_2/R_g$$

With this constraint in place, the common- mode rejection ratio of this circuit is infinitely large, and the output

$$V_{out} = \frac{R_f}{R_1}V_{dif} = \frac{R_f}{R_1}(V_2 - V_1)\,.$$
$$V_{out} = V_2 - V_1.$$

where the simple expression R_f / R_1 represents the closed-loop gain of the differential amplifier.The special case when the closed- loop gain is unity is a differential follower, with:

Inverting amplifier

An inverting amplifier is a special case of the differential amplifier in which that circuit's non- inverting input V_2 is grounded, and inverting input V_1 is identified with V_{in} above. The closed-loop gain is R_f / R_{in}, hence

$$V_{out} = -\frac{R_f}{R_{in}}V_{in}\,.$$

The simplified circuit above is like the differential amplifier in the limit of R_2 and R_g very small. In this case, though, the circuit will be susceptible to input bias current drift because of the mismatch between R_f and R_{in}.To intuitively see the gain equation above, calculate the current in R_{in}:

$$i_{in} = \frac{V_{in}}{R_{in}}$$

then recall that this same current must be passing through R_f, therefore (because $V_- = V_+ = 0$):

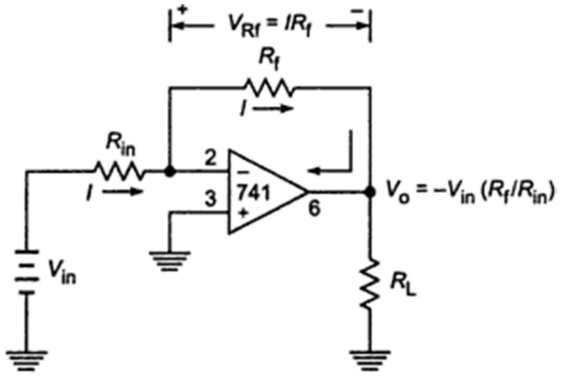


Figure 1.6 Inverting amplifier

A mechanical analogy is a seesaw, with the V_- node (between R_{in} and R_f) as the fulcrum, at ground potential. V_{in} is at a length R_{in} from the fulcrum; V_{out} is at a length R_f. When V_{in} descends "below ground", the output V_{out} rises proportionately to balance the seesaw, and *vice versa*.

Non-inverting amplifier

A non- inverting amplifier is a special case of the differential amplifier in which that circuit's inverting input V_1 is grounded, and non- inverting input V_2 is identified with V_{in} above, with $R_g \gg R_2$. Referring to the circuit immediately above,

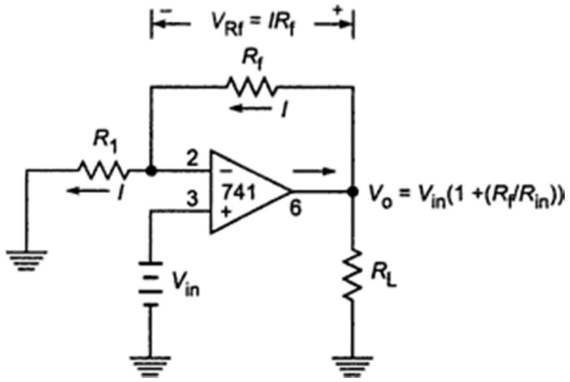


Figure 1.7 Non inverting amplifier

$$V_{\text{out}} = \left(1 + \frac{R_2}{R_1}\right) V_{\text{in}}.$$

To intuitively see this gain equation, use the virtual ground technique to calculate the current in resistor R_1:

$$i_1 = \frac{V_{\text{in}}}{R_1},$$

then recall that this same current must be passing through R_2, therefore:

$$V_{\text{out}} = V_{\text{in}} + i_{\text{in}} R_2 = V_{\text{in}} \left(1 + \frac{R_2}{R_1}\right)$$

A mechanical analogy is a class-2 lever, with one terminal of R_1 as the fulcrum, at ground potential. V_{in} is at a length R_1 from the fulcrum; V_{out} is at a length R_2 further along. When V_{in}ascends "above ground", the output V_{out} rises proportionately with the lever.The input impedance of the simplified non- inverting amplifier is high, of order $R_{dif} \times A_{OL}$ times the closed- loop gain, where R_{dif} is the op amp's input impedance to differential signals, and $A_{OL}$is the open- loop voltage gain of the op amp; in the case of the ideal op amp, with A_{OL} infinite and R_{dif} infinite, the input impedance is infinite. In this case, though, the circuit will be susceptible to input bias current drift because of the mismatch between the impedances driving the V_+ and V_- op amp inputs.

1.6 INSTRUMENTATION AMPLIFIER:

Combines very high input impedance, high common- mode rejection, low DC offset, and other properties used in making very accurate, low- noise measurements Is made by adding a non- inverting buffer to each input of the differential amplifier to increase the input impedance.

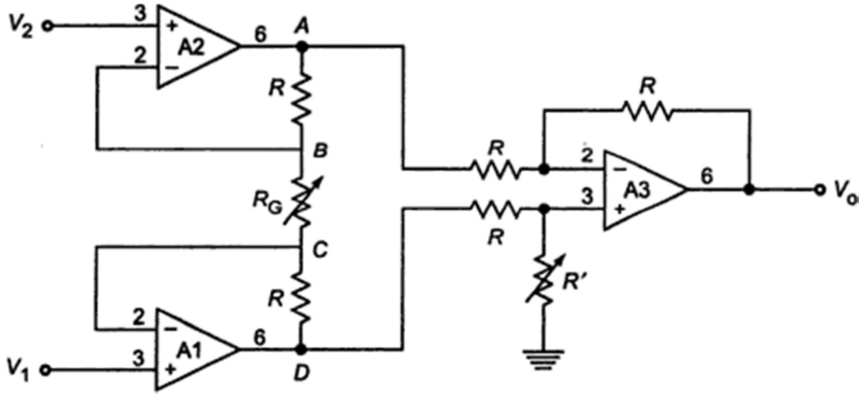


Figure 1.7 Instrumentation amplifier

$$I = \frac{(V_1 - V_2)}{R_G}$$

The same current flows through the two R resistors, which are in series with R_G. It makes the point D to be at voltage $\left\{(V_1 - V_2) \times \left(\frac{R}{R_G}\right)\right\}$ greater than at point C and the point A to be at voltage $\left\{(V_1 - V_2) \times \left(\frac{R}{R_G}\right)\right\}$ less than at point B. Total voltage difference between points D and A is

$$(V_1 - V_2) \times \frac{R}{R_G} + (V_1 - V_2) + (V_1 - V_2) \times \left(\frac{R}{R_G}\right) = (V_1 - V_2)\left(1 + \left(2 \times \frac{R}{R_G}\right)\right)$$

$$= (V_1 - V_2)\left(1 + \frac{2}{a}\right) \qquad \text{where, } a = \frac{R_G}{R} \qquad V_o = (V_1 - V_2)\left(1 + \frac{2}{a}\right)$$

$$\therefore A = \frac{V_o}{(V_1 - V_2)} = \left(1 + \frac{2}{a}\right)$$

BRIDGE CIRCUITS

A **bridge circuit** is a type of electrical circuit in which two circuit branches (usually in parallel with each other) are "bridged" by a third branch connected between the first two branches at some intermediate point along them. The bridge was originally developed for laboratory measurement purposes and one of the intermediate bridging points is often adjustable when so used. Bridge circuits now find many applications, both linear and non-linear, including in instrumentation, filtering and power conversion

The best-known bridge circuit, the Wheatstone bridge, was invented by Samuel Hunter Christie and popularized by Charles Wheatstone, and is used for measuring resistance. It is constructed from four resistors, two of known values R_1 and R_3 (see diagram), one whose resistance is to be determined R_x, and one which is variable and calibrated R_2.

Two opposite vertices are connected to a source of electric current, such as a battery, and a galvanometer is connected across the other two vertices. The variable resistor is adjusted until the galvanometer reads zero. It is then known that the ratio between the variable resistor and its neighbour R1 is equal
to the ratio between the unknown resistor and its neighbour R3, which enables the value of the unknown resistor to be calculated.

$$V_o = V_A - V_B \qquad V_A = \frac{(V_{ex}\, R_1)}{(R_4 + R_1)} \qquad \frac{V_o}{V_{ex}} = \frac{(R_1 R_3 - R_2 R_4)}{\{(R_4 + R_1)(R_3 + R_2)\}}$$

$$V_B = \frac{(V_{ex}\, R_2)}{(R_3 + R_2)}$$

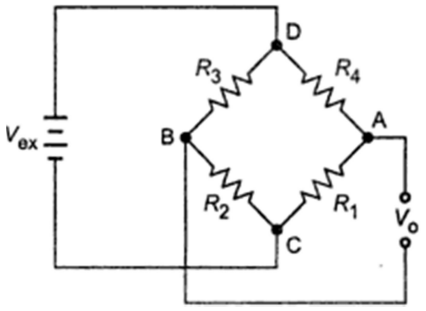


Figure 1.8 Basis bridge circuit

1.7 BRIDGE AMPLIFIER:

This is the most commonly misunderstood mode of operation and it requires additional circuitry to implement if the pair of amplifiers does not have the facility built in. The image shows two identical amplifiers A1 and A2 connected in bridge mode. The signals presented to each amplifier of the pair are caused to be in anti- phase. In other words, as the signal in one amplifier is swinging positively, the signal in the other is swinging negatively. If, for example the maximum output voltage swing of each amplifier is between a peak of + and – 10 volts, when the output of one amplifier is at + 10 volts the output of the other will be at –10 volts, which means that the load (a loudspeaker) now sees a 20 volt peak difference between the ‑hot‖ (normally red) output terminals. The inverting & instrumentation bridge amplifie rs are shown in below.

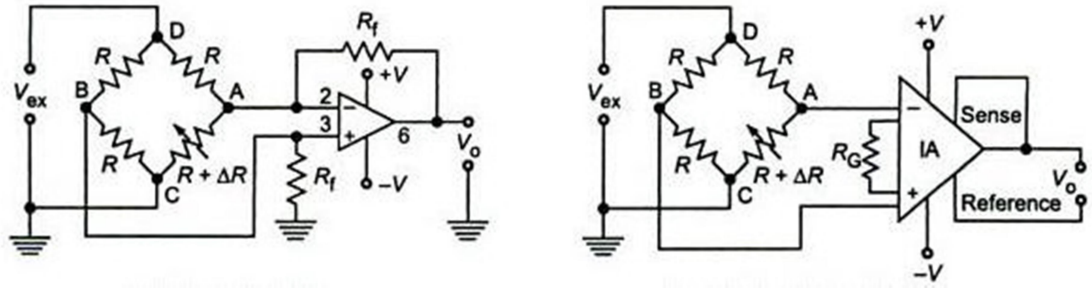


Figure 1.9 Bridge amplifier

1.8 FILTERS

The purpose of the filter as a part of signal conditioning in measurement system is to eliminate the noise or undesired frequencies from the measurement signals without altering the desired information.Basically it can be classified into two categories,

RC filters Active filters

RC FILTERS

Simple filters can be designed using passive elements like resistors and capacitors.In this section we will discuss about the simple RC filter sections.

Low pass filter:

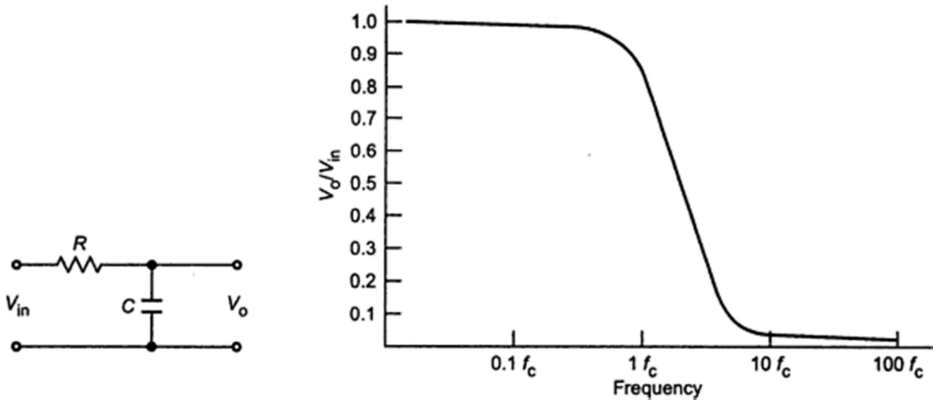


Figure 1.10 Low pass filter

$$V_o = V_{in}\frac{(1/j\omega C)}{\{R+(1/j\omega C)\}} = \frac{V_{in}}{(1+j\omega RC)} \quad \left(\frac{V_o}{V_{in}}\right) = \frac{1}{(1+j2\pi fRC)}$$

$$\left(\frac{V_o}{V_{in}}\right) = \frac{1}{\{1+j(f/f_c)\}}$$

High pass filter:

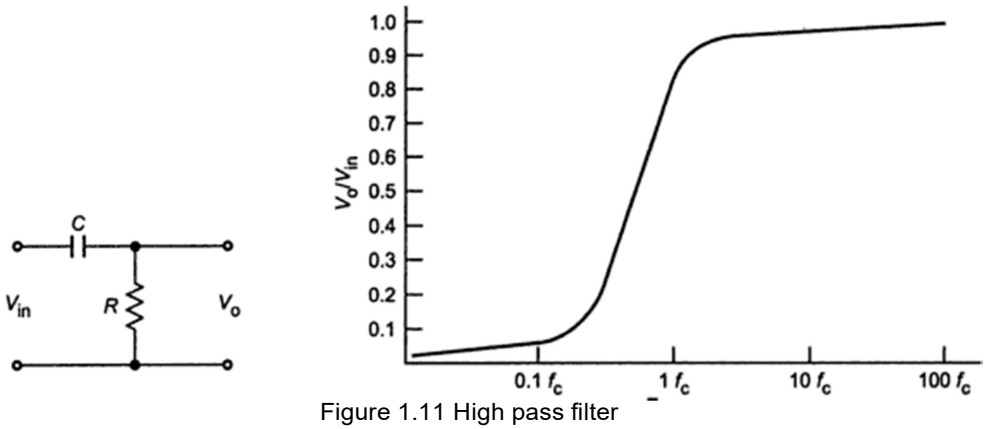


Figure 1.11 High pass filter

$$V_o = \frac{V_{in}(R)}{\{R + (1/j\omega C)\}} \qquad \left(\frac{V_o}{V_{in}}\right) = \frac{j(f/f_c)}{\{1 + j(f/f_c)\}}$$

ACTIVE FILTERS:

Active filters are designed using opamp, resistance and Capacitance. There are four types of filters are existing.

Active Low pass filters: Which can be used to remove low frequency components

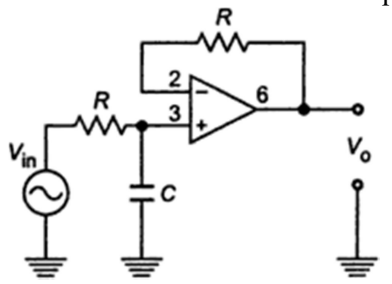
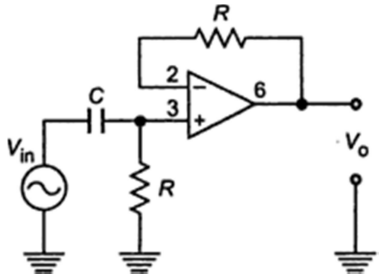


Figure 1.12 : Active High Pass filter

$$\left|\frac{V_o}{V_{in}}\right| = \frac{1}{\{1 + (f/f_c)^2\}^{1/2}} \qquad \phi = -\tan^{-1}\left(\frac{f}{f_c}\right)$$

Active High pass filter: Which can be used to remove High frequency components

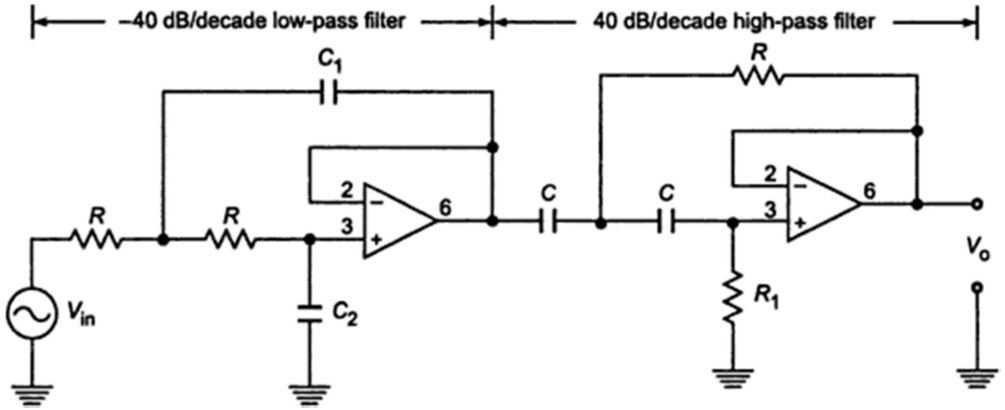


$$\left|\frac{V_o}{V_{in}}\right| = \frac{(f/f_c)}{\{1 + (f/f_c)^2\}^{1/2}}$$

Figure 1.13 band Pass Filter

Band pass filter: Which can be used to remove Particular band of frequency components

Band Reject Filter: Which removes particular band of frequencies



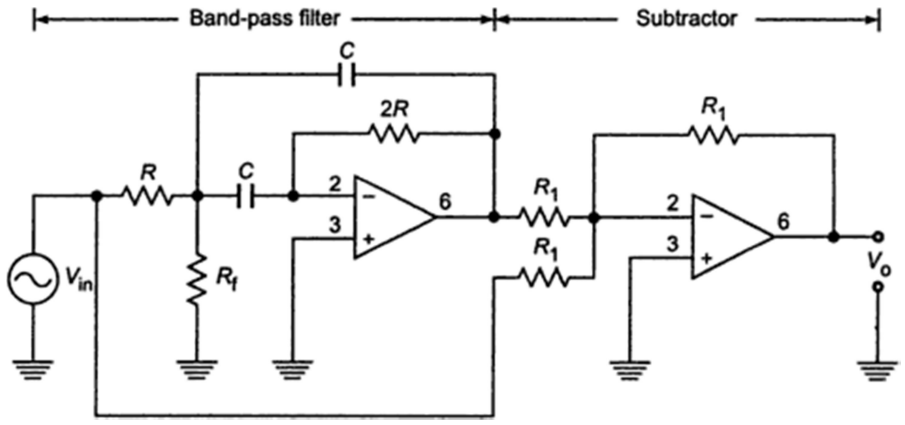


Figure 1.14 Band Stop Filter

$$f_r = \frac{(0.1125/RC)}{\{1+(R/R_f)\}^{1/2}}$$

1.9 NOISE & NOISE REDUCTION TECHNIQUES :

In all signal conditioning unwnted noise signals corrupt the weak measurement signals and reduce the accuracy of the measurement. The parameter which gives the value of the corruption is known as signal to noise ratio (SNR).

The first type of noise originates externally and enters into the system along with the signal.
The second type is intrinsic and is generated by the components used in the construction of a measurement circuit. It results from change in characteristics of components like transistors, op amps, etc., over the period of time. Change in ambient temperature and line voltage fluctuations also cause noise which is usually called drift rather than noise.
The third type is induced noise and is picked up by the circuit through resistive, capacitive or magnetic coupling. Ground loop, electric field interference, magnetic field interference and radio frequency interference are the causes of induced noise.

Induced noise:
There are several sources are there for induced noise.

1. Grounding

2. Shielding Grounding:

System ground plays a major role in internally generated induced noise.
Ground and return lines in a system: Ground in electronic system refers to a reference potential.

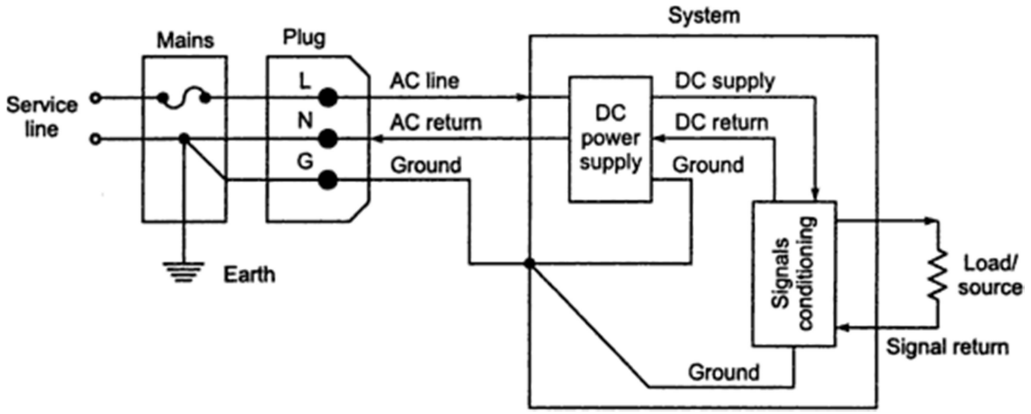


Figure 1.15 Band Stop Filter

Grounding in analog circuits: Providing the separate power supply to the devices drawing higher currents.

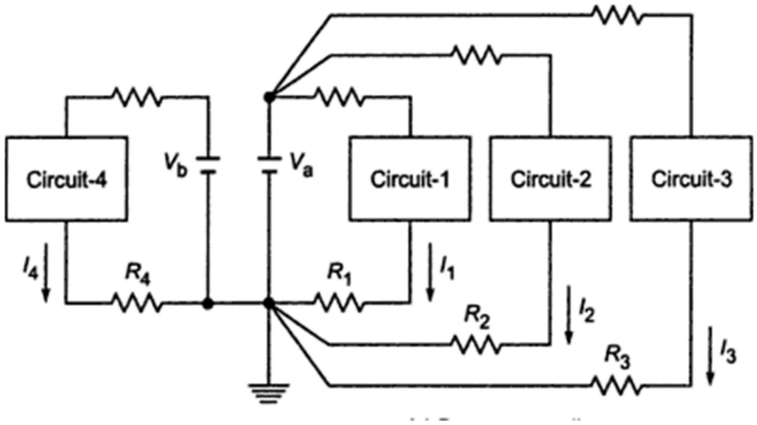


Figure 1.16 Grounding of circuits

SHIELDING:
Electro magnetic interference:

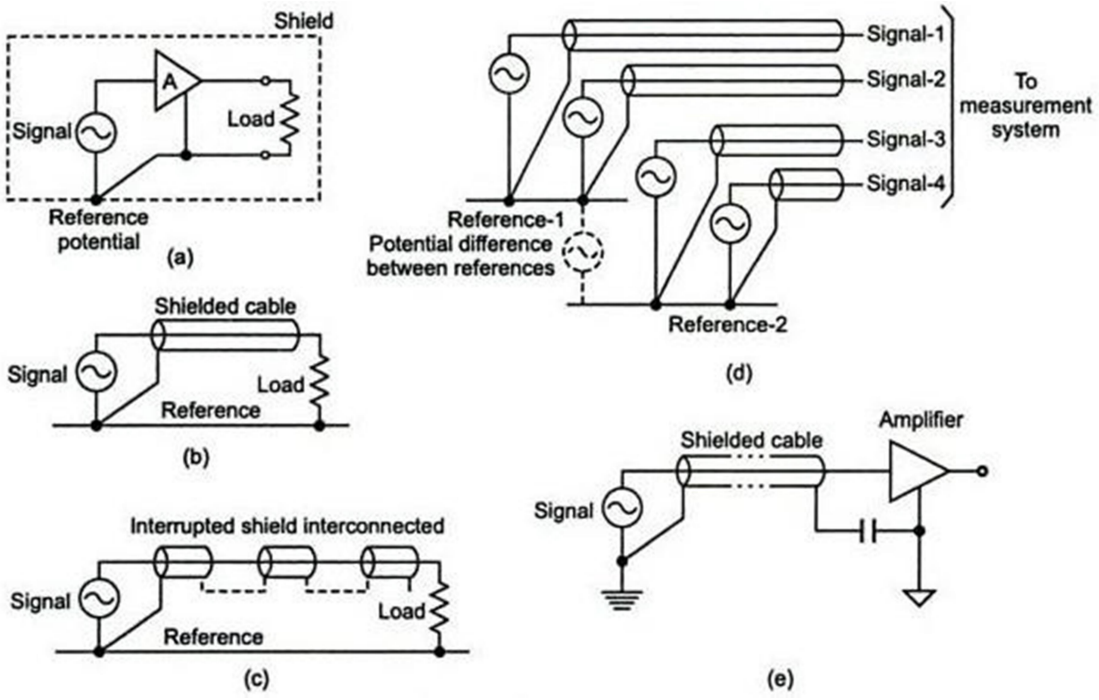


Figure 1.17 Sheiding

The strength of magnetic coupling is minimized by proper wiring and shielding techniques.

UNIT - II

Computer Based Medical Instrumentaion – SBMA7006

Principle of Data Acquisition

Sampling concepts – DAC- Weighted Resistor Network, R-2R Ladder Network, ADC – Successive Approximation ADC - Counting type, Successive approximation, parallel comparator, Data acquisition system – Analog Input , Analog Output

2.1 SAMPLING CONCEPTS:

Signals in digital form are more convenient than analog form for processing and control operations. Computers process information represented by digital data. Sampling is a process of converting continuous time amplitude into discrete time amplitude.

RECONSTRUCTION:

Series of digital data obtained by periodic sampling of a continuous analog signal are the samples of analog signals. The samples represent the analog signal at specific instants. The original analog signal can be reconstructed by applying back the digital data in sequence at the same time intervals to a digital to analog converter. The DAC represent the analog signal in discrete time and amplitude, it cannot faithfully represent and reconstruct the original without error.

SHANNON SAMPLING THEOREM:

The Shannon sampling theorem relates the sampling rate and maximum frequency component present in the signal to completely recover the original signal from its sampled data without loss of information.

DISCRETE TIME REPRESENTATION:

Consider two signals, one varying smoothly in time,x(t), and another a train of equally spaced pulses with constant amplitude,p(t). Let the signal x(t) be band limited and let the highest frequency component in the band limited signal be fH. Let the period of the signal p(t) be T.

FREQUENCY SPECTRUM:

The signals x(t), p(t) and xp(t) are represented in time domain. Fourier transforms are X(f), P(f) and Xp(f) describe the frequency content s of the respective signals. To completely recover the CT signal x(t) from its sampled representation xp(t), the spectrum X(f) should be recovered totally from Xp(f). X(f) is complete ly recovered from Xp(f) by filtering all frequencies. The frequency 2fH is referred to as the Nyquist frequency.

ALIASING:

Sampling at frequency less than twice the high frequency component is known as undersampling. Undersampling results in the phenomena called aliasing.

INTERPOLATION:

A DAC generates quantized analog outputs for the digital inputs. The quantized analog output of the DAC for discrete digital input reconstructs the original signal only in discrete time and there are gaps between the data points. The process of reconstructing the signal between the data point is called interpolation.

2.2 DIGITAL TO ANALOG CONVERTERS

There are two basic type of converters, digital-to-analog (DACs or D/As) and analog- to digital (ADCs or A/Ds). Their purpose is fairly straightforward. In the case of DACs, they output an analog voltage that is a proportion of a reference voltage, the proportion based on the digital word applied. In the case of the ADC, a digital representation of the analog voltage that is applied to the ADCs input is outputted, the representation proportional to a reference voltage. In both cases the digital word is almost always based on a binarily weighted proportion. The digital input or output is arranged in words of varying widths, referred to as bits, typically anywhere from 6 bits to 24 bits. In a binarily weighted system each bit is worth half of the bit to its left and twice the bit to its right. The greater the number of bits in the digital word, the finer the resolution. These bits are typically arranged in groups of four, called bytes, for convenience

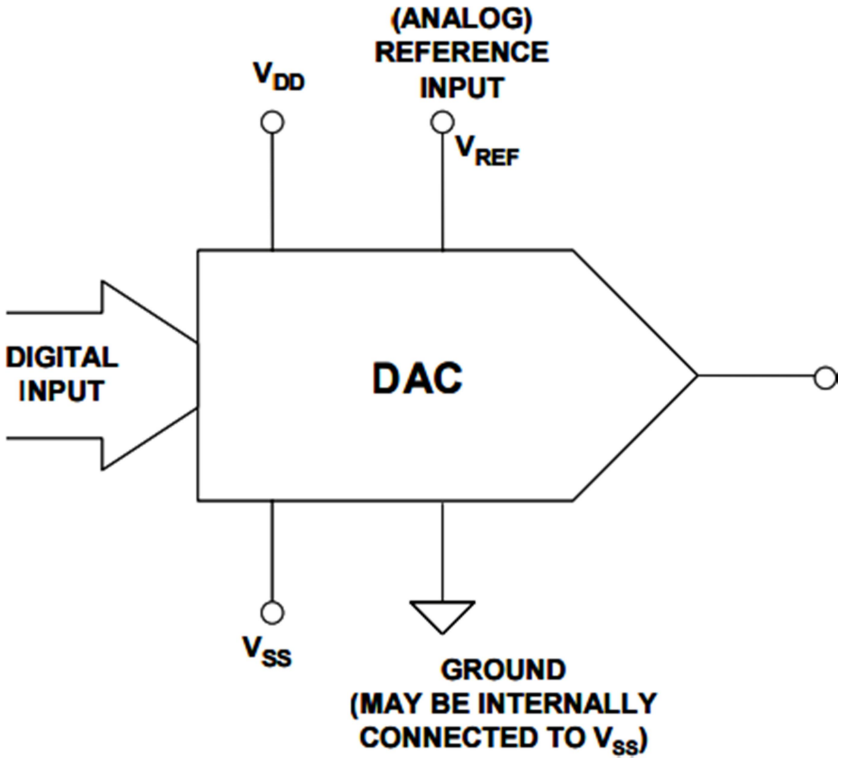


Figure 2.1 basic DAC

BINARY WEIGHTED CURRENT SOURCE

The voltage- mode binary-weighted resistor DAC shown in Figure, is usually the simplest textbook example of a DAC. However, this DAC is not inherently monotonic and is actually quite hard to manufacture successfully at high resolutions due to the large spread in component (resistor) values. In addition, the output impedance of the voltagemode binary DAC changes with the input code.

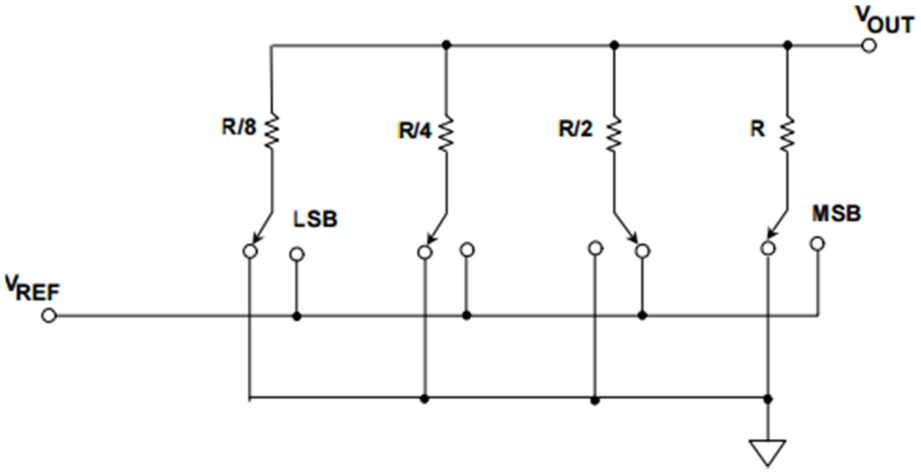


Figure 2.2 Digital Switch

Current- mode binary weighted DACs are shown in Figure (resistor-based), (current- source based). An N-bit DAC of this type consists of N weighted current sources (which may simply be resistors and a voltage reference) in the ratio 1:2:4:8 2N–1. The LSB switches the 2N–1 current, the MSB the 1 current, etc. The theory is simple but the practical problems of manufacturing an IC of an economical size with current or resistor ratios of even 128:1 for an 8-bit DAC are enormous, especially as they must have matched temperature coefficients. This architecture is virtually never used on its own in integrated circuit DACs, although, again, 3-bit or 4-bit versions have been used as components in more complex structures. For example, the AD550 mentioned at the beginning of this section is an example of a binary-weighted DAC.

R-2R LADDER

One of the most common DAC building-block structures is the R-2R resistor ladder network shown in Figure . It uses resistors of only two different values, and their ratio is 2:1. An N-bit DAC requires 2N resistors, and they are quite easily trimmed. There are also relatively few resistors to trim.

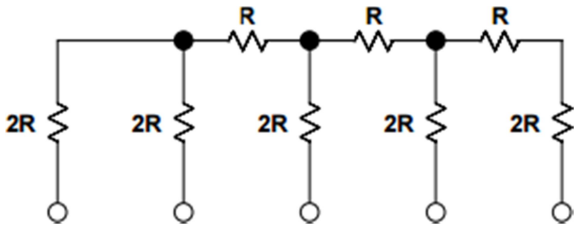


Figure 2.3 R-2R ladder Network

Note the extra resistor added at the RFEEDBACK pin. This is designed to be the feedback resistor for the I/V op amp. This resistor is trimmed along with the rest of the resistors so it tracks. Also, since it is made of the same material as the rest of the resistors, therefore having the same temperature coefficient, and is on the same substrate, hence at the same temperature, it will track over temperature.

2.3 ANALOG TO DIGITAL CONVERTER

The basic ADC function is shown in Figure . This could also be referred to as a quantizer. Most ADC chips also include some of the support circuitry, such as clock oscillator for the sampling clock, reference (REF), the sample and hold function, and output data latches. In addition to these basic functions, some ADCs have additional circuitry built in. These functions could include multiplexers, sequencers, auto- calibration circuits, programmable gain amplifiers (PGAs).

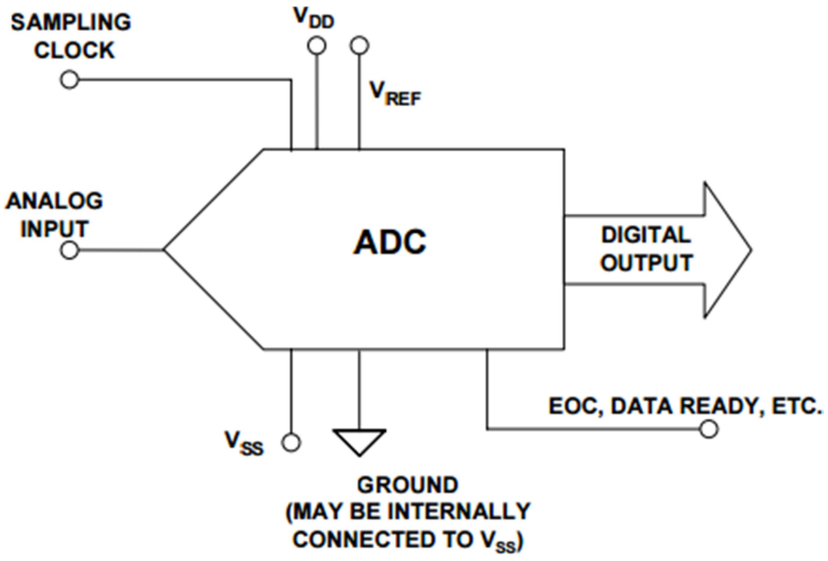


Figure 2.4 basic ADC

The sampling clock input is a critical function in an ADC and a source of some confusion. It could truly be the sampling clock. This frequency would typically be several times higher than the sampling rate of the converter. It could also be a convert start (or encode) command which would happen once per conversion. Pipeline architecture devices and sigma delta (Σ-Δ) converters are continuously converting and have no convert start command.

THE COMPARATOR: A 1-BIT ADC

A comparator is a 1-bit ADC , If the input is above a threshold, the output has one logic value, below it has another. There is no ADC architecture which does not use at least one comparator of some sort. So while a 1 bit ADC is of very limited usefulness it is a building block for other architectures.

Comparators used as building blocks in ADCs need good resolution which implies high gain. This can lead to uncontrolled oscillation when the differential input approaches zero. In order to prevent this, hysteresis is often added to comparators using a small amount of positive feedback. Figure shows the effects of hysteresis on the overall transfer function. Many comparators have a milli volt or two of hysteresis to encourage ―snap‖ action and to prevent local feedback from causing instability in the transition region. Note that the resolution of the comparator can be no less than the hysteresis, so large values of hysteresis are generally not useful.

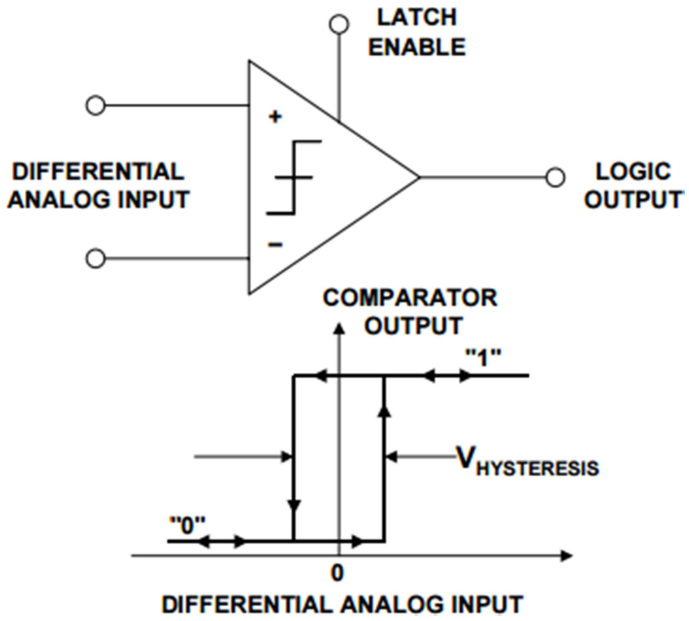


Figure 2.5 Hysteresis Curve

SUCCESSIVE APPROXIMATION ADCS

The successive approximation ADC has been the mainstay of data acquisition for many years. Recent design improvements have extended the sampling frequency of these ADCs into the megahertz region. The basic successive approximation ADC is shown in Figure, It performs conversions on command. On the assertion of the CONVERT START command, the sample-and-hold (SHA) is placed in the hold mode, and all the bits of the successive approximation register (SAR) are reset to –0‖ except the MSB which is set to –1.‖ The SAR output drives the internal DAC. If the DAC output is greater than the analog input, this bit in the SAR is reset, otherwise it is left set. The next most significant bit is then set to –1.‖ If the DAC output is greater than the analog input, this bit in the SAR is reset, otherwise it is left set. The process is repeated with each bit in turn. When all the bits have been set, tested, and reset or not as appropriate, the contents of the SAR correspond to the value of the analog input, and the conversion is complete. These bit –tests‖ can form the basis of a serial output version SAR-based ADC.

The fundamental timing diagram for a typical SAR ADC is shown in Figure, The end of conversion is generally indicated by an end-of-convert (EOC), data-ready (DRDY), or a busy signal (actually, not-BUSY indicates end of conversion). The polarities and name of this signal may be different for different SAR ADCs, but the fundamental concept is the same. At the beginning of the conversion interval, the signal goes high (or low) and remains in that state until the conversion is completed, at which time it goes low (or high). The trailing edge is generally an indication of valid output

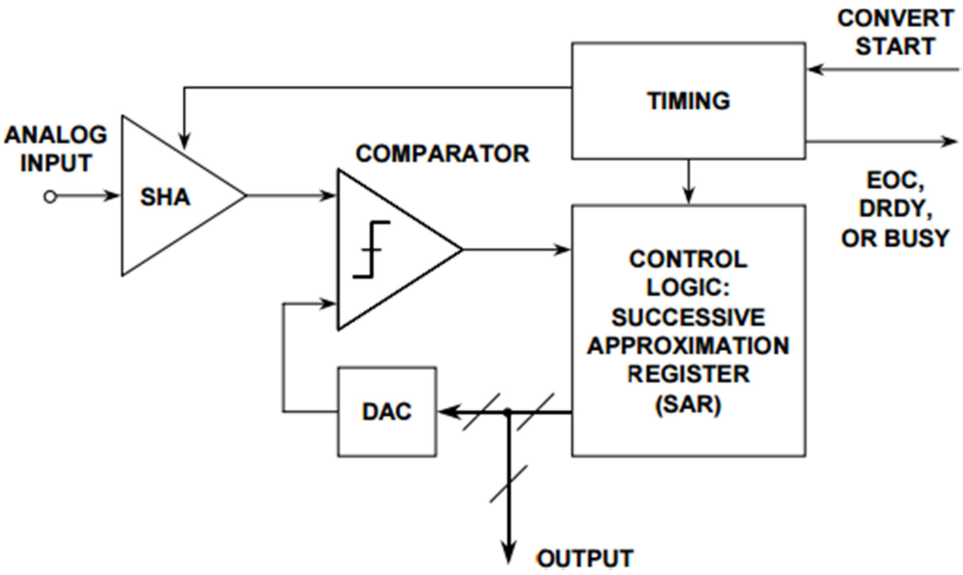


Figure 2.6 SAR ADC

data, but the data sheet should be carefully studied—in some ADCs additional delay is required before the output data is valid.

FLASH CONVERTERS

Flash ADCs (sometimes called parallel ADCs) are the fastest type of ADC and use large numbers of comparators. An N-bit flash ADC consists of 2N resistors and 2N–1 comparators arranged as in Figure Each comparator has a reference voltage which is 1 LSB higher than that of the one below it in the chain. For a given input voltage, all the comparators below a certain point will have their input voltage larger than their reference voltage and a ‑1‖ logic output, and all the comparators above that point will have a reference voltage larger than the input voltage and a ‑0‖ logic output. The 2N– 1 comparator outputs therefore behave in a way analogous to a mercury thermometer, and the output code at this point is sometimes called a thermometer code. Since 2N–1 data outputs are not really practical, they are processed by a decoder to generate an N- bit binary output.

The input signal is applied to all the comparators at once, so the thermometer output is delayed by only one comparator delay from the input, and the encoder N-bit output by only a few gate delays on top of that, so the process is very fast. However, the architecture uses large numbers of resistors and comparators and is limited to low resolutions, and if it is to be fast, each comparator must run at relatively high power levels. Hence, the problems of flash ADCs include limited resolution, high power dissipation because of the large number of high speed comparators (especially at sampling rates greater than 50 MSPS), and relatively large (and therefore expensive) chip sizes. In addition, the resistance of the reference resistor chain must be kept low to supply adequate bias current to the fast comparators, so the voltage reference has to source quite large currents (typically > 10 mA).

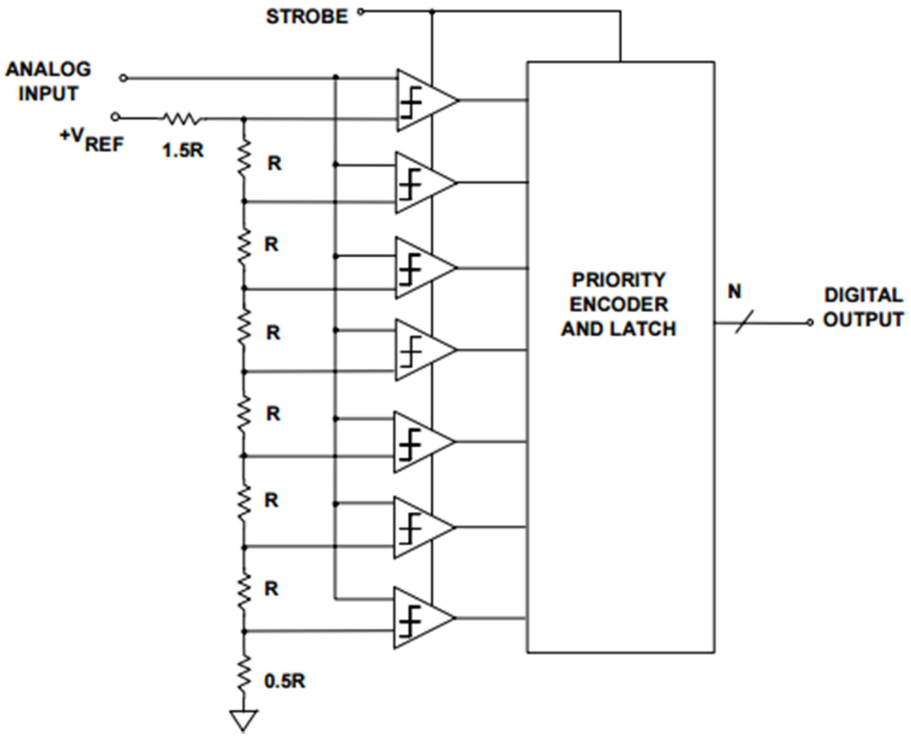


Figure 2.7 Parallel ADC

Each comparator has a voltage-variable junction capacitance, and this signal- dependent capacitance results in most flash ADCs having reduced ENOB and higher distortion at high input frequencies. For this reason, most flash converters must be driven with a wideband op amp which is tolerant to the capacitive load presented by the converter as well as high speed transients developed on the input.

COUNTING AND INTEGRATING ADC ARCHITECTURES

Although counting-based ADCs are not well suited for high speed applications, they are ideal for high resolution, low frequency applications, especially when combined with integrating techniques.

The counting ADC technique basically uses a sampling pulse to take a sample of the analog signal, set an R/S flip- flop, and simultaneously start a controlled ramp voltage. The ramp voltage is compared with the input, and when they are equal, a pulse is generated which resets the R/S flip- flop. The output of the flip- flop is a pulse whose width is proportional to the analog signal at the sampling instant. This pulse width modulated (PWM) pulse controls a gated oscillator, and the number of pulses out of the gated oscillator represents the quantized value of the analog signal. This pulse train can be easily converted to a binary word by driving a counter. In Reeves' system, a master clock of 600 kHz was used, and a 100:1 divider generated the 6-kHz sampling pulses. The system uses a 5-bit counter, and 31 counts (out of the 100 counts between sampling pulses) therefore represents a full-scale signal. The technique can obviously be extended to higher resolutions.

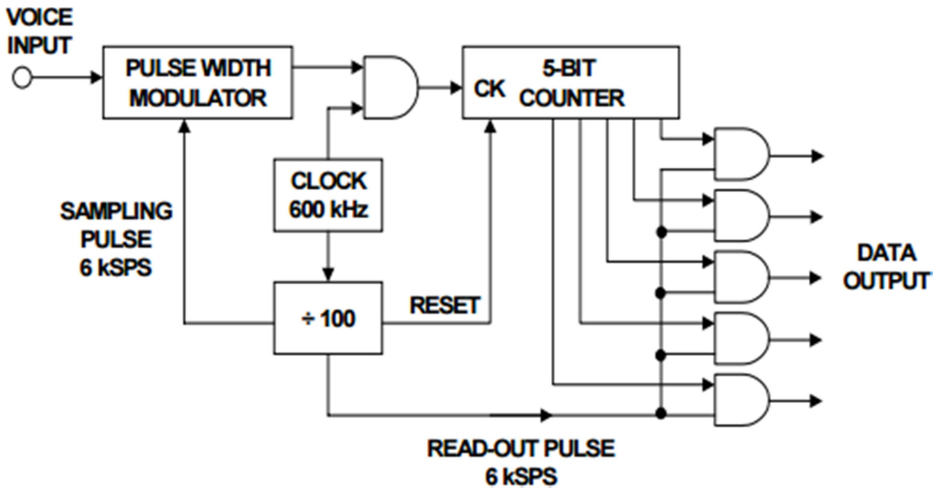


Figure 2.8 Single mode Flash ADC

DUAL SLOPE/MULTI-SLOPE ADCS

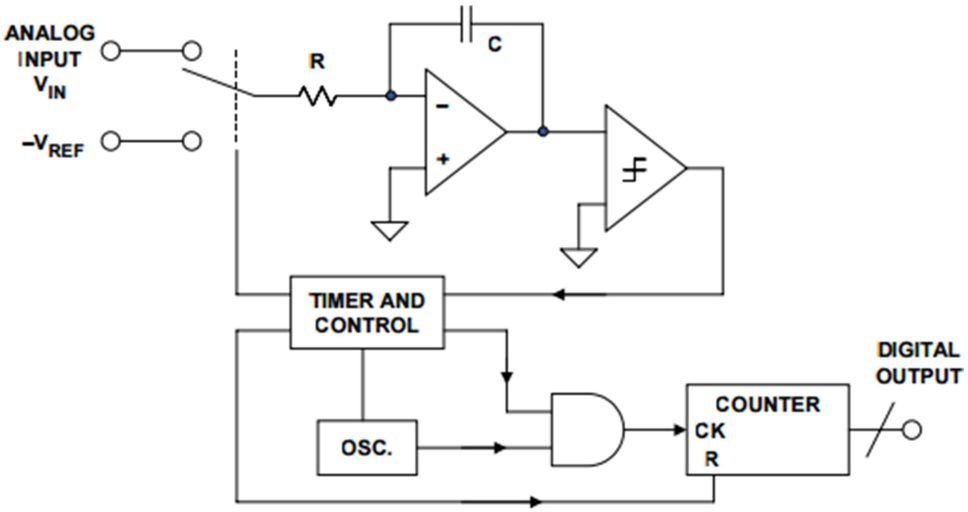


Figure 2.9 Dual mode Flash ADC

The dual-slope ADC architecture was truly a breakthrough in ADCs for high resolution applications such as digital voltmeters (DVMs), etc. A simplified diagram is shown in Figure. the integrating capacitor is proportional to the average value of the input over the interval T. The integral of the reference is an opposite- going ramp having a slope of VREF/RC. At the same time, the counter is again counting from zero. When the integrator output reaches zero, the count is stopped, and the analog circuitry is reset. Since the charge gained is proportional to VIN · T, and the equal amount of charge lost is proportional to VREF · tx, then the number of counts relative to the full scale count is proportional to tx/T, or VIN/VREF. If the output of the counter is a binary number, it will therefore be a binary representation of the input voltage. Dual-slope integration has many advantages. Conversion accuracy is independent of both the capacitance and the clock frequency, because they affect both the up-slope and the down-slope by the same ratio.

UNIT - III

Computer Based Medical Instrumentaion – SBMA7006

HARDWARE ORGANIZATION OF PC

Motherboard components – Microprocessor, memory, Chipset Chips, Interrupts, DMA Channel, System Control Chip , Peripheral Control Chip- Peripherals , Features of ISA and PCI buses.

3.1 MOTHERBOARD COMPONENTS

A PC in general, consists of central processing unit (CPU), monitor, keyboard, mouse and printer. The basic block diagram of motherboard is shown in below. Components directly attached to or part of the motherboard include, The **CPU** (Central Processing Unit) performs most of the calculations which enable a computer to function, and is sometimes referred to as the "brain" of the computer. It is usually cooled by a heat sink and fan. Most newer CPUs include an on-die Graphics Processing Unit (GPU).The **Chipset**, which includes the north bridge, mediates communication between the CPU and the other components of the system, including main memory.The **Random-Access Memory** (RAM) stores the code and data that are being actively accessed by the CPU. The **Read-Only Memory** (ROM) stores the BIOS that runs when the computer is powered on or otherwise begins execution, a process known as Bootstrapping, or "booting" or "booting up". The **BIOS** (Basic Input Output System) includes boot firmware and power management firmware.

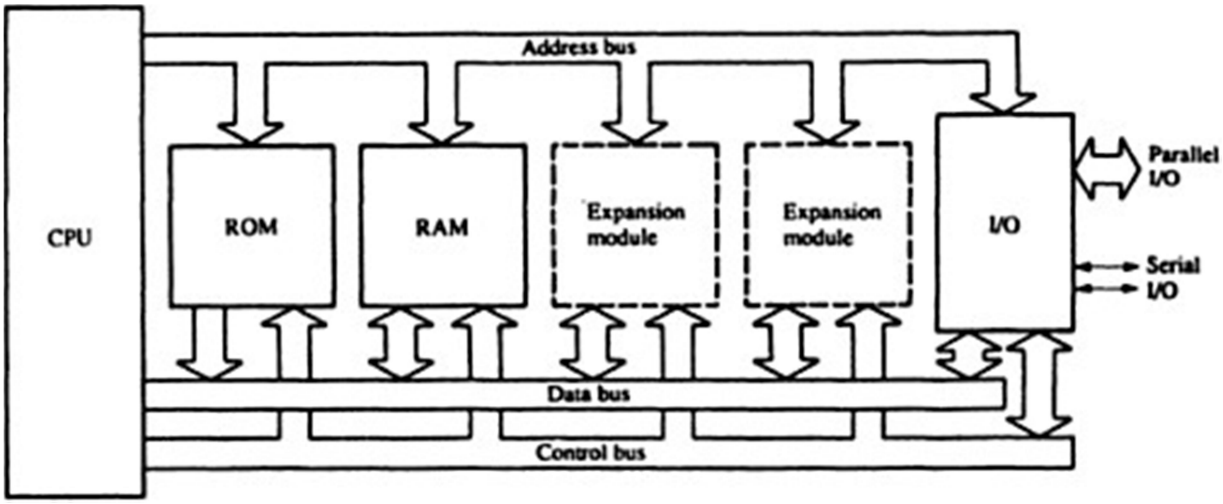


Figure 3.1 GPIB Architecture

Newer motherboards use Unified Extensible Firmware Interface (UEFI) instead of BIOS. **Buses** connect the CPU to various internal components and to expand cards for graphics and sound. The CMOS battery is also attached to the motherboard. This battery is the same as a watch battery or a battery for a remote to a car's central locking system. Most batteries are CR2032, which powers the memory for date and time in the BIOS chip.

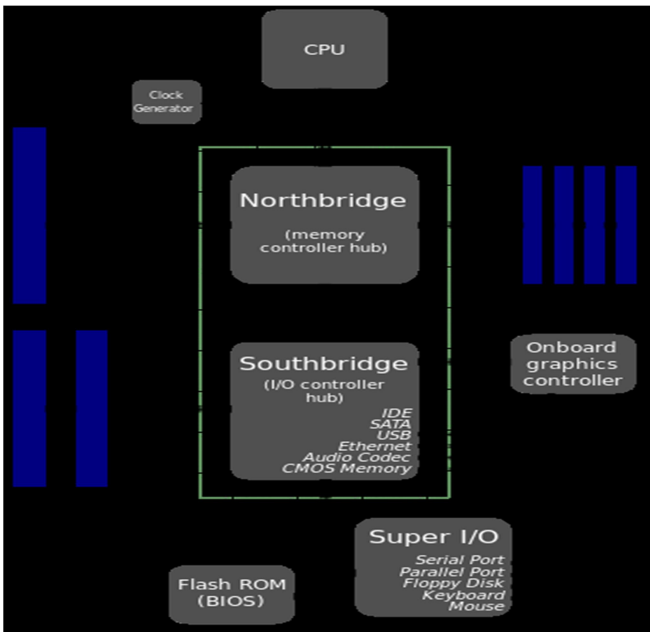


Figure 3.2 Unified Extensible Firmware Interface

3.2 SYSTEM RESOURCES

Systems programming, an **interrupt** is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. An interrupt alerts the processor to a high- priority condition requiring the interruption of the current code the processor is executing. The processor responds by suspending its current activities, saving its state, and executing a function called an *interrupt handler* (or an interrupt service routine, ISR) to deal with the event. This interruption is temporary, and, after the interrupt handler finishes, the processor resumes normal activities. There are two types of interrupts: hardware interrupts and software interrupts.

Hardware interrupts are used by devices to communicate that they require attention from the operating system. Internally, hardware interrupts are implemented using electronic alerting signals that are sent to the processor from an external device, which is either a part of the computer itself, such as a disk controller, or an external peripheral. For example, pressing a key on the keyboard or moving the mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position. Unlike the software type (described below), hardware interrupts are asynchronous and can occur in the middle of instruction execution, requiring additional care in programming. The act of initiating a hardware interrupt is referred to as an interrupt request(IRQ). A **software interrupt** is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed. The former is often called a *trap* or *exception* and is used for errors or events occurring during program

execution that are exceptional enough that they cannot be handled within the program itself. For example, if the processor's arithmetic logic unit is commanded to divide a number by zero, this impossible demand will cause a *divide-by-zero exception*, perhaps causing the computer to abandon the calculation or display an error message. Software interrupt instructions function similarly to subroutine calls and are used for a variety of purposes, such as to request services from low-levelsystem software such as device drivers. For example, computers often use software interrupt instructions to communicate with the disk controller to request data be read or written to the disk.

Each interrupt has its own interrupt handler. The number of hardware interrupts is limited by the number of interrupt request (IRQ) lines to the processor, but there may be hundreds of different software interrupts. Interrupts are a commonly used technique for computer multitasking, especially in real-time computing. Such a system is said to be interrupt-driven.

Level-triggered

A *level-triggered interrupt* is an interrupt signalled by maintaining the interrupt line at a high or low level. A device wishing to signal a Level-triggered interrupt drives the interrupt request line to its active level (high or low), and then holds it at that level until it is serviced. It ceases asserting the line when the CPU commands it to or otherwise handles the condition that caused it to signal the interrupt. Typically, the processor samples the interrupt input at predefined times during each bus cycle such as state T2 for the Z80 microprocessor. If the interrupt isn't active when the processor samples it, the CPU doesn't see it. One possible use for this type of interrupt is to minimize spurious signals from a noisy interrupt line: a spurious pulse will often be so short that it is not noticed.

Multiple devices may share a level-triggered interrupt line if they are designed to. The interrupt line must have a pull-down or pull-up resistor so that when not actively driven it settles to its inactive state. Devices actively assert the line to indicate an outstanding interrupt, but let the line float (do not actively drive it) when not signalling an interrupt. The line is then in its asserted state when any (one or more than one) of the sharing devices is signalling an outstanding interrupt.

Level-triggered interrupt is favored by some because it is easy to share the interrupt request line without losing the interrupts, when multiple shared devices interrupt at the same time. Upon detecting assertion of the interrupt line, the CPU must search through the devices sharing the interrupt request line until one who triggered the interrupt is detected. After servicing this device, the CPU may recheck the interrupt line status to determine whether any other devices also needs service. If the line is now de-asserted, the CPU avoids checking the remaining devices on the line. Since some devices interrupt more frequently than others, and other device interrupts

are particularly expensive, a careful ordering of device checks is employed to increase efficiency. The original PCI standard mandated level-triggered interrupts because of this advantage of sharing interrupts.

There are also serious problems with sharing level-triggered interrupts. As long as any device on the line has an outstanding request for service the line remains asserted, so it is not possible to detect a change in the status of any other device. Deferring servicing a low-priority device is not an option, because this would prevent detection of service requests from higher-priority devices. If there is a device on the line that the CPU does not know how to service, then any interrupt from that device permanently blocks all interrupts from the other devices.

Edge-triggere

An *edge-triggered interrupt* is an interrupt signalled by a level transition on the interrupt line, either a falling edge (high to low) or a rising edge (low to high). A device, wishing to signal an interrupt, drives a pulse onto the line and then releases the line to its inactive state. If the pulse is too short to be detected by polled I/O then special hardware may be required to detect the edge.

Multiple devices may share an edge-triggered interrupt line if they are designed to. The interrupt line must have a pull-down or pull-up resistor so that when not actively driven it settles to one particular state. Devices signal an interrupt by briefly driving the line to its non-default state, and let the line float (do not actively drive it) when not signalling an interrupt. This type of connection is also referred to as open collector. The line then carries all the pulses generated by all the devices. (This is analogous to the pull cord on some buses and trolleys that any passenger can pull to signal the driver that they are requesting a stop.) However, interrupt pulses from different devices may merge if they occur close in time. To avoid losing interrupts the CPU must trigger on the trailing edge of the pulse (e.g. the rising edge if the line is pulled up and driven low). After detecting an interrupt the CPU must check all the devices for service requirements.

Edge-triggered interrupts do not suffer the problems that level-triggered interrupts have with sharing. Service of a low-priority device can be postponed arbitrarily, and interrupts will continue to be received from the high-priority devices that are being serviced. If there is a device that the CPU does not know how to service, it may cause a spurious interrupt, or even periodic spurious interrupts, but it does not interfere with the interrupt signalling of the other devices. However, it is fairly easy for an edge triggered interrupt to be missed - for example if interrupts have to be masked for a period - and unless there is some type of hardware latch that records the event it is impossible to recover. Such problems caused many "lockups" in early computer hardware because the processor did not know it was expected to do something. More modern hardware often has one or more interrupt status registers that latch the interrupt requests; well written edge-driven interrupt software often checks such registers to ensure events are not missed.

The elderly Industry Standard Architecture (ISA) bus uses edge-triggered interrupts, but does not mandate that devices be able to share them. The parallel port also uses edge-triggered interrupts. Many older devices assume that they have exclusive use of their interrupt line, making it electrically unsafe to share them. However, ISA motherboards include pull-up resistors on the IRQ lines, so well-behaved devices share ISA interrupts just fine.

3.3 SYSTEM AND PERIPHERAL CONTROL CHIPS

Keyboard controller

n computing, a **keyboard controller** is a device that interfaces a keyboard to a computer. Its main function is to inform the computer when a key is pressed or released. When data from the keyboard arrives, the controller raises an interrupt (a*keyboard interrupt*) to allow the CPU to handle the input.

If a keyboard is a separate peripheral system unit (such as in most modern desktop computers), the keyboard controller is not directly attached to the keys, but receives scancodes from a microcontroller embedded in the keyboard via some kind ofserial interface. In this case, the controller usually also controls the keyboard's LEDs by sending data back to keyboard through the wire.

The IBM PC AT used an Intel 8042 chip to interface to the keyboard. This computer also controlled access to the A20 line in order to implement a workaround for a chip bug in the Intel 80286.[1] The keyboard controller was also used to initiate a software CPU reset in order to allow the CPU to transition from protected mode to real mode because the 286 did not allow the CPU to go from protected mode to real mode unless the CPU is reset. This was a problem because the BIOS and the operating system services could only be called by programs in real mode. These behaviors have been used by plenty of software that expects this behavior, and therefore keyboard controllers have continued controlling the A20 line and performing software CPU resets even when the need for a reset via the keyboard controller was obviated by the Intel 80386's ability to switch to real mode from protected mode without a CPU reset. The keyboard controller also handles PS/2 mouse input if a PS/2 mouse port is present. Today the keyboard controller is either a unit inside a Super I/O device or is missing, having its keyboard and mouse functions handled by a USB controller and its role in controlling the A20 line handled by the chipset.

3.4 Chip set

In a computer system, a **chipset** is a set of electronic components in an integrated circuit that manages the data flow between the processor, memory and peripherals. It is usually found on the motherboard. Chipsets are usually designed to work with a specific family of microprocessors. Because it controls communications between the processor and external devices, the chipset plays a crucial role in determining system performance. In computing, the term chipset commonly refers to a set of specialized chips on a computer's motherboard or an expansion card. In personal computers, the first chipset for the IBM PC AT of 1984 was the NEAT chipset developed by Chips and Technologies for the Intel 80286 CPU.

3.5 PERIPHERALS

Input devices allow the user to enter information into the system, or control its operation. Most personal computers have a mouse and keyboard, but laptop systems typically use a touchpad instead of a mouse. Other input devices include webcams, microphones, joysticks, and image scanners. A **peripheral** is a "device that is used to put information into or get information out of the computer.

There are two different types of peripherals: input devices, which interact with or send data to the computer (mouse, keyboards, etc.), and output devices, which provide output to the user from the computer (monitors, printers, etc.). Some peripherals, such as touchscreens, can be used both as input and output devices.

A peripheral device is generally defined as any auxiliary device such as a computer mouse or keyboard that connects to and works with the computer in some way. Other examples of peripherals are image scanners, tape drives, microphones, loudspeakers, webcams, and digital cameras. Many modern devices, such as digitalwatches, smartphones and tablet computers, have interfaces that allow them to be used as a peripheral by desktop computers, although they are not host-dependent in the same way as other peripheral devices.

Common input peripherals include keyboards, , graphic tablets, touchscreens, barcode readers,image scanners, microphones, webcams, game controllers, light pens, and digital cameras. Common output peripherals include computer displays, printers,projectors, and computer speakers.

Output device

Output devices display information in a human readable form. Such devices could include printers, speakers, monitors or a Braille embosser. Data is stored by a computer using a variety of media. Hard disk drives are found in virtually all older computers, due to their

high capacity and low cost, but solid-state drives are faster and more power efficient, although currently more expensive than hard drives, so are often found in more expensive computers. Some systems may use a disk array controller for greater performance or reliability.

3.6 BIOS SERVICES

The **BIOS** an acronym for **Basic Input/Output System** and also known as the **System** BIOS, ROM BIOS or PC BIOS) is a type of firmware used during the booting process (power-on startup) on IBM PC compatible computers. The BIOS firmware is built into personal computers (PCs), and it is the first software they run when powered on. The name itself originates from the Basic Input/Output System used in the CP/M operating system in 1975. Originally proprietary to the IBM PC, the BIOS has been reverse engineered by companies looking to create compatible systems and the interface of that original system serves as a *de facto* standard.

The fundamental purposes of the BIOS in modern PCs are to initialize and test the system hardware components, and to load a boot loader or an operating system from a mass memory device. The BIOS additionally provides an abstraction layer for the hardware, i.e., a consistent way for application programs and operating systems to interact with the keyboard, display, and other input/output (I/O) devices. Variations in the system hardware are hidden by the BIOS from programs that use BIOS services instead of directly accessing the hardware. MS-DOS (PC DOS), which was the dominant PC operating system from the early 1980s until the mid 1990s, relied on BIOS services for disk, keyboard, and text display functions. MS Windows NT, Linux, and other protected mode operating systems in general ignore the abstraction layer provided by the BIOS and do not use it after loading, instead accessing the hardware components directly.

Every BIOS implementation is specifically designed to work with a particular computer or motherboard model, by interfacing with various devices that make up the complementary system chipset. Originally, BIOS firmware was stored in a ROM chip on the PC motherboard; in modern computer systems, the BIOS contents are stored on flash memory so it can be rewritten without removing the chip from the motherboard. This allows easy updates to the BIOS firmware so new features can be added or bugs can be fixed, but it also creates a possibility for the computer to become infected with BIOS rootkits.

Unified Extensible Firmware Interface (UEFI) was designed as a successor to BIOS, aiming to address its technical shortcomings. As of 2014, new PC hardware predominantly ships with UEFI firmware.

3.7 EXPANSION BUSES

An expansion card in computing is a printed circuit board that can be inserted into an expansion slot of a computer motherboard or backplane to add functionality to a computer system via the expansion bus. Expansions cards can be used to obtain or expand on features not offered by the motherboard. Computer data storage, often called storage or memory, refers to computer components and recording media that retain digital data. Data storage is a core function and fundamental component of computers. The price of solid-state drives (SSD), which store data on flash memory, has dropped a lot in recent years, making them a better choice than ever to add to a computer to make booting up and accessing files faster.

3.8 PARALLEL PORT

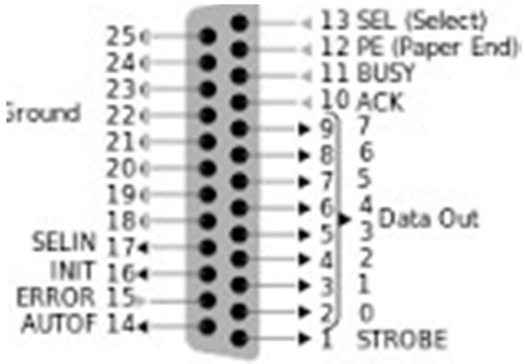


Figure 3.3: Parallel Port Pin Diagram

A **parallel port** is a type of interface found on computers (personal and otherwise) for connecting peripherals. Incomputing, a parallel port is a parallel communication physical interface. It is also known as a **printer port** orCentronics port. It was an industry *de facto* standard for many years, and was finally standardized as IEEE 1284 in the late 1990s, which defined the Enhanced Parallel Port (EPP) and Extended Capability Port (ECP) bi-directional versions. Today, the parallel port interface is seeing decreasing use because of the rise of Universal Serial Bus (USB) devices, along with network printing using Ethernet.

The parallel port interface was originally known as the **Parallel Printer Adapter** on IBM PC-compatible computers. It was primarily designed to operate a line printer that used IBM's 8-bit extended ASCII character set to print text, but could also be used to adapt other peripherals. Graphical printers, along with a host of other devices, have been designed to communicate with the system. Most PC-compatible systems in the 1980s and 1990s had one to three ports, with communication interfaces defined like this:

Logical parallel port 1: I/O port 0x3BC, IRQ 7 (usually in monochrome graphics adapters)
Logical parallel port 2: I/O port 0x378, IRQ 7 (dedicated IO cards or using a controller built into the mainboard)

Logical parallel port 3: I/O port 0x278, IRQ 5 (dedicated IO cards or using a controller built into the mainboard)

If no printer port is present at 0x3BC, the second port in the row (0x378) becomes logical parallel port 1 and 0x278 becomes logical parallel port 2 for the BIOS. Sometimes, printer ports are jumpered to share an interrupt despite having their own IO addresses (i.e. only one can be used interrupt-driven at a time). In some cases, the BIOS supports a fourth printer port as well, but the base address for it differs significantly between vendors. Since the reserved entry for a fourth logical printer port in the BIOS Data Area (BDA) is shared with other uses on PS/2 machines and with S3 compatible graphics cards, it typically requires special drivers in most environments. Under DR-DOS 7.02 the BIOS port assignments can be changed and overridden using the LPT1, LPT2, LPT3 (and optionally LPT4)CONFIG.SYS directives.

3.9 Features of ISA & PCI Buses:

ISA is still a commonplace technology in embedded systems, despite being an obsolete expansion bus technology in the sphere of personal computing. Due to the long lifetimes of embedded systems and the need to re-use existing system peripherals, it is often attractive for system designers to retain ISA compatibility in their system, despite the availability of newer and more advanced expansion bus technology. The purpose of this document is to highlight any limitations in implementing an ISA expansion bus on a modern Intel® Express Chipset that a system designer may face.

ISA Bridge Support and Limitations This chapter summarizes the two methods available to system designers for implementing an ISA bus in their design. It also describes the limitations that system designers will face in the implementation of each method. In both cases, system designers should work with the ISA bridge vendor to fully understand the impact on their design. A list of vendors who provide bridges for each method is also provided. These lists are provided as a reference only and do not constitute a guarantee of operability with the Intel® Express Chipsets.

3.10 PCI/ISA Bridge

PCI to ISA bridge is the most common method of interfacing ISA devices to modern chipsets. In most respects, these devices perform like a standard PCI device. As such, it is a relatively simple for system designers to use such a bridge in their design. However, there are limitations in the PCI interface of Intel® Express chipsets that could limit the usefulness of a PCI to ISA bridge in the system’s application. System designers should be aware of these limitations before proceeding with their design.

ISA DMA ISA DMA or Bus Master transactions are not supported through the standard PCI Bus Master functionality. Instead, PCI/ISA bridges will implement the PC/PCI DMA and/or Distributed DMA specification to fulfill these transactions. As a result, it is necessary for the PCI controller to implement support for at least one of these specifications to facilitate ISA DMA or Bus Master support.

Distributed DMA Distributed DMA is not supported in any of Intel's I/O Controller Hub variants.

3.11 PC/PCI DMA

The PC/PCI DMA protocol is supported on all I/O Controller Hubs from ICH to ICH5 (excluding 6300ESB). These parts have dedicated Request and Grant signals – REQ[A:B] and GNT[A:B] – to implement the hardware aspect of the protocol. From ICH6 onwards these signals have been removed and, therefore, these devices no longer support the PC/PCI protocol. As a result, it is no longer possible to support ISA DMA or Bus Master transactions using a PCI/ISA bridge. A system designer should be aware of this limitation before using such a bridge.

If a system designer does not require ISA DMA or Bus Master functionality then it may still be possible to use the PCI/ISA bridge without the presence of the PC/PCI Request and Grant signals. It is recommended that a system designer works with the bridge vendor to understand if this approach is fea

UNIT - IV

Computer Based Medical Instrumentaion – SBMA7006

COMPUTERISED DATA ACQUISITION

Overview of GPIB – System and Implementation, commands – primary command, secondary commands, evice specific commands, expanding GPIB, Sharing GPIB,SCPI- Generalized Instrument Model.

4.1 overview of GPIB

The GPIB refers to general purpose interface bus. It is one of the standard interfaces available in many standalone, general purpose, high performance instruments for data acquisition and control, employing PC. In 1965, Hewlett-Packard designed the Hewlett-Packard Interface Bus (**HP-IB**) to connect their line of programmable instruments to their computers. Because of its high transfer rates (nominally 1 Mbytes/s), this interface bus quickly gained popularity. It was later accepted as IEEE Standard 488-1975, and has evolved to ANSI/IEEE Standard **488.1**-1987. Today, the name **G** eneral **P**urpose **I**nterface **B**us (**GPIB**) is more widely used than HP-IB. ANSI/IEEE **488.2**-1987 strengthened the original standard by defining precisely how controllers and instruments communicate. **S**tandard **C**ommands for **P**rogrammable **I**nstruments (**SCPI**) took the command structures defined in IEEE 488.2 and created a single, comprehensive programming command set that is used with any SCPI instrument. The below figure shows the minimum requirement of a GPIB system and minimal system configuration respectively.

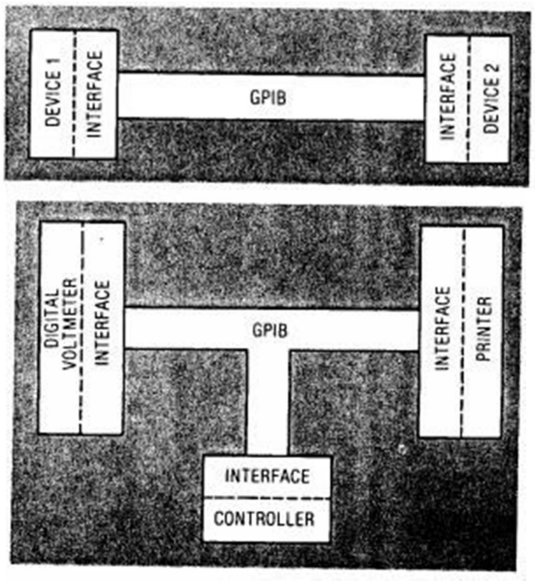


Figure 4.1 : GPIB Architecture

GPIB devices communicate with other GPIB devices by sending device-dependent messages and interface messages through the interface system. **Device-dependent** messages, often called **data** or data messages, contain device-specific information, such as programming instructions, measurement results, machine status, and data files. **Interface** messages manage the bus. Usually called **commands** or command messages, interface messages perform such functions as initializing the bus, addressing and unaddressing devices, and setting device modes for remote or local programming.

The term "command" as used here should not be confused with some device instructions that are also called commands. Such device-specific commands are actually data messages as far as the GPIB interface system itself is concerned.

GPIB Devices can be Talkers, Listeners, and/or Controllers. A **Talker** sends data messages to one or more **Listeners**, which receive the data. The **Controller** manages the flow of information on the GPIB by sending commands to all devices. A digital voltmeter, for example, is a Talker and is also a Listener. The figure shows the general view of GPIB interface, outline of the GPIB interface functions.

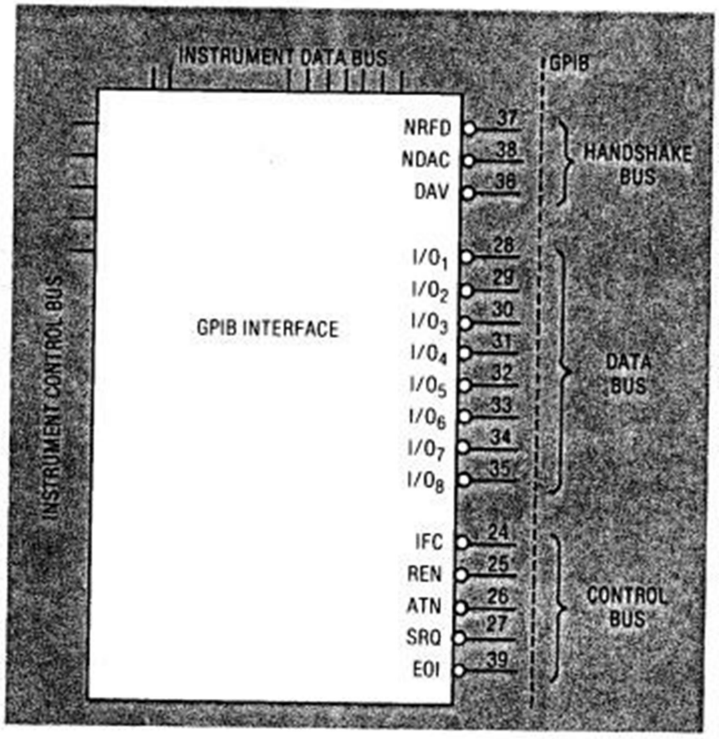


Figure 4.2 : GPIB Interface

The GPIB is like an ordinary computer bus, except that a computer has its circuit cards interconnected via a backplane - the GPIB has stand-alone devices interconnected by standard cables. The role of the GPIB Controller is comparable to the role of a computer CPU, but a better analogy is to compare the Controller to the switching center of a city telephone system.The switching center (Controller) monitors the communications network (GPIB). When the center (Controller) notices that a party (device) wants to make a call (send a data message), it connects the caller (Talker) to the receiver (Listener).The Controller usually addresses (or enables) a Talker and a Listener before the Talker can send its message to the Listener. After the message is transmitted, the Controller may address other Talkers and Listeners.

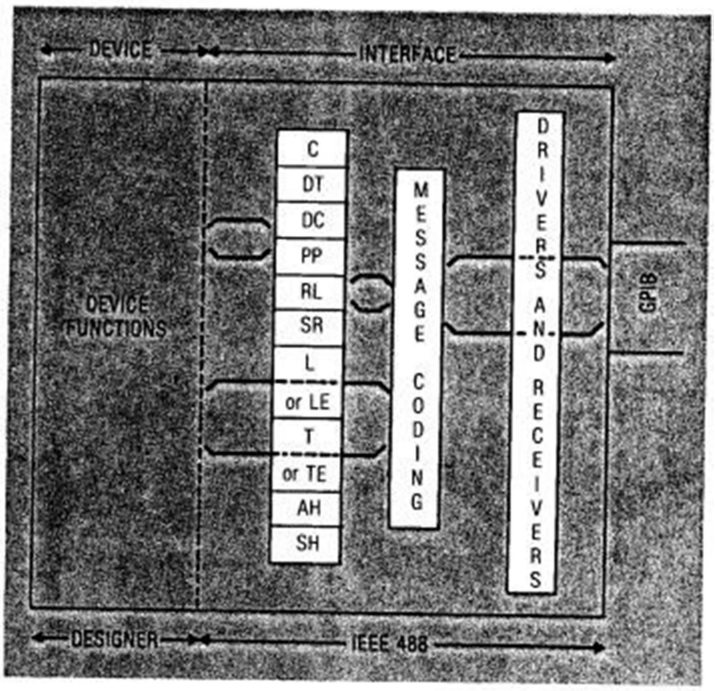


Figure 4.3 : GPIB device function

Some GPIB configurations do not require a Controller. For example, a device that is always a Talker, called a talk-only device, is connected to one or more listen-only devices. A Controller is necessary when the active or addressed Talker or Listener must be changed. The Controller function is usually handled by a computer. A computer with the appropriate hardware and software could perform the roles of Talker/Listener and Controller.

Data Lines

The eight data lines, DIO1 through DIO8, carry both data and command messages. The state of the Attention (ATN) line determines whether the information is data or commands. All commands and most data use the 7-bit ASCII or ISO code set, in which case the eighth bit, DIO8, is either unused or used for parity.

Handshake Lines

Three lines **asynchronously** control the transfer of message bytes between devices. The process is called a 3-wire interlocked handshake. It guarantees that message bytes on the data lines are sent and received without transmission error.

NRFD (not ready for data) - Indicates when a device is ready or not ready to receive a message byte. The line is driven by all devices when receiving commands, by Listeners when receiving data messages, and by the Talker when enabling the HS488 protocol.

NDAC (not data accepted) - Indicates when a device has or has not accepted a message byte. The line is driven by all devices when receiving commands, and by Listeners when receiving data messages.

DAV (data valid) - Tells when the signals on the data lines are stable (valid) and can be accepted safely by devices. The Controller drives DAV when sending commands, and the Talker drives DAV when sending data messages.

The standard IEEE 488.1 3-wire handshake (shown in Figure 9) requires the Listener to unassert Not Ready for Data (NRFD), the Talker to assert the Data Valid (DAV) signal to indicate to the Listener that a data byte is available, and for the Listener to unassert the Not Data Accepted (NDAC) signal when it has accepted that byte. A byte cannot transfer in less than the time it takes for the following events to occur:

- NRFD to propagate to the Talker,
- DAV signal to propagate to all Listeners,
- the Listeners to accept the byte and assert NDAC,
- the NDAC signal to propagate back to the Talker, and
- the Talker to allow a settling time (T1) before asserting DAV again.

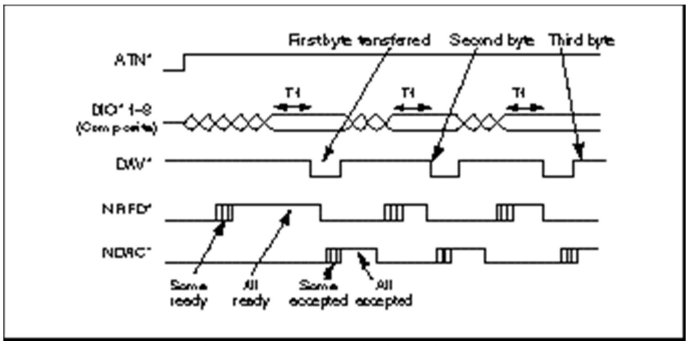


Figure 4.4 : GPIB Timing Diagram

Interface Management Lines

Five lines manage the flow of information across the interface:

- ATN (attention) - The Controller drives ATN true when it uses the data lines to send commands, and drives ATN false when a Talker can send data messages.
- IFC (interface clear) - The System Controller drives the IFC line to initialize the bus and become CIC.
- REN (remote enable) - The System Controller drives the REN line, which is used to place devices in remote or local program mode.
- SRQ (service request) - Any device can drive the SRQ line to asynchronously request service from the Controller.
- EOI (end or identify) - The EOI line has two purposes - The Talker uses the EOI line to mark the end of a message string, and the Controller uses the EOI line to tell devices to identify their response in a parallel poll.

To achieve the high data transfer rate for which the GPIB was designed, the physical distance between devices and the number of devices on the bus are limited.

The following restrictions are typical for normal operation:

A maximum separation of 4 m between any two devices and an average separation of **2 m** over the entire bus. A maximum total cable length of **20 m**. No more than **15 device** loads connected to each bus, with no less than two-thirds powered on For higher speed systems using the 3-wire IEEE 488.1 handshake (T1 delay = 350 ns), and HS488 systems, the following restrictions apply. A maximum total cable length of 15 m with a device load per 1 m cable. All devices should be powered on. All devices should use 48 mA three-state drivers. Device capacitance on each GPIB signal should be less than 50 pF per device.

4.2 COMMANDS

SCPI became defined with the IEEE 488.2 specification. The standard specifies a common syntax, command structure, and data formats, to be used with all instruments. It introduced generic commands (such as CONFigure and MEASure) that could be used with any instrument. These commands are grouped into subsystems. SCPI also defines several classes of instruments. For example, any controllable power supply would implement the same DCPSUPPLY base functionality class. Instrument classes specify which subsystems they implement, as well as any instrument-specific features.

The physical communications link is not defined by SCPI. While originally created for IEEE-488 (GPIB), it can also be used with RS-232, Ethernet, USB, VXIbus,HiSLIP, etc.

SCPI commands are ASCII textual strings, which are sent to the instrument over the physical layer (e.g., IEEE-488). Commands are a series of one or more keywords, many of which take parameters. In the specification, keywords are written CONFigure : The entire keyword can be used, or it can be abbreviated to just the uppercase portion. Responses to query commands are typically ASCII strings. However, for bulk data, binary formats can be used.

Command syntax

SCPI commands to an instrument may either perform a *set* operation (e.g. switching a power supply on) or a *query* operation (e.g. reading a voltage). Queries are issued to an instrument by appending a question-mark to the end of a command. Some commands can be used for both setting and querying an instrument. For example, the data-acquisition mode of an instrument could be set by using the ACQuire:MODe command or it could be queried by using the ACQuire:MODe? command. Some commands can both set and query an instrument at once. Fo r example, the *CAL? command runs a self-calibration routine on some equipment, and then returns the results of the calibration.

Similar commands are grouped into a hierarchy or "tree" structure. For example, any instruction to read a measurement from an instrument will begin with " MEASure ". Specific sub-commands within the hierarchy are nested with a colon (:) character. For example, the command to

"Measure a DC voltage" would take the form `MEASure:VOLTage:DC?`, and the command to "Measure an AC current" would take the form `MEASure:CURRent:AC?` .

Arguments

Some commands require an additional argument. Arguments are given after the command, and are separated by a space. For example, the command to set the trigger mode of an instrument to "normal" may be given as " `TRIGger:MODe NORMal` ". Here, the word " `NORMal` " is used as the argument to the " `TRIGger:MODe` " command. The below figure shows the GPIB commands and its address code.

Table 4.1 GPIB commands

	ATN MOST SIGNIFICANT DIGIT							
	ADDRESS COMMAND GROUP	UNIVERSAL COMMAND GROUP		LISTEN ADDRESS GROUP	TALK ADDRESS GROUP		SECOND COMMAND GROUP	
HEX	0	1	2	3	4	5	6	7
0	—	—	32	48	64	80	96	112
1	GTL	LLO	33	49	65	81	97	113
2	—	—	34	50	66	82	98	114
3	—	—	35	51	67	83	99	115
4	SDC	DCL	36	52	68	84	100	116
5	PPC	PPU	37	53	69	85	101	117
6	—	—	38	54	70	86	102	118
7	—	—	39	55	71	87	103	119
8	GET	SPE	40	56	72	88	104	120
9	TCT	SPD	41	57	73	89	105	121
A	—	—	42	58	74	90	106	122
B	—	—	43	59	75	91	107	123
C	—	—	44	60	76	92	108	124
D	—	—	45	61	77	93	109	125
E	—	—	46	62	78	94	110	126
F	—	—	47	UNL	79	UNT	111	127

LEAST SIGNIFICANT DIGIT

Concatenating commands

Multiple commands can be issued to an instrument in a single string. Each command must be separated by a semicolon character (`;`). Additionally, all commands except the first must be prefixed by a colon (unless they already begin with an asterisk). For example, the command to "Measure a DC voltage then measure an AC current" would be issued as `MEASure:VOLTage:DC?;:MEASure:CURRent:AC?` .

Abbreviating commands

The command syntax shows some characters in a mixture of upper and lower case. Abbreviating the command to only sending the upper case has the same meaning as sending the upper and lower case command.

For example, the command " `SYSTem:COMMunicate:SERial:BAUD 2400` " would set an RS-232 serial communications interface to 2400 bit/s. This could also alternatively be abbreviated "`SYST:COMM:SER:BAUD 2400`".

The query command

" SYSTem:COMMunicate:SERial:BAUD? " or " SYST:COMM:SER:BAUD? " would instruct the instrument to report its current baud rate.. The below figure shows that summary of GPIB address group.

DAG		LAG		TAG		SCG	
0	1	2	3	4	5	6	7
0	16	0	16	0	16	0	16
1	17	1	17	1	17	1	17
2	18	2	18	2	18	2	18
3	19	3	19	3	19	3	19
4	20	4	20	4	20	4	20
5	21	5	21	5	21	5	21
6	22	6	22	6	22	6	22
7	23	7	23	7	23	7	23
8	24	8	24	8	24	8	24
9	25	9	25	9	25	9	25
10	26	10	26	10	26	10	26
11	27	11	27	11	27	11	27
12	28	12	28	12	28	12	28
13	29	13	29	13	29	13	29
14	30	14	30	14	30	14	30
15	31	15	31	15	31	15	31

Figure 4.5 : GPIB Commands

The figure shows the Example of code transfer with a secondary address

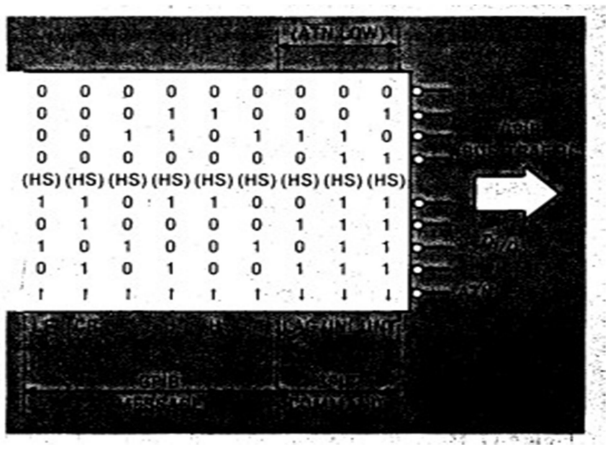


Figure 4.6 : GPIB Commands Examples

PROGRAMMING

The transmission of ASCII characters from talkers to listeners via the GPIB. In practice the situation is more complicated, Because some characters sent down the bus are not the ASCII code. The below figure shows the different types of eight bit characters that can be sent on the GPIB. The type of character sent depends on the voltage condition on the ATN line.

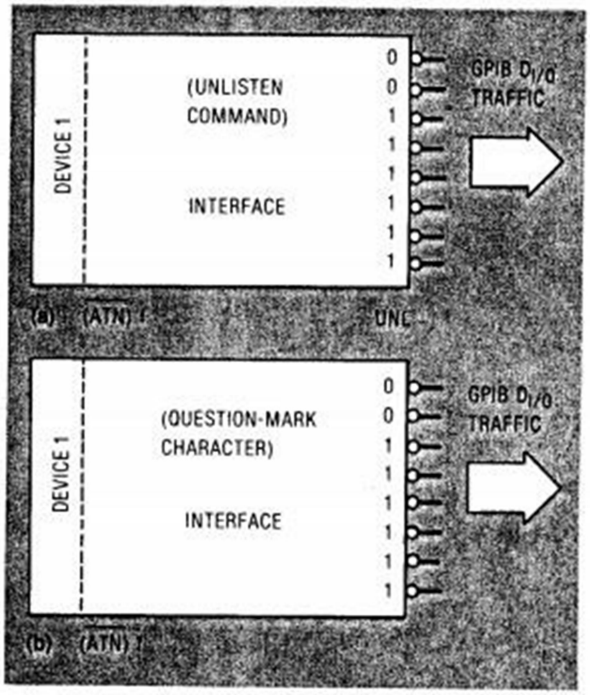


Figure 4.7 : GPIB communication

4.3 EXPANDING GPIB

Five lines manage the flow of information across the interface:

- ATN (attention) - The Controller drives ATN true when it uses the data lines to send commands, and drives ATN false when a Talker can send data messages.
- IFC (interface clear) - The System Controller drives the IFC line to initialize the bus and become CIC.
- REN (remote enable) - The System Controller drives the REN line, which is used to place devices in remote or local program mode.
- SRQ (service request) - Any device can drive the SRQ line to asynchronously request service from the Controller.
- EOI (end or identify) - The EOI line has two purposes - The Talker uses the EOI line to mark the end of a message string, and the Controller uses the EOI line to tell devices to identify their response in a parallel poll.

Devices are usually connected with a shielded 24-conductor cable with both a plug and receptacle connector at each end You can link devices in either a linear configuration a star configuration or a combination of the two.

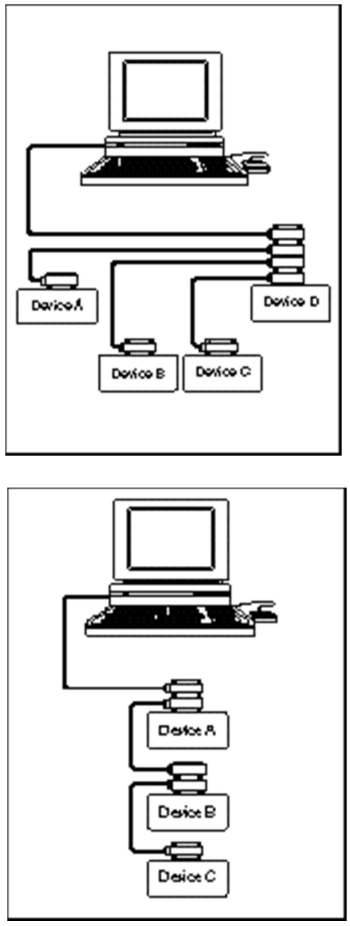


Figure 4.8 : Sharing devices

4.4 SCPI

The SCPI and IEEE 488.2 standards addressed the limitations and ambiguities of the original IEEE 488 standard. **IEEE 488.2** makes it possible to design more compatible and productive test systems. **SCPI**simplifies the programming task by defining a single comprehensive command set for programmable instrumentation, regardless of type or manufacturer. The scope of each of the IEEE 488, IEEE 488.2, and SCPI standards is shown in Figure

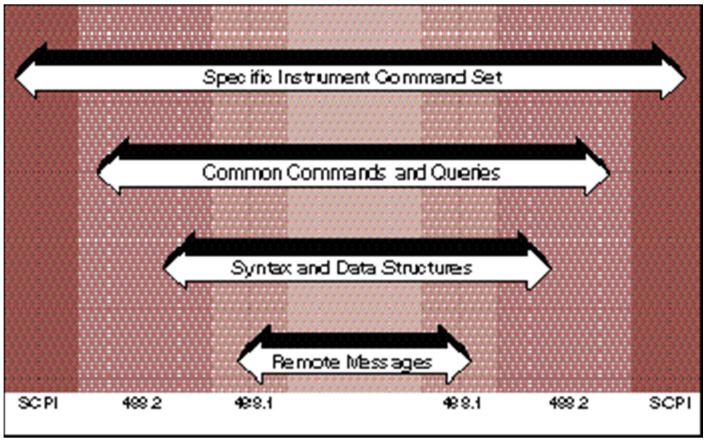


Figure 4.9 : SPCI

The ANSI/IEEE Standard 488-1975, now called **IEEE 488.1**, greatly simplified the interconnection of programmable instrumentation by clearly defining mechanical, electrical, and hardware protocol specifications. For the first time, instruments from different manufacturers were interconnected by a standard cable. Although this standard went a long way towards improving the productivity of test engineers, the standard did have a number of shortcomings. Specifically, IEEE 488.1 did not address data formats, status reporting, message exchange protocol, common configuration commands, or device-specific commands. As a result, each manufacturer implemented these items differently, leaving the test system developer with a formidable task.

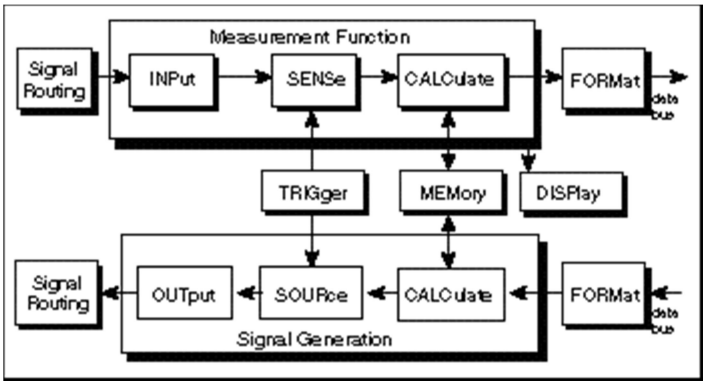


Figure 4.10 : SPCI instrument Model

IEEE 488.2 enhanced and strengthened IEEE 488.1 by standardizing data formats, status reporting, error handling, Controller functionality, and common commands to which all instruments must respond in a defined manner. By standardizing these issues, IEEE 488.2 systems are much more compatible and reliable. The IEEE 488.2 standard focuses mainly on the software protocol issues and thus maintains compatibility with the hardware-oriented IEEE 488.1 standard.

SCPI built on the IEEE 488.2 standard and defined device-specific commands that standardize programming instruments. SCPI systems are much easier to program and maintain. In many cases, you can interchange or upgrade instruments without having to change the test program. The combination of SCPI and IEEE 488.2 offers significant productivity gains, and finally, delivers as sound a software standard as IEEE 488.1 did a hardware standard.

IEEE 488.2

IEEE 488.2-1987 encouraged a new level of growth and acceptance of the IEEE 488 bus or GPIB by addressing problems that had arisen from the original IEEE 488 standard. IEEE 488.2 was drafted on the premise that it stay compatible with the existing IEEE 488.1 standard. The overriding concept used in the IEEE 488.2 specification for the communication between Controllers and instruments is that of "precise talking" and "forgiving listening." In other words, IEEE 488.2 exactly defined how both IEEE 488.2 Controllers and IEEE 488.2 instruments talk so that a completely IEEE 488.2-compatible system can be highly reliable and efficient. The standard also required that IEEE 488.2 devices be able to work with existing IEEE 488.1 devices by accepting a wide range of commands and data formats as a Listener. You obtain the true benefits of IEEE 488.2 when you have a completely IEEE 488.2-compatible system.

On April 23, 1990, a group of instrument manufacturers announced the SCPI specification, which defines a common command set for programming instruments. Before SCPI, each instrument manufacturer developed its own command sets for its programmable instruments. This lack of standardization forced test system developers to learn a number of different command sets and instrument-specific parameters for the various instruments used in an application, leading to programming complexities and resulting in unpredictable schedule delays and development costs. By defining a standard programming command set, SCPI decreases development time and increases the readability of test programs and the ability to interchange instruments.

SCPI is a complete, yet extendable, standard that unifies the software programming commands for instruments. The first version of the standard was released in mid-1990. Today, the SCPI Consortium continues to add commands and functionality to the SCPI standard. SCPI has its own set of required common commands in addition to the mandatory IEEE 488.2 common commands and queries. Although IEEE 488.2 is used as its basis, SCPI defines programming commands that you can use with any type of hardware or communication link.

SCPI specifies standard rules for abbreviating command keywords and uses the IEEE 488.2 message exchange protocol rules to format commands and parameters. You may use command keywords in their long form (MEASure) or their short form shown in capital letters (MEAS).

SCPI offers numerous advantages to the test engineer. One of these is that SCPI provides a comprehensive set of programming functions covering all the major functions of an instrument. This standard command set ensures a higher degree of instrument interchangeability and minimizes the effort involved in designing new test systems. The SCPI command set is hierarchical, so adding commands for more specific or newer functionality is easily accommodated.

UNIT - V

Computer Based Medical Instrumentaion – SBMA7006

DATA ACQUISTION USING SERIAL INTERFACE

Serial Communication–Features and Formats, Interface standard–RS232,RS-422,RS-485,PC,serial port UART, Micro controller serial interface–USB System and USB Transfer..

5.1 SERIAL COMMUNICATION

In serial communication, data bytes from a transmitting system are converted to a stream of bits and transferred to receiving system one bit at a time. The receiving system collects the bits and reassembles them back into original data bytes. In data parallel communication, all bits of a byte are transferred at the same instant. This makes parallel communication faster than serial communication. However, parallel communication cannot be used in all Serial applications. communication is preferred in many applications, particularly when Over transferring data long distances, since it has the following advantages.

Features of serial communication

a) In parallel communication, since all bits of a data byte are transferred at the same instant, separate wires are used to carry each bit. Hence parallel communication cables use many wires. Serial communication cables use only limited number of Wires, usually four: two for transmission and two for reception.

b) Parallel communication cannot use long cables. When signals are carried over long stances using cables, electromagnetic interference creates problems at higher data ister rates. Each wire in the communication cable acts as antenna and captures lot O1Se from environment. It corrupts the data being transferred. Since serial Lunication uses only limited wires, it can use long wires. The signals can be Orected from electromagnetic interference by shielding the wires.

(c) Another problem with parallel communication is that though all bits of a data are transmitted at the same instant by transmitter, the bits do not reach the Toe the same instant. Hence the receiver should wait till all bits to arrive in. It o causes delay in communication.

(d) Parallel communication is not always faster than serial communication. Serial communication today transfers data at faster rate than parallel communication Hence serial communication gains more and more importance today. For example traditional data transfer between hard disk and PC via IDE interface is in parall form at the speed of 133 MB/s. It is being replaced by serial interface known as Serial ATA (SATA) interface, which can transfer at the speed up to 150 MB/s. Similarly, PCI bus is being transformed into serial bus called PCI Express. SCSI bus is also being transformed into serial form

A popular way to transfer commands and data between a personal computer and a microcontroller is the use of standard interface, like the one described by protocols RS232 (olde R) or USB (newer). This chapter is devoted to communication conforming to RS232 protocol, th e hardware for such interface is provided on board. An example will be presented showing the pr ocessing of commands received through RS232 interface, and sending of a string of numbers usi ng the same interface.

The protocol RS232 defines the signals used in communication, and the hardware to transfer signals between devices. The time diagram of the typical signal used to transfer character 'A' (A SCII: 6510 or 0x41) from device A to device B is given in Fig. 1, and would appear on the upper line TX -> RX between devices.

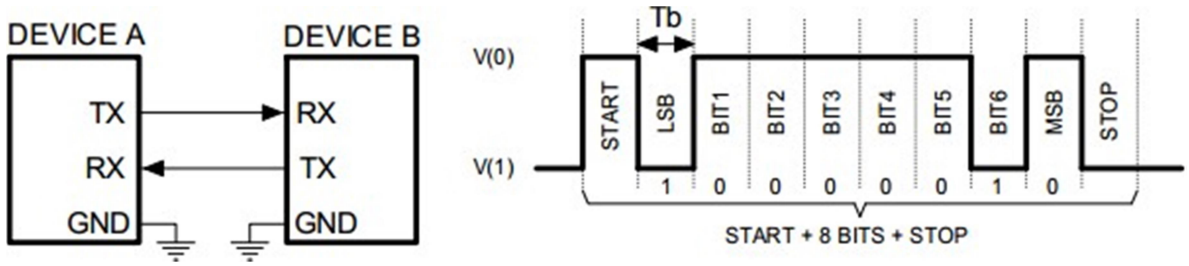


Figure 5.1 Serial Communication Architecture

In serial communication, data is in the form of binary pulses. In other words, we can say Binary One represents a logic HIGH or 5 Volts, and zero represents a logic LOW or 0 Volts. Serial communication can take many forms depending on the type of transmission mode and data transfer. The **transmission modes** are classified as Simplex, Half Duplex, and Full Duplex. There will be a source (also known as a *sender)* and destination (also called *a receiver*) for each transmission mode.

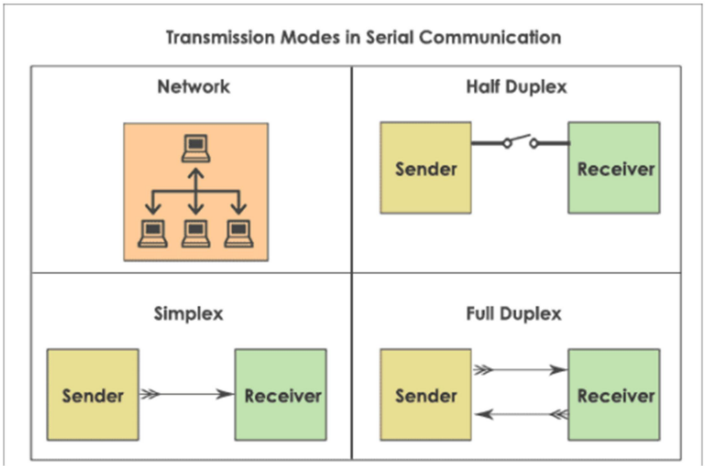


Figure 5.2 : serial communication transmission modes

The **Simplex method** is a one-way communication technique. Only one client (either the sender or receiver is active at a time). If a sender transmits, the receiver can only accept. Radio and Television transmission are the examples of simplex mode.

In **Half Duplex mode**, both sender and receiver are active but not at a time, i.e. if a sender transmits, the receiver can accept but cannot send and vice versa. A good example is an internet. If a client (laptop) sends a request for a web page, the web server processes the application and sends back the information.

The **Full Duplex mode** is widely used communication in the world. Here both sender and receiver can transmit and receive at the same time. An example is your smartphone. Beyond the transmission modes, we have to consider the endianness and protocol design of the host computer (sender or receiver). **Endianness** is the way of storing the data at a particular memory address. Depending on the data alignment endian is classified as

- Little Endian and
- Big Endian.

Take this example to understand the concept of endianness. Suppose, we have a 32-bit hexadecimal data **ABCD87E2**. How is this data stored in memory? To have a clear idea, I have explained the **difference between Little Endian and Big Endian**.

Figure 5.3 Little and Big Indian

Data transfer can happen in two ways. They are serial communication and parallel communication. Serial communication is a technique used to send data bit by bit using a two-wires i.e. transmitter (sender) and receiver.

For example, I want to send an 8-bit binary data **11001110** from the transmitter to the receiver. But, which bit goes out first? Most Significant Bit – MSB (7^{th} bit) or Least Significant Bit- LSB (0^{th} Bit). We cannot say. Here I am considering LSB is moving first (for little Endian).

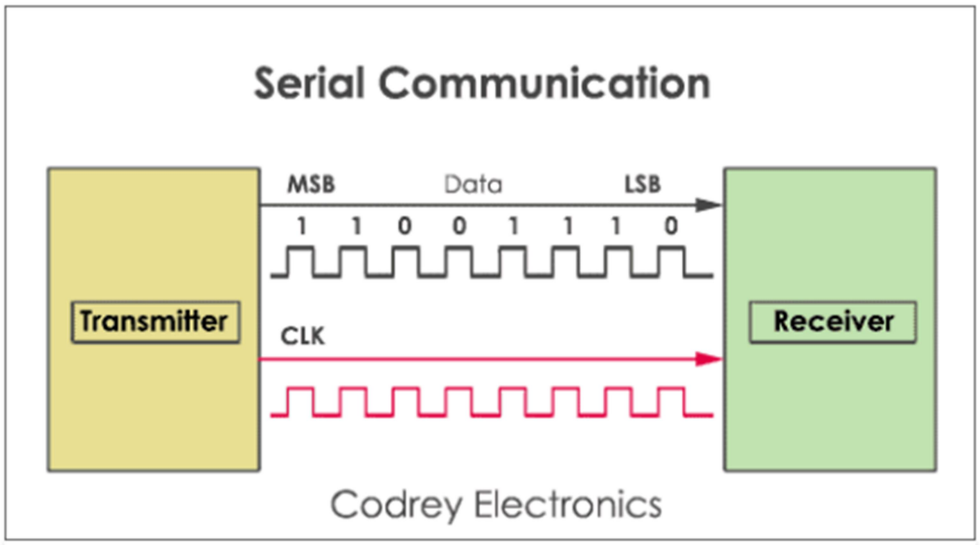


Figure 5.4 Serial Communication Example

From the above diagram, for every clock pulse; the transmitter sends a single bit of data to the receiver.

Parallel communication moves 8,16, or 32 bits of data at a time. Printers and Xerox machines use parallel communication for faster data transfer.

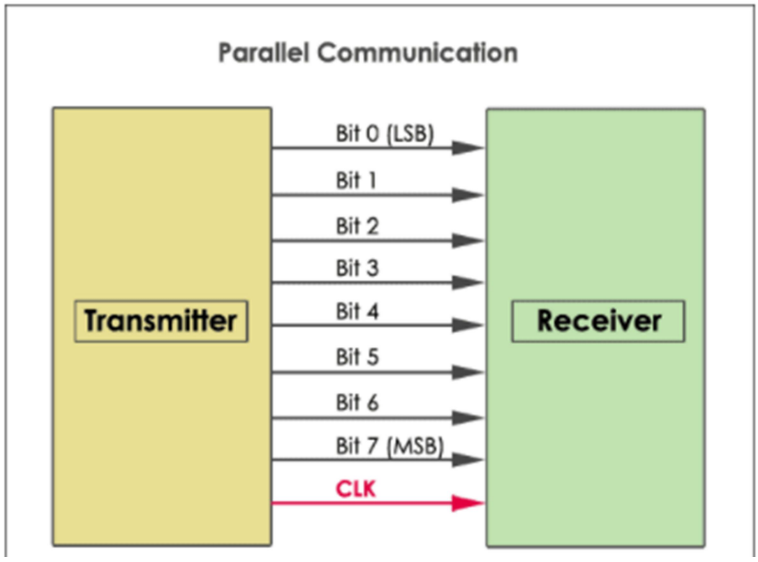


Figure 5.5 Parallel Communication

Difference between Serial and Parallel communication

Serial communication sends only one bit at a time. so, these require fewer I/O (input-output) lines. Hence, occupying less space and more resistant to cross-talk. The main advantage of serial communication is, the cost of the entire embedded system becomes cheap and transmits the information over a long distance. Serial transfer is used in DCE (Data communication Equipment) devices like a modem.

In parallel communication, a chunk of data (8,16 or 32 bit) is sent at a time. So, each bit of data requires a separate physical I/O line. The advantage of parallel communication is it is fast but its drawback is it use more number of I/O (input-output) lines. Parallel transfer is used in PC (personal computer) for interconnecting CPU (central processing unit), RAM (random access memory), modems, audio, video and network hardware.

For easy understanding, here is the comparison of serial and parallel communication.

Serial Communication	Parallel Communication
Sends data bit by bit at one clock pulse	Transfers a chunk of data at a time
Requires one wire to transmit the data	Requires 'n' number of lines for transmitting 'n' bits
Communication speed is slow	Communication speed is fast
Installation cost is low	Installation cost is high
Preferred for long distance communication	Used for short distance communication
Example: Computer to Computer	Computer to multi function printer

Clock Synchronization

For efficient working of serial devices, the clock is the primary source. Malfunction of the clock may lead to unexpected results. The clock signal is different for each serial device, and it is categorized as synchronous protocol and asynchronous protocol.

Synchronous serial interface

All the devices on *Synchronous* serial interface use the single CPU bus to share both clock and data. Due to this fact, data transfer is faster. The advantage is there will be no mismatch in baud rate. Moreover, fewer I/O (input-output) lines are required to interface components. Examples are I2C, SPI etc.

Asynchronous serial interface
The *asynchronous* interface does not have an external clock signal, and it relies on four parameters namely

1. Baud rate control
2. Data flow control
3. Transmission and reception control
4. Error control.

Asynchronous protocols are suitable for stable communication. These are used for long distance applications. Examples of asynchronous protocols are RS-232, RS-422, and RS-485.

Asynchronous Serial Protocols

The most common question that will come to mind when you start working on the embedded system is why to use Asynchronous protocols?

- To move around the information at a longer distance and
- For more reliable data transfer.

Some of the asynchronous communication protocols are:

RS-232 protocol

- RS232 is the first serial protocol used for connecting modems for telephony. RS stands for *Recommended Standard,* and now it has changed to EIA (*Electronic Industries Alliance*) / TIA (*Telecommunication Industry Association).*
- It is also used in modem, mouse, and CNC (computed numerical computing) machines. You can connect only a single transmitter to a single receiver.
- It supports full duplex communication and allows baud rate up to 1Mbps.
- Cable length is limited to 50 feet.

As you know, the data stored in the memory are in the form of bytes. You may have a doubt How is the byte-wise data converted to binary bits? The answer is a Serial port.

The serial port has an internal chip called **UART**. UART is an acronym for Universal Asynchronous Receiver Transmitter which converts the parallel data (byte) into the bitwise serial form.

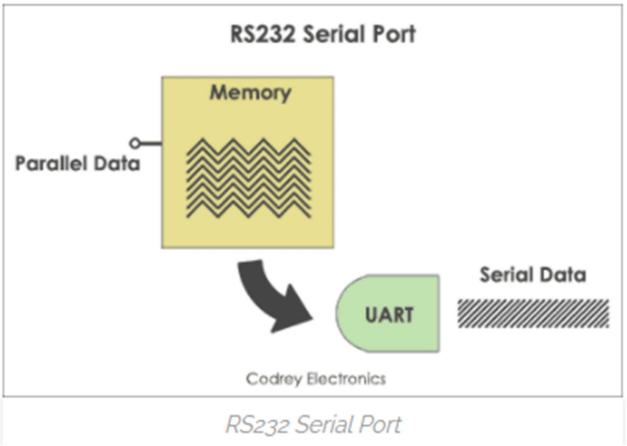


Figure 5.6 RS232 Serial Port

RS-232 Wiring Connection

The RS232 serial port has nine pins, male or female type models. *RS 232C serial communication interface* is the later version of RS232.

All the features present in RS232 is present in the RS232C model except it has 25 pins. Out of 25 or 9 pins, we use only three pins for the connection of terminal devices.

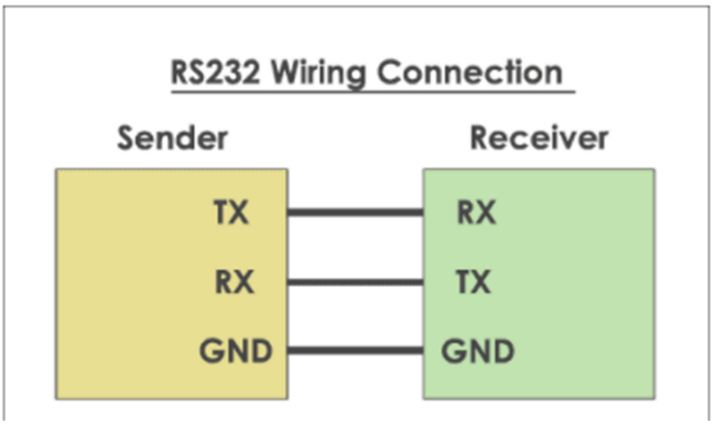


Figure 5.7 RS232 Wiring

RS422 Interface

We can transfer data only up to 1Mpbs limit using RS232. To overcome this problem RS422 comes into the picture. RS422 is a multi-drop serial interface. we can connect ten transmitters to 10 receivers at a time using the single bus. It sends data using two twisted pair cables (***differential configuration***). Cable length is 4000 feet with a baud rate of 10Mbps.

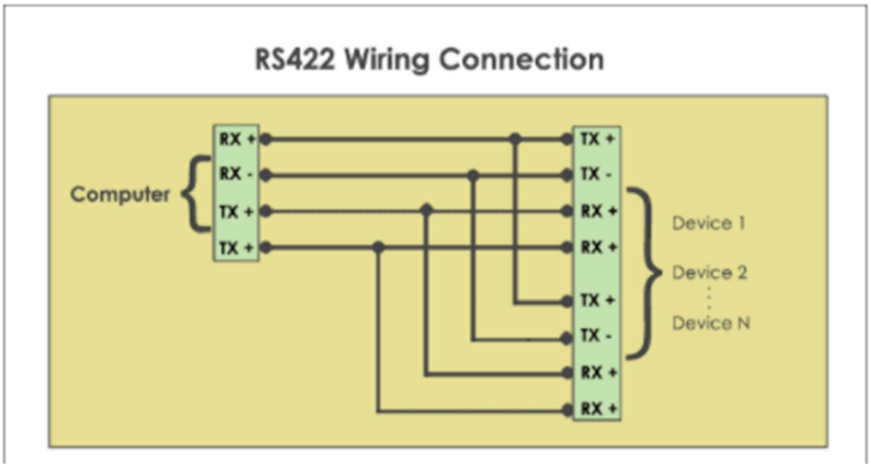


Figure 5.8 RS422 Wiring

RS485 Interface

RS485 is the industry preferred protocol. Unlike RS422, you can connect 32 line drivers and 32 receivers in a differential configuration. The transmitter is also called *Line driver*. However, only one transmitter is active at a time.

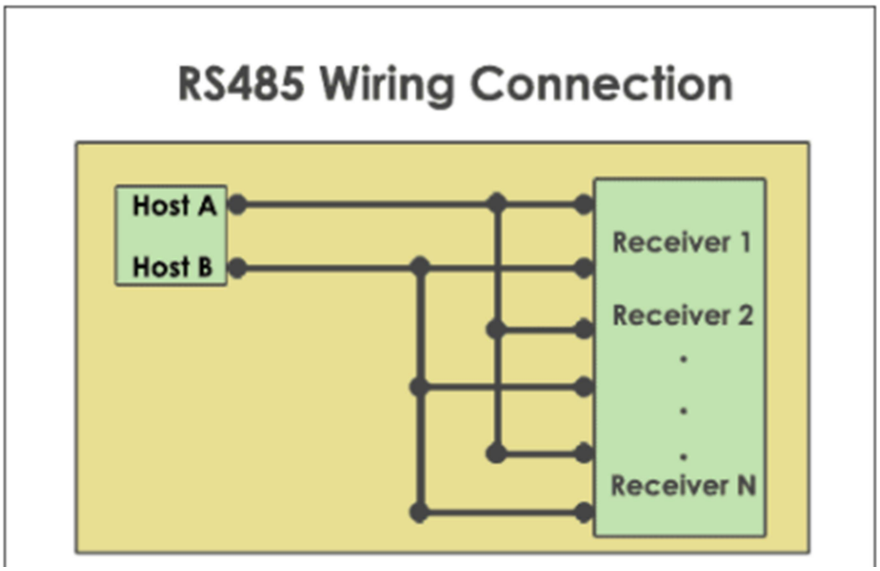


Figure 5.9 RS485 Wiring

5.2 PC SERIAL PORT

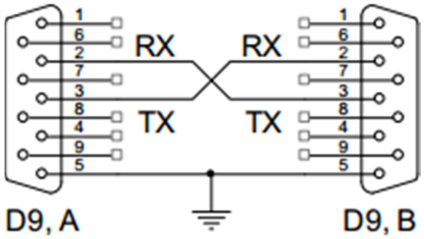


Figure 5.10 PC serial port

The standard defines voltage levels V(0) to be at least +5V at the transmitting end of the line TX, and can be degraded along the line to become at least +3V at the receiving and of the line. Simila rly voltage level V(1) must be at least -5V at TX, and at least -3V at RX. The standard also defin ed the upper limit for these voltages to be up to ±15V. Logic high is transferred as V(0). The mic rocontroller cannot handle such voltage levels, so typically a voltage level translator is inserted between the microcontroller and the connector where the RS232 signals are available. T he connectors are typically so-called D9 connectors, and the electric wiring in between two conn ectors at devices A and B is shown in Fig. 2, for two female type connectors at both devices.

To verify the validity of the transmission the protocol RS232 provides a so called “parity bit”. A single bit is added by the transmitter before the stop bit, and its value is selected to give either od d or even number of ones in a string. The number of ones in a string can be verified at the receivi ng end, and if it does not match the required value, the transmission should be repeated. Other, hi gher level protocols to ensure the valid transmission, can be implemented in software. The protocol

RS232 also accounts for testing if the receiver is capable of receiving incoming bytes and defines two additional wires called RTS (Request To Send) and CTS (Clear To Send) between devices. We will not use either of these options in our experiments.

To verify the validity of the transmission the protocol RS232 provides a so called “parity bit”. A single bit is added by the transmitter before the stop bit, and its value is selected to give either od d or even number of ones in a string. The number of ones in a string can be verified at the receivi ng end, and if it does not match the required value, the transmission should be repeated. Other, hi gher level protocols to ensure the valid transmission, can be implemented in software. The protocol RS232 also accounts for testing if the receiver is capable of receiving incoming bytes an d defines two additional wires called RTS (Request To Send) and CTS (Clear To Send) between devices. We will not use either of these options in our experiments.

5.3 INTERFACE STANDARD & MICROCONTROLLER SERIAL INTERFACE:

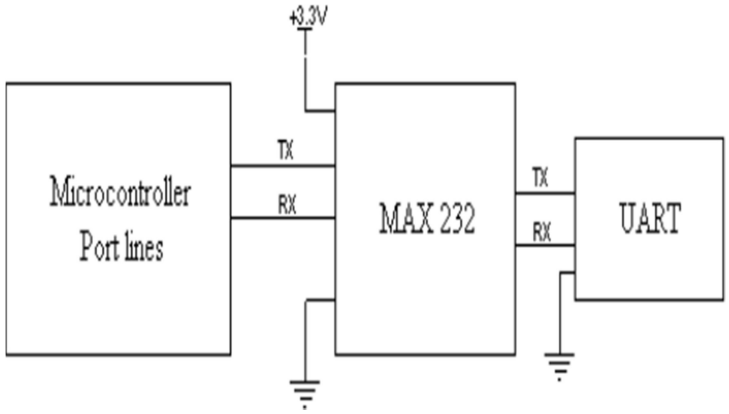


Figure 5.11 Interface of MAX to microcontroller

Fig. 1 shows how to interface the UART to microcontroller. To communicate over UART or USART, we just need three basic signals which are namely, RXD (receive), TXD (transmit), GND (common ground). So to interface UART with 8051, we just need the basic signals. Fig. 1 shows how to interface the UART to microcontroller. To communicate over UART or USART, we just need three basic signals which are namely, RXD (receive), TXD (transmit), GND (common ground). So to interface UART with 8051, we just need the basic signals.

Logic thresholds are dictated by the process and design implemented on each product. On some products, SPI inputs are 5 V or 3.3 V tolerant. However, other products may only accept smaller voltages (for example, 1.8 V). In this case, it might be necessary to incorporate a voltage level translator to adapt the levels from the microcontroller to levels acceptable to the inputs of the DUT. It is recommended that the customer uses the ADG3304 which is a four-channel bidirectional level translator. The translator can be customized to perform bidirectional logic level translation without an additional signal to set the direction in which the translation takes place. For instance, if the microcontroller is operating at 5 V supply and the SPI port is 1.8 V

tolerant, the ADG3304 can be set so that it translates the SCLK, SDI, and CSB signals from 5 V to 1.8 V. The ADG3304 is an easy-to-use solution that requires very few external components. Moreover, the user can disable the outputs from the translator at any time by pulling Pin 8 (EN) low. Figure 5 shows the implementation of the SPI boot circuit with the voltage level translator.