



Your spring board to future success

**AKSHAYA INSTITUTE OF TECHNOLOGY, TUMKUR**

Department of Electronics & Communication Engineering



Module 1 Notes for

**“EMBEDDED SYSTEM DESIGN”**

**[BEC601]**

**Prepared by:**

**SINDHU R**

**Assistant Professor**

**Department of ECE.**

**Akshaya Institute of Technology**

**Tumakuru**

# AKSHAYA INSTITUTE OF TECHNOLOGY

Lingapura, Obalapura Post, Koratagere Road, Tumakuru - 572106

## DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING



### VISION

To produce competent engineering professionals in the field of Electronics and Communication Engineering by imparting value based quality technical education to meet the societal needs and to develop socially responsible citizens.



### MISSION

**M1:** To provide strong fundamentals and technical skills in the field of Electronics and Communication Engineering through effective teaching learning process.

**M2:** Enhancing employability of the students by providing skills in the fields of VLSI, Embedded systems, Signal processing, etc., through Centre of Excellence.

**M3:** Encourage the students to participate in co-curricular and extra-curricular activities that creates a spirit of social responsibility and leadership qualities.



### Program Specific Outcomes (PSOs)

*After Successful Completion of Electronics and Communication Engineering Program Students will be able to*

1. Apply fundamental knowledge of core. Electronics and Communication Engineering in the analysis, design and development of Electronics Systems as well as to interpret and synthesize experimental data leading to valid conclusions.
2. Exhibit the skills gathered to analyze, design, develop software applications and hardware products in the field of embedded systems and allied areas.



### Program Educational Objectives (PEOs)

**PEO1:** Graduates exhibit their innovative ideas and management skills to meet the day to day technical challenges.

**PEO2:** Graduates utilize their knowledge and skills for the development of optimal solutions to the problems in the field of Electronics and Communication Engineering..

**PEO3:** Graduates exhibit good interpersonal skills, leadership qualities and adapt themselves for life-long Learning



Embedded System Design		Semester	06
Course Code	BEC601	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:2:0	SEE Marks	50
Total Hours of Pedagogy	40 hours Theory + 8-10 Lab slots	Total Marks	100
Credits	04	Exam Hours	
Examination nature (SEE)	Theory		
<p><b>Course objectives:</b></p> <ul style="list-style-type: none"> <li>● Identify various components, their purpose, and their application to the embedded system's applicability.</li> <li>● Program various embedded components using the embedded C program.</li> <li>● Understand the embedded system's real-time operating system and its application in IoT</li> <li>● Understand the fundamentals of ARM-based systems, including architecture and its units like registers, debug interface, stack, MPU, Interrupts etc</li> <li>● Use the various instructions to program the ARM controller.</li> </ul>			
<p><b>Teaching-Learning Process (General Instructions)</b></p> <p>These are sample Strategies; that teachers can use to accelerate the attainment of the various course outcomes.</p> <ol style="list-style-type: none"> <li>1. In addition to the traditional lecture method, different types of innovative teaching methods may be adopted so that the delivered lessons shall develop students' theoretical and applied Mathematical skills.</li> <li>2. Provide real-life examples.</li> <li>3. Support and guide the students for self-study.</li> <li>4. You will assign homework, grading assignments and quizzes, and documenting students' progress.</li> <li>5. Encourage the students to group learning to improve their creative and analytical skills.</li> <li>6. Show short related video lectures in the following ways: <ul style="list-style-type: none"> <li>● As an introduction to new topics (pre-lecture activity).</li> <li>● As a revision of topics (post-lecture activity).</li> <li>● As additional examples (post-lecture activity).</li> <li>● As an additional material of challenging topics (pre-and post-lecture activity).</li> <li>● As a model solution of some exercises (post-lecture activity).</li> </ul> </li> </ol>			
<b>MODULE-1</b>			
<p><b>Introduction to Embedded System:</b> What is an Embedded Systems? Embedded systems Vs General computing systems, History of Embedded Systems, Classification of Embedded systems, Major Application Areas of Embedded Systems. Purpose of Embedded Systems, The Typical Embedded System, Microprocessor Vs Microcontroller, Differences between RISC and CISC, Harvard V/s Von-Neumann Processor/Controller Architecture, Big-endian V/s Little-endian processors, Memory (ROM and RAM types), Sensors &amp; Actuators, The I/O Subsystem – I/O Devices, Light Emitting Diode (LED), 7-Segment LED Display, Optocoupler, Relay, Piezo buzzer, Push button switch, Communication Interfaces, On-board Communication Interface, External Communication Interface, Embedded Firmware, Other System Components</p> <p style="text-align: right;"><b>(Text 1: All the Topics from Ch-1 and Ch-2.)</b></p>			
<b>MODULE-2</b>			
<p><b>Embedded System Design Concepts:</b> Characteristics and Quality Attributes of Embedded Systems, Operational and non-operational quality attributes, Embedded Systems-Application and Domain specific, Hardware Software Co-Design and Program Modeling (excluding UML), Embedded firmware design and development (excluding C language).</p>			

**Text 1: Ch-3, Ch-4 (4.1, 4.2.1 and 4.2.2 only), Ch-7 (Sections 7.1, 7.2 only), Ch-9 (Sections 9.1, 9.2, 9.3.1, 9.3.2 only)**

**MODULE-3**

**RTOS and IDE for Embedded System Design:** Operating System basics, Types of operating systems, Task, process and threads (Only POSIX Threads with an example program), Thread preemption, Preemptive Task scheduling techniques, Task Communication, Task synchronization issues – Racing and Deadlock. How to choose an RTOS, Integration and testing of Embedded hardware and firmware, Embedded system Development Environment – Block diagram (excluding Keil).  
**(Text 1: Ch-10 (Sections 10.1, 10.2, 10.3, 10.5.2, 10.7, 10.8.1.1, 10.8.1.2 only), Ch-12, Ch-13 (a block diagram before 13.1, only).**

**MODULE-4**

**ARM Embedded Systems:**

Introduction, RISC design philosophy, ARM design philosophy, Embedded system hardware – AMBA bus protocol, ARM bus technology, Memory, Peripherals, Embedded system software – Initialization (BOOT) code, Operating System, Applications.

ARM Processor Fundamentals, ARM core dataflow model, registers, current program status register, Pipeline, Exceptions, Interrupts and Vector Table, Core extensions.

**Text 2: Chapter 1, 2**

**MODULE-5**

**Introduction to the ARM Instruction set:** Introduction, Data processing instructions, Load – Store instruction, Software interrupt instructions, Program status register instructions, Loading constants, ARMv5E extensions, Conditional Execution.

**Text 2: Chapter 3**

**PRACTICAL COMPONENT OF IPCC**

Conduct the following experiments on an ARM CORTEX M3 evaluation board to learn Assembly Language Program and using evaluation version of Embedded 'C' & Keil uVision-4 tool/compiler.

Sl.NO	Experiments
1	Write a program to find the sum of the first 10 integer numbers.
2	Write an Assembly Language Program (ALP) to i) Multiply two 16-bit numbers. ii) Add two 32-bit numbers.
3	Write a program to find the factorial of a number.
4	Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM.
5	Write a program to find the square of a number (1 to 10) using a look-up table.
6	Write a program to find the largest or smallest number in an array of 32 numbers.
7	Write a program to arrange a series of 32 bit numbers in ascending/descending order.
8	Write a program to count the number of ones and zeros in two consecutive memory locations.
9	Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.
10	Interface a DAC and generate Triangular and Square waveforms.
11	Display the Hex digits 0 to F on a 7-segment LED interface, with a suitable delay in between.
12	Interface a simple Switch and display its status through Relay, Buzzer and LED

**Course outcomes (Course Skill Set):**

At the end of the course, the student will be able to:

- Describe the architectural features and instructions of 32-bit microcontroller ARM Cortex M3.
- Apply the knowledge gained for Programming ARM Cortex M3 for different applications.
- Understand the basic hardware components and their selection method based on the characteristics and attributes of an embedded system.
- Understand the hardware software co-design and firmware design approaches.
- Explain the need of real time operating system for embedded system applications.

**Assessment Details (both CIE and SEE)**

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

**CIE for the theory component of the IPCC (maximum marks 50)**

- IPCC means practical portion integrated with the theory of the course.
- CIE marks for the theory component are **25 marks** and that for the practical component is **25 marks**.
- 25 marks for the theory component are split into **15 marks** for two Internal Assessment Tests (Two Tests, each of 15 Marks with 01-hour duration, are to be conducted) and **10 marks** for other assessment methods mentioned in 22OB4.2. The first test at the end of 40-50% coverage of the syllabus and the second test after covering 85-90% of the syllabus.
- Scaled-down marks of the sum of two tests and other assessment methods will be CIE marks for the theory component of IPCC (that is for **25 marks**).
- The student has to secure 40% of 25 marks to qualify in the CIE of the theory component of IPCC.

**CIE for the practical component of the IPCC**

- **15 marks** for the conduction of the experiment and preparation of laboratory record, and **10 marks** for the test to be conducted after the completion of all the laboratory sessions.
- On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to **15 marks**.
- The laboratory test (**duration 02/03 hours**) after completion of all the experiments shall be conducted for 50 marks and scaled down to **10 marks**.
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for **25 marks**.
- The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

**SEE for IPCC**

Theory SEE will be conducted by University as per the scheduled timetable, with common question papers for the course (**duration 03 hours**)

1. The question paper will have ten questions. Each question is set for 20 marks.
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics** under that module.
3. The students have to answer 5 full questions, selecting one full question from each module.

4. Marks scored by the student shall be proportionally scaled down to 50 Marks

**The theory portion of the IPCC shall be for both CIE and SEE, whereas the practical portion will have a CIE component only. Questions mentioned in the SEE paper may include questions from the practical component.**

**Suggested Learning Resources:**

**Text Books**

1. Shibu K V, "Introduction to Embedded Systems", Tata McGraw Hill Education
2. Andrew N Sloss, Dominic System and Chris Wright, "ARM System Developers Guide", Elsevier, Morgan Kaufman publisher, 1st Edition, 2008.

**Reference Book**

1. Raj Kamal, "Embedded Systems: Architecture and Programming", Tata McGraw Hill, 2008.

**Web links and Video Lectures (e-Resources):**

1. <https://archive.nptel.ac.in/courses/106/105/106105193/>
2. <https://developer.arm.com/documentation/dui0068/b/ARM-Instruction-Reference>
3. <https://www.udemy.com/course/introduction-to-arm-cortex-m3-and-m4-processors/>
4. [www.Nuvoton.com](http://www.Nuvoton.com)/websites on Advanced ARM Cortex Processors
5. <https://alison.com/tag/embedded-systems>

**Activity Based Learning (Suggested Activities in Class)/ Practical Based learning**

Programming Assignments / Mini Projects can be given to improve programming skills

## Module – 1

# Embedded System Components

### What is Embedded System?

An embedded system is a combination of 3 types of components: a. Hardware b. Software c. Mechanical Components and it is supposed to do one specific task only.

**Or**

An Electronic/Electro mechanical system which is designed to perform a specific function and is a combination of both hardware and firmware (Software)

E.g. Electronic Toys, Mobile Handsets, Washing Machines, Air Conditioners, Automotive Control Units, Set Top Box, DVD Player etc...

### Embedded System & General-Purpose Computing system

- The Embedded System and the General-purpose computing system are at two extremes.
- The embedded system is designed to perform a specific task whereas as per definition the general-purpose computer is meant for general use. It can be used for playing games, watching movies, creating software, work on documents or spreadsheets etc.
- Following are certain specific points that differentiates between embedded systems and general-purpose computers:

General Purpose Computing System	Embedded System
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
Contains a General Purpose Operating System (GPOS)	May or may not contain an operating system for functioning
Applications are alterable (programmable) by the user (It is possible for the end user to re-install the operating system, 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 (There may be exceptions for system supporting OS kernel image flashing through special hardware settings)
Performance is the key deciding factor in the selection of the system. Always, 'Faster is Better'	Application-specific requirements (like performance, power requirements, memory usage, etc.) are the key deciding factors
Less/not at all tailored towards reduced operating power requirements, options for different levels of power management	Highly tailored to take advantage of the power saving modes supported by the hardware and the operating system
Response requirements are not time-critical	For certain category of embedded systems like mission critical systems, the response time requirement is highly critical
Need not be deterministic in execution behaviour	Execution behaviour is deterministic for certain types of embedded systems like 'Hard Real Time' systems

## Classification of Embedded Systems

It is possible to have classifications for embedded systems, based on different criteria.

Some of the criteria used in the classification of embedded systems are:

- Complexity & performance requirements
- Based on generation
- Based on deterministic behavior
- Based on triggering

### Based on generation

First Generation:

- The early embedded systems built around 8bit microprocessors like 8085 and Z80 and 4bit microcontrollers.
- Examples: Digital telephone keypads.

Second Generation:

- Embedded Systems built around 16 bit microprocessors and 8 or 16bit microcontrollers, following the first generation embedded systems.
- Examples: SCADA systems

Third Generation:

- Embedded Systems built around high performance 16/32 bit Microprocessors/controllers.
- Application Specific Instruction set processors like Digital Signal Processors (DSPs), and Application Specific Integrated Circuits (ASICs).
- Examples: Robotics, Media, etc.

Fourth Generation:

- Embedded Systems built around System on Chips (SoCs), Re configurable processors and multicore processors.
- Highly complex & very powerful.
- Examples: Smart P

## **Based on Complexity & performance requirements**

Small-scale:

- Simple in application need
- Performance not time-critical.
- Built around low performance & low cost 8 or 16 bit  $\mu\text{p}/\mu\text{c}$ . Example: an electronic toy

Medium-scale

- Slightly complex in hardware & firmware requirement.
- Built around medium performance & low cost 16 or 32 bit  $\mu\text{p}/\mu\text{c}$ .
- Usually contain operating system.
- Examples: Industrial machines

Large-scale:

- Highly complex hardware & firmware.
- Built around 32 or 64 bit RISC  $\mu\text{p}/\mu\text{c}$  or PLDs or Multicore-Processors.
- Response is time-critic

## **Based on deterministic behaviour**

This classification is applicable for “Real Time” systems.

- The task execution behaviour for an embedded system may be deterministic or non- deterministic.
- Based on execution behaviour Real Time embedded systems are divided into Hard and Soft.

## **Based on triggering**

Embedded systems which are “Reactive” in nature can be classify based on triggering.

Reactive systems can be:

- Event triggered
- Time triggered

## **Application Area of Embedded System**

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 found their job for a livelihood.

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.
2. Household appliances: Washing machine, Refrigerator.
3. Automotive industry: Anti-lock breaking system(ABS), engine control.
4. Home automation & security systems: Air conditioners, sprinklers, fire alarms.
5. Telecom: Cellular phones, telephone switches.
6. Computer peripherals: Printers, scanners.
7. Computer networking systems: Network routers and switches.
8. Healthcare: EEG, ECG machines.
9. Banking & Retail: Automatic teller machines, point of sales.
10. Card Readers: Barcode, smart card readers.

## **Purpose of Embedded System**

Each Embedded Systems is designed to serve the purpose of any one or a combination of the following tasks.

1. Data Collection/Storage/Representation
2. Data Communication
3. Data (Signal) Processing
4. Monitoring
5. Control
6. Application Specific User Interface

## Data Collection/Storage/Representation

- Embedded system designed for the purpose of data collection performs acquisition of data from the external world.
- Data collection is usually done for storage, analysis, manipulation and transmission.
- Data can be analog or digital.
- Embedded systems with analog data capturing techniques collect data directly in the form of analog signal whereas embedded systems with digital data collection mechanism converts the analog signal to the digital signal using analog to digital converters.
- If the data is digital it can be directly captured by digital embedded system.
- A digital camera is a typical example of an embedded System with data collection/storage/representation of data.
- Images are captured and the captured image may be stored within the memory of the camera. The captured image can also be presented to the user through a graphic LCD unit.



**Digital Camera for Image capturing/storage/display**  
Photo Courtesy of Casio -Model EXILIM ex-Z850  
([www.casio.com](http://www.casio.com))

## Data communication

- Embedded data communication systems are deployed in applications from complex satellite communication to simple home networking systems.
- The transmission of data is achieved either by a wire-line medium or by a wire-less medium. Data can either be transmitted by analog means or by digital means.
- Wireless modules-Bluetooth, Wi-Fi.
- Wire-line modules-USB, TCP/IP.
- Network hubs, routers, switches are examples of dedicated data transmission embedded systems.

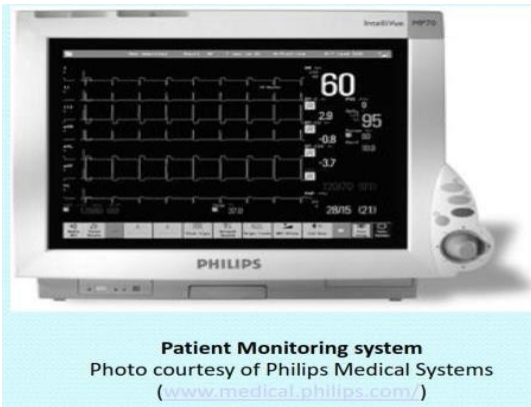
## Data signal processing

- Embedded systems with signal processing functionalities are employed in applications demanding signal processing like speech coding, audio video codec, transmission applications etc.
- A digital hearing aid is a typical example of an embedded system employing data processing. Digital hearing aid improves the hearing capacity of hearing impaired person.



## Monitoring

- All embedded products coming under the medical domain are with monitoring functions. Electro cardiogram machine is intended to do the monitoring of the heartbeat of a patient but it cannot impose control over the heartbeat.
- Other examples with monitoring function are digital CRO, digital multi-meters, and logic analysers.



## Control

- Embedded systems with control functionalities are used for imposing control over some variables according to the changes in input variables.
- A system with control functionality contains both sensors and actuators
- Sensors are connected to the input port for capturing the changes in environmental variable and the actuators connected to the output port are controlled according to the changes in the input variable.
- Air conditioner system used to control the room temperature to a specified limit is a typical example for control purpose.



## Application specific user interface

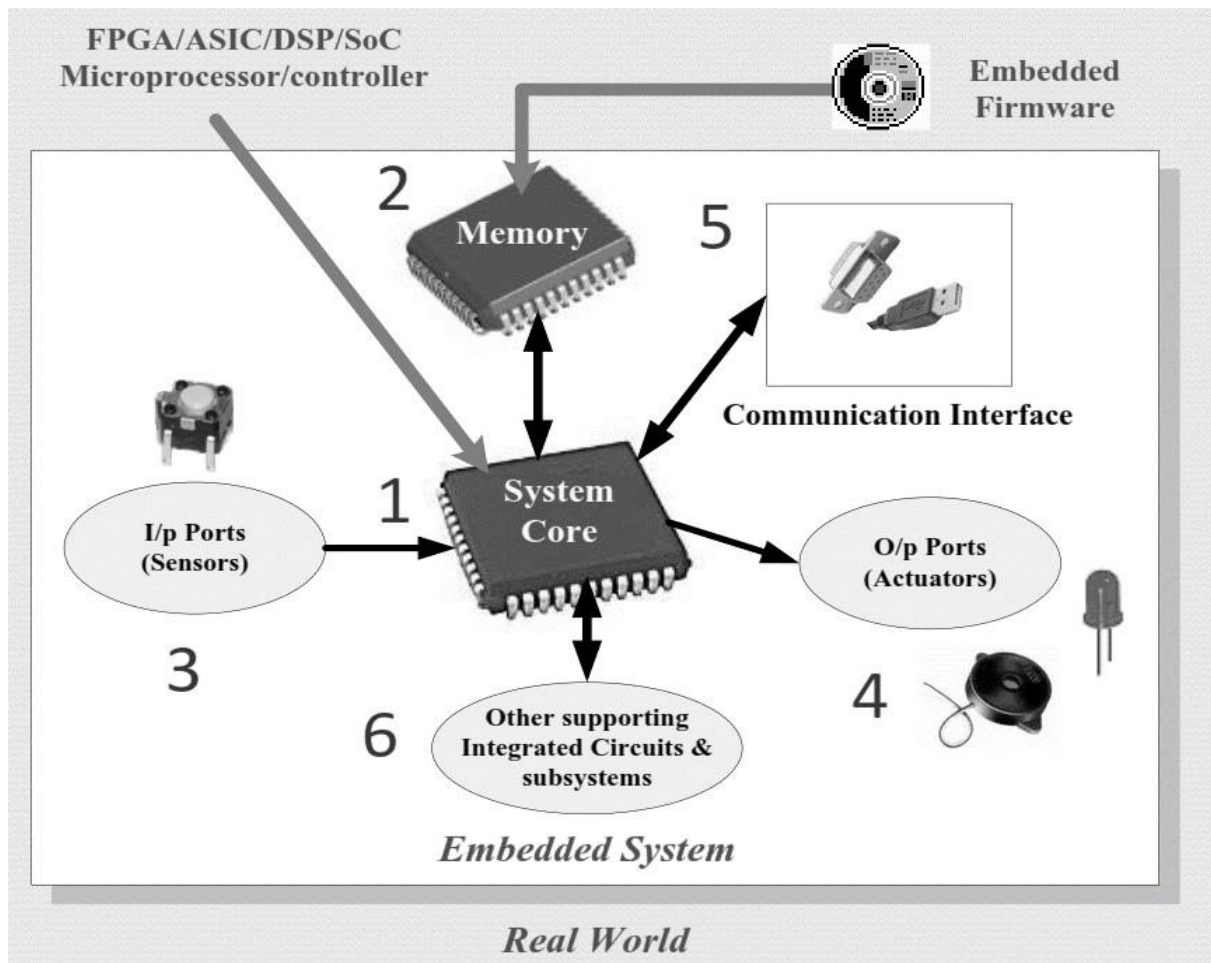
- Embedded systems which are designed for a specific application
- Buttons, switches, keypad, lights, bells, display units etc. are application specific user interfaces.
- Mobile phone, Control units in industry applications are example of application specific user interface.
- In mobile phone the user interface is provided through the keypad, system speaker, vibration alert etc.



## **‘Smart’ running shoes from Adidas – The Innovative bonding of Life Style with Embedded Technology**

- Shoe developed by Adidas, which constantly adapts its shock-absorbing characteristics to customize its value to the individual runner, depending on running style, pace, body weight, and running surface
- It contains sensors, actuators and a microprocessor unit which runs the algorithm for adapting the shock-absorbing characteristics of the shoe
- A ‘Hall effect sensor’ placed at the top of the “cushioning element” senses the compression and passes it to the Microprocessor
- A micro motor actuator controls the cushioning as per the commands from the MPU, based on the compression sensed by the ‘Hall effect sensor’





Embedded systems are basically designed to regulate a physical variable (such as Microwave Oven) or to manipulate the state of some devices by sending some signals to the actuators or devices connected to the output port system (such as temperature in Air Conditioner), in response to the input signal provided by the end users or sensors which are connected to the input ports. Hence the embedded systems can be viewed as a reactive system.

The control is achieved by processing the information coming from the sensors and user interfaces and controlling some actuators that regulate the physical variable. Keyboards, push button, switches, etc. are Examples of common user interface input devices and LEDs, LCDs, Piezoelectric buzzers, etc examples for common user interface output devices for a typical

embedded system. The requirement of type of user interface changes from application to application based on domain.

Some embedded systems do not require any manual intervention for their operation. They automatically sense the input parameters from real world through sensors which are connected at input port. The sensor information is passed to the processor after signal conditioning and digitization. The core of the system performs some predefined operations on input data with the help of embedded firmware in the system and sends some actuating signals to the actuator connect connected to the output port of the system.

The memory of the system is responsible for holding the code (control algorithm and other important configuration details). There are two types of memories are used in any embedded system. Fixed memory (ROM) is used for storing code or program. The user cannot change the firmware in this type of memory. The most common types of memories used in embedded systems for control algorithm storage are OTP, PROM, UVEPROM, EEPROM and FLASH. An embedded system without code (i.e. the control algorithm) implemented memory has all the peripherals but is not capable of making decisions depending on the situational as well as real world changes.

Memory for implementing the code may be present on the processor or may be implemented as a separate chip interfacing the processor. In a controller based embedded system, the controller may contain internal memory for storing code such controllers are called Micro-controllers with on- chip ROM, eg. Atmel AT89C51.

## **Core of Embedded Systems**

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. Programmable Logic Devices (PLDs)
3. Application Specific Integrated Circuits (ASICs)
4. Commercial off the shelf Components (COTS)

## **General Purpose and Domain Specific Processors**

- Almost 80% of the embedded systems are processor/ controller based.
- The processor may be microprocessor or a microcontroller or digital signal processor, depending on the domain and application.

### **Microprocessor:**

- A silicon chip representing a Central Processing Unit (CPU), which is capable of performing arithmetic as well as logical operations according to a pre-defined set of Instructions, which is specific to the manufacturer.
- In general the CPU contains the Arithmetic and Logic Unit (ALU), Control Unit and Working registers Microprocessor is a dependent unit and it requires the combination of other hardware like Memory, Timer Unit, and Interrupt Controller etc for proper functioning.
- Intel claims the credit for developing the first Microprocessor unit Intel 4004, a 4 bit processor which was released in Nov 1971
- Developers of microprocessors.
  - Intel – Intel 4004 – November 1971(4-bit)
  - Intel – Intel 4040.
  - Intel – Intel 8008 – April 1972.
  - Intel – Intel 8080 – April 1974(8-bit).
  - Motorola – Motorola 6800.
  - Intel – Intel 8085 – 1976.
  - Zilog - Z80 – July 1976

### **General Purpose Processor (GPP) Vs Application Specific Instruction Set Processor (ASIP)**

- General Purpose Processor or GPP is a processor designed for general computational tasks
- GPPs are produced in large volumes and targeting the general market. Due to the high volume production, the per unit cost for a chip is low compared to ASIC or other other specific Ics

- A typical general purpose processor contains an Arithmetic and Logic Unit (ALU) and Control Unit (CU)
- Application Specific Instruction Set processors (ASIPs) are processors with architecture and instruction set optimized to specific domain/application requirements like Network processing, Automotive, Telecom, media applications, digital signal processing, control applications etc.
- ASIPs fill the architectural spectrum between General Purpose Processors and Application Specific Integrated Circuits (ASICs).
- The need for an ASIP arises when the traditional general purpose processor are unable to meet the increasing application needs.
- Some Microcontrollers (like Automotive AVR, USB AVR from Atmel), System on Chips, Digital Signal Processors etc are examples of Application Specific Instruction Set Processors (ASIPs)
- ASIPs incorporate a processor and on-chip peripherals, demanded by the application requirement, program and data memory

### **Microcontroller:**

- A highly integrated silicon chip containing a CPU, scratch pad RAM, Special and General- purpose Register Arrays, On Chip ROM/FLASH memory for program storage, Timer and Interrupt control units and dedicated I/O ports
- Microcontrollers can be considered as a super set of Microprocessors
- Microcontroller can be general purpose (like Intel 8051, designed for generic applications and domains) or application specific (Like Automotive AVR from Atmel Corporation. Designed specifically for automotive applications)
- Since a microcontroller contains all the necessary functional blocks for independent working, they found greater place in the embedded domain in place of microprocessors
- Microcontrollers are cheap, cost effective and are readily available in the market
- Texas Instruments TMS 1000 is considered as the world's first microcontroller

## Microprocessor Vs Microcontroller

Microprocessor	Microcontroller
A silicon chip representing a Central Processing Unit (CPU), which is capable of performing arithmetic as well as logical operations according to a pre-defined set of Instructions	A microcontroller is a highly integrated chip that contains a CPU, scratch pad RAM, Special and General purpose Register Arrays, On Chip ROM/FLASH memory for program storage, Timer and Interrupt control units and dedicated I/O ports
It is a dependent unit. It requires the combination of other chips like Timers, Program and data memory chips, Interrupt controllers etc for functioning	It is a self contained unit and it doesn't require external Interrupt Controller, Timer, UART etc for its functioning
Most of the time general purpose in design and operation	Mostly application oriented or domain specific
Doesn't contain a built in I/O port. The I/O Port functionality needs to be implemented with the help of external Programmable Peripheral Interface Chips like 8255	Most of the processors contain multiple built-in I/O ports which can be operated as a single 8 or 16 or 32 bit Port or as individual port pins
Targeted for high end market where performance is important	Targeted for embedded market where performance is not so critical (At present this demarcation is invalid)
Limited power saving options compared to microcontrollers	Includes lot of power saving features

## Digital Signal Processors

- DSP are powerful special purpose 8/16/32 bit microprocessor designed to meet the computational demands and power constraints of today's embedded audio, video and communication applications. DSP are 2 to 3 times faster than general purpose microprocessors in signal processing applications.
- This is because of the architectural difference between DSP and general purpose microprocessors.
- DSPs implement algorithms in hardware which speeds up the execution whereas general purpose processor implement the algorithm in software and the speed of execution depends primarily on the clock for the processors.
- DSP includes following key units:
  - i. Program memory: It is a memory for storing the program required by DSP to process the data.
  - ii. Data memory: It is a working memory for storing temporary variables and data/signal to be processed.
  - iii. Computational engine: It performs the signal processing in accordance with the stored program memory computational engine incorporated many specialized arithmetic units and each of them operates simultaneously to increase the execution speed. It also includes multiple hardware

shifting operands and saves execution time. iv. I/O unit: It acts as an interface between the outside world and DSP. It is responsible for capturing signals to be processed and delivering the processed signals.

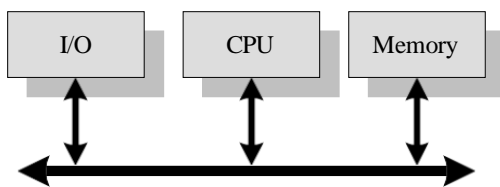
- Examples: Audio video signal processing, telecommunication and multimedia applications. SOP(Sum of Products) calculation, convolution, FFT(Fast Fourier Transform), DFT(Discrete Fourier Transform), etc are some of the operation performed by DSP.

### RISC Vs CISC Processors/Controllers

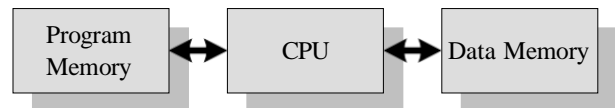
CISC	RISC
Greater no. of Instructions	Lesser no. of instructions
Generally no instruction pipelining feature	Instruction Pipelining and increased execution speed
Non Orthogonal Instruction Set (All instructions are not allowed to operate on any register and use any addressing mode. It is instruction specific)	Orthogonal Instruction Set (Allows each instruction to operate on any register and use any addressing mode)
Operations are performed on registers or memory depending on the instruction	Operations are performed on registers only, the only memory operations are load and store
Limited no. of general purpose registers	Large number of registers are available
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	Programmer needs to write more code to execute a task since the instructions are simpler ones
Variable length Instructions	Single, Fixed length Instructions
More silicon usage since more additional decoder logic is required to implement the complex instruction decoding.	Less Silicon usage and pin count
Can be Harvard or Von-Neumann Architecture	With Harvard Architecture

## Harvard V/s Von-Neumann Processor/Controller Architecture

- 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
- 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 on 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”)



Single shared Bus



Harvard Architecture
Separate buses for Instruction and Data fetching
Easier to Pipeline, so high performance can be achieved
Comparatively high cost
No memory alignment problems
Since data memory and program memory are stored physically in different locations, no chances for accidental corruption of program memory

Von-Neumann Architecture
Single shared bus for Instruction and Data fetching
Low performance Compared to Harvard Architecture
Cheaper
Allows self modifying codes
Since data memory and program memory are stored physically in same chip, chances for accidental corruption of program memory

## Big-endian V/s Little-endian processors:

- Endianness specifies the order in which the data is stored in the memory by processor operations in a multi byte system (Processors whose word size is greater than one byte). Suppose the word length is two byte then data can be stored in memory in two different ways
  - Higher order of data byte at the higher memory and lower order of data byte at location just below the higher memory
  - Lower order of data byte at the higher memory and higher order of data byte at location just below the higher memory
- *Little-endian* means the lower-order byte of the data is stored in memory at the lowest address, and the higher-order byte at the highest address. (The little end comes first)
- *Big-endian* means the higher-order byte of the data is stored in memory at the lowest address, and the lower-order byte at the highest address. (The big end comes first.)

### Little-endian Operation

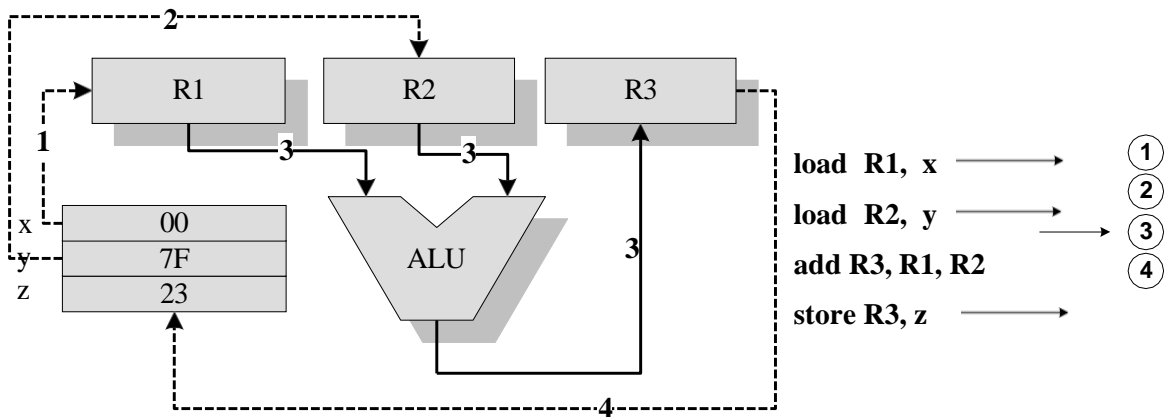
Base Address + 0	Byte 0	Byte 0	0x20000 (Base Address)
Base Address + 1	Byte 1	Byte 1	0x20001 (Base Address + 1)
Base Address + 2	Byte 2	Byte 2	0x20002 (Base Address + 2)
Base Address + 3	Byte 3	Byte 3	0x20003 (Base Address + 3)

### Big-endian Operation

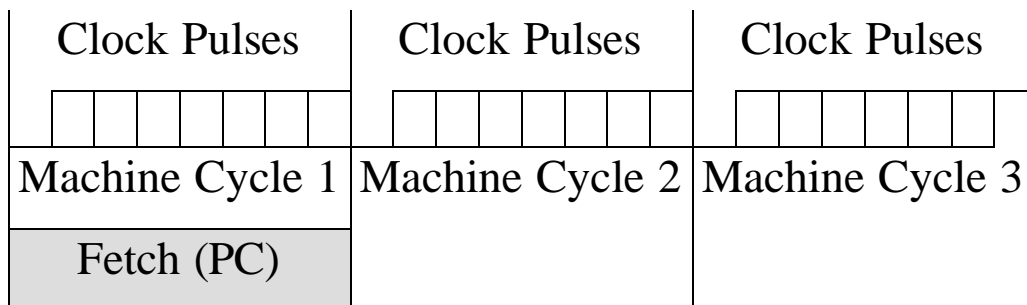
Base Address + 0	Byte 3	Byte 3	0x20000 (Base Address)
Base Address + 1	Byte 2	Byte 2	0x20001 (Base Address + 1)
Base Address + 2	Byte 1	Byte 1	0x20002 (Base Address + 2)
Base Address + 3	Byte 0	Byte 0	0x20003 (Base Address + 3)

### Load Store Operation & Instruction Pipelining:

- The RISC processor instruction set is orthogonal and it operates on registers. The memory access related operations are performed by the special instructions *load* and *store*. If the operand is specified as memory location, the content of it is loaded to a register using the *load* instruction.
- The instruction *store* stores data from a specified register to a specified memory location



- The conventional instruction execution by the processor follows the Fetch-Decode-Execute sequence
- During the decode operation the memory address bus is available and it is possible to effectively utilize it for an instruction fetch, the processing speed can be increased
- In its simplest form instruction pipelining refers to the overlapped execution of instructions



Execute (PC - 1)	Fetch (PC+1)	
	Execute (PC)	Fetch (PC+2)
PC : Program Counter		Execute (PC+1)

The Single stage pipelining concept

### Application Specific Integrated Circuit (ASIC):

- A microchip designed to perform a specific or unique application. It is used as replacement to conventional general purpose logic chips.
- ASIC integrates several functions into a single chip and thereby reduces the system development cost
- Most of the ASICs are proprietary products. As a single chip, ASIC consumes very small area in the total system and thereby helps in the design of smaller systems with high capabilities/functionalities
- ASICs can be pre-fabricated for a special application or it can be custom fabricated by using the components from a re-usable „*building block*“ library of components for a particular customer application
- Fabrication of ASICs requires a non-refundable initial investment (Non Recurring Engineering (NRE) charges) for the process technology and configuration expenses
- If the Non-Recurring Engineering Charges (NRE) is born by a third party and the Application Specific Integrated Circuit (ASIC) is made openly available in the market, the ASIC is referred as Application Specific Standard Product (ASSP)
- The ASSP is marketed to multiple customers just as a general-purpose product , but to a smaller number of customers since it is for a specific application.

## Programmable Logic Devices (PLDs)

- Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform.
- Logic devices can be classified into two broad categories - Fixed and Programmable. The circuits in a fixed logic device are permanent, they perform one function or set of functions
  - once manufactured, they cannot be changed
- Programmable logic devices (PLDs) offer customers a wide range of logic capacity, features, speed, and voltage characteristics - and these devices can be re-configured to perform any number of functions at any time
- Designers can use inexpensive software tools to quickly develop, simulate, and test their logic designs in PLD based design. The design can be quickly programmed into a device, and immediately tested in a live circuit
- PLDs are based on re-writable memory technology and the device is reprogrammed to change the design
- Two major types of programmable logic devices

Field Programmable Gate Arrays (FPGAs) and  
Complex Programmable Logic Devices (CPLDs)

FPGAs:

- FPGAs offer the highest amount of logic density, the most features, and the highest performance.
- These advanced FPGA devices also offer features such as built-in hardwired processors (such as the IBM Power PC), substantial amounts of memory, clock management systems, and support for many of the latest, very fast device-to-device signaling technologies
- FPGAs are used in a wide variety of applications ranging from data processing and storage, to instrumentation, telecommunications, and digital signal processing

CPLDs:

- CPLDs, by contrast, offer much smaller amounts of logic - up to about 10,000 gates
- CPLDs offer very predictable timing characteristics and are therefore ideal for critical control applications
- CPLDs such as the Xilinx CoolRunner series also require extremely low amounts of power and are very inexpensive, making them ideal for cost-sensitive, battery-operated, portable applications such as mobile phones and digital handheld assistants

### **Commercial off the Shelf Component (COTS)**

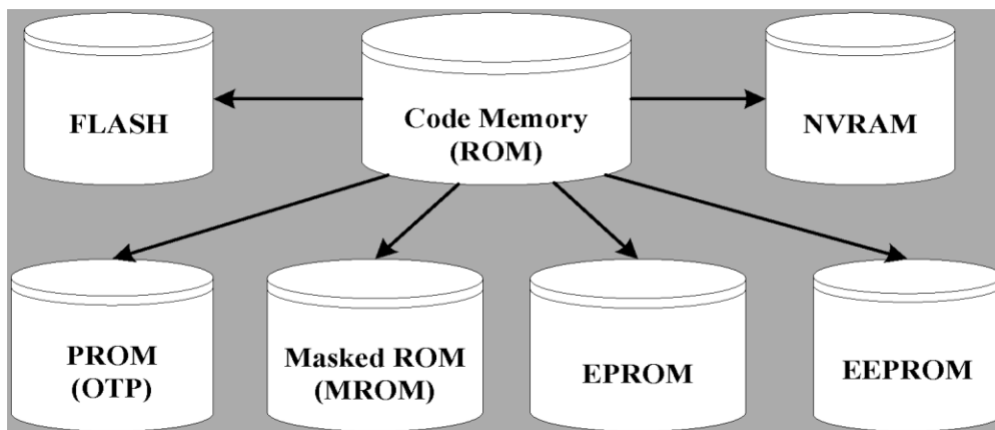
- A Commercial off-the-shelf (COTS) product is one which is used '*as-is*'.
- COTS products are designed in such a way to provide easy integration and interoperability with existing system components
- Typical examples for the COTS hardware unit are Remote Controlled Toy Car control unit including the RF Circuitry part, High performance, high frequency microwave electronics (2 to 200 GHz), High bandwidth analog-to-digital converters, Devices and components for operation at very high temperatures, Electro-optic IR imaging arrays, UV/IR Detectors etc.
- A COTS component in turn contains a GPP / ASIP / ASIC / ASSP / PLD
- **Advantages:** Readily available in the market, cheap and cuts down development time

### **Memory**

- Memory is an important part of an embedded system. The memory used in embedded system can be either
  1. Program Storage Memory (ROM)
  2. Data memory (RAM)
- Certain Embedded processors/controllers contain built in program memory and data memory and this memory is known as on-chip memory

## Program Storage Memory

- Stores the program instructions (CODE)
- Retains its contents even after the power to it is turned off. It is generally known as Nonvolatile storage memory
- Depending on the fabrication, erasing and programming techniques they are classified into



### Masked ROM (MROM)

- One-time programmable memory. Uses hardwired technology for storing data. The device is factory programmed by masking and metallization process according to the data provided by the end user
- The primary advantage of MROM is low cost for high volume production. They are the least expensive type of solid-state memory
- Different mechanisms are used for the masking process of the ROM, like
  - Creation of an enhancement or depletion mode transistor through channel implant
  - By creating the memory cell either using a standard transistor or a high threshold transistor. In the high threshold mode, the supply voltage required to turn ON the transistor is above the normal ROM IC operating voltage. This ensures that the transistor is always off and the memory cell stores always logic 0.
- The limitation with MROM based firmware storage is the inability to modify the device firmware against firmware upgrades.

## **PROM / OTP**

- Unlike MROM it is not pre-programmed by the manufacturer
- PROM/OTP has *nichrome* or *polysilicon* wires arranged in a matrix, these wires can be functionally viewed as fuses
- It is programmed by a PROM programmer which selectively burns the fuses according to the bit pattern to be stored
- Fuses which are not blown/burned represents a logic “1” where as fuses which are blown/burned represents a logic “0”. The default state is logic “1”
- OTP is widely used for commercial production of embedded systems whose prototyped versions are proven and the code is finalized
- It is a low-cost solution for commercial production.
- OTPs cannot be reprogrammed

## **EPROM**

- Erasable Programmable Read Only (EPROM) memory gives the flexibility to re-program the same chip
- EPROM stores the bit information by charging the floating gate of an FET
- Bit information is stored by using an EPROM Programmer, which applies high voltage to charge the floating gate
- EPROM contains a quartz crystal window for erasing the stored information. If the window is exposed to Ultra violet rays for a fixed duration, the entire memory will be erased
- Even though the EPROM chip is flexible in terms of re-programmability, it needs to be taken out of the circuit board and needs to be put in a UV eraser device for 20 to 30 minutes

## **EEPROM**

- Erasable Programmable Read Only (EPROM) memory gives the flexibility to re-program the same chip using electrical signals
- The information contained in the EEPROM memory can be altered by using electrical signals at the register/Byte level
- They can be erased and reprogrammed within the circuit
- These chips include a chip erase mode and in this mode they can be erased in a

few milliseconds

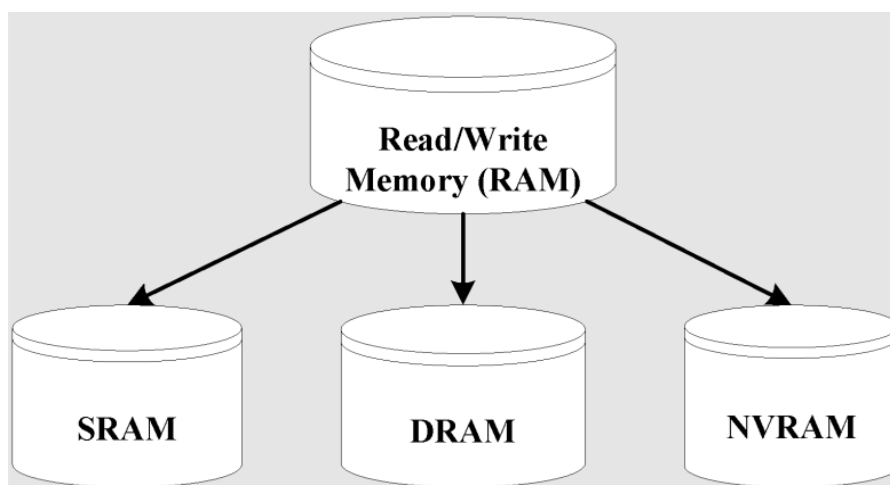
- It provides greater flexibility for system design
- The only limitation is their capacity is limited when compared with the standard ROM (A few kilobytes).

## **FLASH**

- FLASH memory is a variation of EEPROM technology
- It combines the re-programmability of EEPROM and the high capacity of standard ROMs
- FLASH memory is organized as sectors (blocks) or pages
- FLASH memory stores information in an array of floating gate MOSFET transistors
- The erasing of memory can be done at sector level or page level without affecting the other sectors or pages
- Each sector/page should be erased before re-programming

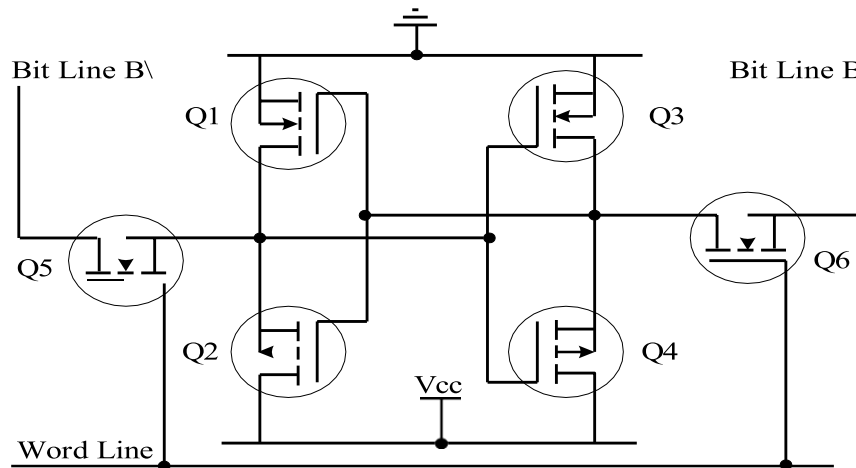
## **Read-Write Memory/Random Access Memory (RAM)**

- RAM is the data memory or working memory of the controller/processor
- RAM is volatile, meaning when the power is turned off, all the contents are destroyed
- RAM is a direct access memory, meaning we can access the desired memory location directly without the need for traversing through the entire memory locations to reach the desired memory position (i.e. Random Access of memory location)
- 



## Static RAM (SRAM)

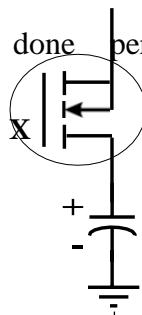
- Static RAM stores data in the form of Voltage. They are made up of flip-flops
- In typical implementation, an SRAM cell (bit) is realized using 6 transistors (or 6 MOSFETs). Four of the transistors are used for building the latch (flip-flop) part of the memory cell and 2 for controlling the access.
- Static RAM is the fastest form of RAM available. SRAM is fast in operation due to



its resistive networking and switching capabilities

## Dynamic RAM (DRAM)

- Dynamic RAM stores data in the form of charge. They are made up of MOS transistor gates
- The advantages of DRAM are its high density and low cost compared to SRAM
- The disadvantage is that since the information is stored as charge it gets leaked off with time and to prevent this, they need to be refreshed periodically
- Special circuits called DRAM controllers are used for the refreshing operation. The refresh operation is done periodically in milliseconds inter



## SRAM Vs DRAM

SRAM Cell	DRAM Cell
Made up of 6 CMOS transistors (MOSFET)	Made up of a MOSFET and a capacitor
Doesn't Require refreshing	Requires refreshing
Low capacity (Less dense)	High Capacity (Highly dense)
More expensive	Less Expensive
Fast in operation. Typical access time is 10ns	Slow in operation due to refresh requirements. Typical access time is 60ns. Write operation is faster than read operation.

### Non Volatile RAM (NVRAM)

- Random access memory with battery backup
- It contains Static RAM based memory and a minute battery for providing supply to the memory in the absence of external power supply
- The memory and battery are packed together in a single package
- NVRAM is used for the non volatile storage of results of operations or for setting up of flags etc
- The life span of NVRAM is expected to be around 10 years
- DS1744 from Maxim/Dallas is an example for 32KB NVRAM
- Memory selection for Embedded Systems: Selection of suitable memory is very much essential step-in high-performance applications, because the challenges and limitations of the system performance are often decided upon the type of memory architecture.
- Systems memory requirement depend primarily on the nature of the application that is planned to run on the system.
- Memory performance and capacity requirement for low cost systems are small, whereas memory throughput can be the most critical requirement in a complex, high performance system.
- Following are the factors that are to be considered while selecting the memory devices,
  - Speed
  - Data storage size and capacity
  - Bus width
  - Power consumption
  - Cost

## **Sensors & Actuators:**

- Embedded system is in constant interaction with the real world
- Controlling/monitoring functions executed by the embedded system is achieved in accordance with the changes happening to the Real World.
- The changes in the system environment or variables are detected by the sensors connected to the input port of the embedded system.
- If the embedded system is designed for any controlling purpose, the system will produce some changes in controlling variable to bring the controlled variable to the desired value.
- It is achieved through an actuator connected to the out port of the embedded system.

### **Sensor:**

- A transducer device which converts energy from one form to another for any measurement or control purpose. Sensors acts as input device Eg. Hall Effect Sensor which measures the distance between the cushion and magnet in the Smart Running shoes from adidas
- Example: IR, humidity , PIR(passive infra red) , ultrasonic , piezoelectric , smoke sensors

### **Actuator:**

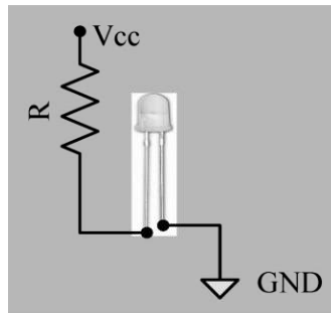
- A form of transducer device (mechanical or electrical) which converts signals to corresponding physical action (motion).
- Actuator acts as an output device
- Example: Micro motor actuator which adjusts the position of the cushioning element in the Smart Running shoes from adidas.

## **The I/O Subsystem:**

- The I/O subsystem of the embedded system facilitates the interaction of the embedded system with external world
- The interaction happens through the sensors and actuators connected to the Input and output ports respectively of the embedded system
- The sensors may not be directly interfaced to the Input ports, instead they may be interfaced through signal conditioning and translating systems like ADC.

## I/O Devices - Light Emitting Diode (LED):

- LED is a p-n junction diode and contains a CATHODE and ANODE. For functioning the anode is connected to +ve end of power supply and cathode is connected to -ve end of power supply.
- The maximum current flowing through the LED is limited by connecting a RESISTOR in series between the power supply and LED as shown in the figure below

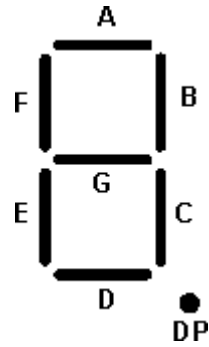
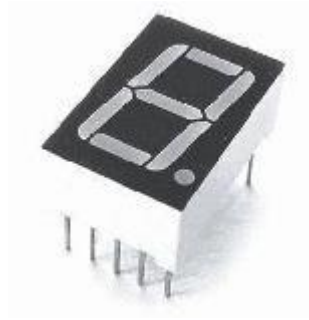


There are two ways to interface an LED to a microprocessor/microcontroller:

- The Anode of LED is connected to the port pin and cathode to Ground : In this approach the port pin sources the current to the LED when it is at logic high (ie. 1).
- The Cathode of LED is connected to the port pin and Anode to Vcc : In this approach the port pin sources the current to the LED when it is at logic high (ie. 1). Here the port pin sinks the current and the LED is turned ON when the port pin is at Logic low (ie. 0)

## I/O Devices – 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
- The LED segments are named A to G and the decimal point LED segment is named as DP
- The LED Segments A to G and DP should be lit accordingly to display numbers and characters

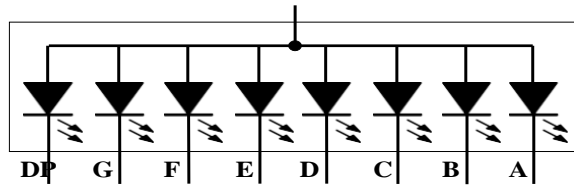


- The 7 – segment LED displays are available in two different configurations, namely; Common anode and Common cathode
- In the Common anode configuration, the anodes of the 8 segments are connected commonly whereas in the Common cathode configuration, the 8 LED segments share a common cathode line

Based on the configuration of the 7 – segment LED unit, the LED segment 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.

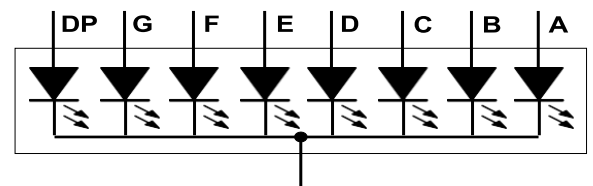
- The current flow through each of the LED segments should be limited to the maximum value supported by the LED display unit

#### Anode



Common Anode LED Display

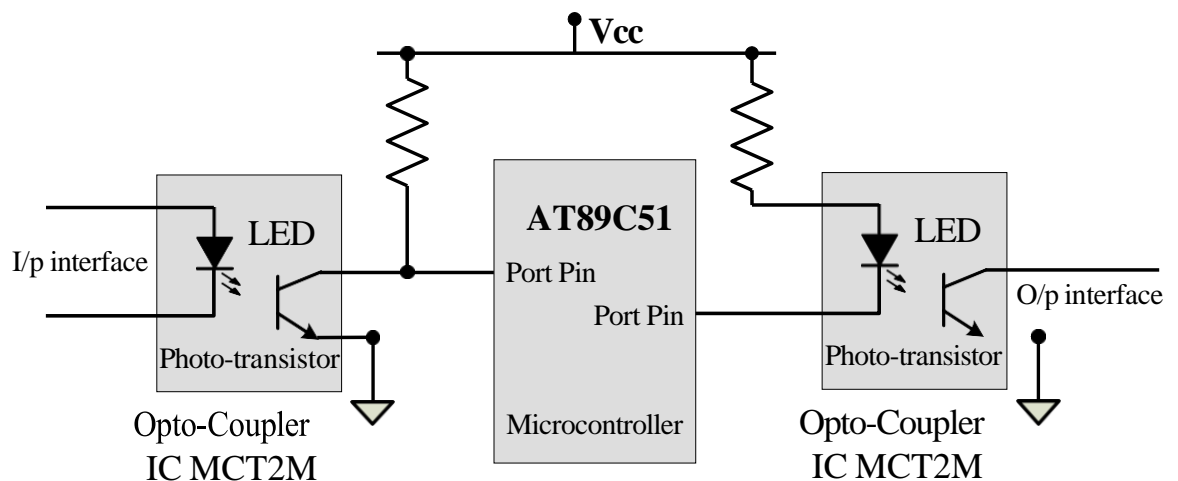
#### Common Cathode LED Display



Cathode

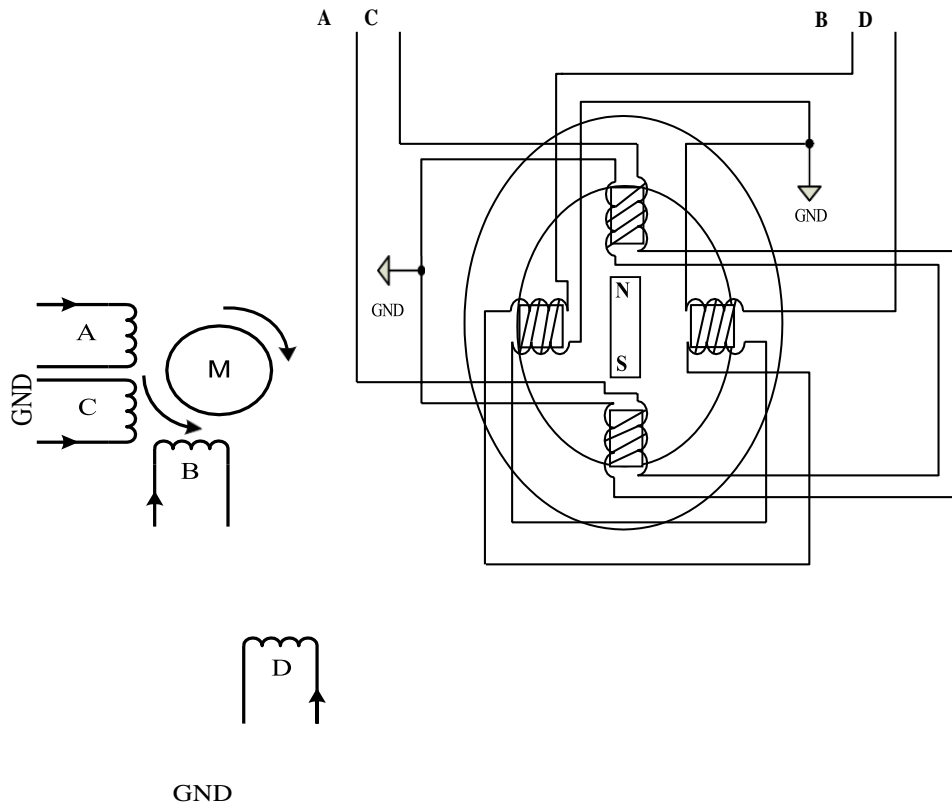
### I/O Devices – Optocoupler

- Optocoupler is a solid state device to isolate two parts of a circuit.
- Optocoupler combines an LED and a photo-transistor in a single housing (package)
- In electronic circuits, optocoupler is used for suppressing interference in data communication, circuit isolation, High voltage separation, simultaneous separation and intensification signal etc.



## **I/O Devices – Stepper Motor:**

- Stepper motor is an electro mechanical device which generates discrete displacement (motion) in response to dc electrical signals LED O/P interface
- 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
  - Unipolar
  - Bipolar



**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.

**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.

## 2 Phase Unipolar Stepper Motor – Stepping Sequence

**Full Step:**

In the full step mode both the phases are energized simultaneously. The coils A, B, C and D are energized in the 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

**Wave Step:**

only one phase is energized at a time & each coils of the phase are energized alternatively. The coils A, B, C and D are energized in the 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

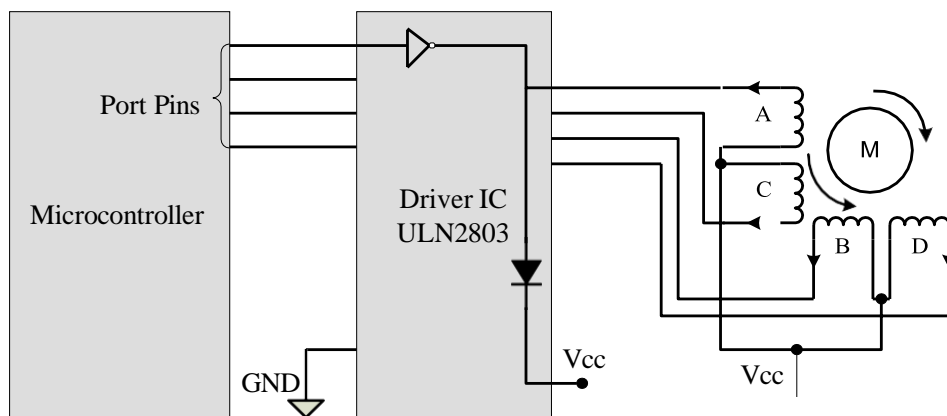
### Half Step:

Half step uses the combination of wave and full step. It has the highest torque and stability. The coils A, B, C and D are energized in the order

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

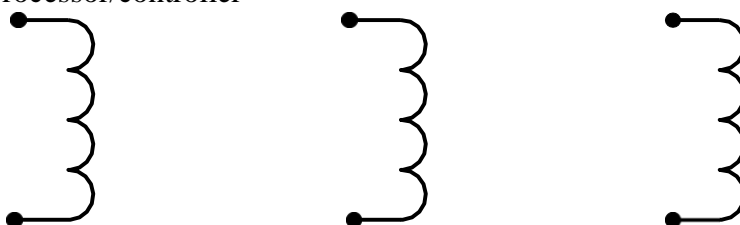
### 2 Phase Unipolar Stepper Motor – Interfacing

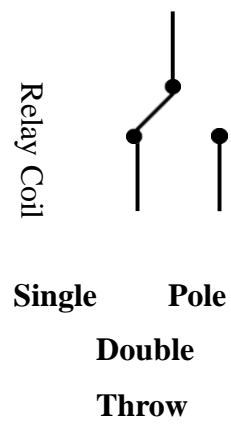
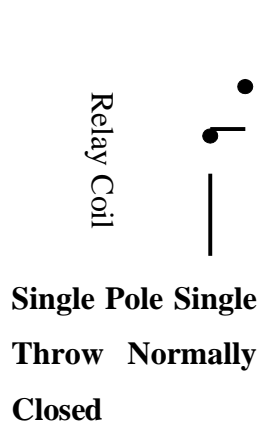
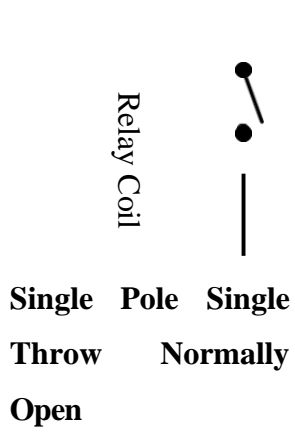
- Depending on the current and voltage requirements, special driving circuits are required to interface the stepper motor with microcontroller.
- Stepper motor driving ICs like ULN2803 or simple transistor based driving circuit can be used for interfacing stepper motors with processor/controller

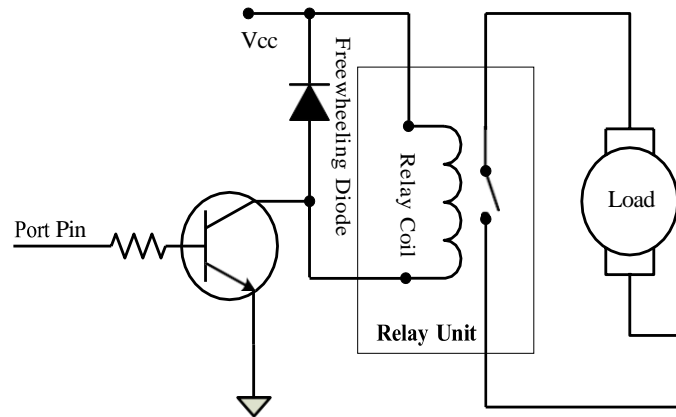


### I/O Devices – Relay:

- An electro mechanical device which 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.
- The Relay is normally controlled using a relay driver circuit connected to the port pin of the processor/controller







### I/O Devices -Piezo Buzzer:

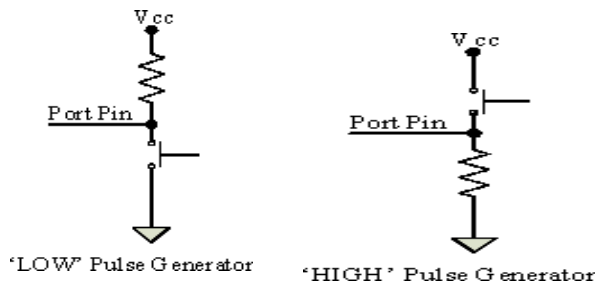
- It is a piezoelectric device for generating audio indications in embedded applications.
- A Piezo buzzer contains a piezoelectric diaphragm which produces audible sound in response to the voltage applied to it.
- Piezoelectric buzzers are available in two types
  1. Self-driving
  2. External driving
- Self-driving contains are the necessary components to generate sound at a predefined tone.
- 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/Controller.



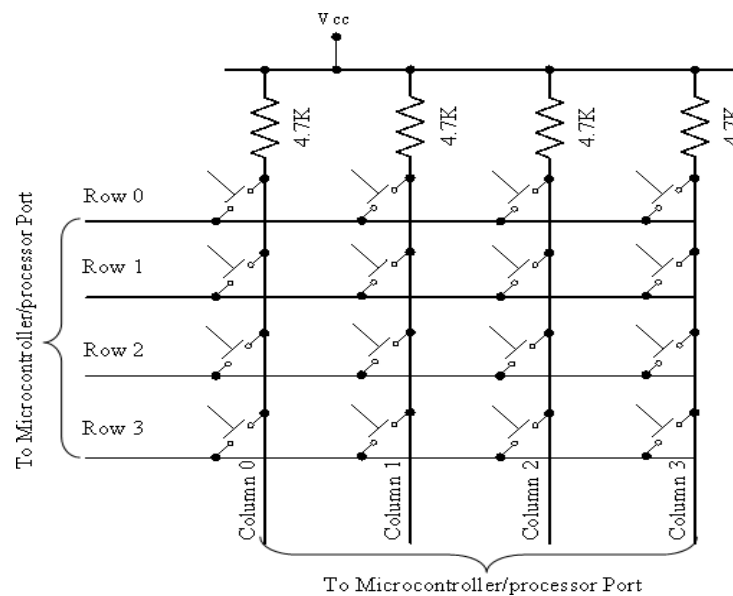
### I/O Devices – Push button switch:

- Push Button switch is an input device.

- Push button switch comes in two configurations, namely “Push to Make” and “Push to Break”
- The switch is normally in the open state and it makes a circuit contact when it is pushed or pressed in the “Push to Make” configuration.
- In the “Push to Break” configuration, the switch normally in the closed state and it breaks the circuit contact when it is pushed or pressed

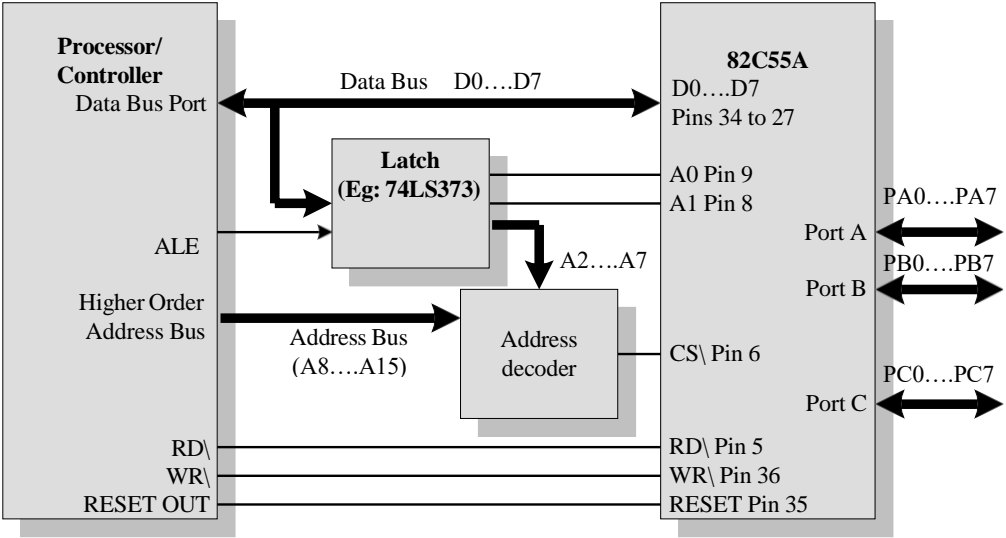


- If the number of keys required is very limited, push button switches can be used and they can be directly interfaced to the port pins for reading.
- Matrix keyboard is an optimum solution for handling large number of key requirements



### I/O Devices – Programmable Peripheral Interface (PPI)

- PPI devices are used for extending the I/O capabilities of processors /controllers
- 8255A is a popular PPI device for 8 bit processors/controllers
- 8255A supports 24 I/O pins and these I/O pins can be grouped as either
  - Three 8-bit parallel ports (Port A, Port B and Port C) or
  - Two 8 bit parallel ports (Port A and Port B) with Port C in any one of the following configurations
    - As 8 individual I/O pins
    - Two 4bit Ports namely Port C<sub>UPPER</sub> (C<sub>U</sub>) and Port C<sub>LOWER</sub> (C<sub>L</sub>)
- The Configuration of ports is done through the Control Register of 8255A



Bit	Description
D0	<b>Port C Lower (<math>C_L</math>) I/O mode selector</b> D0 = 1; Sets $C_L$ as Input Port D0 = 0; Sets $C_L$ as Output Port
D1	<b>Port B I/O mode selector</b> D1 = 1; Sets Port B as Input Port D1 = 0; Sets Port B as Output Port
D2	<b>Mode Selector for Port C lower and Port B</b> D2 = 0; Mode 0 – Port B functions as 8bit I/O Port. Port C Lower functions as 4 bit Port. D2 = 1; Mode 1 – Handshake Mode. Port B uses 3 bits of Port C as handshake signals
D3	<b>Port C Upper (<math>C_U</math>) I/O mode selector</b> D3 = 1; Sets $C_U$ as Input Port D3 = 0; Sets $C_U$ as Output Port
D4	<b>Port A I/O mode selector</b> D4 = 1; Sets Port A as Input Port D4 = 0; Sets Port A as Output Port
D5, D6	<b>Mode Selector for Port C upper and Port A</b> D6 D5 = 00; Mode 0 – Simple I/O mode D6 D5 = 01; Mode 1 – Handshake Mode. Port A uses 3 bits of Port C as handshake signals D6 D5 = 1X; Mode 2. X can be 0 or 1 – Port A functions as Bi-directional Port
D7	<b>Control/Data mode selector for Port C</b> D7 = 1; I/O mode. D7 = 0; Bit Set/Reset (BSR) mode.

## 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:
  - Onboard Communication Interface (Device/board level communication interface):
    - The communication channel which interconnects the various components within an embedded product is referred as Device/board level communication interface (Onboard Communication Interface)
    - E.g.: Serial interfaces like I2C, SPI, UART, 1-Wire, etc. and parallel bus interface.
  - External Communication Interface (Product level communication interface):
    - The communication channel which interconnects the embedded product with the external world is called Product level communication interface (External Communication Interface)
    - The external communication interface can be either wired media or wireless media and it can be a serial or parallel interface.

- E.g.: Wireless interfaces like Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves (RF), GPRS, etc. and wired interfaces like RS-232C/RS-422/RS-485, USB, Ethernet IEEE 1394 port, Parallel port, CF-II interface, SDIO, PCMCIA, etc.

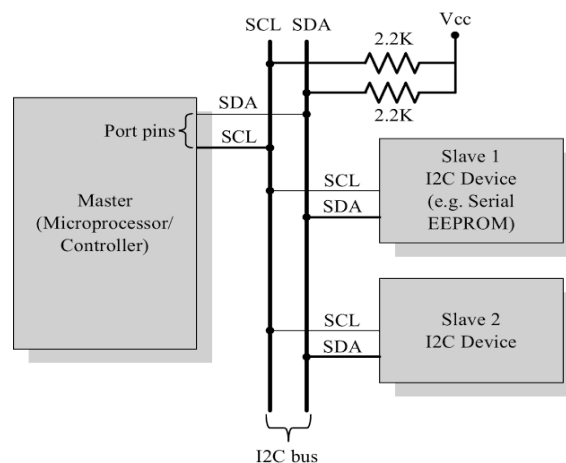
**Device/board level or on board communication interfaces:**

**I2C (Inter Integrated Circuit) Bus:**

- The Inter Integrated Circuit Bus (I2C or I<sup>2</sup>C Pronounced 'I square C') is a synchronous bi-directional half duplex two wire serial interface bus.

- (Half duplex - one-directional communication at a given point of time)
- The concept of I2C bus was developed by Philips Semiconductors in the early 1980s.
- The original intention of I2C was to provide an easy way of connection between a microprocessor/microcontroller system and the peripheral chips in television sets.
- The I2C bus comprise of two bus lines:
  - Serial Clock (SCL line) – responsible for generating synchronisation clock pulses
  - Serial Data (SDA line) – responsible for transmitting the serial data across devices
- I2C bus is a shared bus system to which many number of I2C devices can be connected. Devices connected to the I2C bus can act as either ‘Master’ device or ‘Slave’ device
- The ‘Master’ device is responsible for controlling the communication by initiating/terminating data transfer, sending data and generating necessary synchronization clock pulses
- ‘Slave’ devices wait for the commands from the master and respond upon receiving the commands
- ‘Master’ and ‘Slave’ devices can act as either transmitter or receiver
- Regardless whether a master is acting as transmitter or receiver, the synchronization clock signal is generated by the ‘Master’ device only
- I2C supports multi masters on the same bus

The following bus interface diagram illustrates the connection of master and slave devices on the I2C bus



- The I2C bus interface is built around an input buffer and an open drain or collector transistor.
- When the bus is in the idle state, the open drain/collector transistor will be in the floating state and the output lines (SDA and SCL) switch to the 'High Impedance' state.
- For proper operation of the bus, both the bus lines should be pulled to the supply voltage (+5 V for TTL family and +3.3V for CMOS family devices) using pull-up resistors.
- With pull-up resistors, the output lines of the bus in the idle state will be 'HIGH'
- The address of a I2C device is assigned by hardwiring the address lines of the device to the desired logic level. Done at the time of designing the embedded hardware.
- The sequence of operations for communicating with an I2C slave device is listed below:
  1. The master device pulls the clock line (SCL) of the bus to 'HIGH'
  2. The master device pulls the data line (SDA) 'LOW', when the SCL line is at logic 'HIGH' (This is the 'Start' condition for data transfer)

3. The master device sends the address (7 bit or 10 bit wide) of the 'slave' device to which it wants to communicate, over the SDA line.
  - Clock pulses are generated at the SCL line for synchronizing the bit reception by the slave device.
  - The MSB of the data is always transmitted first.
  - The data in the bus is valid during the 'HIGH' period of the clock signal
4. The master device sends the Read or Write bit (Bit value = 1 Read operation; Bit value = 0 Write operation) according to the requirement
5. The master device waits for the acknowledgement bit from the slave device whose address is sent on the bus along with the Read/ Write operation command.
 

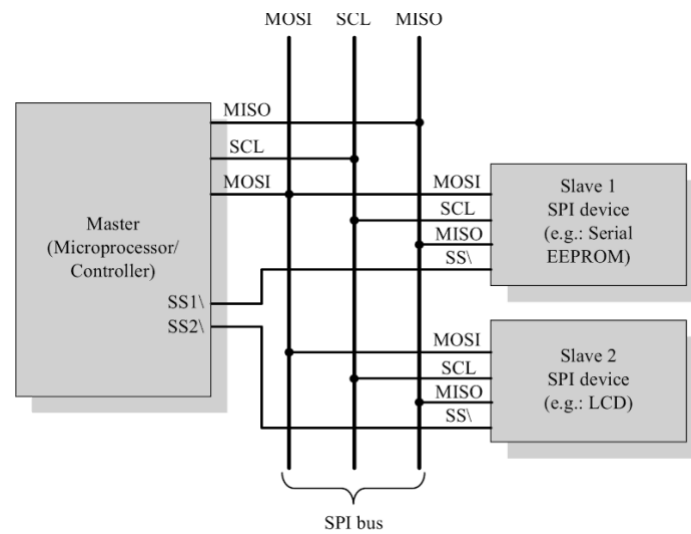
Slave devices connected to the bus compares the address received with the address assigned to them
6. The slave device with the address requested by the master device responds by sending an acknowledge bit (Bit value 1) over the SDA line
7. Upon receiving the acknowledge bit, the Master device sends the 8 bit data to the slave device over SDA line, if the requested operation is 'Write to device'.
 

If the requested operation is 'Read from device', the slave device sends data to the master over the SDA line
8. The master device waits for the acknowledgement bit from the device upon byte transfer complete for a write operation and sends an acknowledge bit to the Slave device for a read operation
9. The master device terminates the transfer by pulling the SDA line 'HIGH' when the clock line SCL is at logic 'HIGH' (Indicating the 'STOP' condition)

### **Serial Peripheral Interface (SPI) Bus:**

- The Serial Peripheral Interface Bus (SPI) is a synchronous bi-directional full duplex four wire serial interface bus.
- The concept of SPI is introduced by Motorola.
- SPI is a single master multi-slave system. It is possible to have a system where more than one SPI device can be master, provided the condition only one master device is active at any given point of time, is satisfied.
- SPI requires four signal lines for communication.

- Master Out Slave In (MOSI): Signal line carrying the data from master to slave device. It is also known as Slave Input/Slave Data In (SI/SDI)
  - Master In Slave Out (MISO): Signal line carrying the data from slave to master device. It is also known as Slave Output (SO/SDO)
  - Serial Clock (SCLK) : Signal line carrying the clock signals
  - Slave Select (SS/) : Signal line for slave device select. It is an active low signal
- The bus interface diagram shown in the figure illustrates the connection of master and slave devices on the SPI bus.



- The master device is responsible for generating the clock signal.
- Master device selects the required slave device by asserting the corresponding slave device's slave select signal 'LOW'. The data out line (MISO) of all the slave devices when not selected floats at high impedance state
- The serial data transmission through SPI Bus is fully configurable.
- The Serial Peripheral Control Register holds the various configuration parameters like master/slave selection for the device, baudrate selection for communication, clock signal control etc. The status register holds the status of various conditions for transmission and reception.
- SPI works on the principle of 'Shift Register'. The master and slave devices contain a special shift register for the data to transmit or receive. The size of the shift register is device dependent. Normally it is a multiple of 8.
- During transmission from the master to slave, the data in the master's shift register is shifted out to the MOSI pin and it enters the shift register of the slave device through the MOSI pin of the slave device. At the same time the shifted out data bit from the slave device's shift register enters the shift register of the master device through MISO pin

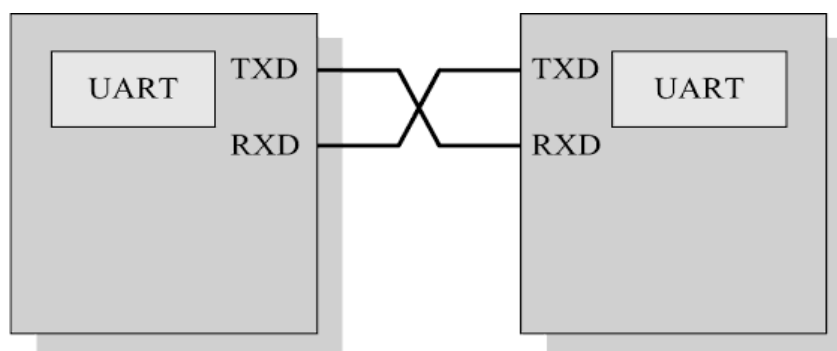
## Universal Asynchronous Receiver Transmitter (UART)

- Universal Asynchronous Receiver Transmitter (UART) based data transmission is an

asynchronous form of serial data transmission.

- It doesn't require a clock signal to synchronise the transmitting end and receiving end for transmission.
- Instead it relies upon the pre-defined agreement between the transmitting 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 are 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 baud rate settings.
- 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 1s in the transmitted bit stream and 0 for even number of 1s).
- 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.
- In addition to the serial data transmission function, UART provides hardware handshaking signal support for controlling the serial data flow.

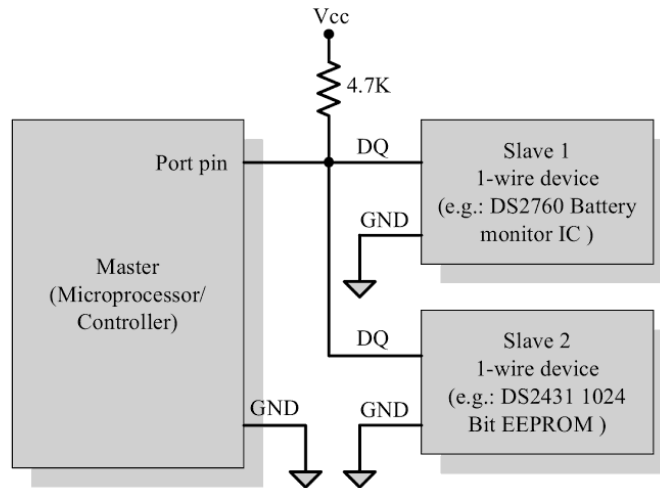


TXD: Transmitter line  
RXD: Receiver line

## 1- Wire Interface

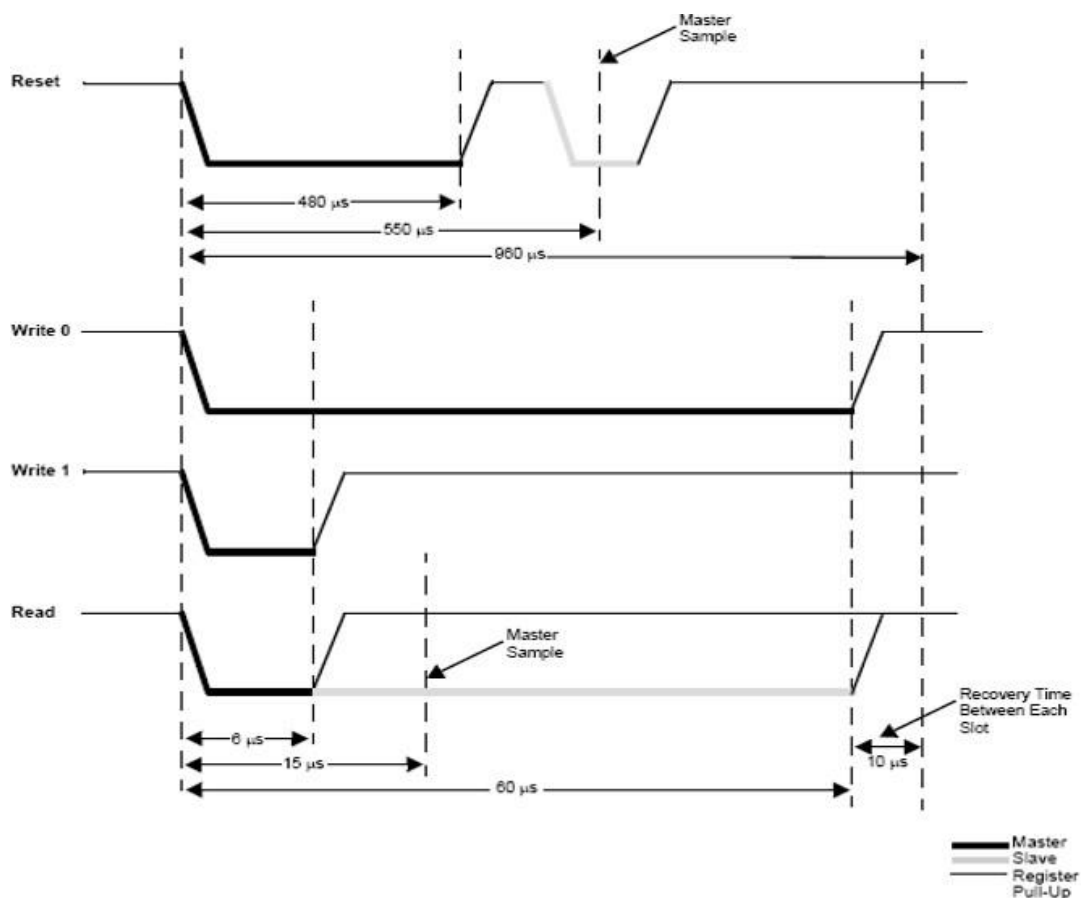
- 1-wire interface is an asynchronous half-duplex communication protocol developed by Maxim Dallas Semiconductor.
- It is also known as Dallas 1-Wire protocol.
- It makes use of only a single signal line (wire) called DQ for communication and follows the master-slave communication model.

- One of the key feature of 1-wire bus is that it allows power to be sent along the signal wire as well.
- The slave devices incorporate internal capacitor (typically of the order of 800 pF) to power the device from the signal line.
- The 1-wire interface supports a single master and one or more slave devices on the bus.
- The bus interface diagram shown in the figure illustrates the connection of master and slave devices on the 1-wire bus.



- Every 1-wire device contains a globally unique 64bit identification number stored within it.
- This unique identification number can be used for addressing individual devices present on the bus in case there are multiple slave devices connected to the 1-wire bus.
- The identifier has three parts: an 8-bit family code, a 48-bit serial number and an 8-bit CRC computed from the first 56 bits.
- The sequence of operation for communicating with a 1-wire slave device is listed below:
  1. The master device sends a 'Reset' pulse on the 1-wire bus.
  2. The slave device(s) present on the bus respond with a 'Presence' pulse.
  3. The master device sends a ROM command (Net Address Command followed by the 64 bit address of the device). This addresses the slave device(s) to which it wants to initiate a communication.
  4. The master device sends a read/write function command to read/write the internal memory or register of the slave device.
  5. The master initiates a Read data/Write data from the device or to the device.
- All communication over the 1-wire bus is master initiated.
- The communication over the 1-wire bus is divided into timeslots of 60 microseconds.
- The 'Reset' pulse occupies 8 time slots. For starting a communication, the master asserts the reset pulse by pulling the 1-wire bus 'LOW' for at least 8 time slots (480  $\mu$ s).

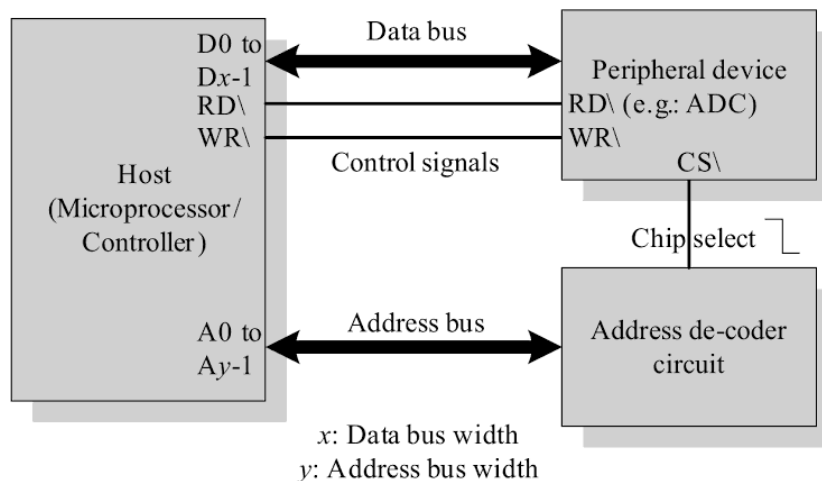
- If a 'slave' device is present on the bus and is ready for communication it should respond to the master with a 'Presence' pulse, within 60  $\mu\text{s}$  of the release of the 'Reset' pulse by the master.
- The slave device(s) responds with a 'Presence' pulse by pulling the 1-wire bus 'LOW' for a minimum of 1 time slot (60  $\mu\text{s}$ ).
- For writing a bit value of 1 on the 1-wire bus, the bus master pulls the bus for 1 to 15  $\mu\text{s}$  and then releases the bus for the rest of the time slot.
- A bit value of '0' is written on the bus by master pulling the bus for a minimum of 1 time slot (60  $\mu\text{s}$ ) and a maximum of 2 time slots (120  $\mu\text{s}$ ).
- To Read a bit from the slave device, the master pulls the bus 'LOW' for 1 to 15  $\mu\text{s}$ .
- If the slave wants to send a bit value '1' in response to the read request from the master, it simply releases the bus for the rest of the time slot.
- If the slave wants to send a bit value '0', it pulls the bus 'LOW' for the rest of the time slot.



## 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).
- The bus interface diagram shown in the figure illustrates the interfacing of devices through parallel interface.



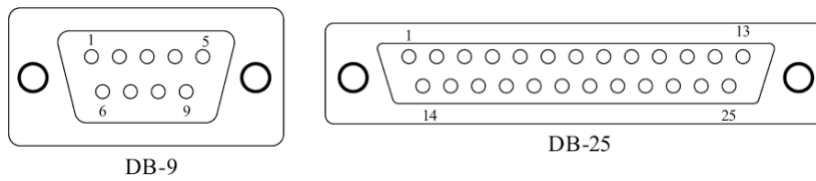
- Parallel communication is host processor initiated.
- If a device wants to initiate the communication, it can inform the same to the processor through interrupts.
- For this, the interrupt line of the device is connected to the interrupt line of the processor and the corresponding interrupt is enabled in the host processor.
- The width of the parallel interface is determined by the data bus width of the host processor.
- It can be 4 bit, 8 bit, 16 bit, 32 bit or 64 bit etc.
- The bus width supported by the device should be same as that of the host processor.
- Parallel data communication offers the highest speed for data transfer.

## External Communication Interfaces

### RS-232 C & RS-485

- RS-232 C (Recommended Standard number 232, revision C) is a legacy, full duplex, wired, asynchronous serial communication interface.
- The RS-232 interface was developed by the Electronics Industries Association (EIA) during the early 1960s.
- RS-232 extends the UART communication signals for external data communication.
- UART uses the standard TTL/CMOS logic (Logic 'High' corresponds to bit value 1 and Logic 'Low' corresponds to bit value 0) for bit transmission whereas RS-232 follows the EIA standard for bit transmission.
- As per the EIA standard, a logic '0' is represented with voltage between +3 and +25V and a logic '1' is represented with voltage between -3 and -25 V.

- In EIA standard, logic '0' is known as 'Space' and logic '1' as 'Mark'.
- The RS-232 interface defines various handshaking and control signals for communication apart from the 'Transmit' and 'Receive' signal lines for data communication.
- RS-232 supports two different types of connectors:
  - DB-9: 9-Pin connector
  - DB-25: 25-Pin connector.



- The pin details for the two connectors are explained in the following table:

Pin Name	Pin no: (For DB-9 Connector)	Pin no: (For DB-25 Connector)	Description
TXD	3	2	Transmit Pin for Transmitting Serial Data
RXD	2	3	Receive Pin for Receiving Serial Data
RTS	7	4	Request to send.
CTS	8	5	Clear To Send
DSR	6	6	Data Set Ready
GND	5	7	Signal Ground
DCD	1	8	Data Carrier Detect
DTR	4	20	Data Terminal Ready
RI	9	22	Ring Indicator
FG		1	Frame Ground
SDCD		12	Secondary DCD
SCTS		13	Secondary CTS
STXD		14	Secondary TXD
TC		15	Transmission Signal Element Timing
SRXD		16	Secondary RXD
RC		17	Receiver Signal Element Timing
SRTS		19	Secondary RTS
SQ		21	Signal Quality detector
NC		9	No Connection
NC		10	No Connection
NC		11	No Connection
NC		18	No Connection
NC		23	No Connection
NC		24	No Connection
NC		25	No Connection

- RS-232 is a point-to-point communication interface and the devices involved in RS-232 communication are called 'Data Terminal Equipment (DTE)' and 'Data Communication Equipment (DCE)'.
- If no data flow control is required, only TXD and RXD signal lines and ground line (GND) are required for data transmission and reception.
- The RXD pin of DCE should be connected to the TXD pin of DTE and vice versa for proper data transmission.
- If hardware data flow control is required for serial transmission, various control signal lines of the RS-232 connection are used appropriately.
- The Request To Send (RTS) and Clear To Send (CTS) signals co-ordinate the communication between DTE and DCE.
- Whenever the DTE has a data to send, it activates the RTS line and if the DCE is ready to accept the data, it activates the CTS line.
- The Data Terminal Ready (DTR) signal is activated by DTE when it is ready to

accept data.

- The Data Set Ready (DSR) is activated by DCE when it is ready for establishing a communication link.
- DTR should be in the activated state before the activation of DSR. The Data Carrier Detect (DCD) control signal is used by the DCE to indicate the DTE that a good signal is being received.
- Ring Indicator (RI) is a modem specific signal line for indicating an incoming call on the telephone line.

- As per the EIA standard RS-232 C supports baudrates up to 20Kbps (Upper limit 19.2 Kbps)
- The commonly used baudrates by devices are 300bps, 1200bps, 2400bps, 9600bps, 11.52Kbps and 19.2Kbps. 9600 is the popular baudrate setting used for PC communication.
- The maximum operating distance supported by RS-232 is 50 feet at the highest supported baudrate.
- RS-422 is another serial interface standard from EIA for differential data communication. It supports data rates up to 100Kbps and distance up to 400 ft.
- RS-422 supports multi-drop communication with one transmitter device and receiver devices up to 10.
- RS-485 is the enhanced version of RS-422 and it supports multi-drop communication with up to 32 transmitting devices (drivers) and 32 receiving devices on the bus.
- The communication between devices in the bus uses the 'addressing' mechanism to identify slave devices.

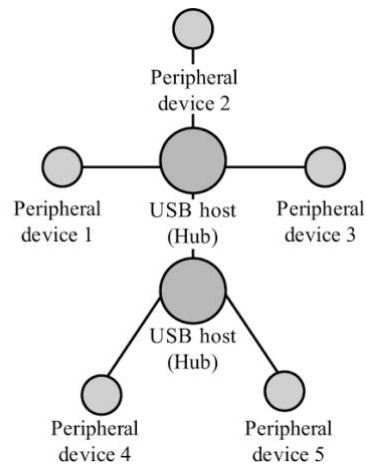
### **Universal Serial Bus (USB)**

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.

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.

Figure illustrates the star topology for USB device connection.



- 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:
  - Open Host Control Interface (OHCI)

- Universal Host Control Interface (UHCI)

- The physical connection between a USB peripheral device 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.

Pin no:	Pin name	Description
1	V <sub>BUS</sub>	Carries power (5V)
2	D-	Differential data carrier line
3	D+	Differential data carrier line
4	GND	Ground signal line



Type A Connector



Type B Connector



Type C Connector

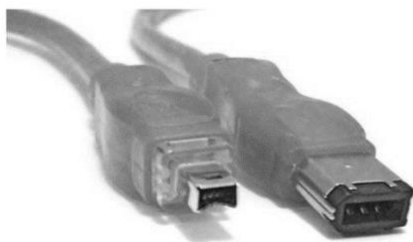
- Both Type A and Type B connectors contain 4 pins for communication.
- Each USB device contains a Product ID (PID) and a Vendor ID (VID).
  - Embedded into the USB chip by the USB device manufacturer.
  - The VID for a device is supplied by the USB standards forum.
  - PID and VID are essential for loading the drivers corresponding to a USB device for communication.
- USB supports four different types of data transfers:
- Control transfer : Used by USB system software to query, configure and issue commands to the USB device.

- Bulk transfer : Used for sending a block of data to a device.
  - Supports error checking and correction.
  - Transferring data to a printer is an example for bulk transfer.
- Isochronous data transfer : Used for real-time data communication.
  - Data is transmitted as streams in real-time.
  - 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 : 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.

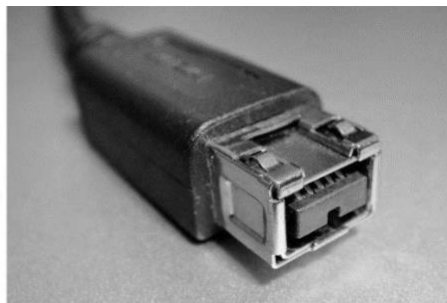
- Devices like Mouse and Keyboard, which transmits fewer amounts of data, uses Interrupt transfer.

## IEEE 1394 (Firewire)

- IEEE 1394 is a wired, isochronous high speed serial communication bus.
- It is also known as High Performance Serial Bus (HPSB).
- The research on 1394 was started by Apple Inc. in 1985 and the standard for this was coined by IEEE.
- The implementation of 1394 is available from various players with different names:
  - Firewire is the implementation from Apple Inc
  - i.LINK is the implementation from Sony Corporation
  - Lynx is the implementation from Texas Instruments
- 1394 supports peer-to-peer connection and point-to-multipoint communication allowing 63 devices to be connected on the bus in a tree topology.
- 1394 is a wired serial interface and it can support a cable length of up to 15 feet for interconnection.
- The 1394 standard supports a data rate of 400 to 3200 Mbits/second.
- The IEEE 1394 uses differential data transfer.
  - It increases the noise immunity.
- The interface cable supports 3 types of connectors, namely; 4-pin connector, 6-pin connector (alpha connector) and 9 pin connector (beta connector).
  - The 6 and 9 pin connectors carry power also to support external devices.
  - It can supply unregulated power in the range of 24 to 30V.



4-pin and 6-pin Connectors



9-pin Connector

- The table given below illustrates the pin details for 4, 6 and 9 pin connectors.

Pin name	Pin no: (4 Pin Connector)	Pin no: (6 Pin Connector)	Pin no: (9 Pin Connector)	Description
Power		1	8	Unregulated DC supply. 24 to 30V
Signal Ground		2	6	Ground connection
TPB-	1	3	1	Differential Signal line for Signal line B
TPB+	2	4	2	Differential Signal line for Signal line B
TPA-	3	5	3	Differential Signal line for Signal line A
TPA+	4	6	4	Differential Signal line for Signal line A
TPA(S)			5	Shield for the differential signal line A. Normally grounded
TPB(S)			9	Shield for the differential signal line B. Normally grounded
NC			7	No connection

- There are two differential data transfer lines A and B per connector.

- In a 1394 cable, normally the differential lines of A are connected to B (TPA+ to TPB+ and TPA- to TPB- ) and vice versa.
- 1394 is a popular communication interface for connecting embedded devices like Digital Camera, Camcorder, Scanners to desktop computers for data transfer and storage.
- IEEE 1394 doesn't require a host for communicating between devices. For example, you can directly connect a scanner with a printer for printing.

### **Infrared (IrDA)**

- Infrared (IrDA) is a serial, half duplex, line of sight based wireless technology for data communication between devices.
- It is in use from the olden days of communication and you may be very familiar with it. E.g.: The remote control of TV, VCD player, etc. works on Infrared.
- Infrared communication technique uses infrared waves of the electromagnetic spectrum for transmitting the data.
- It supports point-point and point-to-multipoint communication, provided all devices involved in the communication are within the line of sight.
- The typical communication range for IrDA lies in the range 10 cm to 1 m. The range can be increased by increasing the transmitting power of the IR device.
- IR supports data rates ranging from 9600bits/second to 16Mbps.
- Depending on the speed of data transmission IR is classified into:
  - Serial IR (SIR) – supports data rates ranging from 9600bps to 115.2kbps.
  - Medium IR (MIR) – supports data rates of 0.576Mbps and 1.152Mbps.
  - Fast IR (FIR) – supports data rates up to 4Mbps.
  - Very Fast IR (VFIR) – supports high data rates up to 16Mbps.
  - Ultra Fast IR (UFIR) – targeted to support a data rate up to 100Mbps.
- IrDA communication involves a transmitter unit for transmitting the data over IR and a receiver for receiving the data.
- Infrared Light Emitting Diode (LED) is the IR source for transmitter and at the receiving end a photodiode acts as the receiver.
- Both transmitter and receiver unit will be present in each device supporting IrDA communication for bidirectional data transfer. Such IR units are known as 'Transceiver'.

- Certain devices like a TV remote control always require unidirectional communication and so they contain either the transmitter or receiver unit. The remote control unit contains the transmitter unit and TV contains the receiver unit.
- Infrared Data Association (IrDA) is the regulatory body responsible for defining and licensing the specifications for IR data communication.
- IR communication has two essential parts: a physical link part and a protocol part.
- The physical link is responsible for the physical transmission of data between devices supporting IR communication. Protocol part is responsible for defining the rules of communication.
- The physical link works on the wireless principle making use of Infrared for communication.
- IrDA is a popular interface for file exchange and data transfer in low cost devices.
- IrDA was the prominent communication channel in mobile phones before Bluetooth's existence.

## **Bluetooth (BT)**

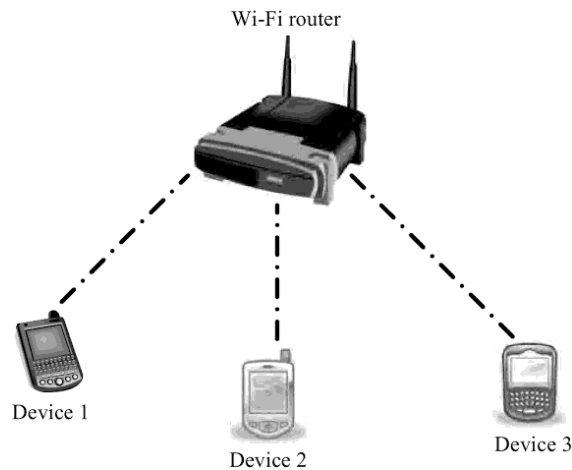
- Bluetooth is a low cost, low power, short range wireless technology for data and voice communication.
- Bluetooth was first proposed by Ericsson in 1994.
- Bluetooth operates at 2.4GHz of the Radio Frequency spectrum and uses the Frequency Hopping Spread Spectrum (FHSS) technique for communication.
- It supports a data rate of up to 1Mbps and a range of approximately 30 feet for data communication.
- Bluetooth communication has two essential parts – a physical link part and a protocol part.
- The physical link is responsible for the physical transmission of data between devices supporting Bluetooth communication. The protocol part is responsible for defining the rules of communication.
- The physical link works on the wireless principle making use of RF waves for communication.
- Bluetooth enabled devices essentially contain a Bluetooth wireless radio for the transmission and reception of data.
- The rules governing the Bluetooth communication is implemented in the 'Bluetooth protocol stack'. The Bluetooth communication IC holds the stack.
- Each Bluetooth device will have a 48 bit unique identification number.
- Bluetooth communication follows packet-based data transfer.
- Bluetooth supports point-to-point (device to device) and point-to-multipoint (device to multiple device broadcasting) wireless communication.
- The point-to-point communication follows the master-slave relationship. A Bluetooth device can function as either master or slave.
- When a network is formed with one Bluetooth device as master and more than one device as slaves, it is called a Piconet. A Piconet supports a maximum of seven slave devices.
- Bluetooth is the favourite choice for short range data communication in handheld embedded devices.
- Bluetooth technology is very popular among cell phone users as they are the easiest communication channel for transferring ringtones, music files, pictures, media files, etc. between neighbouring Bluetooth enabled phones.
- The Bluetooth standard specifies the minimum requirements that a Bluetooth device must support for a specific usage scenario.

- The specifications for Bluetooth communication is defined and licensed by the standards body 'Bluetooth Special Interest Group (SIG)'.

## **Wi-Fi**

- Wi-Fi or Wireless Fidelity is the popular wireless communication technique for networked communication of devices.
- Wi-Fi follows the IEEE 802.11 standard.
- Wi-Fi is intended for network 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 2.4 GHz or 5 GHz of radio spectrum and they co-exist with other ISM band devices like Bluetooth.
- Figure 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 1 Mbps to 1.73 Gbps depending on the standards (802.11a/b/g/n) and access/modulation method.
- Depending on the type of antenna and usage location (indoor/outdoor), Wi-Fi offers a range of 100 to 300 feet.

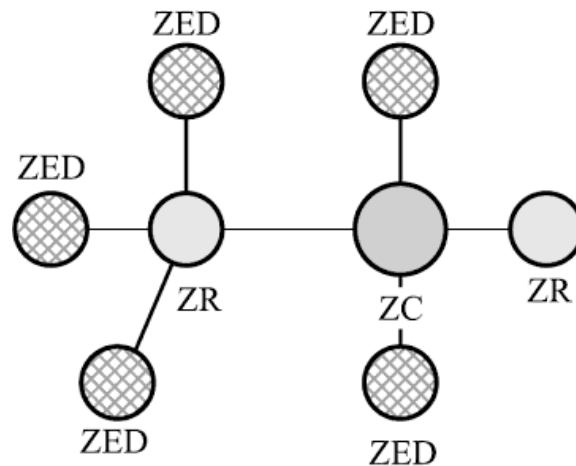
## **ZigBee**

- ZigBee is targeted for low power, low cost, low data rate and secure applications for Wireless Personal Area Networking (WPAN)
- The ZigBee specifications support a robust mesh network containing multiple nodes. This networking strategy makes the network reliable by permitting messages to

travel through a number of different paths to get from one node to another.

- ZigBee operates worldwide at the unlicensed bands of Radio spectrum, mainly at 2.400 to 2.484 GHz, 902 to 928 MHz and 868.0 to 868.6 MHz
- ZigBee Supports an operating distance of up to 100 meters and a data rate of 20 to 250Kbps
- ZigBee is primarily targeting application areas like Home & Industrial Automation, Energy Management, Home control/security, Medical/Patient tracking, Logistics & Asset tracking and sensor networks & active RFID
- In the ZigBee terminology, each ZigBee device falls under any one of the following ZigBee device category:
- ZigBee Coordinator (ZC)/Network Coordinator. The ZigBee coordinator acts as the root of the ZigBee network. The ZC is responsible for initiating the ZigBee network and it has the capability to store information about the network.

- ZigBee Router (ZR)/Full function Device (FFD) Responsible for passing information from device to another device or to another ZR.
- ZigBee End Device (ZED)/Reduced Function Device (RFD): End device containing ZigBee functionality for data communication. It can talk only with a ZR or ZC and doesn't have the capability to act as a mediator for transferring data from one device to another.



- The specifications for ZigBee is developed and managed by the ZigBee Alliance, a non- profit consortium of leading semiconductor manufacturers, technology providers, OEMs and end users worldwide.

### General Packet Radio Service (GPRS)

- General Packet Radio Service (GPRS) is a communication technique for transferring data over a mobile communication network like GSM.
- 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 171.2 kbps.
- 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 EDGE, High Speed Downlink Packet Access (HSDPA), Long Term Evolution (LTE), etc. which offers higher bandwidths for communication.

## **Embedded Firmware**

- Embedded firmware refers to the control algorithm (Program instructions) and or the configuration settings that an embedded system developer dumps into the code (Program) memory of the embedded system.
- It is an un-avoidable part of an embedded system.
- There are various methods available for developing the embedded firmware:
  1. Write the program in high level languages like Embedded C/C++ using an Integrated Development Environment (IDE). The IDE will contain an editor, compiler, linker, debugger, simulator, etc. IDEs are different for different family of processors/controllers. For example, Keil  $\mu$ Vision 4 IDE is used for all family members of 8051 microcontroller, since it contains the generic 8051 compiler C51.
  2. Write the program in Assembly language using the instructions supported by your application's target processor/controller.
- The program written in high level language or assembly code should be converted into a processor understandable machine code before loading it into the program memory.
- The process of converting the program written in either a high level language or processor/controller specific Assembly code to machine readable binary code is called 'HEX File Creation'.
- The methods used for 'HEX File Creation' is different depending on the programming techniques used. If the program is written in Embedded C/C++ using an IDE, the cross compiler included in the IDE converts it into corresponding processor/controller understandable 'HEX File'. If Assembly language based programming technique is used, the utilities supplied by the processor/controller vendors can be used to convert the source code into 'HEX File'. Also, third party tools are available, which may be of free of cost, for this conversion.
- For a beginner in the embedded software field, it is strongly recommended to use the high level language based development technique. Writing codes in a high level language is easy.
- The code written in high level language is highly portable. The same code can be used to run on different processor/controller with little or less modification. The only thing you need to do is re-compile the program with the required processor's IDE, after replacing the include files for that particular processor.

- The programs written in high level languages are not developer dependent. Any skilled programmer can trace out the functionalities of the program by just having a look at the program. It will be much easier if the source code contains necessary comments and documentation lines. It is very easy to debug and the overall system development time will be reduced to a greater extent.
- The embedded software development process in assembly language is tedious and time consuming.
- The developer needs to know about all the instruction sets of the processor/controller or at least he should carry an instruction set reference manual with him.
- A programmer using assembly language technique writes the program according to his view and taste. Often, he may be writing a method or functionality which can be achieved through a single instruction as an experienced person's point of view, by two or three instructions in his own style. So, the program will be highly dependent on the developer.
- It is very difficult for a second person to understand the code written in Assembly even if it is well documented.
- Two types of control algorithm design exist in embedded firmware development:  
The first type of control algorithm development is known as the infinite loop or 'super loop' based approach, where the control flow runs from top to bottom and then jumps back to the top of the program in a conventional procedure. It is similar to the while (1) {  
}; based technique in C.

- The second method deals with splitting the functions to be executed into tasks and running these tasks using a scheduler which is part of a General Purpose or Real Time Embedded Operating System (GPOS/RTOS).

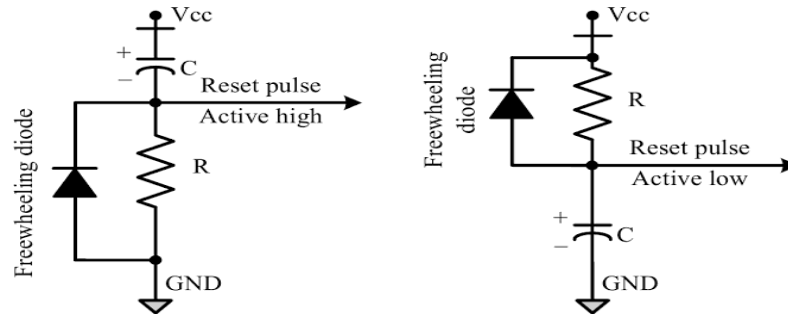
## **Other System Components**

- The other system components refer to the components/circuits/ICs which are necessary for the proper functioning of the embedded system.
- Some of these circuits may be essential for the proper functioning of the processor/controller and firmware execution. E.g.: Watchdog timer, Reset IC (or passive circuit), brown-out protection IC (or passive circuit), etc.
- Some of the controllers or SoCs integrate these components within a single IC and doesn't require such components externally connected to the chip for proper functioning.

## **Reset Circuit**

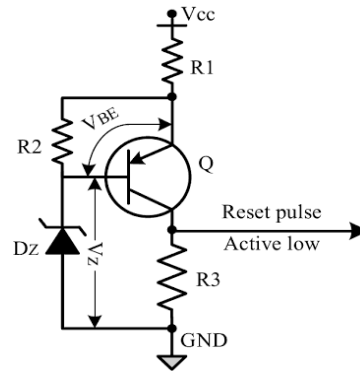
- The reset circuit is essential to ensure that the device is not operating at a voltage level where the device is not guaranteed to operate, during system power ON.
- The reset signal brings the internal registers and the different hardware systems of the processor/controller to a known state and starts the firmware execution from the reset vector. Normally from vector address 0x0000 for conventional processors/controllers.
- The reset signal can be either active high or active low.
- Since the processor operation is synchronised to a clock signal, the reset pulse should be wide enough to give time for the clock oscillator to stabilise before the internal reset state starts.
- The reset signal to the processor can be applied at power ON through an external passive reset circuit comprising a Capacitor and Resistor or through a standard Reset IC like MAX810 from Maxim Dallas.
- Select the reset IC based on the type of reset signal and logic level (CMOS/TTL) supported by the processor/controller in use.
- Some microprocessors/controllers contain built-in internal reset circuitry and they don't require external reset circuitry.

- Figure illustrates a resistor capacitor based passive reset circuit for active high and low configurations.
- The reset pulse width can be adjusted by changing the resistance value  $R$  and capacitance value  $C$ .



### Brown-out Protection Circuit

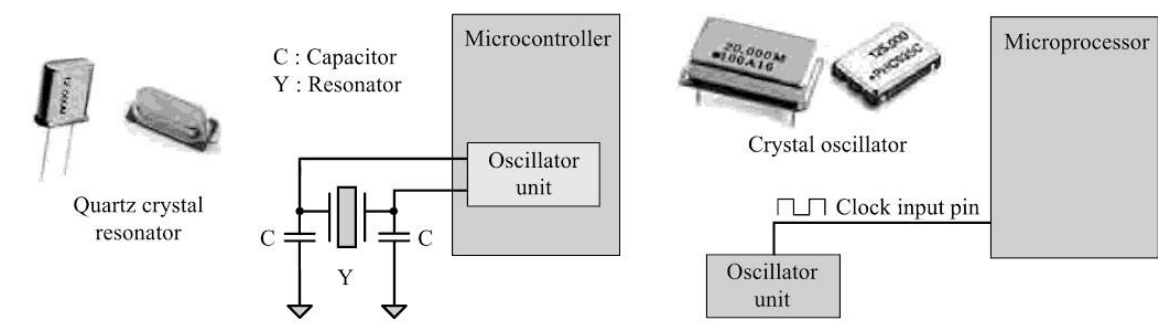
- Brown-out protection circuit prevents the processor/controller from unexpected program execution behaviour when the supply voltage to the processor/controller falls below a specified voltage.
- It is essential for battery powered devices since there are greater chances for the battery voltage to drop below the required threshold.
  - The processor behaviour may not be predictable if the supply voltage falls below the recommended operating voltage.
  - It may lead to situations like data corruption.
- A brown-out protection circuit holds the processor/controller in reset state, when the operating voltage falls below the threshold, until it rises above the threshold voltage.
- Certain processors/controllers support built in brown-out protection circuit which monitors the supply voltage internally.
- If the processor/controller doesn't integrate a built-in brown-out protection circuit, the same can be implemented using external passive circuits or supervisor ICs.
- Figure illustrates a brown-out circuit implementation using Zener diode and transistor for processor/controller with active low Reset logic.
- The Zener diode  $D_Z$  and transistor  $Q$  forms the heart of this circuit.
- The transistor conducts always when the supply voltage  $V$  greater than that of the sum of  $V_{BE}$  and  $V_Z$  (Zener voltage).
- The transistor stops conducting when the supply voltage falls below the sum of  $V_{BE}$  and  $V_Z$ .
- Select the Zener diode with required voltage for setting the low threshold value for  $V_{CC}$ .
- The values of  $R_1$ ,  $R_2$ , and  $R_3$  can be selected based on the electrical characteristics of the transistor in use.
- Microprocessor Supervisor ICs like DS1232 from Maxim also provides Brown-out protection.



## Oscillator Unit

- A microprocessor/microcontroller is a digital device made up of digital combinational and sequential circuits.
- The instruction execution of a microprocessor/controller occurs in sync with a clock signal.
- The oscillator unit of the embedded system is responsible for generating the precise clock for the processor. Analogous to the heart in living beings which produces heart beats.
- Certain processors/controllers integrate a built-in oscillator unit and simply require an external ceramic resonator/quartz crystal for producing the necessary clock signals.
- Quartz crystals and ceramic resonators are equivalent in operation; however, they possess physical difference.

- A quartz crystal is normally mounted in a hermetically sealed metal case with two leads protruding out of the case.
- Certain devices may not contain a built-in oscillator unit and require the clock pulses to be generated and supplied externally. Quartz crystal Oscillators are available in the form of chips and they can be used for generating the clock pulses in such cases.
- The speed of operation of a processor is primarily dependent on the clock frequency. However, we cannot increase the clock frequency blindly for increasing the speed of execution. The logical circuits lying inside the processor always have an upper threshold value for the maximum clock at which the system can run, beyond which the system becomes unstable and non functional.
- The total system power consumption is directly proportional to the clock frequency.
- The power consumption increases with increase in clock frequency. The accuracy of program execution depends on the accuracy of the clock signal.
- Figure illustrates the usage of quartz crystal/ceramic resonator and external oscillator chip for clock generation.



## Real-Time Clock (RTC)

- Real-Time Clock (RTC) is a system component responsible for keeping track of time.
- RTC holds information like current time (In hours, minutes and seconds) in 12 hours/24 hour format, date, month, year, day of the week, etc. and supplies timing reference to the system.
- RTC is intended to function even in the absence of power.
- RTCs are available in the form of Integrated Circuits from different semiconductor manufacturers like Maxim/Dallas, ST Microelectronics etc.

- The RTC chip contains a microchip for holding the time and date related information and backup battery cell for functioning in the absence of power, in a single IC package.
- The RTC chip is interfaced to the processor or controller of the embedded system.
- For Operating System based embedded devices, a timing reference is essential for synchronizing the operations of the OS kernel.
- The RTC can interrupt the OS kernel by asserting the interrupt line of the processor/controller to which the RTC interrupt line is connected.
- The OS kernel identifies the interrupt in terms of the Interrupt Request (IRQ) number generated by an interrupt controller.
- One IRQ can be assigned to the RTC interrupt and the kernel can perform necessary operations like system date time updation, managing software timers, etc. when an RTC timer tick interrupt occurs.

The RTC can be configured to interrupt the processor at predefined intervals or to interrupt the processor when the RTC register reaches a specified value (used as alarm interrupt).

## **Watchdog Timer**

- A watchdog timer, or simply a watchdog, is a hardware timer for monitoring the firmware execution and resetting the system processor/microcontroller when the program execution hangs up.
- Depending on the internal implementation, the watchdog timer increments or decrements a free running counter with each clock pulse and generates a reset signal to reset the processor if the count reaches zero for a down counting watchdog, or the highest count value for an up counting watchdog.
- If the watchdog counter is in the enabled state, the firmware can write a zero (for up counting watchdog implementation) to it before starting the execution of a piece of code (which is susceptible to execution hang up) and the watchdog will start counting.
- If the firmware execution doesn't complete due to malfunctioning, within the time required by the watchdog to reach the maximum count, the counter will generate a reset pulse and this will reset the processor.
- If the firmware execution completes before the expiration of the watchdog timer you can reset the count by writing a 0 (for an up counting watchdog timer) to the watchdog timer register.
- Most of the processors implement watchdog as a built-in component and provides status register to control the watchdog timer (like enabling and disabling watchdog functioning) and watchdog timer register for writing the count value.
- If the processor/controller doesn't contain a built-in watchdog timer, the same can be implemented using an external watchdog timer IC circuit.
- The external watchdog timer uses hardware logic for enabling/disabling, resetting the watchdog count, etc. instead of the firmware based 'writing' to the status and watchdog timer register.
- The Microprocessor supervisor IC DS1232 integrates a hardware watchdog timer in it.
- In modern systems running on embedded operating systems, the watchdog can be implemented in such a way that when a watchdog timeout occurs, an interrupt is

generated instead of resetting the processor.

- The interrupt handler for this handles the situation in an appropriate fashion.
- Figure illustrates the implementation of an external watchdog timer based microprocessor supervisor circuit for a small scale embedded system.

