Rashtreeya Sikshana Samithi Trust's

Rashtreeya Vidyalaya Institute of Technology and Management (RVITM), Bengaluru

BASIC ELECTRONICS & COMMUNICATION ENGINEERING (21ELN14/24)

SEMESTER-I

Module-III

Prepared by

Dr. Madhumathy P
Associate Professor,
Dept.of E&C Engg,
RVITM, Bengaluru

Dr. Sushama
Assistant Professor
Dept.of E&C Engg
RVITM, Bengaluru

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Module 3	RBT Levels
Embedded Systems – Definition, Embedded systems vs general computing systems, Classification of Embedded Systems, Major application areas of Embedded Systems, Elements of an Embedded System, Core of the Embedded System, Microprocessor vs Microcontroller, RISC vs CISC, Harvard vs Von-Neumann. **Sensors and Interfacing** – Instrumentation and control systems, Transducers, Sensors. **Actuators**, LED, 7-Segment LED Display, Stepper Motor, Relay, Piezo Buzzer, Push Button Switch, Keyboard. **Communication Interface,** UART, Parallel Interface, USB, Wi-Fi, GPRS.	L2,L3

3.1. EMBEDDED SYSTEMS

3.1.1. WHAT IS AN EMBEDDED SYSTEM?

An embedded system is an electronic/electro-mechanical system designed to perform a specific function and is a combination of both hardware and firmware (software).

Every embedded system is unique, and the hardware as well as the firmware is highly specialised to the application domain. Embedded systems are becoming an inevitable part of any product or equipment in all fields including household appliances, telecommunications, medical equipment, industrial control, consumer products, etc.

3.1.2. EMBEDDED SYSTEMS vs. GENERAL COMPUTING SYSTEMS

The computing revolution began with the general purpose computing requirements. Later it was realized that the general computing requirements are not sufficient for the embedded computing requirements. The embedded computing requirements demand 'something special' in terms of response to stimuli, meeting the computational deadlines, power efficiency, limited memory availability, etc. Let's take the case of your personal computer, which may be either a desktop PC or a laptop PC or a palmtop PC. It is built around a general purpose processor like an Intel® Centrino or a Duo/Quad core or an AMD Turion™ processor and is designed to support a set of multiple peripherals like multiple USB 2.0 ports, Wi-Fi, ethernet, video port, IEEE1394, SD/CF/MMC external interfaces, Bluetooth, etc and with additional interfaces like a CD read/writer, on-board Hard Disk Drive (HDD), gigabytes of RAM, etc. You can load any supported operating system (like Windows® XP/Vista/7, or Red Hat Linux/Ubuntu Linux, UNIX etc) into the hard disk of your PC. You can write or purchase a multitude of applications for your PC and can use your PC for running a large number of applications (like printing your dear's photo using a printer device connected to the PC and printer software, creating a document using Microsoft® Office Word tool, etc.) Now let us think about the DVD player you use for playing DVD movies. Is it possible for you to change the operating system of your DVD? Is it possible for you to write an ap¬ plication and download it to your DVD player for executing? Is it possible for you to add a printer soft¬ ware to your DVD player and connect a printer to your DVD player to take a printout? Is it possible for you to change the functioning of your DVD player to a television by changing the embedded software? The answers to all these questions are 'NO'. Can you see any general purpose interface like Bluetooth or Wi-Fi

on your DVD player? Of course 'NO'. The only interface you can find out on the DVD player is the interface for connecting the DVD player with the display screen and one for controlling the DVD player through a remote (May be an IR or any other specific wireless interface). Indeed your DVD player is an embedded system designed specifically for decoding digital video and generating a video signal as output to your TV or any other display screen which supports the display interface supported by the DVD Player. Let us summarise our findings from the comparison of embedded system and general purpose computing system with the help of a Table 3.1.

Table 3.1 Comparison of embedded system and general purpose computing system

Criteria	General Purpose Computing System	Embedded System
Contents	A system which is a combination of a generic hardware and a General Purpose Operating System for executing a variety of applications.	A system which is a combination of special purpose hardware and embedded OS for executing a specific set of applications.
OS	It contains a general purpose operating system (GPOS).	It may or not contain an operating system for functioning.
Alterations	Applications are alterable (programmable) by the user. (It is possible for the end user to re-install the OS and also add or remove user applications.)	The firmware of the embedded system is pre-programmed and it is non-alterable by the end-user.
Key factor	Performance is the key deciding factor in the selection of the system. Faster is better.	Application specific requirements (like performance, power requirements, memory usage, etc.) are key deciding factors.
Power Consumption	More	Less
Response Time	Not critical	Critical for some applications
Execution	Need not be deterministic	Deterministic for certain types of ES like 'Hard Real Time' systems.

However, the demarcation between desktop systems and embedded systems in certain areas of embedded applications are shrinking in certain contexts. Smart phones are typical examples of this. Nowadays smart phones are available with RAM up to 256 MB and users can extend most of their desktop applications to the smart phones and it waives the clause "Embedded systems are designed for a specific application" from the characteristics of the embedded system for the mobile embedded device category. However, smart phones come with a built-in operating system and it is not modifiable by the end user. It makes the clause: "The firmware of the embedded system is unalterable by the end user", still a valid clause in the mobile embedded device category.

3.1.3. HISTORY OF EMBEDDED SYSTEMS

Embedded systems were in existence even before the IT revolution. In the olden days embedded systems were built around the old vacuum tube and transistor technologies and the embedded algorithm was developed in low level languages. Advances in semiconductor and nano-technology and IT revolution gave way to the development of miniature embedded systems. The first recognised modem embedded system is the Apollo Guidance Computer (AGC) developed by the MIT Instrumentation Laboratory for the lunar expedition. They ran the inertial guidance systems of both the Command Module (CM) and. the Lunar Excursion Module (LEM). The Command Module was designed to encircle the moon while the Lunar Module and its crew were designed to go down to the moon surface and land there safely. The Lunar Module featured in total 18 engines. There were 16 reaction control thrusters, a descent engine and an ascent engine. The descent engine was 'designed to' provide thrust to the lunar module out of the lunar orbit and land it safely on the moon. MIT's original design was based on 4K words of fixed memory (Read Only Memory) and 256 words of erasable memory (Random Access Memory). By June 1963, the figures reached 10K of fixed and IK of erasable memory. The final configuration was 36K words of fixed memory and 2K words of erasable memory. The clock frequency of the. first microchip proto model used in AGC was 1.024 MHz and it was derived from a 2.048 MHz crystal clock. The computing unit of AGC consisted of approximately 11 instructions and 16 bit word logic. Around 5000 ICs (3-input NOR gates, RTL logic) supplied by Fairchild Semiconductor were used in this design. The user interface unit of AGC is known as DSKY (display/keyboard). DSKY looked like a calculator type keypad with an array of numerals. It was used for inputting the commands to the module numerically.

The first mass-produced embedded system was the guidance computer for the Minuteman-I missile in 1961. It was the 'Autonetics D-l7 guidance computer, built using discrete transistor logic and a hard-disk for main memory. The first integrated circuit was produced in September 1958 but computers using them didn't begin to appear until 1963. Some of their early uses were in embedded systems, notably used by NASA for the Apollo Guidance Computer and by the US military in the Minuteman-II intercontinental ballistic missile.

3.1.4. CLASSIFICATION OF EMBEDDED SYSTEMS

It is possible to have a multitude of classifications for embedded systems, based on different criteria. Some of the criteria used in the classification of embedded systems are:

1. Based on generation
2. Complexity and performance requirements
3. Based on deterministic behaviour
4. Based on triggering.

The classification based on deterministic system behaviour is applicable for 'Real Time' systems. The application/task execution behaviour for an embedded system can be either deterministic or non- deterministic. Based on the execution behaviour, Real Time embedded systems are classified into Hard and Soft. We will discuss about hard and soft real time systems in a later chapter. Embedded Systems which are 'Reactive' in nature (Like process control systems in industrial control applications) can be classified based on the trigger. Reactive systems can be either event triggered or time triggered.

3.1.4.1. CLASSIFICATION BASED ON GENERATION

This classification is based on the order in which the embedded processing systems evolved from the first version to where they are today. As per this criterion, embedded systems can be classified into:

3.1.4.1.1. First Generation The early embedded systems were built around 8bit microprocessors like 8085 and Z80, and 4bit microcontrollers. Simple in hardware circuits with firmware developed in Assembly code. Digital telephone keypads, stepper motor control units etc. are examples of this.

3.1.4.1.2. Second Generation These are embedded systems built around 16bit microprocessors and 8 or 16 bit microcontrollers, following the first generation embedded systems. The instruction set for the second generation processors/controllers were much more complex and powerful than the first generation processors/controllers. Some of the second generation embedded systems contained embedded operating systems for their operation. Data Acquisition Systems, SCADA systems, etc. are examples of second generation embedded systems.

3.1.4.1.3. Third Generation With advances in processor technology, embedded system developers started making use of powerful 32bit processors and 16bit microcontrollers for their design. A new concept of application and domain specific processors/controllers like Digital Signal Processors (DSP) and Application Specific Integrated Circuits (ASICs) came into the picture. The instruction set of processors became more complex and powerful and the concept of instruction pipelining also evolved. The processor market was flooded with different types of processors from different vendors. Processors like Intel Pentium, Motorola 68K, etc. gained attention in high performance embedded requirements. Dedicated embedded real time and general purpose operating systems entered into the embedded market. Embedded systems spread its ground to areas like robotics, media, industrial process control, networking, etc.

3.1.4.1.4. Fourth Generation The advent of System on Chips (SoC), reconfigurable processors and multicore processors are bringing high performance, tight integration and miniaturisation into the embedded device market. The SoC technique implements a total system on a chip by integrating different functionalities with a processor core on an integrated circuit. We will discuss about SoCs in a later chapter. The fourth generation embedded systems are making use of high performance real time embedded operating systems for their functioning. Smart phone devices, mobile internet devices (MIDs), etc. are examples of fourth generation embedded systems.

3.1.4.1.5. What Next? The processor and embedded market is highly dynamic and demanding. So 'what will be the next smart move in the next embedded generation?' Let's wait and see.

3.1.4.2 CLASSIFICATION BASED ON COMPLEXITY AND PERFORMANCE

This classification is based on the complexity and system performance requirements. According to this classification, embedded systems can be grouped into:

3.1.4.2.1 Small-Scale Embedded Systems Embedded systems which are simple in application needs and where the performance requirements are not time critical fall under this category. An electronic toy is a typical example of a small-scale embedded system. Small-scale embedded systems are usually built around low performance and low cost 8 or 16 bit microprocessors/microcontrollers. A small-scale embedded system may or may not contain an operating system for its functioning.

3.1.4.2.2 Medium-Scale Embedded Systems Embedded systems which are slightly complex in hardware and firmware (software) requirements fall under this category. Medium-scale embedded systems are usually built around medium performance, low cost 16 or 32 bit microprocessors/microcontrollers or digital signal processors. They usually contain an embedded operating system (either general, purpose or real time operating system) for functioning.

3.1.4.2.3 Large-Scale Embedded Systems/Complex Systems Embedded systems which involve highly complex hardware and firmware requirements fall under this category. They are employed in mission critical applications demanding high performance. Such systems are commonly built around high performance 32 or 64 bit RISC processors/controllers or Reconfigurable System on Chip (RSoC) or multi-core processors and programmable logic devices. They may contain multiple processors/controllers and co-units/hardware accelerators for offloading the processing requirements from the main processor of the system. Decoding/encoding of media, cryptographic function implementation, etc. are examples for processing requirements which can be implemented using a co-processor/hard- ware accelerator. Complex embedded systems usually contain a-high-performance Real-Time Operating System (RTOS) for task scheduling, prioritization and management.

3.1.5. MAJOR APPLICATION AREAS OF EMBEDDED SYSTEMS

We are living in a world where embedded systems play a vital role in our day-to-day life, starting from home to the computer industry, where most of .the people find their job for a livelihood. Embedded technology has acquired a new dimension from its first generation model, the Apollo guidance computer, to the latest radio navigation system combined with in-car entertainment technology and the microprocessor based “Smart” running shoes launched by Adidas in April 2005. The application areas and the products in the embedded domain are countless. A few of the important domains and products are listed below:

1. Consumer electronics: Camcorders, cameras, etc.
2. Household appliances: Television, DVD players, washing machine, fridge, microwave oven, etc.
3. Home automation and security systems: Air conditioners, sprinklers, intruder detection alarms, closed circuit television cameras, fire alarms, etc.

4. Automotive industry: Anti-lock breaking systems (ABS), engine control, ignition systems, automatic navigation systems, etc.
5. Telecom: Cellular telephones, telephone switches, handset multimedia applications, etc.
6. Computer peripherals: Printers, scanners, fax machines, etc.
7. Computer networking systems: Network routers, switches, hubs, firewalls, etc.
8. Healthcare: Different kinds of scanners, EEG, ECG machines etc.
9. Measurement & Instrumentation: Digital multi meters, digital CROs, logic analyzers PLC systems, etc.
10. Banking & Retail: Automatic teller machines (ATM) and currency counters, point of sales (POS)
11. Card Readers: Barcode, smart card readers, hand held devices, etc.

3.1.6. CORE OF THE EMBEDDED SYSTEM

Embedded systems are domain and application specific and are built around a central core. The core of the embedded system falls into any one of the following categories:

1. General Purpose and Domain Specific Processors
1.1 Microprocessors
1.2 Microcontrollers
1.3 Digital Signal Processors
2. Application Specific Integrated Circuits (ASICs)
3. Programmable Logic Devices (PLDs)
4. Commercial off-the-shelf Components (COTS)

If you examine any embedded system you will find that it is built around any of the core units mentioned above.

3.1.7. RISC VS. CISC PROCESSORS/CONTROLLERS

The term RISC stands for Reduced Instruction Set Computing. As the name implies, all RISC processors/controllers possess lesser number of instructions, typically in the range of 30 to 40. CISC stands for Complex Instruction Set Computing. From the definition itself it is clear that the instruction set is complex and instructions are high in number. From a programmers point of view RISC processors are comfortable since s/he needs to learn only a few instructions, whereas for a CISC processor s/he needs to learn more number of instructions and should

understand the context of usage of each instruction (This scenario is explained on the basis of a programmer following Assembly Language coding. For a programmer following C coding it doesn't matter since the cross-compiler is responsible for the conversion of the high level language instructions to machine dependent code). Atmel AVR microcontroller is an example for a RISC processor and its instruction set contains only 32 instructions. The original version of 8051 microcontroller (e.g. AT89C51) is a CISC controller and its instruction set contains 255 instructions. Remember it is not the number of instructions that determines whether a processor/controller is CISC or RISC. There are some other factors like pipelining features, instruction set type, etc. for determining the RISC/CISC criteria. Some of the important criteria are listed below:

Table 3.2 RISC vs. CISC

RISC	CISC
Lesser number of instructions	Greater number of Instructions
Instruction pipelining and increased execution speed	Generally no instruction pipelining feature
Orthogonal instruction set (Allows each instruction to operate on any register and use any addressing mode)	Non-orthogonal instruction set (All instructions are not allowed to operate on any register and use any addressing mode. It is instruction-specific)
Operations are performed on registers only, the only memory operations are load and store	Operations are performed on registers or memory depending on the instruction
A large number of registers are available	Limited number of general purpose registers
Programmer needs to write more code to execute a task since the instructions are simpler ones	Instructions are like macros in C language. A programmer can achieve the desired functionality with a single instruction which in turn provides the effect of using more simpler single instructions in RISC
Single, fixed length instructions	Variable length instructions
Less silicon usage and pin count	More silicon usage since more additional decoder logic is required to implement the complex instruction decoding.
With Harvard Architecture	Can be Harvard or Von-Neumann Architecture

3.1.8. HARVARD VS. VON-NEUMANN PROCESSOR/CONTROLLER ARCHITECTURE

The terms Harvard and Von-Neumann refers to the processor architecture design.

Microprocessors/controllers based on the Von-Neumann architecture shares a single common bus for fetching both instructions and data. Program instructions and data are stored in a common main memory. Von-Neumann architecture based processors/controllers first fetch an instruction and then fetch the data to support the instruction from code memory. The two

separate fetches slows down the controller's operation. Von-Neumann architecture is also referred as Princeton architecture, since it was developed by the Princeton University.

Microprocessors/controllers based on the Harvard architecture will have separate data bus and instruction bus. This allows the data transfer and program fetching to occur simultaneously oh both buses. With Harvard architecture, the data memory, can be read and written while the program memory is being accessed. These separated data memory and code memory buses allow one instruction to execute while the next instruction is fetched ("pre-fetching"). The pre-fetch theoretically allows much faster execution than Von-Neumann architecture. Since some additional hardware logic is required for the generation of control signals for this type of operation it adds silicon complexity to the system. Fig. 3.1 explains the Harvard and Von-Neumann architecture concept.

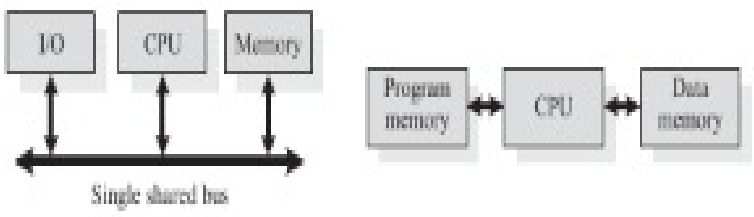


Fig. 3.1 Harvard vs Von-Neumann architecture

The following table highlights the differences between Harvard and Von-Neumann architecture

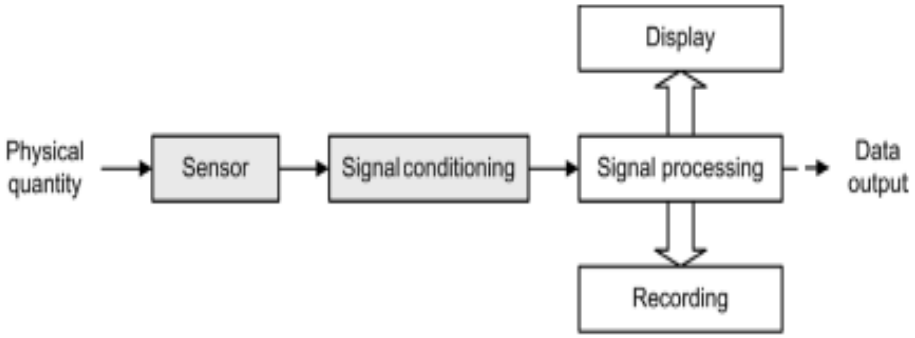
Table 3.3 Comparison of Harvard and Von Neumann Architecture

Harvard Architecture	Von-Neumann Architecture
Separate buses for instruction and data fetching	Single shared bus for instruction and data fetching
Easier to pipeline, so high performance can be achieved	Low performance compared to Harvard architecture
Comparatively high cost	Cheaper
No memory alignment problems	Allows self modifying codes
Since data memory and program memory are stored physically in different locations, no chances for accidental corruption of program memory	Since data memory and program memory are stored physically in the same chip, chances for accidental corruption of program memory

3.2 Sensors and Interfacing

3.2.1. INSTRUMENTATION AND CONTROL SYSTEMS

Fig. 3.2 shows the arrangement of an instrumentation system. The physical quantity to be measured (e.g. temperature) acts upon a sensor that produces an electrical output signal. This signal is an electrical analogue of the physical input but note that there may not be a linear relationship between the physical quantity and its electrical equivalent. Because of this and since the output produced by the sensor may be small or may suffer from the presence of noise (i.e. unwanted signals) further signal conditioning will be required before the signal will be at an acceptable level and in an acceptable form for signal processing, display and recording. Furthermore, because the signal processing may use digital rather than analogue signals an additional stage of analogue-to-analogue conversion may be required. Fig. 3.2 (b) shows the arrangement of a control system. This uses negative feedback in order to regulate and stabilize the output. It thus becomes possible to set the input or demand (i.e. what we desire the output to be) and leave the system to regulate itself by comparing it with a signal derived from the output (via a sensor and appropriate signal conditioning). A comparator is used to sense the difference in these two signals and where any discrepancy is detected the input to the power amplifier is adjusted accordingly. This signal is referred to as an error signal (it should be zero when the output exactly matches the demand). The input (demand) is often derived from a simple potentiometer connected across a stable d.c. voltage source while the controlled device can take many forms (e.g. a d.c. motor, linear actuator, heater, etc.).



(a) An irstrumentation system

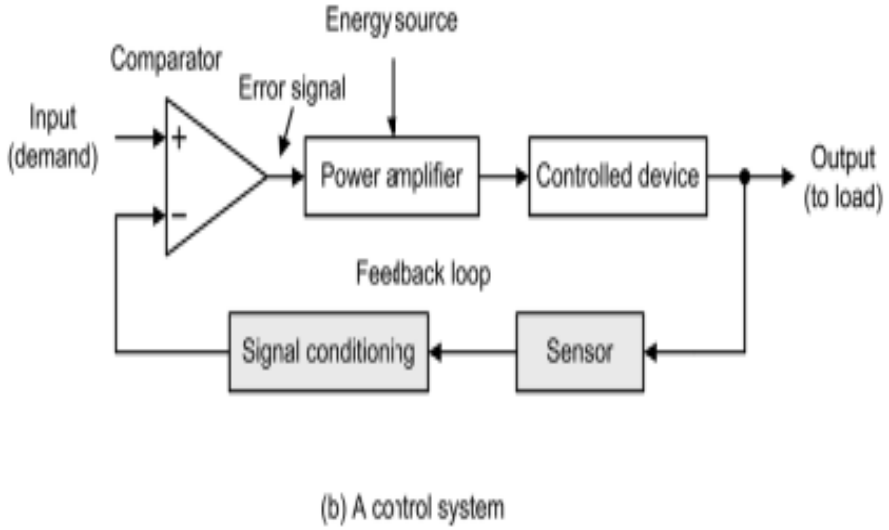


Fig. 3.2 Instrumentation and control systems

3.2.2. TRANSDUCERS

Transducers are devices that convert energy in the form of sound, light, heat, etc., into an equivalent electrical signal, or vice versa. Before we go further, let's consider a couple of examples that you will already be familiar with. A loudspeaker is a transducer that converts lowfrequency electric current into audible sounds. A microphone, on the other hand, is a transducer that performs the reverse function, i.e. that of converting sound pressure variations into voltage or current. Loudspeakers and microphones can thus be considered as complementary transducers.

Transducers may be used both as inputs to electronic circuits and outputs from them. From the two previous examples, it should be obvious that a loudspeaker is an output transducer designed for use in conjunction with an audio system. A microphone is an input transducer designed for use with a recording or sound reinforcing system. There are many different types of transducer and Tables 3.3 and 3.4 provide some examples of transducers that can be used to input and output three important physical quantities; sound, temperature and angular position.

Table 3.3 Some examples of input transducers

Physical quantity	Input transducer	Notes
Sound (pressure change)	Dynamic microphone (see Fig. 15.3)	Diaphragm attached to a coil is suspended in a magnetic field. Movement of the diaphragm causes current to be induced in the coil.
Temperature	Thermocouple (see Fig. 15.2)	Small e.m.f. generated at the junction between two dissimilar metals (e.g. copper and constantan). Requires reference junction and compensated cables for accurate measurement.
Angular position	Rotary potentiometer	Fine wire resistive element is wound around a circular former. Slider attached to the control shaft makes contact with the resistive element. A stable d.c. voltage source is connected across the ends of the potentiometer. Voltage appearing at the slider will then be proportional to angular position.

Table 3.4 Some examples of output transducers

Physical quantity	Output transducer	Notes
Sound (pressure change)	Loudspeaker (see Fig. 15.3)	Diaphragm attached to a coil is suspended in a magnetic field. Current in the coil causes movement of the diaphragm which alternately compresses and rarefies the air mass in front of it.
Temperature	Heating element (resistor)	Metallic conductor is wound onto a ceramic or mica former. Current flowing in the conductor produces heat.
Angular position	Rotary potentiometer	Multi-phase motor provides precise rotation in discrete steps of 15° (24 steps per revolution), 7.5° (48 steps per revolution) and 1.8° (200 steps per revolution).

Fig. 3.3 A selection of thermocouple probes

Fig. 3.4 A selection of audible transducers

3.2.2. SENSORS

A sensor is a special kind of transducer that is used to generate an input signal to a measurement, instrumentation or control system. The signal produced by a sensor is an electrical analogy of a physical quantity, such as distance, velocity, acceleration, temperature, pressure, light level, etc. The signals returned from a sensor, together with control inputs from the user or controller (as appropriate) will subsequently be used to determine the output from the system. The choice of sensor is governed by a number of factors including accuracy, resolution, cost and physical size. Sensors can be categorized as either active or passive. An active sensor generates a current or voltage output. A passive transducer requires a source of current or voltage and it modifies this in some way (e.g. by virtue of a change in the sensor's resistance). The result may still be a voltage or current but it is not generated by the sensor on its own. Sensors can also be classed as either digital or analogue. The output of a digital sensor can exist in only two discrete states, either 'on' or 'off', 'low' or 'high', 'logic 1' or 'logic 0', etc. The output of an analogue sensor can take any one of an infinite number of voltage or current levels. It is thus said to be continuously variable. Table 3.5 provides details of some common types of sensor.

Fig. 3.5 Various switch sensors

Fig. 3.6 Liquid level float switch

Fig. 3.7 Resistive linear position sensor

Fig. 3.8 Various optical and light sensors

Fig. 3.9 Various temperature and gas sensors

Fig. 3.10 Liquid flow sensor (digital output)

3.3 Actuators

3.3.1. ACTUATORS

Actuator is a form of transducer device (mechanical or electrical) which converts signals to corresponding physical action (motion). Actuator acts as an output device.

Looking back to the "Smart" running shoe example given at the end of Chapter 1, we can see that the actuator used for adjusting the position of the cushioning element is a micro stepper motor (Please refer back).

3.3.2. LED

Light Emitting Diode (LED) is an important output device for visual indication in any embedded system. LED can be used as an indicator for the status of various signals or

situations. Typical examples are indicating the presence of power conditions like 'Device ON', 'Battery low' or 'Charging of battery' for a battery operated handheld embedded devices.

Light Emitting Diode is a p-n junction diode (Refer Analog Electronics fundamentals to refresh your memory for p-n junction diode ©) and it contains an anode and a cathode. For proper functioning of the LED, the anode of it should be connected to +ve terminal of the supply voltage and cathode to the -ve terminal of supply voltage. The current flowing through the LED must be limited to a value below the maximum current that it can conduct. A resister is used in series between the power supply and the LED to limit the current through the LED. The ideal LED interfacing circuit is shown in Fig. 3.11.

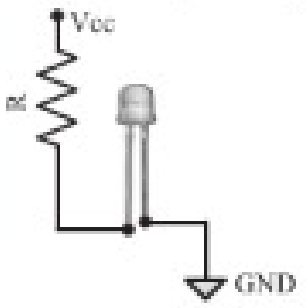


Fig. 3.11 LED interfacing

LEDs can be interfaced to the port pin of a processor/controller in two ways. In the first method, the anode is directly connected to the port pin and the port pin drives the LED. In this approach the port pin 'sources' current to the LED when the port pin is at logic High (Logic '1'). In the second method, the cathode of the LED is connected to the port pin of the processor/controller and the anode to the supply voltage through a current limiting resistor. The LED is turned on when the port pin is at logic Low (Logic '0'). Here the port pin 'sinks' current. If the LED is directly connected to the port pin, depending on the maximum current that a port pin can source, the brightness of LED may not be to the required level. In the second approach, the current is directly sourced by the power supply and the port pin acts as the sink for current. Here we will get the required brightness for the LED.

3.3.2. 7-SEGMENT LED DISPLAY

The 7-segment LED display is an output device for displaying alpha numeric characters. It contains 8 light-emitting diode (LED) segments arranged in a special form. Out of the 8 LED segments, 7 are used for displaying alpha numeric characters and 1 is used for representing

'decimal point' in decimal number display. Figure 3.12 explains the arrangement of LED segments in a 7-segment LED display.

The LED segments are named A to G and the decimal point LED segment is named as DR The LED segments A to G and DP should be lit accordingly to display numbers and characters. For example, for displaying the number 4, the segments F, G, B and C are lit, For displaying 3, the segments A, B, C, D, G and DP are lit. For displaying the character 'd', the segments B, C, D, E and G are lit. All these 8 LED segments need to be connected to one port of the processor/controller for displaying alpha numeric digits. The 7-segment LED displays are available in two different configurations, namely; Common Anode and Common Cathode. In the common anode configuration, the anodes of the 8 segments are connected commonly whereas in the common cathode configuration, the 8 LED segments share a common cathode line. Fig. 3.13 illustrates the Common Anode and Cathode configurations.

Based on the configuration of the 7-segment LED unit, the LED segment's anode or cathode is connected to the port of the processor/controller in the order 'A' segment to the least significant port pin and DP segment to the most significant port pin.

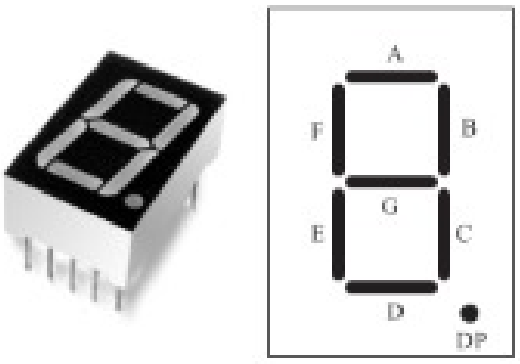


Fig. 3.12 7-segment LED display

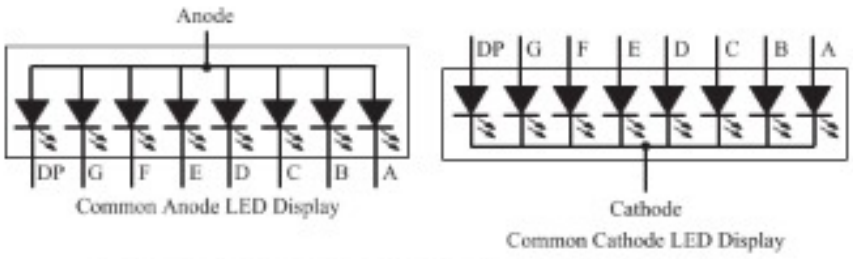


Fig. 3.13 Common Anode and Cathode configurations of a 7-segment LED display

The current flow through each of the LED segments should be limited to the maximum value supported by the LED display unit. The typical value for the current falls within the range of

20mA. The current through each segment can be limited by connecting a current limiting resistor to the anode or cathode of each segment. The value for the current limiting resistors can be calculated using the current value from the electrical parameter listing of the LED display.

For common cathode configurations, the anode of each LED segment is connected to the port pins of the port to which the display is interfaced. The anode of the common anode LED display is connected to the 5V supply voltage through a current limiting resistor and the cathode of each LED segment is connected to the respective port pin lines. For an LED segment to lit in the Common anode LED configuration, the port pin to which the cathode of the LED segment is connected should be set at logic 0.

7-segment LED display is a popular choice for low cost embedded applications like, Public telephone call monitoring devices, point of sale terminals, etc.

3.3.3. STEPPER MOTOR

A stepper motor is an electro-mechanical device which generates discrete displacement (motion) in response, to dc electrical signals. It differs from the normal dc motor in its operation. The dc motor produces continuous rotation on applying dc voltage whereas a stepper motor produces discrete rotation in response to the dc voltage applied to it. Stepper motors are widely used in industrial embedded applications, consumer electronic products and robotics control systems. The paper feed mechanism of a printer/fax makes use of stepper motors for its functioning.

Based on the coil winding arrangements, a two-phase stepper motor is classified into two. They are:

1. Unipolar
2. Bipolar

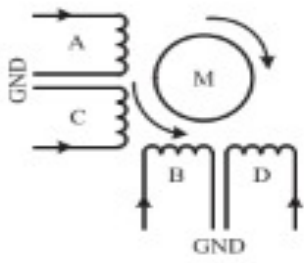


Fig. 3.14 2- phase unipolar stepper motor

1. Unipolar A unipolar stepper motor contains two windings per phase. The direction of rotation (clockwise or anticlockwise) of a stepper motor is controlled by changing the direction of current flow. Current in one direction flows through one coil and in the opposite direction flows through the other coil. It is easy to shift the direction of rotation by just switching the terminals to which the coils are connected. Fig. 3.14 illustrates the working of a two- phase unipolar stepper motor.

The coils are represented as A, B, C and D. Coils A and C carry current in opposite directions for phase 1 (only one of them will be carrying current at a time). Similarly, B and D carry current in opposite directions for phase 2 (only one of them will be carrying current at a time).

2. Bipolar A bipolar stepper motor contains single winding per phase. For reversing the motor rotation the current flow through the windings is reversed dynamically. It requires complex circuitry for current flow reversal. The stator winding details for a two phase unipolar stepper motor is shown in Fig. 3.15.

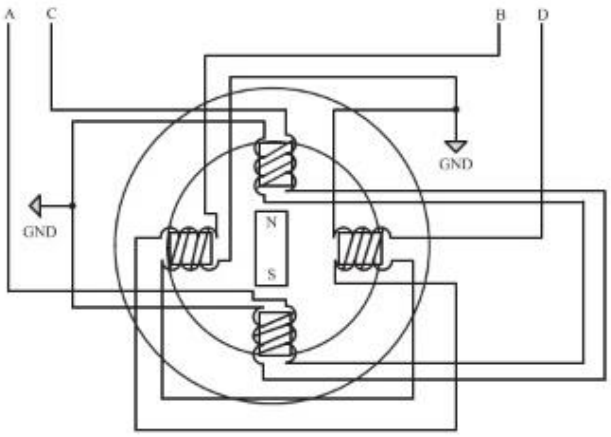


Fig. 3.15 Stator winding details for a 2-phase unipolar stepper motor

The stepping of stepper motor can be implemented in different ways by changing the sequence of activation of the stator windings. The different stepping modes supported by stepper motor are explained below.

Full Step: In the full step mode both the phases are energised simultaneously. The coils A, B, C and D are energised in the following order:

Step	Coil A	Coil B	Coil C	Coil D
1	H	H	L	L
2	L	H	H	L
3	L	L	H	H
4	H	L	L	H

It should be noted that out of the two windings, only one winding of a phase is energised at a time.

Wave Step: In the wave step mode only one phase is energised at a time and each coils of the phase is energised alternatively. The coils A, B, C and D are energised in the following order:

Step	Coil A	Coil B	Coil C	Coil D
1	H	L	L	L
2	L	H	L	L
3	L	L	H	L
4	L	L	L	H

Half Step: It uses the combination of wave and lull step. It has the highest torque and stability. The coil energising sequence for half step is given below.

Step	Coil A	Coil B	Coil C	Coil D
1	H	L	L	L
2	H	H	L	L
3	L	H	L	L
4	L	H	H	L
5	L	L	H	L
6	L	L	H	H
7	L	L	L	H
8	H	L	L	H

The rotation of the stepper motor can be reversed by reversing the order in which the coil is energised.

Two-phase unipolar stepper motors are the popular choice for embedded applications. The current requirement for stepper motor is little high and hence the port pins of a microcontroller/processor may' not be able to drive them directly. Also, the supply voltage required to operate stepper motor varies normally in the range 5V to 24 V. Depending on the current and voltage requirements, special driving circuits are required to interface the stepper motor with microcontroller/processors. Commercial off- the-shelf stepper motor driver ICs are

available in the market and they can be directly interfaced to the microcontroller port. ULN2803 is an octal peripheral driver array available from ON semiconductors and ST microelectronics for driving a 5V stepper motor. Simple driving circuit can also be built using transistors.

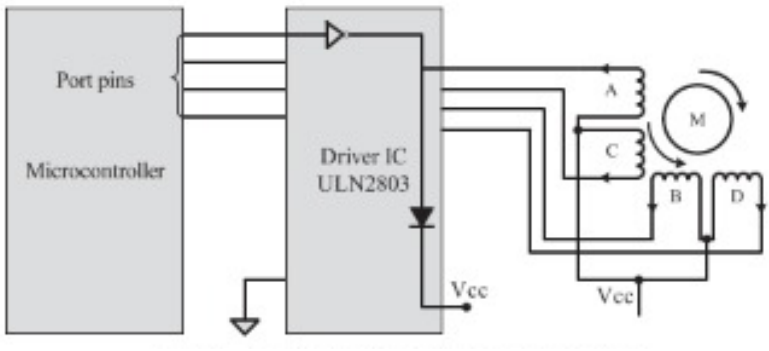


Fig. 3.16 Interfacing of a stepper motor through a driver circuit

The following circuit diagram (Fig. 3.16) illustrates the interfacing of a stepper motor through a driver circuit connected to the port pins of a microcontroller/processor.

3.3.4. RELAY

Relay is an electro-mechanical device. In embedded application, the 'Relay' unit acts as dynamic path selectors for signals and power. The 'Relay' unit contains a relay coil made up of insulated wire on a metal core and a metal armature with one or more contacts.

'Relay' works on electromagnetic principle. When a voltage is applied to the relay coil, current flows through the coil, which in turn generates a magnetic field. The magnetic field attracts the armature core and moves the contact point. The movement of the contact point changes the power/signal flow path. 'Relays' are available in different configurations. Fig.3.17 given below illustrates the widely used relay configurations for embedded applications.

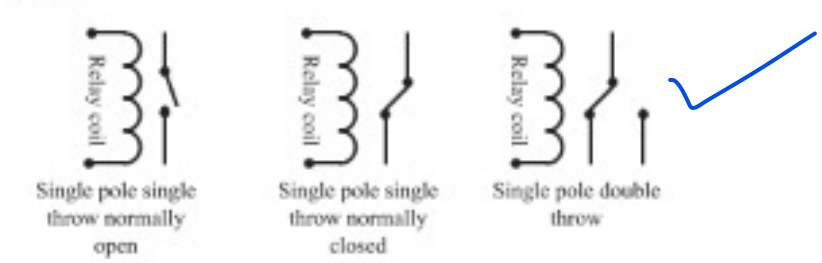


Fig.3.17 Relay configurations

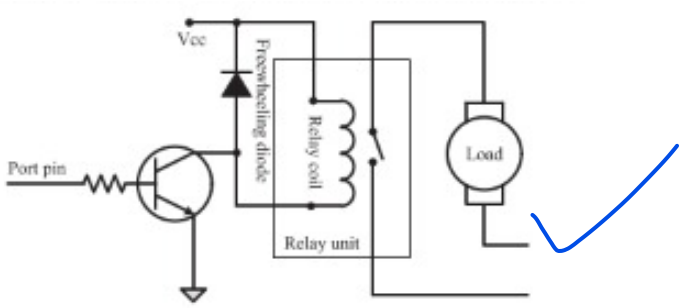


Fig. 3.18 Transistor based Relay driving circuit

The Single Pole Single Throw configuration has only one path for information flow. The path is either open or closed in normal condition. For normally Open Single Pole Single Throw relay, the circuit is normally open and it becomes closed when the relay is energised. For normally closed Single Pole Single Throw configuration, the circuit is normally closed and it becomes open when the relay is energised. For Single Pole Double Throw Relay, there are two paths for information flow and they are selected by energising or de-energising the relay.

The Relay is normally controlled using a relay driver circuit connected to the port pin of the processor/controller. A transistor is used for building the relay driver circuit. Fig. 3.18 illustrates the same.

A free-wheeling diode is used for free-wheeling the voltage produced in the opposite direction when the relay coil is de-energised. The freewheeling diode is essential for protecting the relay and the transistor.

Most of the industrial relays are bulky and requires high voltage to operate. Special relays called 'Reed' relays are available for embedded application requiring switching of low voltage DC signals.

3.3.5. PIEZO BUZZER

Piezo buzzer is a piezoelectric device for generating audio indications in embedded application. A piezoelectric buzzer contains a piezoelectric diaphragm which produces audible sound in response to the voltage applied to it. Piezoelectric buzzers are available in two types. 'Self- driving' and 'External driving'. The 'Self-driving' circuit contains all the necessary components to generate sound at a predefined tone. It will generate a tone on applying the voltage. External driving piezo buzzers supports the generation of different tones. The tone can be varied by applying a variable pulse train to the piezoelectric buzzer. A piezo buzzer can

be directly interfaced to the port pin of the processor/control. Depending on the driving current requirements, the piezo buzzer can also be interfaced using a transistor based driver circuit as in the case of a 'Relay'.

3.3.6. PUSH BUTTON SWITCH

It is an input device. Push button switch comes in two configurations, namely 'Push to Make' and 'Push to Break'. In the 'Push to Make' configuration, the switch is normally in the open state and it makes a circuit contact when it is pushed or pressed. In the 'Push to Break' con¬ figuration, the switch is normally in the closed state and it breaks the circuit contact when it is pushed or pressed. The push button stays in the 'closed' (For Push to Make type) or 'open' (For Push to Break type) state as long as it is kept in the pushed state and it breaks/makes the circuit connection when it is released. Push button is used for generating a momentary pulse. In embedded application push button is generally used as reset and start switch and pulse generator. The Push button is normally connected to the port pin of the host processor/controller. Depending on the way in which the push button interfaced to the controller, it can generate either a 'HIGH' pulse or a 'LOW' pulse. Fig. 3.19 illustrates how the push button can be used for generating 'LOW' and 'HIGH' pulses.

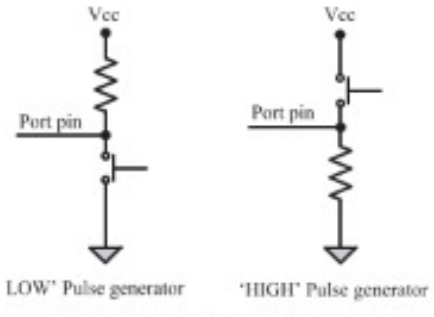


Fig. 3.19 Push button switch configuration

3.3.7. KEYBOARD

Keyboard is an input device for user interfacing. If the number of keys required is very limited, push button switches-can.be used and they can be directly interfaced to the port pins for reading. However, there may be situations demanding a large number of keys for user input (e.g. PDA device with alpha-numeric keypad for user data entry). In such situations it may not be possible to interface each keys to a port pin due to the limitation in the number of general

purpose port pins available for the processor/controller in use and moreover it is wastage of port pins. Matrix keyboard is an optimum solution for handling large key requirements. It greatly reduces the number of interface connections. For example, for interfacing 16 keys, in the direct interfacing technique 16 port pins are required, whereas in the matrix keyboard only 8 lines are required. The 16 keys are arranged in a 4 column x 4 Row matrix. Fig. 3.20 illustrates the connection of keys in a matrix keyboard.

In a matrix keyboard, the keys are arranged in matrix fashion (i.e. they are connected in a row and column style). For detecting a key press, the keyboard uses the scanning technique, where each row of the matrix is pulled low and the columns are read. After reading the status of each columns corresponding to a row, the row is pulled high and the next row is pulled low and the status of the columns are read. This process is repeated until the scanning for all rows are completed. When a row is pulled low and if a key connected to the row is pressed, reading the column to which the key is connected will give logic 0. Since keys are mechanical devices, there is a possibility for de-bounce issues, which may give multiple key press effect for a single key press. To prevent this, a proper key de-bouncing technique should be applied.

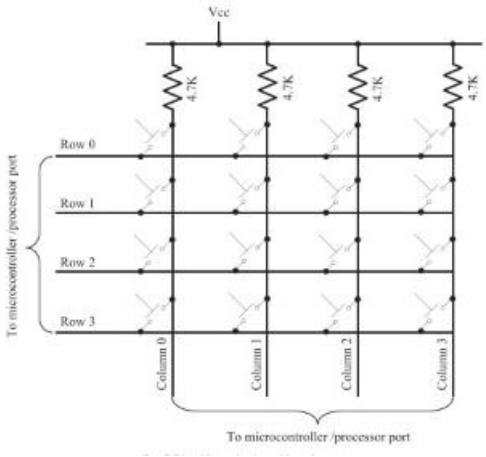


Fig. 3.20 Matrix keyboard Interfacing

Hardware key de-bouncer circuits and software key de-bounce techniques are the key debouncing techniques available. The software key de-bouncing technique doesn't require any additional hardware and is easy to implement. In the software de-bouncing technique, on detecting a key-press, the key is read again after a de-bounce delay. If the key press is a genuine one, the state of the key will remain as 'pressed' on the second read also. Pull-up resistors are connected to the column lines to limit the current that flows to the Row line on a key press.

3.4 Communication Interface

Communication interface is essential for communicating with various subsystems of the embedded system and with the external world. For an embedded product, the communication interface can be viewed in two different perspectives; namely; Device/board level communication interface (Onboard Communication Interface) and Product level communication, interface (External Communication Interface). Embedded product is a combination of different types of components (chips/devices) arranged on a printed circuit board (PCB). The communication channel which interconnects the various components within an embedded product is referred as device/board level communication interface (onboard communication interface). Serial interfaces like I2C, SPI, UART, 1-Wire, etc and parallel bus interface, are examples of 'Onboard Communication Interface'.

Some embedded systems are self-contained units and they don't require any interaction and data transfer with other sub-systems or external world. On the other hand, certain embedded systems may be a part of a large distributed system and they require interaction and data transfer between various devices and sub-modules. The 'Product level communication interface' (External Communication Interface.) is responsible for data transfer between the embedded system and other devices or modules. The external communication interface can be either a wired media or a wireless media and it can be a serial or a parallel interface. Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves (RF), GPRS, etc. are examples for wireless communication interface. RS-232C/RS-422/RS-485, USB, Ethernet IEEE 1394 port, Parallel port, CF-II interface, SDIO, PCMCIA, etc. are examples for wired interfaces. It is not mandatory that an embedded system should contain an external communication interface. Mobile communication equipment is an example for embedded system with external communication interface.

The following section gives you an overview of the various 'Onboard' mid 'External' communication interfaces for an embedded product. We will discuss about the various physical interface, firmware requirements and initialisation and communication sequence for these interfaces in a dedicated book titled 'Device Interfacing', which is planned under this series.

3.4.1. UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART)

Universal Asynchronous Receiver Transmitter (UART) based data transmission is an asynchronous form of serial data transmission. UART based serial data transmission doesn't require a clock signal to synchronise the transmitting end arid receiving end for transmission. Instead it relies upon the pre-defined agreement between the trans¬ mitting device and receiving device. The serial communication settings (Baudrate, number of bits per byte, parity, number of start bits and stop bit and flow control) for both transmitter and receiver should be set as identical. The start and stop of communication is indicated through inserting special bits in the data stream. While sending a byte of data, a start bit is added first and a stop bit is added at the end of the bit stream. The least significant bit of the data byte follows the 'start' bit.

The 'start' bit informs the receiver that a data byte is about to arrive. The receiver device starts polling its 'receive line' as per the baudrate settings. If the baudrate is A' bits per second, the time slot available for one bit is 1/x seconds. The receiver unit polls the receiver line at exactly half of the time slot available for the bit. If parity is enabled for communication, the UART of the transmitting device adds a parity bit (bit value is 1 for odd number of Is in the transmitted bit stream and 0 for even number of Is). The UART of the receiving device calculates the parity of the bits received and compares it with the received parity bit for error checking. The UART of the receiving device discards the 'Start', 'Stop' and 'Parity' bit from the received bit stream and converts the received serial bit data to a word (In the case of 8 bits/byte, the byte is formed with the received 8 bits with the first received bit as the LSB and last received data bit as MSB). For proper communication, the 'Transmit line' of the sending device should be connected to the 'Receive line' of the receiving device. Fig.3.21 illustrates the same.

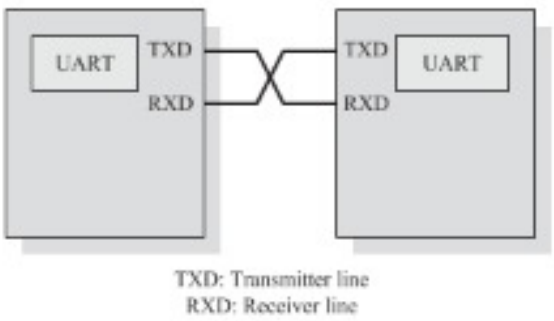


Fig.3.21 UART Interfacing

In addition to the serial data transmission function, UART provides hardware handshaking signal support for controlling the serial data flow. UART chips are available from different semiconductor manufacturers. National

Semiconductor's 8250 UART chip is considered as the standard setting UART. It was used in the original IBM PC.

Nowadays most of the microprocessors/controllers are available with integrated UART functionality and they provide built-in instruction support for serial data transmission and reception.

3.4.2. PARALLEL INTERFACE

The on-board parallel interface is normally used for communicating with peripheral devices which are memory mapped to the host of the system. The host processor/controller of the embedded system contains a parallel bus and the device which supports parallel bus can directly connect to this bus system. The communication through the parallel bus is controlled by the control signal interface between the device and the host. The 'Control Signals' for communication includes 'Read/ Write' signal and device select signal. The device normally contains a device select line and the device becomes active only when this line is asserted by the host processor. The direction of data transfer (Host to Device or Device to Host) can be controlled through the control signal lines for 'Read' and 'Write'. Only the host processor has control over the 'Read' and 'Write' control signals. The device is normally memory mapped to the host processor and a range of address is assigned to it. An address decoder circuit is used for generating the chip select signal for the device. When the address selected by the processor is within the range assigned for the device, the decoder circuit activates the chip select line and thereby the device becomes active. The processor then can read or write from or to the device by asserting the corresponding control line (RD\ and WR\ respectively). Strict timing characteristics are followed for parallel communication. As mentioned earlier, parallel communication is host processor initiated. If a device wants to initiate the communication, it can inform the same to the processor through interrupts. For this, the interrupt line of the device is connected to the interrupt line of the processor and the corresponding interrupt is enabled in the host processor. The width of the parallel interface is determined by the data bus width of the host processor. It can be 4bit, 8bit, 16bit, 32bit or 64bit etc. The bus width supported by

the device should be same as that of the host processor. The bus interface diagram shown in Fig. 3.22 illustrates the interfacing of devices through parallel interface.

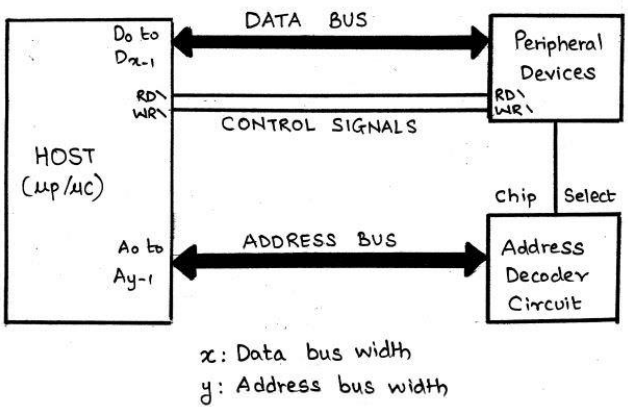


Fig. 3.22 Parallel Bus Interface

Parallel data communication offers the highest speed for the data transfer.

3.4.3. UNIVERSAL SERIAL BUS (USB)

Universal Serial Bus (USB) is a wired high speed serial bus for data communication. The first version of USB (USB 1.0) was released in 1995 and was created by the USB core group members consisting of Intel, Microsoft, IBM, Compaq, Digital and Northern Telecom. The USB communication system follows a star topology with a USB host at the centre and one or more USB peripheral devices/USB hosts connected to it. A USB host can support connections up to 127, including slave peripheral devices and other USB hosts. Fig.3.23 illustrates the star topology for USB device connection.

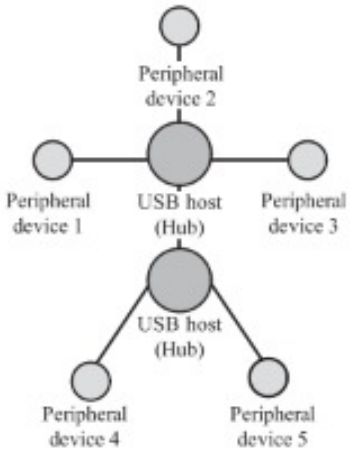


Fig.3.23 USB device connection topology

USB transmits data in packet format. Each data packet has a standard format. The USB communication is a host initiated one. The USB host contains a host controller which is responsible for controlling the data communication, including establishing connectivity with USB slave devices, packetizing and formatting the data. There are different standards for implementing the USB Host Control interface; namely Open Host Control Interface (OHCI) and Universal Host Control Interface (UHCI).

The physical connection between a USB peripheral de¬ vice and master device is established with a USB cable. The USB cable supports communication distance of up to 5 metres. The USB standard uses two different types of connector at the ends of the USB cable for connecting the USB peripheral device and host device. 'Type A' connector is used for upstream connection (connection with host) and Type B connector is used for downstream connection(connection with slave device). The USB connector present in desktop PCs or laptops are examples for 'Type A' USB connector. Both Type A and Type B connectors contain 4 pins for communication. The Pin details for the connectors are listed in the table given below.

Table 3.5 Pin details

Pin No	Pin Name	Description
1	V_{BUS}	Carries Power (5V)
2	D-	Differential data carrier line
3	D+	Differential data carrier line
4	GND	Ground signal line

USB uses differential signals for data transmission. It improves the noise immunity. USB interface has the ability to supply power to the connecting devices. Two connection lines (Ground and Power) of the USB interface are dedicated for carrying power. It can supply power up to 500 mA at 5 V. It is sufficient to operate low power devices. Mini and Micro USB connectors are available for small form factor devices like 7 portable media players.

Each USB device contains a Product ID (PID) and a Vendor ID (VID). The PID and VID are embedded into the USB chip by the USB device manufacturer. The VID for a device is supplied by the USB standards forum. PID and VID are essential for loading the drivers corresponding to a USB device for communication.

USB supports four different types of data transfers, namely; Control, Bulk, Isochronous and Interrupt. Control transfer is used by USB system software to query, configure and issue commands to the USB device. Bulk transfer is used for sending a block of data to a device. Bulk transfer supports error checking and correction. Transferring data to a printer is an example for bulk transfer. Isochronous data transfer is used for real-time data communication. In Isochronous transfer, data is transmitted as streams in real-time. Isochronous transfer doesn't support error checking and re-transmission of data in case of any transmission loss. All streaming devices like audio devices and medical equipment for data collection make use of the isochronous transfer. Interrupt transfer is used for transferring small amount of data. Interrupt transfer mechanism makes use of polling technique to see whether the USB device has any data to send. The frequency of polling is determined by the USB device and it varies from 1 to 255 milliseconds. Devices like Mouse and Keyboard, which transmits fewer amounts of data, uses Interrupt transfer.

USB 3.x is the latest version of the USB standard for peripheral connectivity. USB 3.x brings a number of improvements over USB 2.0 and adds new transfer mode called SuperSpeed (SS), capable of transferring data at speeds upto 4.8 Gbps, which is more than 10 times as fast as the 480 Mbps high speed of USB 2.0.

3.4.4. Wi-Fi

Wi-Fi or Wireless Fidelity is the popular wireless communication technique for net¬ worked communication of devices. Wi-Fi follows the IEEE 802.11 standard. Wi-Fi is intended for net¬ work communication and it supports Internet Protocol (IP) based communication. It is essential to have device identities in a multipoint communication to address specific devices for data communication. In an IP based communication each device is identified by an IP address, which is unique to each device on the network. Wi-Fi based communications require an intermediate agent called Wi-Fi router/Wireless Access point to manage the communications. The Wi-Fi router is responsible for restricting the access to a network, assigning IP address to devices on the network, routing data packets to the intended devices on the network. Wi-Fi enabled devices contain a wireless adaptor for transmitting and receiving data in the form of radio signals through an antenna. The hardware part of it is known as Wi-Fi Radio. Wi-Fi operates at 2.4GHz or 5GHz of radio spectrum and they co-exist with other ISM band devices like Bluetooth. Fig.3.24 illustrates the typical interfacing of devices in a Wi-Fi network.

For communicating with devices over a Wi-Fi network, the device when its Wi-Fi radio is turned ON, searches the available Wi-Fi network in its vicinity and lists out the Service Set Identifier (SSID) of the available networks. If the network is security enabled, a password may be required to connect to a particular SSID. Wi-Fi employs different security mechanisms like Wired Equivalency Privacy (WEP) Wireless Protected Access (WPA), etc. for securing the data communication. Wi-Fi supports data rates ranging from 1Mbps to 150Mbps (Growing towards higher rates as technology progresses) depending on the standards (802.11 a/b/g/n/ac) and access/modulation method. Depending On the type of antenna and usage location (indoor/outdoor), Wi-Fi offers a range of 100 to 1000 feet.

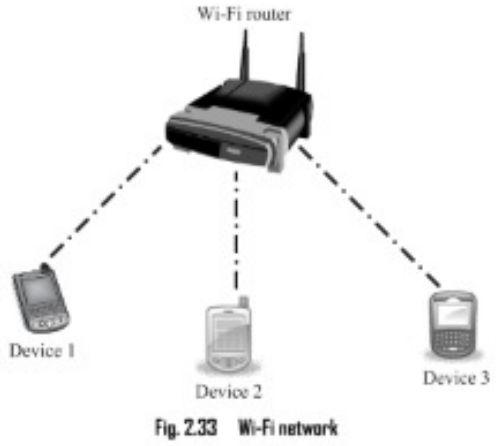


Fig.3.24 Wi-Fi network

3.4.4. GENERAL PACKET RADIO SERVICE (GPRS), 3G, 4G, LTE

General Packet Radio Service (GPRS), 3G, 4G, and LTE are communication technique for transferring data over a mobile communication network like GSM and CDMA. Data is sent as packets in GPRS communication. The transmitting device splits the data into several related packets. At the receiving end the data is re-constructed by combining the received data packets. GPRS supports a theoretical maximum transfer rate of 171.2kbps. In GPRS communication, the radio channel is concurrently shared between several users instead of dedicating a radio channel to a cell phone user. The GPRS communication divides the channel into 8 timeslots and transmits data over the available channel. GPRS supports Internet Protocol (IP), Point to Point Protocol (PPP) and X.25 protocols for communication. GPRS is mainly used by mobile enabled embedded devices for data communication. The device should support the necessary GPRS hardware like GPRS modem and GPRS radio. To accomplish GPRS based

communication, the carrier network also should have support for GPRS communication. GPRS is an old technology and it is being replaced by new generation data communication techniques like 3G (3rd Generation), High Speed Downlink Packet Access (HSDPA), 4G (4th Generation), LTE (Long Term Evolution) etc. which offers higher bandwidths for communication. 3G offers data rates ranging from 144 Kbps to 2Mbps or higher, whereas 4G gives a practical data throughput of 2 to 100+ Mbps depending on the network and underlying technology.