MICROCONTROLLER AND EMBEDDED SYSTEMS

18CS44

MODULE 3

EMBEDDED VS GENERAL COMPUTING SYSTEM:

https://en.wikibooks.org/wiki/Embedded_Systems/Embedded_Systems_Introduction

https://www.youtube.com/watch?v=oPn_adlC1Q0

What is an Embedded System? Give the characteristics of an embedded system.

An embedded system is an electronic/ electro-mechanical system designed to perform a specific function and is a combination of both hardware and firmware (software). Every embedded system is unique, and the hardware as well as the firmware is highly specialized to the application domain. Embedded systems are becoming an inevitable part of any product or equipment in all fields including household appliances, telecommunications, medical equipment, industrial control, consumer products, etc. Characteristics of Embedded Systems:

- Embedded Systems must be highly reliable and stable.
- Embedded Systems have minimal or no user interface.
- Embedded Systems are usually feedback oriented or reactive.
- Embedded Systems are typically designed to meet real time constraints.
- Embedded Systems have limited memory and limited number of peripherals.
- Embedded Systems are typically designed for specific application or purpose.
- Embedded Systems are designed for low power consumption, as they use battery power.

Differentiate Embedded System versus General Computing Systems

General Computing System	Embedded System
1. A combination of generic hardware and a General Purpose Operating System (GPOS) for executing a variety of applications.	1. A combination of special purpose hardware embedded OS for executing a specific set of applications.
2. Applications are alterable (programmable) by the user.	2. The firmware is pre-programmed and it is non-alterable by the end-user (there may be exceptions).
3. Performance is the key deciding factor in the selection of the system. Always, 'Faster is Better'.	3. Application-specific requirements (like performance, power requirements, memory usage, etc.).
4. Less/ not at all tailored towards reduced operating power requirements.	4. Highly tailored to take advantage of the power saving modes supported by the hardware and the operating system.
5. Need not be deterministic in execution behavior; response requirements are not time critical.	5. Execution behavior is deterministic for certain types of embedded systems like 'Hard Real Time' systems.

HISTORY OF EMBEDDED SYSTEMS:

https://en.wikipedia.org/wiki/Embedded_system

• Embedded systems were existing even before the Information Technology revolution. Initially, embedded systems were built around vacuum tube and transistor technologies; and embedded algorithm was developed by using low level programming languages. o Advances in semiconductor and nano-technology and IT revolution gave way to development of miniature embedded systems.

• ***Apollo Guidance Computer*** (AGC) developed (during 1960) by MIT Instrumentation Laboratory for the lunar expedition is the first recognized modern embedded system. o AGC included both Command Module (CM-to encircle the moon) and Lunar Excursion Module (LEM-to go down to the moon surface and land there safely).

o There were 16 reaction control thrusters, a descent engine (designed to provide thrust to the lunar model out of the lunar orbit and land it safely on moon) and an ascent engine.

o Original design was based on 4K words of fixed memory (ROM) and 256 words of erasable memory (RAM); which has been enhanced (during 1963) to 10K fixed and 1K erasable memory. The clock frequency was 1.024 MHz.

o The computing unit of AGC consisted of approximately 11 instructions on 16-bit word logic.

o A calculator type user interface was given and is known as DSKY (display/ keyboard).

• The first mass-produced embedded system was the guidance computer, ***Autonetics D-17***, for the Minuteman-I missile in 1961; built using discrete transistor logic and a hard-disk for main memory.

• The first microprocessor, the Intel 4004, was designed for calculators and other small systems; but still required many external memory and support chips.

• First microcontroller, TMS 1000, developed in 1974 by Texas Instruments. It had ROM, RAM, and clock circuitry on the chip along with the processing chip.

• In 1980, Intel introduced 8051 MCU and called it MCS-51 architecture.

• Laser and Inkjet printers emerged during 1980s; and early 1990, cell phones having five or six DSPs and CPUs emerged.

CLASSIFICATION OF EMBEDDED SYSTEMS:

https://www.watelectronics.com/classification-of-embedded-systems/

https://www.youtube.com/watch?v=FUWnYHBGw1o

What is an Embedded System? Explain the different classifications of embedded systems. Give example for each.

Classification Based on Generation:

Generation with Example	Description
First Generation (1G) - Digital telephone keypads - Stepper motor	✓ 8-bit microprocessor and 4-bit microcontroller like 8085 and Z80 was used in 1G. ✓ Hardware circuit was simple. ✓ Assembly code is used for developing firmware.
Second Generation (2G) - Data acquisition systems like ADC, SCADA system	✓ Uses 16-bit microprocessor and 8-bit microcontroller. ✓ They are more complex and powerful than 1G microprocessor and microcontroller.
Third Generation (3G) - Robotics	✓ Uses 32-bit microprocessor and 16-bit microcontroller. ✓ Domain specific processor and controllers are used.
Fourth Generation (4G) - Smart phones	✓ Uses 64-bit microprocessor and 32-bit microcontroller. ✓ The concept of system on chips, multi-core processors evolved. ✓ Highly complex and very powerful.

Classification Based on Complexity and Performance:

1. ***Small-Scale Embedded Systems:*** o Embedded systems which are simple in application needs and the performance parameters are not time critical (E.g.: Electronic toy).

o Small-scale embedded systems are usually built around low performance and low cost 8 or 16 bit microprocessors/ microcontrollers.

o It may or may not contain an operating system for its functioning.

2. ***Medium-Scale Embedded Systems:*** o Embedded systems which are slightly complex in hardware and firmware (software) requirements.

o Medium-scale embedded systems are usually built around medium performance, low cost 16 or 32 bit microprocessors/ microcontrollers or digital signal processors.

o They usually contain an embedded operating system (general purpose/ real-time).

3. ***Large-Scale Embedded Systems/ Complex Systems:*** o Embedded systems which involve highly complex hardware and firmware. They are employed in mission critical applications demanding high performance.

o Large-scale embedded systems are commonly built around high performance 32 or 64 bit RISC processors/ controllers or Reconfigurable System on Chip (RSoC) or multi-core processors and programmable logic devices.

o They usually contain a high performance Real Time Operating System (RTOS) for task scheduling, prioritization, and management.

MAJOR APPLICATIONS AREAS OF EMBEDDED SYSTEMS:

https://www.elprocus.com/embedded-systems-real-time-applications/

What is an Embedded System? Explain the different applications of embedded systems.

Embedded systems play a vital role in our day-to-day life, starting from home to computer industry. Embedded technology has acquired a new dimension from its first generation model, the Apollo Guidance Computer, to the latest radio navigation system combined with in-car entertainment technology and wearable computing devices (Apple watch, Microsoft band, Fitbit fitness trackers, etc.). The application areas and the products in the embedded domain are countless. A few of the important domains and products are listed below:

1. *Consumer Electronics:* Camcorders, cameras, etc.

2. *Household Appliances:* Television, DVD players, washing machine, fridge, microwave oven, etc.

3. *Home Automation and Security Systems:* Air conditioners, sprinklers, intruder detection alarms, closed circuit television cameras, fire alarms, etc.

4. *Automotive Industry:* Anti-lock breaking systems (ABS), engine control, ignition systems, automatic navigation systems, etc.

5. *Telecom:* Cellular telephones, telephone switches, handset multimedia applications, etc.

6. *Computer Peripherals:* Printers, scanners, fax machines, etc.

7. *Computer Networking Systems:* Network routers, switches, hubs, firewalls, etc.

8. *Healthcare:* Different kinds of scanners, EEG, ECG machines, etc.

9. *Measurement & Instrumentation:* Digital multi meters, digital CROs, logic analyzers PLC systems, etc.

10. *Banking & Retail:* Automatic teller machines (ATM) and currency counters, point of sales (POS).

11. *Card Readers:* Barcode, smart card readers, hand held devices, etc.

12. *Wearable Devices:* Health and fitness trackers, Smartphone screen extension for notifications, etc.

13. Cloud Computing and Internet of Things (IoT).

PURPOSE OF EMBEDDED SYSTEMS:

What is Embedded System? Illustrate any four purpose of embedded systems.

As mentioned in the previous section, embedded systems are used in various domains like consumer electronics, home automation, telecommunications, automotive industry, healthcare, control & instrumentation, retail and banking applications, etc. Each embedded system is designed to serve the purpose of any one or a combination o the following tasks:

1. Data Collection, Storage, Representation

• Data is collection of facts, such as values or measurements. It can be numbers, words, measurements, observations, or even just description of things.

• Purpose of embedded system design is data collection. It 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 convert the analog signal to corresponding digital signal using analog to digital (A/D) converters.

• If the data is digital, it can be directly captured by digital embedded system.

o A digital camera is a typical example of an embedded system with data collection, storage, and representation of data. Images are captured and 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 (Liquid Crystal Display) unit.

2. Data Communication

• Embedded data communication systems are deployed in applications ranging from simple home networking systems to complex satellite communication systems.

o Network hubs, routers, switches are examples of dedicated data transmission embedded systems.

• Data transmission is in the form of wire medium or wireless medium. Initially wired medium is used by embedded systems; and as technology changes, wireless medium becomes de-facto standard in embedded systems.

o USB, TCP/ IP are examples of wired communication; and BlueTooth, ZigBee and Wi-Fi are examples for wireless communication.

• Data can be transmitted by analog means or by digital means.

3. 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.

o A digital hearing aid is a typical example of an embedded system employing data processing.

4. Monitoring

• Almost all embedded products coming under the medical domain are with monitoring functions.

o Patient heart beat is monitored by Electro cardiogram (ECG) machine.

• Digital CRO, digital multi-meters, and logic analyzers are examples of monitoring embedded systems.

5. Control

• Sensors and actuators are used for controlling the system.

o Sensors are connected to the input port for capturing the changes in environmental variable or measuring variable.

o Actuators connected to output port are controlled according to the changes in input variable.

• Air conditioner system used in our home to control the room temperature to a specified limit is a typical example for embedded system for control purpose. The air conditioner's compressor unit (actuator) is controlled according to the current room temperature (sensor) and the desired room temperature set by the user.

6. Application Specific User Interface

• These are embedded systems with application-specific user interfaces like buttons, switches, keypad, lights, bells, display units, etc.

• Mobile phone is an example for this. In mobile phone, the user interface is provided through the keypad, graphic LCD module, system speaker, vibration alert, etc.

CORE OF AN EMBEDDED SYSTEM INCLUDING ALL TYPES OF PROCESSOR/CONTROLLER:

Explain the components of a typical embedded system in detail.

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

1) General Purpose and Domain Specific Processors

a. Microprocessors

b. Microcontrollers

c. Digital Signal Processors

2) Application Specific Integrated Circuits (ASICs)

3) Programmable Logic Devices (PLDs)

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 a microprocessor or a microcontroller or a digital signal processor, depending on the domain and application.

Microprocessors: A Microprocessor is a silicon chip representing a central processing unit (CPU), which is capable of performing arithmetic as well as logical operations. In general, the CPU contains the Arithmetic and Logic Unit (ALU), control unit and working registers. A 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 November 1971. It was designed for older day's calculators. In April 1974, Intel launched the first 8-bit processor, the Intel 8080, with 16-bit address bus and program counter and seven 8-bit registers. Intel 8080 was the most commonly used processors for industrial control and other embedded applications in the 1975s. Immediately after the release of Intel 8080, Motorola also entered the market with their processor, Motorola 6800 with a different architecture and instruction set compared to 8080. In 1976 Intel came up with the upgraded version of 8080 – Intel 8085, with two newly added instructions, three interrupt pins and serial I/0. Clock generator and bus controller circuits were built-in and the power supply part was modified to a single +5 V supply. In July 1976 Zilog entered the microprocessor market with its Z80 processor as competitor to Intel. Intel, AMD, Freescale, GLOBALFOUNDRIES, TI, Cyrix, NVIDIA, Qualcomm, MediaTek, etc. are the key players in the processor market. Intel still leads the market with cutting edge technologies in the processor industry.

Microcontrollers: 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. Microcontrollers can be considered as a super set of microprocessors. Since a microcontroller contains all the necessary functional blocks for independent working, they found greater place in the embedded domain in place of microprocessors. Apart from this, they are cheap, cost effective and are readily available in the market. Texas Instrument's TMS 1000 (1974) is considered as the world's first microcontroller. TI followed Intel's 4004, 4-bit processor design and added some amount of RAM, program storage memory (ROM) and 1/O support on a single chip, there by eliminated the requirement of multiple hardware chips for self-functioning.

In 1977 Intel entered the microcontroller market with a family of controllers coming under one umbrella named MCS-48™ family. Eventually Intel came out with its most fruitful design in the 8-bit microcontroller domain-the 8051 family and its derivatives. 8051 is the most popular and powerful 8-bit microcontroller ever built. It was developed in the 1980s and was put under the family MCS-51. Almost 75% of the microcontrollers used in the embedded domain were 8051 family based controllers during the 1980-90s. 8051 processor cores are used in more than 100 devices by more than 20 independent manufacturers like Maxim, Philips, Atmel, etc. under the license from Intel. Due to the low cost, wide availability, memory efficient instruction set, mature development tools and Boolean processing (bit manipulation operation) capability, 8051 family derivative microcontrollers are much used in high-volume consumer electronic devices, entertainment industry and other gadgets where cost-cutting is essential.

What is Digital Signal Processor (DSP)? Explain the role of DSP in embedded system design.

https://en.wikipedia.org/wiki/Digital_signal_processor

https://embeddedartistry.com/fieldmanual-terms/digital-signal-processor/

Digital Signal Processors are powerful special purpose 8/ 16/ 32 bit microprocessors designed specifically to meet the computational demands and power constraints of today's embedded audio, video, and communications applications.

Digital signal processors are 2 to 3 times faster than the general purpose microprocessors in signal processing applications. This is because of the architectural difference between the two. DSPs implement algorithms in hardware which speeds up the execution, whereas general purpose processors implement the algorithm in firmware and the speed of execution depends primarily on the clock for the processors. In general, DSP can be viewed as a microchip designed for performing high speed computational operations for 'addition', 'subtraction', 'multiplication' and 'division'. A typical digital signal processor incorporates the following four key units:

1. Program Memory: Memory for storing the program required by DSP to process the data.
2. Data Memory: Working memory for storing temporary variables and data/ signal to be processed.
3. Computational Engine: Performs the signal processing in accordance with the stored program memory. Computational Engine incorporates many specialized arithmetic units and each of them operates simultaneously to increase the execution speed. It also incorporates multiple hardware shifters for shifting operands and thereby saves execution time.
4. I/O Unit: Acts as an interface between the outside world and DSP. It is responsible for capturing signals to be processed and delivering the processed signals.

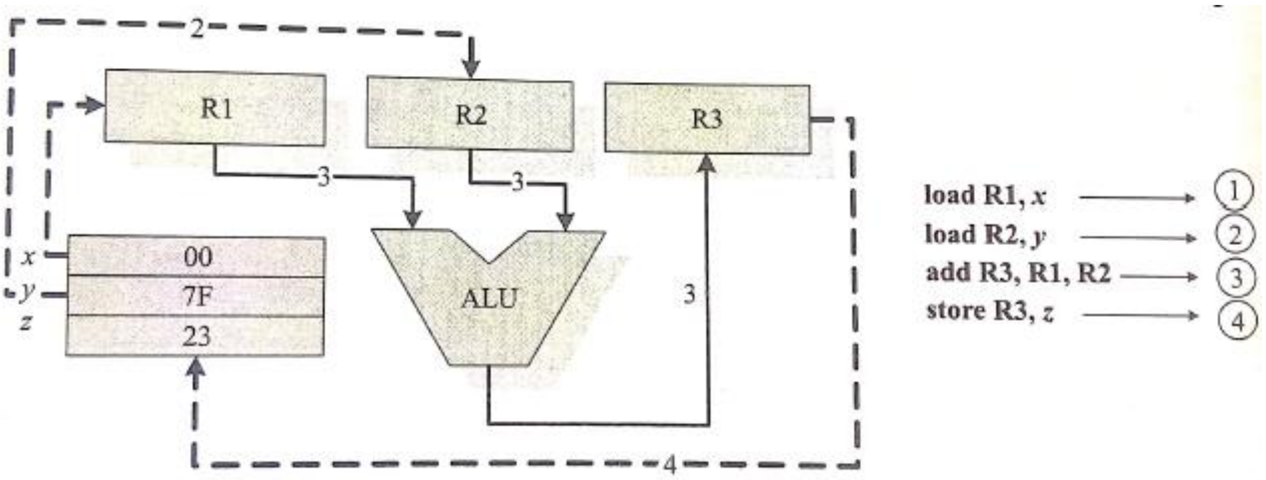
Audio video signal processing, telecommunication and multimedia applications are typical examples where DSP is employed. Digital signal processing employs a large amount of real-time calculations. Sum of Products (SOP) calculation, Convolution, Fast Fourier Transform (FFT), Discrete Fourier Transform (DFT), etc, are some of the operations performed by digital signal processors. Blackfin® processors from Analog Devices is an example of DSP which delivers breakthrough signal processing performance and power efficiency while also offering a full 32-bit RISC MCU programming model.

Explain the concept of Load-Store architecture and Instruction Pipelining

https://en.wikipedia.org/wiki/Load%E2%80%93store_architecture

https://www.youtube.com/watch?v=YhGv5AOcz1s

Load Store Operation and Instruction Pipelining: As mentioned earlier, the RISC processor instruction set is orthogonal, meaning 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 concept of Load Store Architecture is illustrated with the following example: Suppose x, y and z are memory locations and we want to add the contents of x and y and store the result in location z. Under the load store architecture the same is achieved with 4 instructions as shown in following Figure.

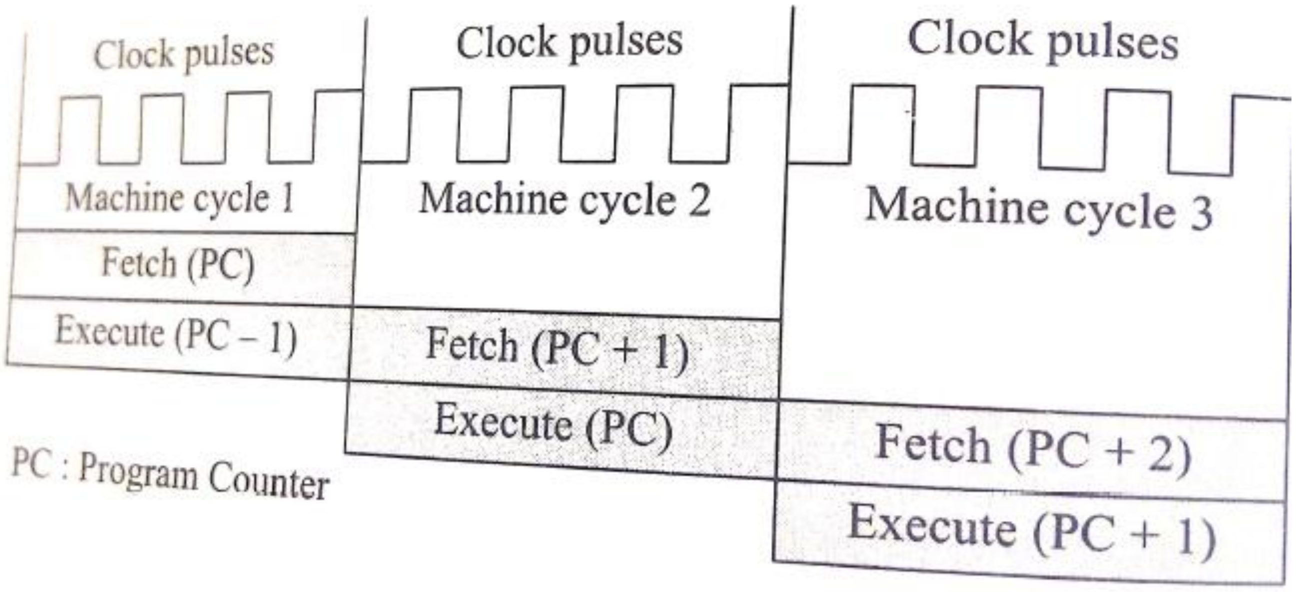


The first instruction load Rl, x loads the register R1 with the content of memory location x, the second instruction load R2, y loads the register R2 with the content of memory location y. The instruction add R3, R1, R2 adds the content of register R1 and R2 and store the result in register R3. The next instruction store R3, z stores the content of register R3 in memory location z.

The conventional instruction execution by the processor follows the fetch-decode-execute sequence; where the 'fetch' part fetches the instruction from program memory or code memory, the ‘decode’ part decodes the instruction to generate the necessary control signals and the ‘execute’ stage reads the operands, perform ALU operations and stores the result.

In conventional program execution, the fetch and decode operations are performed in sequence. Whenever the current instruction is executing the program counter will be loaded with the address of the next instruction. In case of jump or branch instruction, the new location is known only after completion of the jump or branch instruction.

Depending on the stages involved in an instruction (fetch, read register and decode, execute instruction, access an operand in data memory, write back the result to register, etc.), there can be multiple levels of instruction pipelining. The following Figure illustrates the concept of Instruction pipelining for single stage pipelining.



What is an Programmable Logic Device (PLD)? What are the different types of PLDs? Explain advantages of PLDs in embedded system design.

https://www.youtube.com/watch?v=8MwIgsCSTdc

https://en.wikipedia.org/wiki/Programmable_logic_device

https://www.electrical4u.com/programmable-logic-devices/

Programmable Logic Devices: 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.

o As the name indicates, the circuits in a fixed logic device are permanent, they perform one function or set of functions-once manufactured, they cannot be changed.

o On the other hand, 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.

Advantages of PLDs:

Programmable logic devices offer a number of advantages over fixed logic devices, including:

• PLDs offer customers much more flexibility during the design cycle because design iterations are simply a matter of changing the programming file, and results of design changes can be seen immediately in working parts.

• PLDs do not require long lead times for prototypes or production parts-the PLDs are already on a distributor's shelf and ready for shipment.

• PLDs do not require customers to pay for large NRE costs and purchase expensive mask sets-PLD suppliers incur those costs when they design their programmable devices.

• PLDs allow customers to order just the number of parts they need, when they need them, allowing them to control inventory.

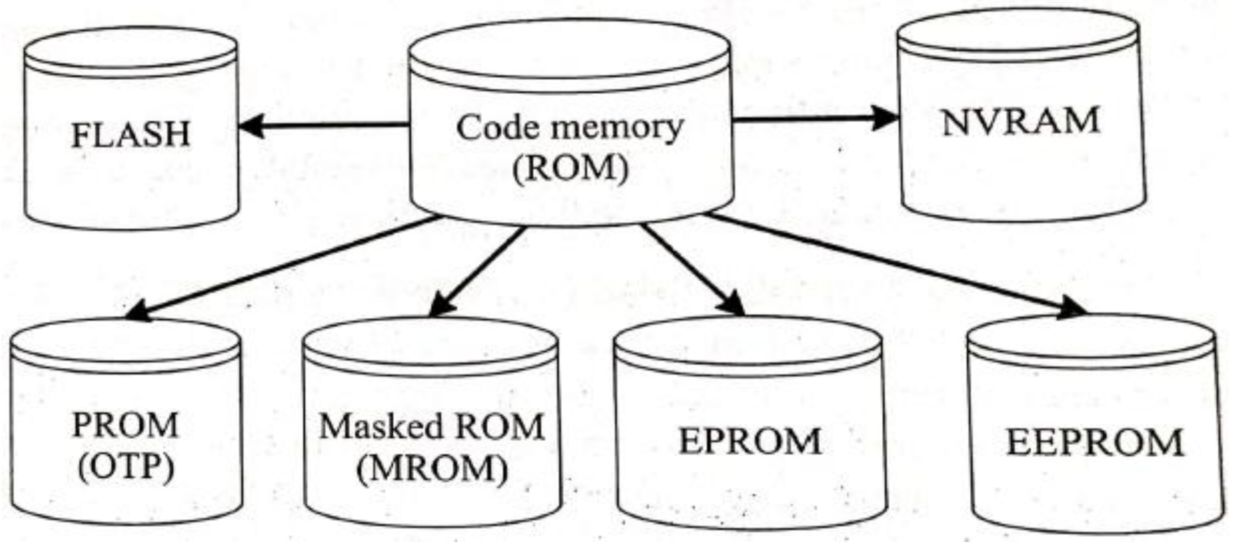
• PLDs can be reprogrammed even after a piece of equipment is shipped to a customer.

MEMORY:

What are different types of memories used in embedded system design?

Memory is an important part of a processor/ controller based embedded systems. Some of the processors/ controllers contain built in memory and this memory is referred as on-chip memory. Others do not contain any memory inside the chip and requires external memory to be connected with the controller/processor to store the control algorithm. It is called off-chip memory. Also some working memory is required for holding data temporarily during certain operations.

Program Storage Memory (ROM): The program memory or code storage memory of an embedded system stores the program instructions and it can be classified into different types as per the block diagram representation given in the following Figure.



The code memory retains its contents even after the power to it is turned off. It is generally known as non-volatile storage memory. Depending on the fabrication, erasing and programming techniques, they are classified into the following types:

1. Masked Memory (MROM): Masked ROM is a one-time programmable device. Masked ROM makes use of the hardwired technology storing data. The device is factory programmed by masking and metallization process at the time of production itself, as per the data provided by the end user.

o The primary advantage of this 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

(1) Creation of an enhancement or depletion mode transistor through channel implant.

(2) By creating the memory cell either using a standard transistor or a high threshold transistor.

o Masked ROM is a good candidate for storing the embedded firmware for low cost embedded devices. Once the design is proven and the firmware requirements are tested and frozen, the binary data (The firmware cross compiled/assembled to target processor specific machine code) corresponding to it can be given to the MROM fabricator.

o The limitation with MROM based firmware storage is the inability to modify the device firmware against firmware upgrades. Since the MROM is permanent in bit storage, it is not possible to alter the bit information.

2. Programmable Read Only Memory (PROM)/ One Time Programmable Memory (OTP): PROM is not pre-programmed by the manufacturer. The end user is responsible for programming these devices.

o This memory 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 logic "1"; whereas fuses which are blown/ burned represents a logic "0". The default state is logic "1".

o OTP is widely used for commercial production of embedded systems whose proto-typed versions are proven and the code is finalized. It is a low cost solution for commercial production. OTPs cannot be reprogrammed.

o Limitations: OTPs are not useful and worth for development purpose. During the development phase, the code is subject to continuous changes and using an OTP each time to load the code is not economical.

3. Erasable Programmable Read Only Memory (EPROM): EPROM gives the flexibility to reprogram the same chip.

o 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.

o EPROM contains a quartz crystal window for erasing the stored information. If the window is exposed to ultraviolet rays for a fixed duration, the entire memory will be erased.

o Limitations: Even though the EPROM chip is flexible in terms of re-programmability, it needs to be taken out of the circuit board and put in a UV eraser device for 20 to 30 minutes. So it is a tedious and time-consuming process.

4. Electrically Erasable Programmable Read Only Memory (EEPROM): Electrically Erasable Programmable Read Only Memory indicates; 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 in-circuit. These chips include a chip erase mode and in this mode they can be erased in a few milliseconds.

o It provide s greater flexibility for system design.

o The only limitation is their capacity is limited (only few kilobytes) when compared with the standard ROM.

5. FLASH: FLASH is the latest ROM technology and is the most popular ROM technology used in today's embedded designs. FLASH memory is a variation of EEPROM technology. It combines the re-programmability of EEPROM and the high capability of standard ROMs.

o 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 one at sector level or page level without affecting the other sectors or pages. Each sector/ page should be erased before re-programming. The typical erasable capacity of FLASH is 1000 cycles.

6. NVRAM: Non-volatile RAM is a 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.

o The life span of NVRAM is expected to be around 10 years. DSJ644 from Maxim/ Dallas is an example of 32KB NVRAM.

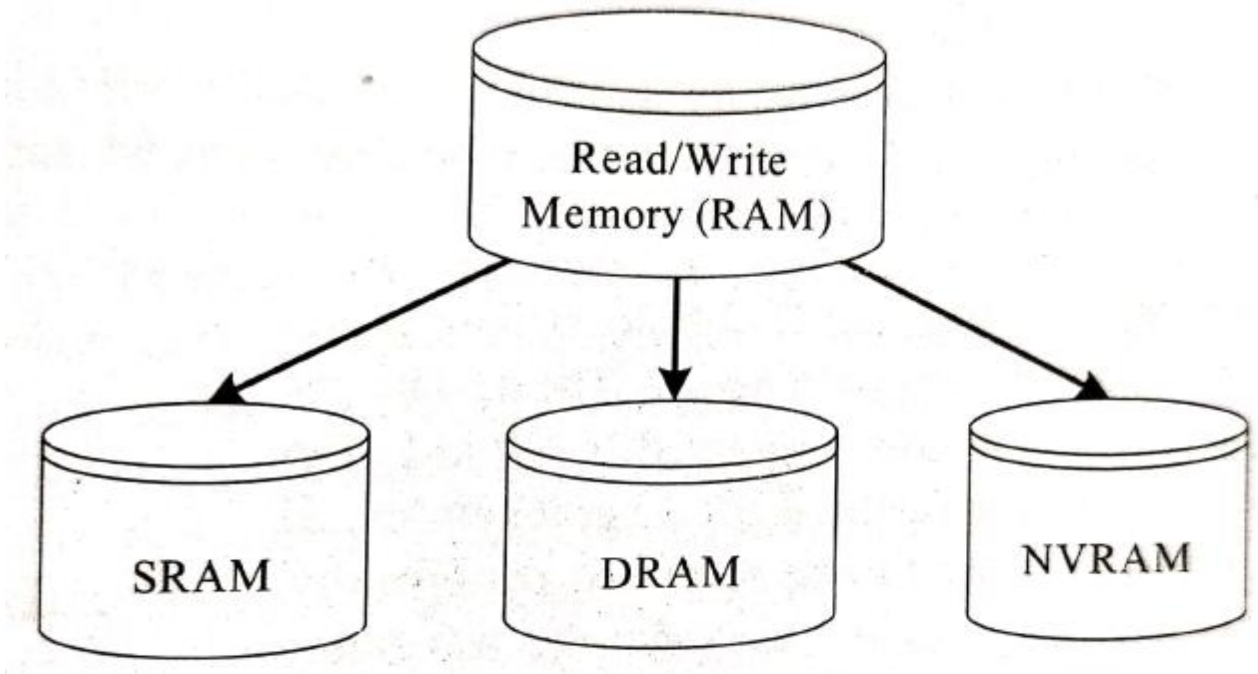
Read-Write Memory/ Random Access memory (RAM): RAM is the data memory or working memory of the controller/ processor. Controller/ processor can read from it and write to it.

• 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).

o This is in contrast to the Sequential Access Memory (SAM), where the desired memory location is accessed by either traversing through the entire memory or through a 'seek' method. Magnetic tapes, CD ROMs, etc. are examples of sequential access memories.

• RAM generally falls into three categories: Static RAM (SRAM), dynamic RAM (DRAM) and non-volatile RAM (NVRAM).

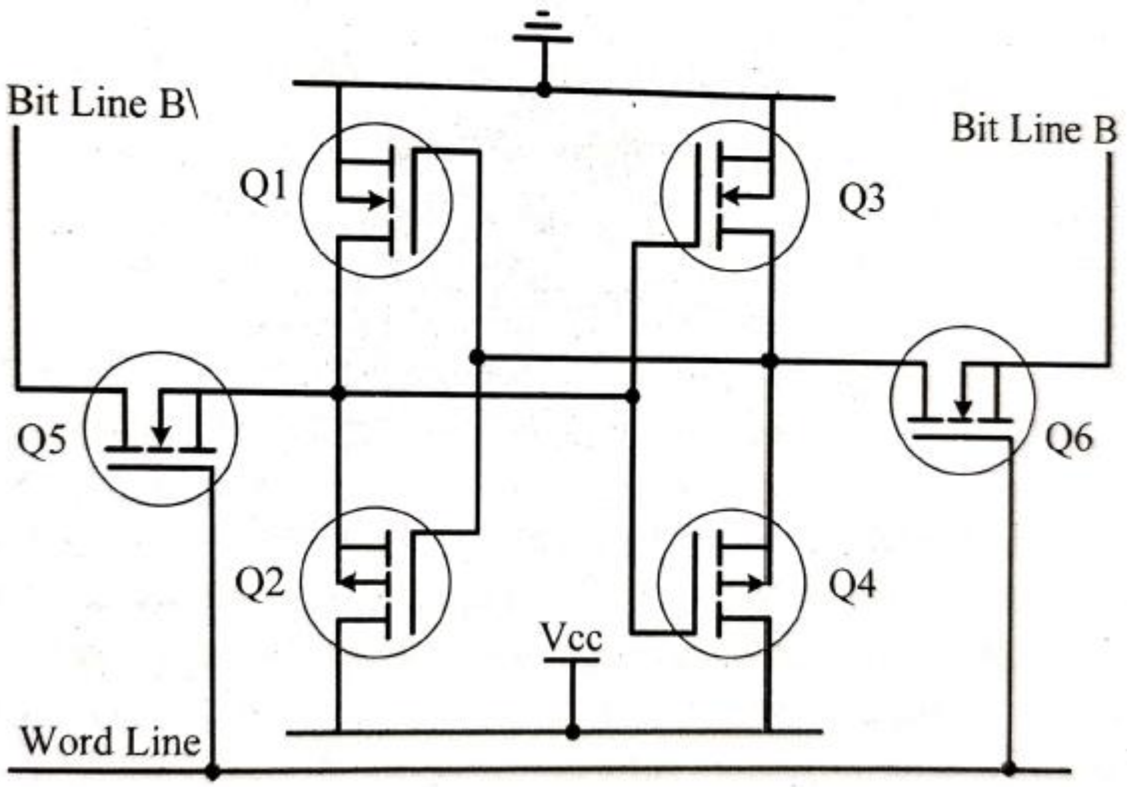


1. Static RAM (SRAM): Static RAM stores data in the form of voltage. They are made up of flip-flops. Static RAM is the fastest form of RAM available.

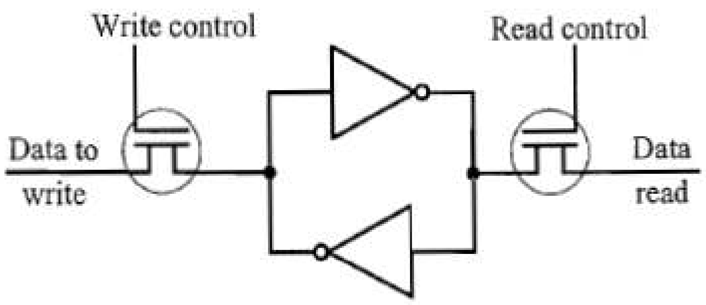
o In typical implementation, an SRAM cell (bit) is realized using six transistors (or 6 MOSFETs). Four of the transistors are used for building the latch (flip-flop) part of the memory cell and two for controlling the access.

o SRAM is fast in operation due to its resistive networking and switching capabilities.

o In simplest representation an SRAM cell can be visualized as shown in the following Figure:



o This implementation in its simpler form can be visualized as two-cross coupled inverters with read/ write control through transistors. The four transistors in the middle form the cross-coupled inverters. This can be visualized as shown in the following Figure:



o From the SRAM implementation diagram, it is clear that access to the memory cell is controlled by the line Word Line, which controls the access transistors (MOSFETs) Q5 and Q6. The access transistors control the connection to bit lines B & B\.

o In order to write a value to the memory cell, apply the desired value to the bit control lines (For writing 1, make B = 1 and B\ =0; For writing 0, make B = 0 and B\ =1) and assert the Word Line (Make Word line high). This operation latches the bit written in the flip-flop.

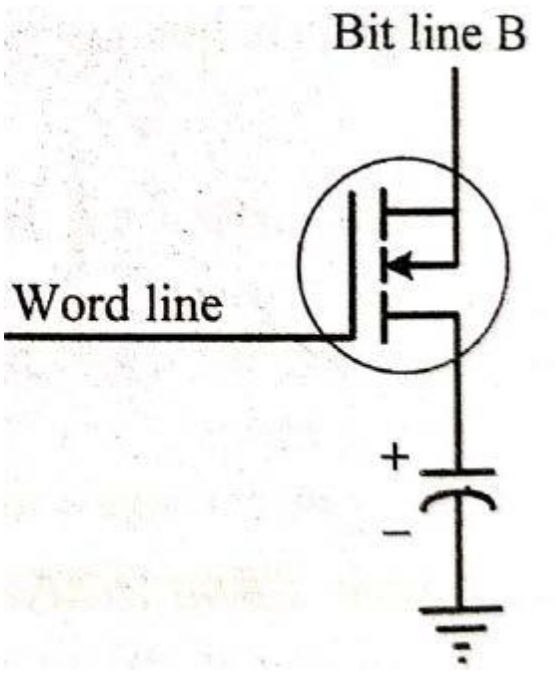
o For reading the content of the memory cell, assert both B and B\ bit lines to 1 and set the Word line to 1.

o The major limitations of SRAM are low capacity and high cost. Since a minimum of six transistors are required to build a single memory cell, imagine how many memory cells we can fabricate on a silicon wafer.

2. Dynamic RAM (DRAM): Dynamic RAM stores data in the form of charge. They are made up of MOS transistor gates.

o The advantages of DRAM are its high density and low cost compared to SRAM.

o 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 interval. The following Figure illustrates the typical implementation of a DRAM cell.



o The MOSFET acts as the gate for the incoming and outgoing data, whereas the capacitor acts as the bit storage unit.

3. NVRAM: Non-volatile RAM is a 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.

o The life span of NVRAM is expected to be around 10 years. DSJ644 from Maxim/ Dallas is an example of 32KB NVRAM.

What is memory shadowing? What is its advantage?

Explain the concept of memory shadowing.

https://people.cs.pitt.edu/~childers/papers/DATE11.pdf

https://www.youtube.com/watch?v=jLNqkbvyYEk

Generally the execution of a program or a configuration from a Read Only Memory (ROM) is very slow (120 to 200 ns) compared to the execution from a random access memory (40 to 70 ns). From the timing parameters, it is obvious that RAM access is about three times as fast as ROM access.

Shadowing of memory is a technique adopted to solve the execution speed problem in processor-based systems. In computer systems and video systems, there will be a configuration holding ROM called Basic Input Output Configuration ROM or simply BIOS.

• In personal computer system, BIOS stores the hardware configuration information like the address assigned for various serial ports and other non-plug 'n' play devices, etc. Usually it is read and the system is configured accordingly to it during system boot up and it is time consuming.

• Now, the manufactures included a RAM behind the logical layer of BIOS at its same address as a shadow to the BIOS; and the following steps happens:

  o During the boot up, BIOS is copied to the shadowed RAM

  o RAM is write protected

  o BIOS reading is disabled.

• Why both RAM and ROM are needed for holding the same data?

  o The answer is: RAM is volatile and it cannot hold the configuration data which is copied from the BIOS when the power supply is switched off. Only a ROM can hold it permanently. But for high system performance, it should be accessed from a RAM instead of accessing from a ROM.

SENSORS AND ACTUATORS, LED, 7 SEGMENT LED DISPLAY, STEPPER MOTOR, KEYBOARD, PUSH BUTTON SWITCH:

What is sensor? Illustrate the role of sensor in embedded system design with example

What is actuator? Illustrate the role of sensor in embedded system design with example

Differentiate sensors versus actuators.

An embedded system is in constant interaction with the Real world, and the controlling/ monitoring functions, executed by the embedded system is achieved in accordance with the changes happening to the Real world. The changes in system environment or variables are detected by the sensors connected to the input port of the embedded system.

• A sensor is a transducer device that converts energy from one form to another, for any measurement or control purpose.

o Sensor which counts steps for pedometer functionality is an Accelerometer sensor.

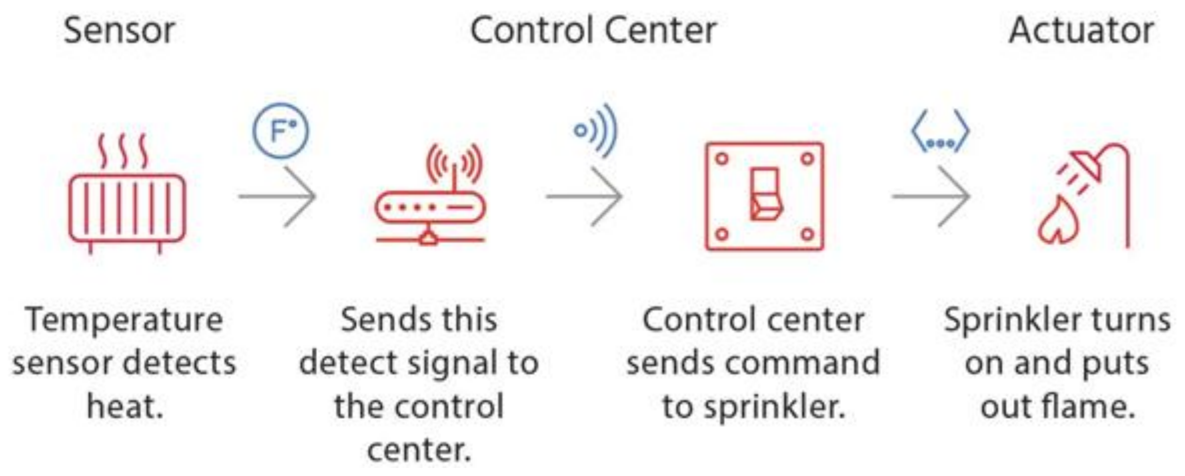
o Sensor used in smart watch devices to measure the high intensity is an Ambient Light Sensor (ALS).

If the embedded system is designed for any controlling purpose, the system will produce some changes in the controlling variable to bring the controlled variable to the desired value. It is achieved through an actuator connected to the output port of the embedded system.

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

o Smart watches use Ambient Light Sensor to detect the surrounding light intensity and uses an electrical/ electronic actuator circuit to adjust the screen brightness.

The following Figure shows the sensor to actuator flow:


Sensor
Control Center
Actuator
F°
Temperature sensor detects heat.
Sends this detect signal to the control center.
Control center sends command to sprinkler.
Sprinkler turns on and puts out flame.

If the embedded system is designed for monitoring purpose only, then there is no need for including an actuator in the system. For example, take the case of an ECG machine. It is designed to monitor the heart beat status of a patient and it cannot impose a control over the patient's heart beat and its order. The sensors used here are the different electrode sets connected to the body of the patient. The variations are captured and presented to the user (may be a doctor) through a visual display or some printed chart.

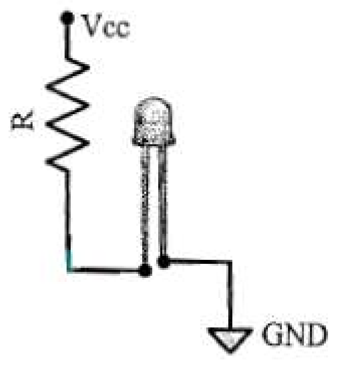
Sensors	Actuators
1. Sensor is an input device	1. Actuator is an output device
2. Convert a physical parameter to an electrical output	2. Convert an electrical signal to a physical output
3. A device that detects events or changes in the environment and send the information to another electronic device	3. A component of a machine that is responsible for moving and controlling mechanisms
4. Sensor help to monitor the changes in the environment	4. Actuator helps to control the environment or physical changes

What is 7-Segment LED display? What are two different configurations of 7-segment LED display? Explain

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

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

Light Emitting Diode is a p-n junction diode and it contains an anode and a cathode. For proper functioning of the LED, the anode of it should be connected to +ve terminal of the supply voltage and cathode to the –ve terminal of supply voltage. A resister is used in series between the power supply and the LED to limit the current through the LED. The ideal LED interfacing circuit is shown in the following Figure.

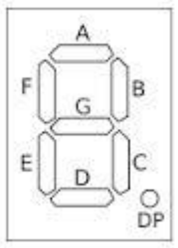


LEDs can be interfaced to the port pin of a processor/ controller in two ways.

• In the first method, the anode is directly connected to the port pin and the port pin drives the LED. In this approach, the port pin 'sources' current to the LED when the port pin is at logic High (Logic '1').

• In the second method, the cathode of the LED is connected to the port pin of processor/ controller and the anode to the supply voltage through a current limiting resistor. LED is turned on when the port pin is at logic Low (Logic '0').

7-Segment LED Display: The 7-segment LED display is an output device used for displaying alpha-numeric characters. It contains 8 light-emitting diode (LED) segments arranged in a special form. Out of the 8 LED segments, 7 are used for displaying alpha-numeric characters and 1 is used for representing ‘decimal point'. The following Figure explains the arrangement of LED segments in 7-segment LED display.

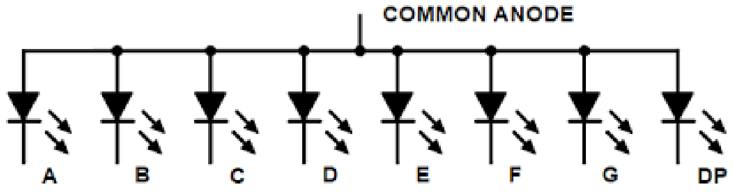


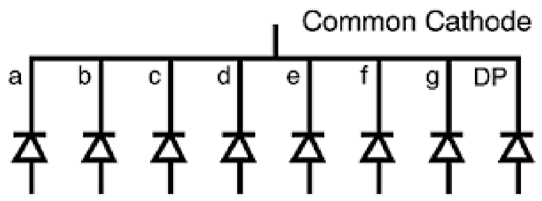
The LED segments are named A to G and the 'decimal point LED segment is named as DP. For displaying the number 4, the segments F, G, B and C are lit. For displaying 3, the segments A, B, C, D, G are lit. All these 8 LED segments need to be connected to one port of the processor/ controller for displaying alpha-numeric digits. The 7-segment LED displays are available in two different configurations, namely; Common Anode and Common Cathode.

• In common anode configuration, the anodes of the 8 segments are connected commonly

• In common cathode configuration, the 8 LED segments share a common cathode line.

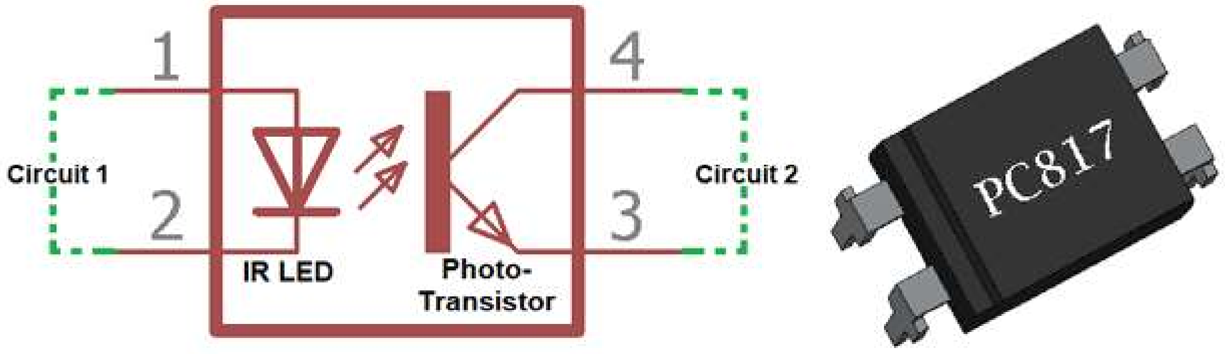
The following Figure illustrates the Common Anode and Cathode configurations.





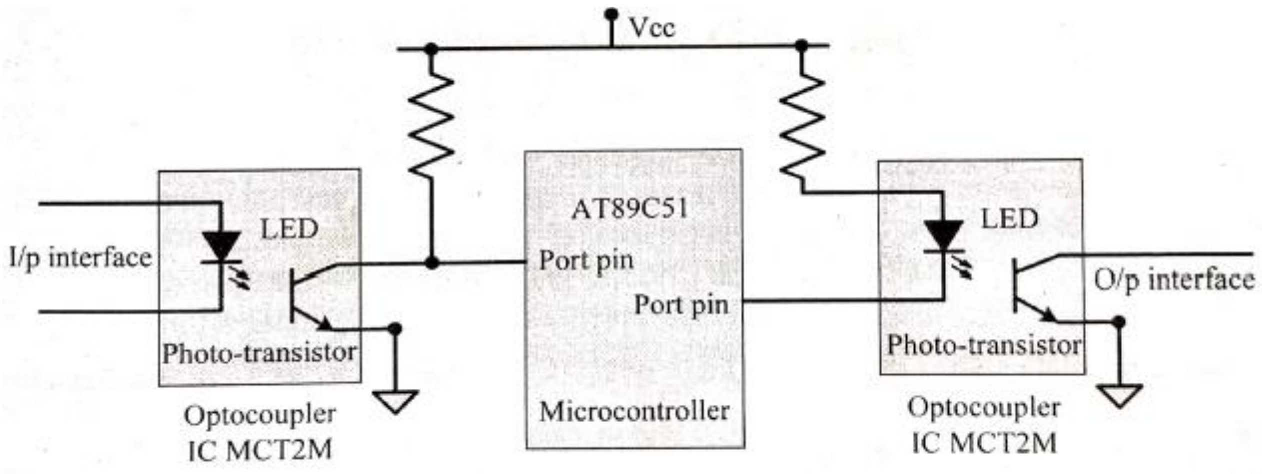
What is Optocopuler? Explain the role of Optocopuler in embedded applications.

Optocopuler: Optocopuler is a solid state device to isolate two parts of a circuit. Optocopuler combines an LED and a photo-transistor in a single housing (package). The following Figure illustrates the functioning of an Optocopuler device.



In electronic circuits, an optocoupler is used for suppressing interference in data communication, circuit isolation, high voltage separation, simultaneous separation and signal intensification, etc. Optocouplers can be used in either input circuits or in output circuits.

The following Figure illustrates the usage of optocoupler in input circuit and output circuit of an embedded system with a microcontroller as the system core.

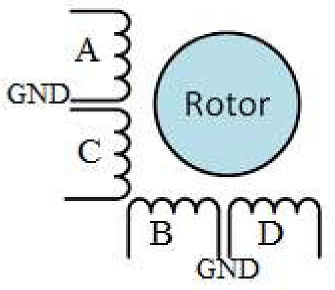


Optocoupler is available as ICs from different semiconductor manufacturers. The MCT2M IC from Fairchild semiconductor is an example for optocoupler IC.

What is Stepper Motor? Explain different step modes. Also, explain the role of stepper motor in embedded applications.

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

1. Unipolar: 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. The following Figure illustrates the working of a two-phase unipolar stepper motor.



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

2. Bipolar: A bipolar stepper motor contains single winding per phase. For reversing the motor rotation the current flow through the windings is reversed dynamically. It requires complex circuitry for current flow reversal.

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

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

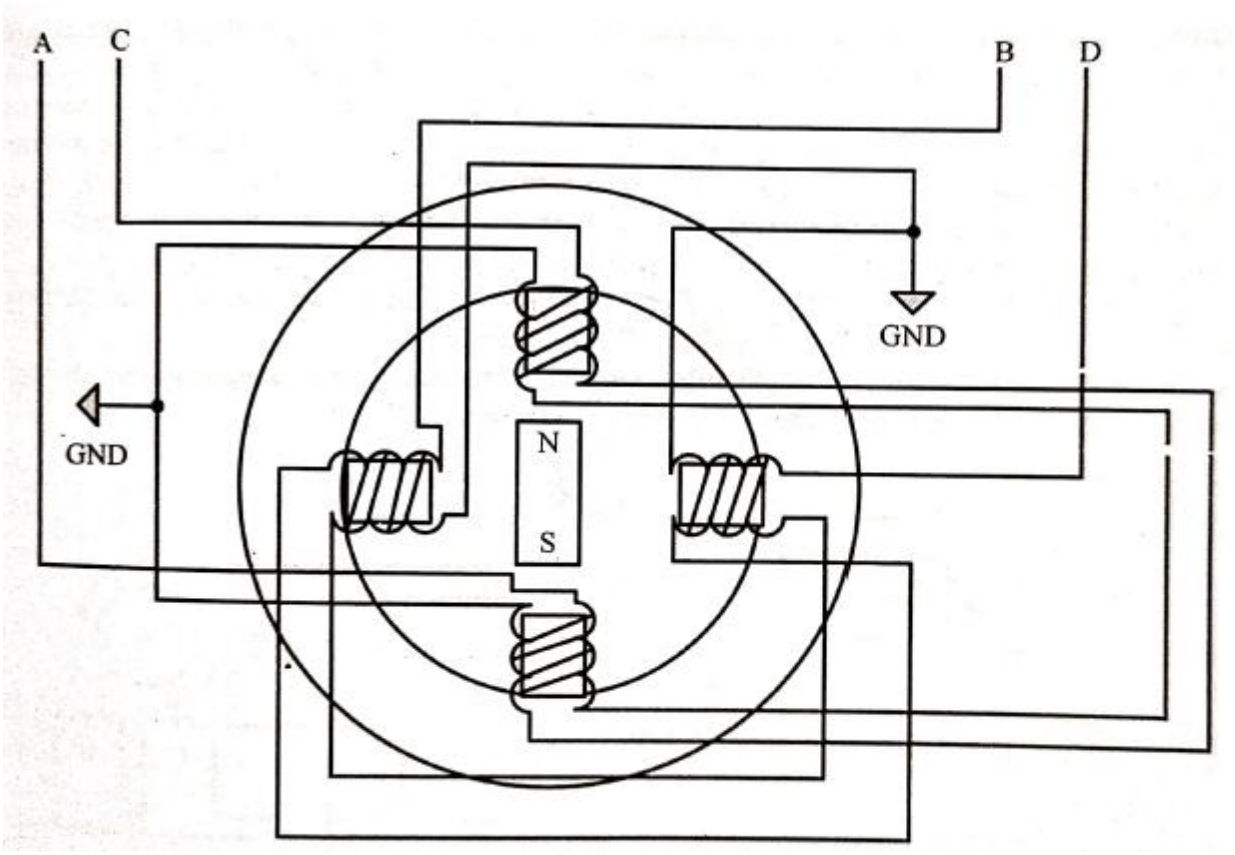
Wave Step: In the wave step mode, only one phase is energized at a time and each coils of the phase is energized alternatively. The A, B, C and D are energized in the order, as shown in the following Table.

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

Half Step: It uses the combination of wave and full step. It has the highest torque and stability. The coil energizing sequence for half step is given in the Table

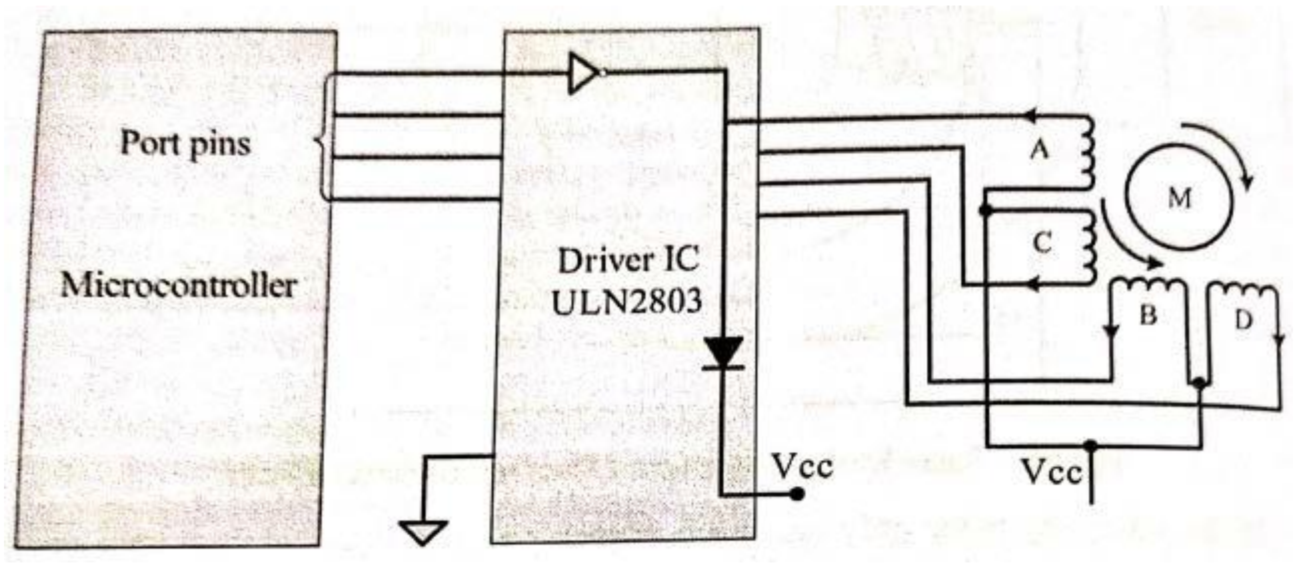
Step	Coil A	Coil B	Coil C	Coil D
1	H	L	L	L
2	H	H	L	L
3	L	H	L	L
4	L	H	H	L
5	L	L	H	L
6	L	L	H	H
7	L	L	L	H
8	H	L	L	H

The rotation of the stepper motor can be reversed by reversing the order in which the coil is energized. The following Figure shows the stator winding details of Stepper motor:



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

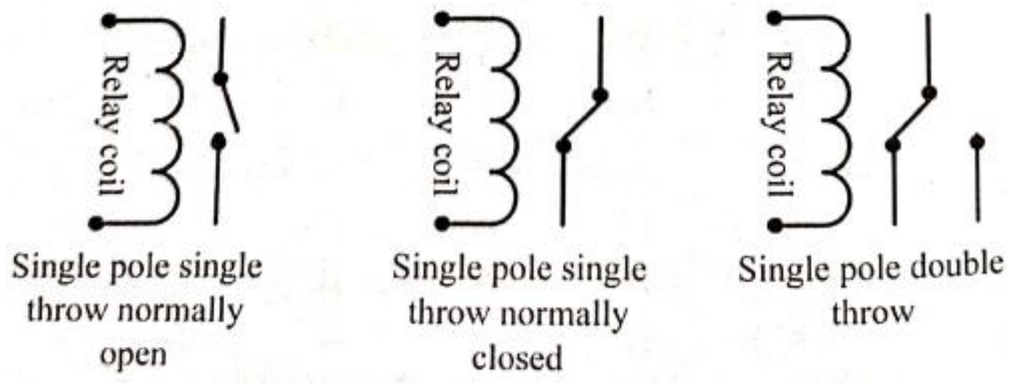
ULN2803 is an octal peripheral driver array available from Texas Instruments and ST microelectronics for driving a 5V stepper motor. Simple driving circuit can also be built using transistors. The following circuit diagram illustrates the interfacing of a stepper motor through a driver circuit connected to the port pins of a microcontroller/ processor.



What is Relay? What are different types of relays available? Explain the role of relay in embedded applications.

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

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

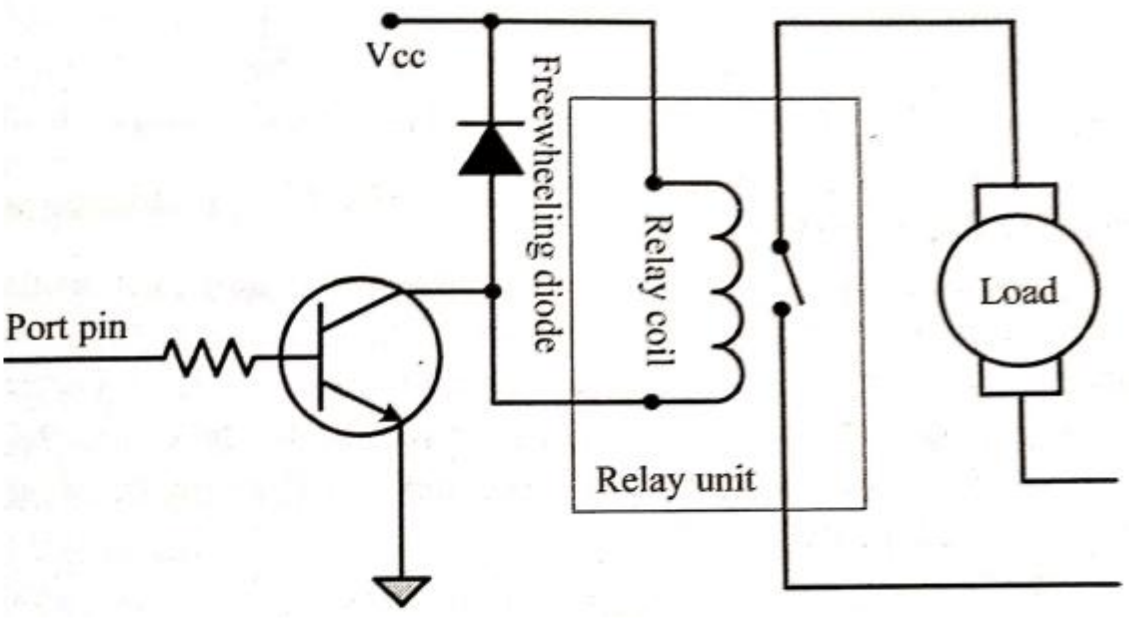

Relay coil
Single pole single throw normally open
Relay coil
Single pole single throw normally closed
Relay coil
Single pole double throw

The Single Pole Single Throw configuration has only one path for information flow. The path is either open or dosed in normal condition.

• For normally open Single Pole Single Throw relay, the circuit is normally open and it becomes closed when the relay is energized.

• For normally closed Single Pole Single Throw configuration, the circuit is normally closed and it becomes open when the relay is energized.

For Single Pole Double Throw Relay, there are two paths for information flow and they are selected by energizing or de-energizing the relay. The Relay is normally controlled using a relay driver circuit connected to the port pin of the processor/ controller. A transistor is used for building the relay driver circuit. The following Figure illustrates the same.



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

Write a short note on: Piezo electric buzzers & Push button switches.

Piezo Buzzer: Piezo buzzer is a piezoelectric device for generating audio indications in embedded application.

o A piezoelectric buzzer contains a piezoelectric diaphragm which produces audible sound in response to the voltage applied to it.

Piezoelectric buzzers are available in two types. 'Selfdriving' and 'External driving'.

• The 'Self-driving' circuit contains all the necessary components to generate sound at a predefined tone. It will generate a tone on applying the voltage.

• External driving piezo buzzers supports the generation of different tones. The tone can be varied by applying a variable pulse train to the piezoelectric buzzer.

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

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'.

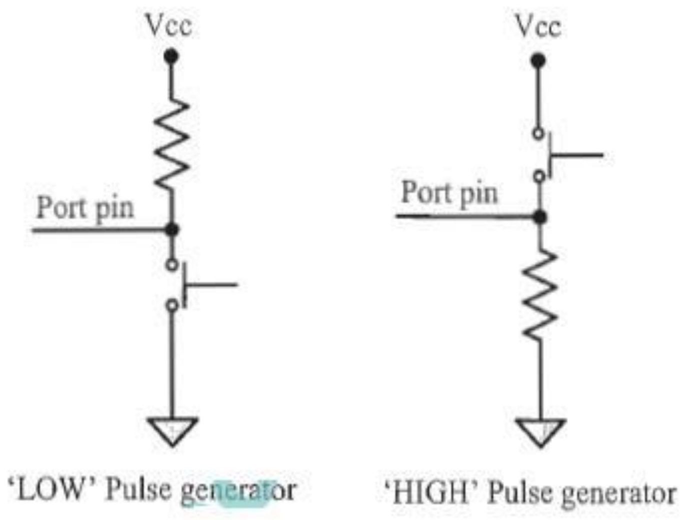
• In the 'Push to Make' configuration, the switch is normally in the open state and it makes a circuit contact when it is pushed or pressed.

• In the 'Push to Break' configuration, the switch is normally in the closed state and it breaks the circuit contact when it is pushed or pressed.

o The push button stays in the 'closed' (for Push to Make type) or 'open' (For Push to Break type) state as long as it is kept in the pushed state and it breaks/ makes the circuit connection when it is released.

• Push button is used for generating a momentary pulse. In embedded application push button is generally used as reset and start switch and pulse generator. The Push button is normally connected to the port pin of the host processor/ controller.

Depending on the way in which the push button interfaced to the controller, it can generate either a 'HIGH' pulse or a 'LOW' pulse. The following Figure Illustrates how the push button can be used for generating 'LOW' and 'HIGH' pulses.



Explain the operation of Matrix Keyboard

Keyboard: Keyboard is an input device 'HIGH' Pulse generator for user interfacing.

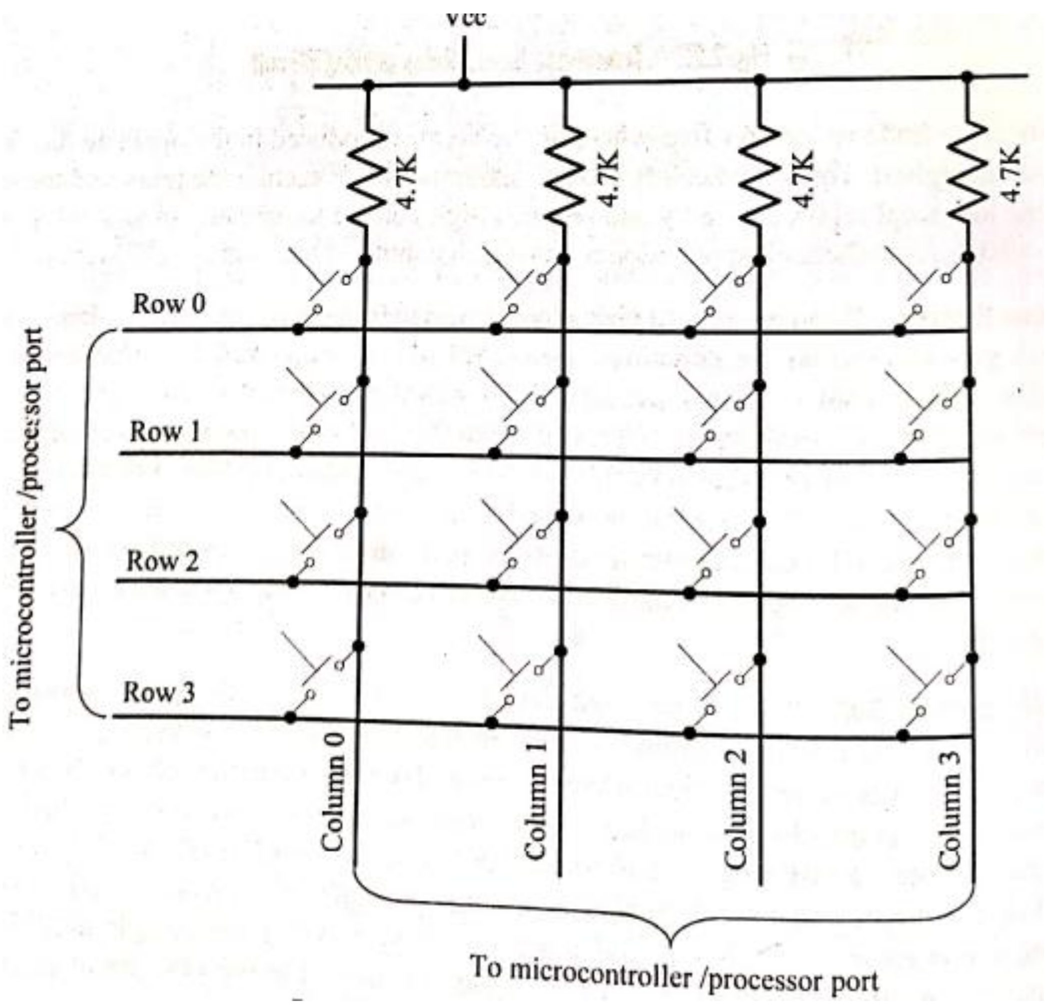
• If the number of keys required is very limited, push button switches can be used and they can be directly interfaced to the port pins for reading.

• However, there may be situations demanding a large number of keys for user input (e.g. PDA device with alpha-numeric keypad for user data entry).

o In such situations it may not be possible to interface each keys to a port pin due to the limitation in the number of general purpose port pins available for the processor/ controller in use and moreover it is wastage of port pins.

o Matrix keyboard is an optimum solution for handling large key requirement. It greatly reduces the number of interface connections.

• For example, for interfacing 16 keys, in the direct interfacing technique, 16 port pins are required, whereas in the matrix keyboard only 8 lines are required. The 16 keys are arranged in a 4 column x 4 Row matrix. The following Figure illustrates the connection o keys in a matrix keyboard.



In a matrix keyboard, the keys are arranged in matrix fashion. For detecting a key press, the keyboard uses the scanning technique, where each row of the matrix is pulled low and the columns are read. After reading the status of each columns corresponding to a row, the row is pulled high and the next row is pulled low and the status of the columns are read.

This process is repeated until the scanning for all rows are completed. When a row is pulled low and if a key connected to the row is pressed, reading the column to which the key is connected will give logic 0. Since keys are mechanical devices, proper key de-bouncing technique should be applied.

What is Programmable Peripheral Interface (PPI)? Explain the control word format and different modes of 8255 PPI.

What is Programmable Peripheral Interface (PPI)? Explain the interfacing of 8255 PPI with an 8-bit processor/ controller.

https://www.tutorialspoint.com/microprocessor/microprocessor_intel_8255a_programmable_peripheral_interface.htm

https://www.youtube.com/watch?v=pphUIgjvqJ8

Programmable Peripheral Interface (PPI): Programmable Peripheral Interface devices are used for extending the I/O capabilities of processors/ controllers.

Most of the processors/ controllers provide very limited number of I/O and data ports and at times it may require more number of I/O ports than the one supported by the controller/ processor.

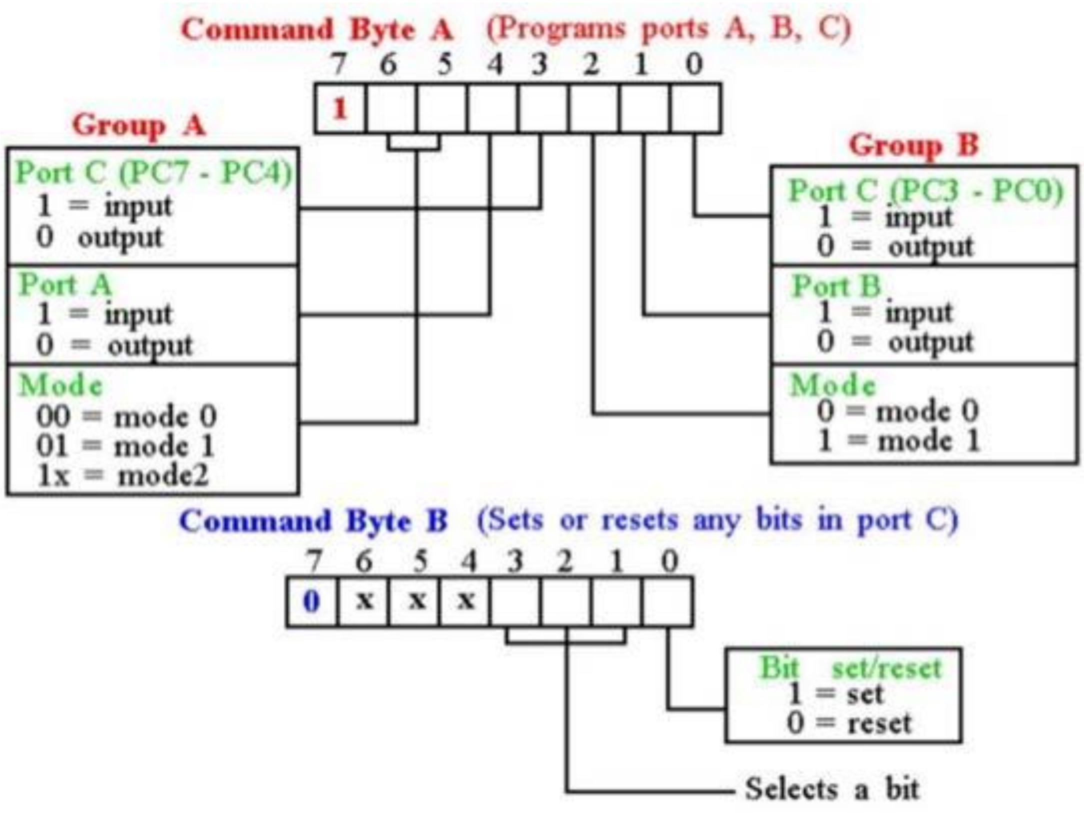
A programmable peripheral interface device expands the I/O capabilities of the processor/ controller. 8255A is a popular PPI device for 8-bit processors/ controllers.

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:

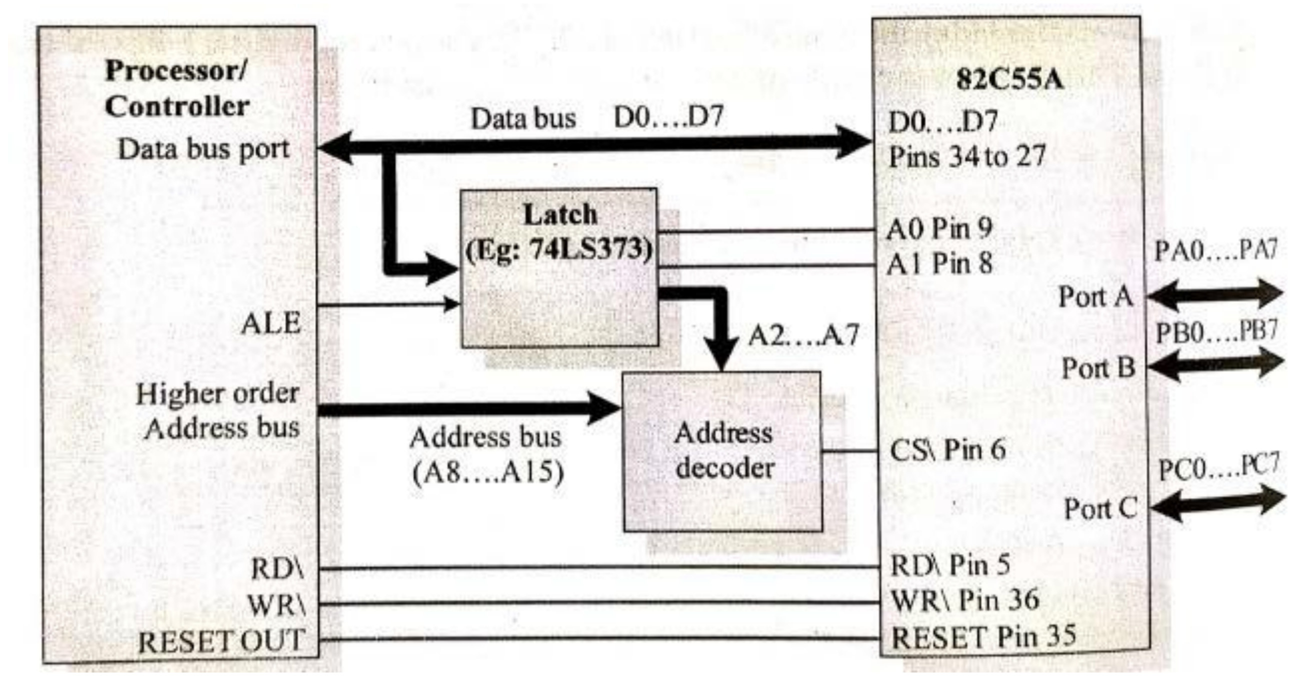
(1) As 8 individual I/O pins

(2) Two 4-bit ports; namely Port CUPPER (Cu) and Port CLOWER (CL).

This is configured by manipulating the control register of 8255A. The control register holds the configuration for Port A, Port B and Port C. The bit details of control register is given in the following Figure:



The following Figure illustrates the generic interfacing of a 8255A device with an 8-bit processor/ controller with 16-bit address bus (Lower order Address bus is multiplexed with data bus).



COMMUNICATION INTERFACE (ONBOARD AND EXTERNAL TYPES):

Explain the different on-board communication interfaces in brief

Explain the sequence of operations for communicating with an I2C slave device.

Give the differences between I2C and SPI communication interface.

https://www.youtube.com/watch?v=qeJN_80CiMU

https://www.youtube.com/watch?v=ba0SQwjTQfw

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 –

• Device/ board level communication interface (Onboard Communication Interface)

o Embedded product is a combination of different types of components (chips/ devices) arranged on a printed circuit board (PCB). The communication channel which interconnects the various components within an embedded product is referred as Device/ Board Level Communication Interface (Onboard Communication Interface).

o Serial interfaces like I2C, SPI, DART, 1-Wire, etc., and parallel bus interface are examples of 'Onboard Communication Interface'.

• Product level communication interface (External Communication Interface)

o Some embedded systems are self-contained units and they don't require any interaction and data transfer with other sub-systems or external world. On the other hand, certain embedded systems may be a part of a large distributed system and they require interaction and data transfer between various devices and sub-modules. The 'Product level communication interface' (External Communication Interface) is responsible for data transfer between the embedded system and other devices or modules.

o The external communication interface can be either a wired medium or a wireless media and it can be a serial or a. parallel interface.

o Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves (RF), GPRS/ 3G/ 4GLTE, etc. are examples for wireless communication interface.

o RS-232C/ RS-422/ RS-485, USB, Ethernet IEEE 1394 port, Parallel port, CF-II interface, SDIO, PCMCIA/ PCIex, etc., are examples for wired interfaces.

Onboard Communication Interfaces: Onboard Communication Interface refers to the different communication channels/ buses for interconnecting the various integrated circuits and other peripherals within the embedded system.

Inter Integrated Circuit (I2C) Bus: The Inter Integrated Circuit Bus (I2C-Pronounced 'I square C') is a synchronous bidirectional half duplex (one-directional communication at a given point of time) two wire serial interface bus.

• 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, namely; Serial Clock-SCL and Serial Data-SDA.

o SCL line is responsible for generating synchronization clock pulses.

o SDA is 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.

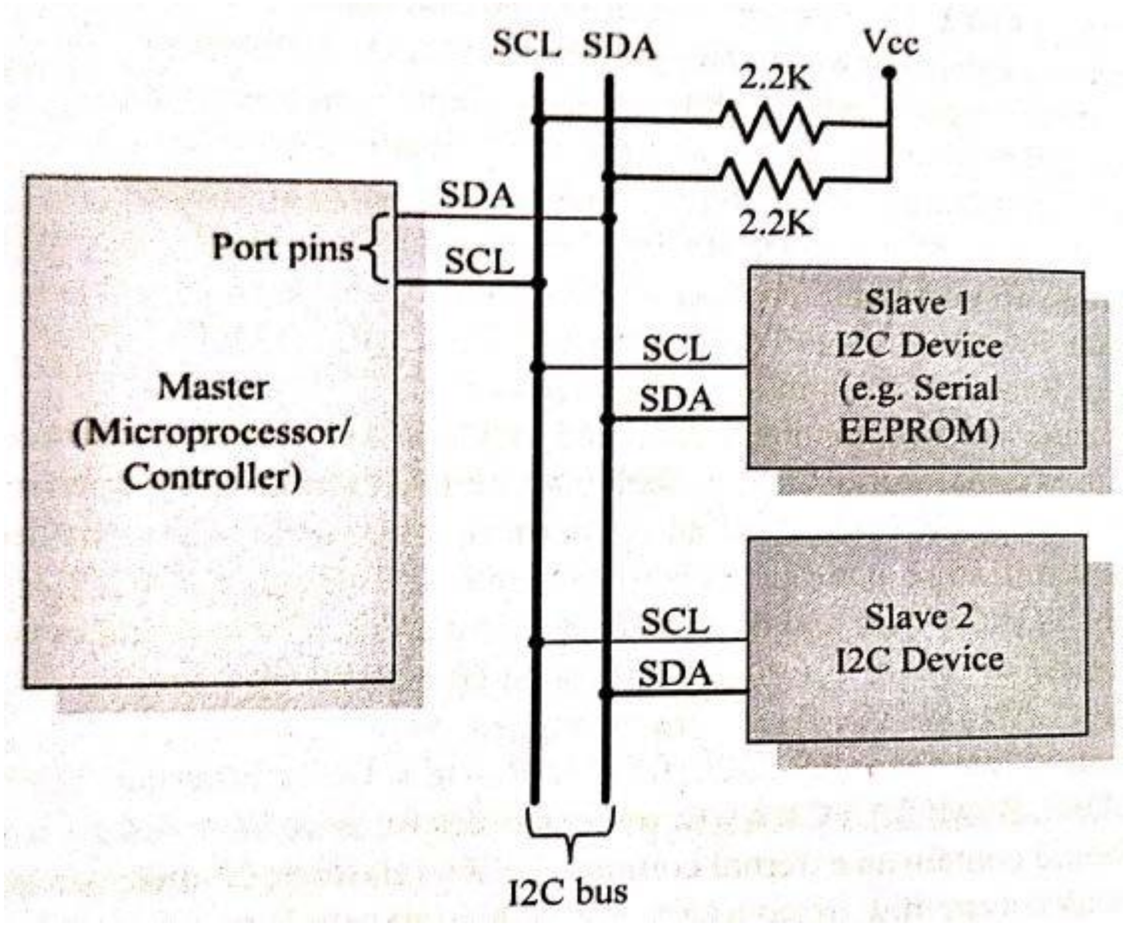
o The 'Master' device is responsible for controlling the communication by initiating/ terminating data transfer, sending data and generating necessary synchronization clock pulses.

o '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 Figure shows bus interface diagram, which illustrates the connection of master and slave devices on the I2C bus.


SCL
SDA
Vcc
2.2K
2.2K
SDA
SCL
Port pins
Master (Microprocessor/ Controller)
SCL
SDA
Slave 1 I2C Device (e.g. Serial EEPROM)
SCL
SDA
Slave 2 I2C Device
I2C bus

The address to various I2C devices in an embedded device is assigned and hardwired 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): Serial Peripheral Interface Bus (SPI) is asynchronous bi-directional full duplex four-wire serial interface bus. The concept of SPI was 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. They are:

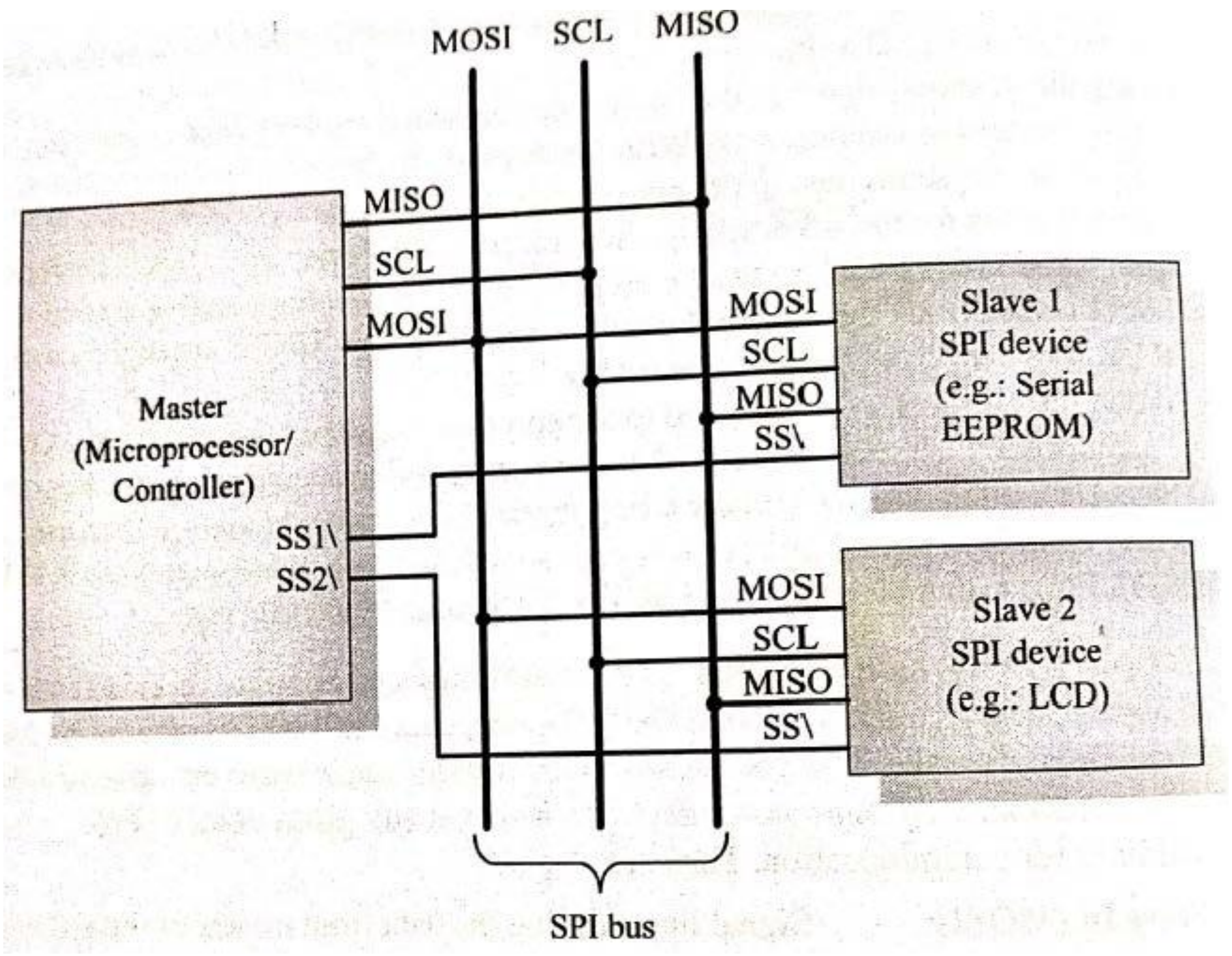
o Master Out Slave In (MOSI): Signal line carrying the data from master to slave device. It is also known as Slave Input/Slave Data In (SI/SDI).

o Master In Slave Out (MISO): Signal line carrying the data from slave to master device. It is also known as Slave Output (SO/ SDO).

o Serial Clock (SCL): Signal line carrying the clock signals

o Slave Select (SS): Signal line for slave device select. It is an active low signal.

The bus interface diagram is shown in the following Figure, illustrates the connection of master and slave devices on the SPI bus.


MOSI
SCL
MISO
MISO
SCL
MOSI
Master (Microprocessor/ Controller)
SS1\
SS2\
MOSI
SCL
MISO
SS\
Slave 1 SPI device (e.g.: Serial EEPROM)
MOSI
SCL
MISO
SS\
Slave 2 SPI device (e.g.: LCD)
SPI bus

The master device is responsible for generating the clock signal. It 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. 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. In summary, the shift registers of 'master' and 'slave' devices form a circular buffer.

When compared to I2C, SPI bus is most suitable for applications requiring transfer of data in 'streams'. The only limitation is SPI doesn’t support an acknowledgement mechanism.

Explain the sequence of operation for communicating with a 1-Wire slave device.

https://en.wikipedia.org/wiki/1-Wire

https://www.youtube.com/watch?v=6IAkYpmA1DQ

Universal Asynchronous Receiver Transmitter (UART): Universal Asynchronous Receiver Transmitter (UART) based data transmission is an asynchronous form of serial data transmission.

• UART based serial data transmission doesn't require a clock signal to synchronize the transmitting end arid receiving end for transmission. Instead it relies upon the pre-defined agreement between the transmitting device and receiving device.

• The serial communication settings (Baud rate, number of bits per byte, parity, number of start bits and stop bit and flow control) for both transmitter and receiver should be set as identical.

• The start and stop of communication is indicated through inserting special bits in the data stream. While sending a byte of data, a start bit is added first and a stop bit is added at the end of the bit stream. The least significant bit of the data byte follows the 'start' bit.

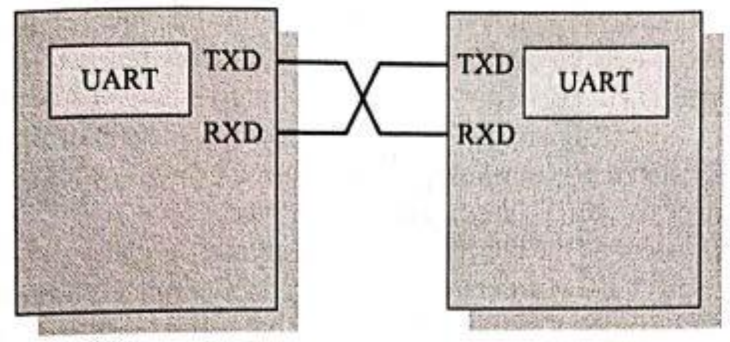
• The 'start' bit informs the receiver that a data byte is about to arrive. The receiver device starts polling its 'receive line' as per the baud rate settings. If the baud rate is 'x' bits per second, the time slot available for one bit is 1/x seconds.

• The receiver unit polls the receiver line at exactly half of the time slot available for the bit.

• If parity is enabled for communication, the UART of the transmitting device adds a parity bit (bit value is 1 for odd number of 1s in the transmitted bit stream and 0 for even number of 1s).

• The DART 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.

For proper communication, the 'Transmit line' of the sending device should be connected to the 'Receive line' of the receiving device. The following Figure illustrates the same.



TXD: Transmitter line
RXD: Receiver line

In addition to the serial data transmission function, UART provides hardware handshaking signal support for controlling the serial data flow.

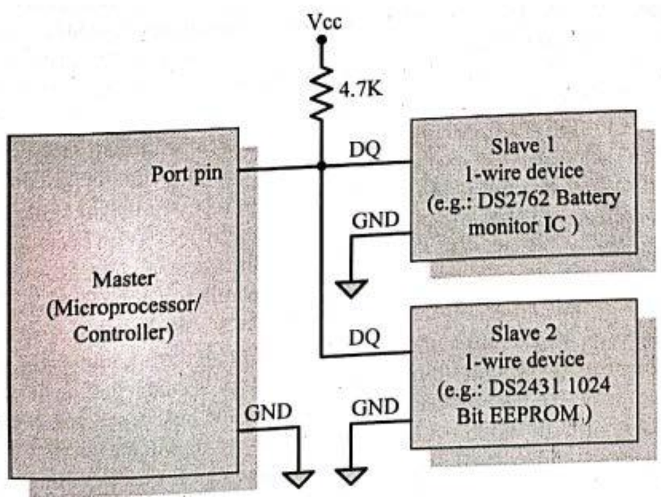
UART chips are available from different semiconductor manufacturers. National Semiconductor's 8250 UART chip is considered as the standard setting UART. It was used in the original IBM PC.

<u>1-Wire Interface:</u> 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 1-wire 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 following Figure illustrates the connection of master and slave devices on the 1-wire bus.


Vcc
4.7K
Port pin
DQ
Slave 1
1-wire device
(e.g.: DS2762 Battery monitor IC)
GND
Master
(Microprocessor/ Controller)
DQ
Slave 2
1-wire device
(e.g.: DS2431 1024 Bit EEPROM)
GND
GND

Every 1-wire device contains a globally unique 64-bit 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 for regular speed mode of operation (16.3Kbps).

Explain the merits and limitations of Parallel port over Serial RS-232 interface.

https://en.wikipedia.org/wiki/Parallel_port

https://www.youtube.com/watch?v=eo9dbnrpspM

https://www.newhavendisplay.com/app_notes/parallel-serial.pdf

Parallel Interface: The on-board parallel interface is normally used for communicating with peripheral devices which are memory mapped to the host of the system.

• The host processor/ controller of the embedded system contains a parallel bus and the device which supports parallel bus can directly connect to this bus system. The communication through the parallel bus is controlled by the control signal interface between the device and the host.

• The 'Control Signals' for communication includes 'Read/ Write' signal and device select signal. The device normally contains a device select line and the device becomes active only when this line is asserted by the host processor.

• The direction of data transfer (Host to Device or Device to Host) can be controlled through the control signal lines for 'Read' and 'Write'. Only the host processor has control over the 'Read' and 'Write' control signals.

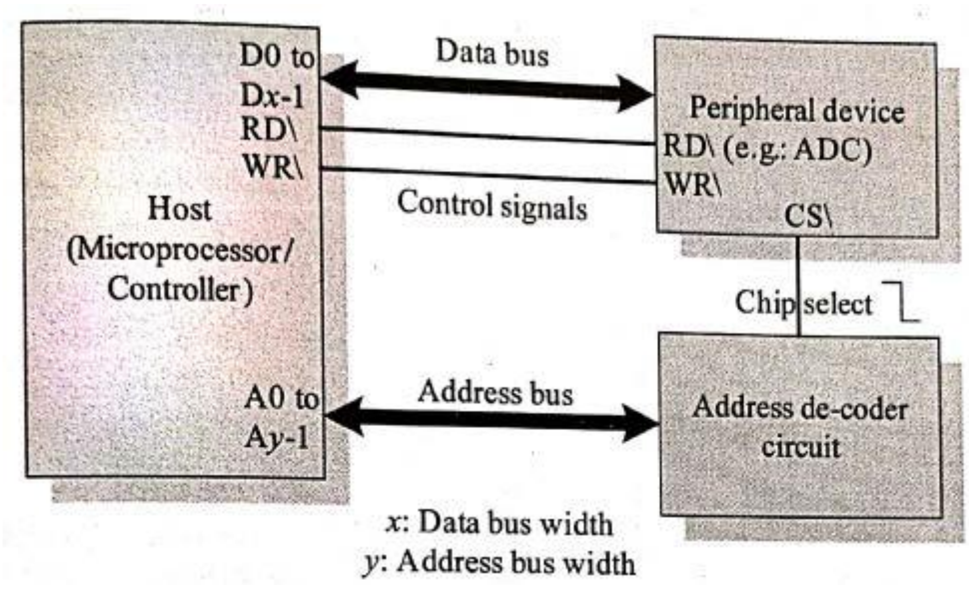
• The device is normally memory mapped to the host processor and a range of address is assigned to it. An address decoder circuit is used for generating the chip select signal for the device. When the address selected by the processor is within the range assigned for the device, the decoder circuit activates the chip select line and thereby the device becomes active. The processor then can read or write from or to the device by asserting the corresponding control line (RD\ and WR\ respectively). Strict timing characteristics are followed for parallel communication.

• As mentioned earlier, parallel communication is host processor initiated. If a device wants to initiate the communication, it can inform the same to the processor through interrupts. For this, the interrupt line of the device is connected to the interrupt line of the processor and the corresponding interrupt is enabled in the host processor.

• The width of the parallel interface is determined by the data bus width of the host processor. It can be 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.

The bus interface diagram shown in the following Figure, illustrates the interfacing of devices through parallel interface.



External Communication Interfaces: The External Communication Interface refers to the different communication channels/ buses used by the embedded system to communicate with the external world.

RS-232 C & RS-485: RS-232 C (Recommended Standard number 232, revision C) from the Electronic Industry Association is a legacy, full duplex, wired, asynchronous serial communication interface.

• The RS-232 interface is 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.

  o 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 -25V. In EIA standard, logic '0' is known as 'Space' and logic '1' as 'Mark'.

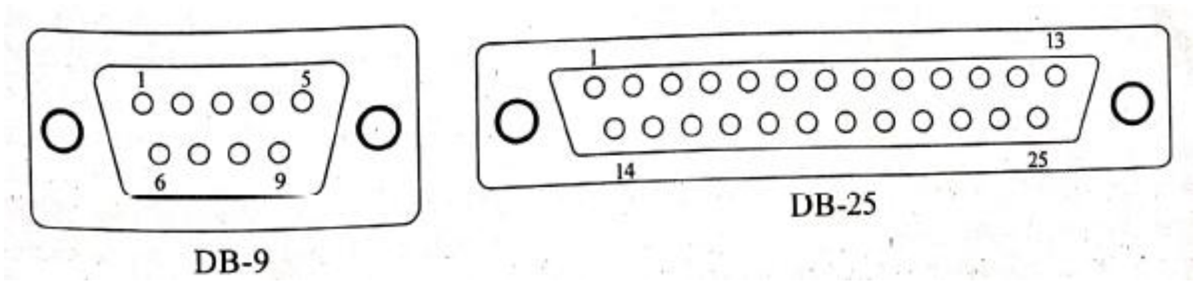
• The RS-232 interface defines various handshaking and control signals for communication apart from the 'Transmit' and. 'Receive' signal lines for data communication.

• RS-232 supports two different types of connectors:

  o DB-9: 9-Pin connector and

  o DB-25: 25-Pin connector.

• The following Figure illustrates the connector details for DB-9 and DB-25.



RS-232 is a point-to-point communication interface and the device involved in RS-232 communication is called 'Data Terminal Equipment (DTE)' and 'Data Communication Equipment (DCE)'.

• 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.

• RS-232 supports only point-to-point communication and not suitable for multi-drop communication. It uses single ended data transfer technique for signal transmission and thereby more susceptible to noise and it greatly reduces the operating distance.

• RS-422 is another serial interface standard from EIA for differential data communication. It supports data rates up to l00Kbps 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.

Explain the merits and limitations of IEEE-1394 over USB.

https://en.wikipedia.org/wiki/IEEE_1394

https://www.lifewire.com/what-is-firewire-2625918

https://www.youtube.com/watch?v=F7NlCaaL3yU

https://www.usb.org/

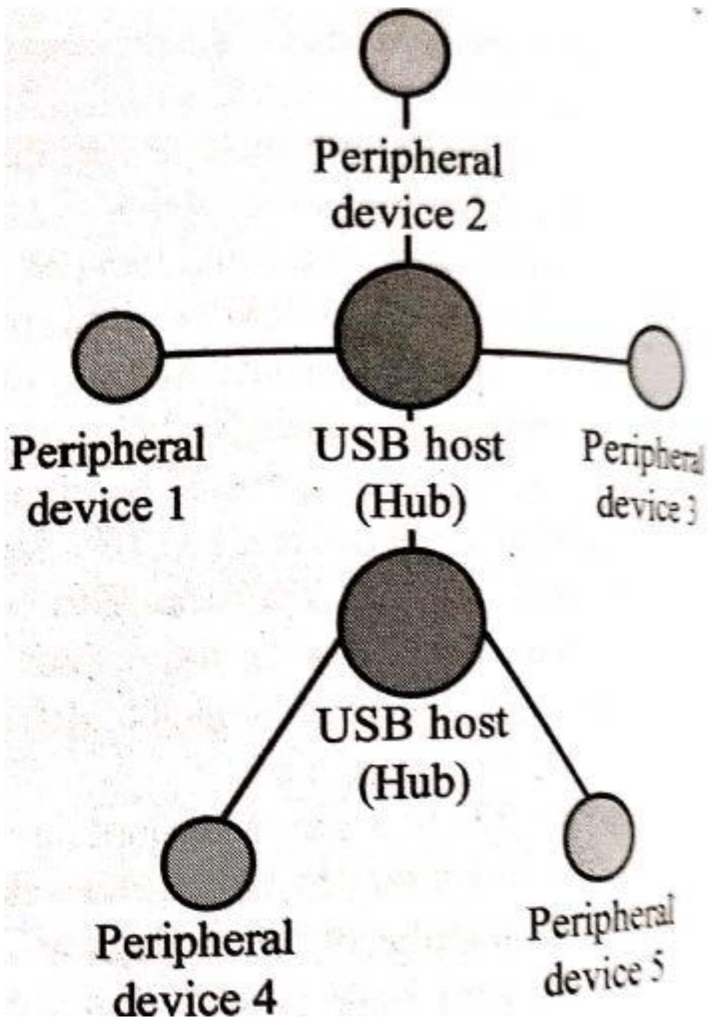
Universal Serial Bus (USB): Universal Serial Bus is a wired high speed serial bus for data communication.

• The first version of USB (USB 1.0) was released in 1995 and was created by the USB core group members consisting of Intel, Microsoft, IBM, Compaq, Digital and Northern Telecom.

• The USB communication system follows a star topology with a USB host at the centre and one or more USB peripheral devices/ USB hosts connected to it.

• A USB 2.0 host can support connections up to 127, including slave peripheral devices and other USB hosts.

The following 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; namely Open Host Control Interface (OHCI) and 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 in USB 2.0 supports communication distance of up to 5 meters.

• The USB 2.0 standard uses two different types of connector at the ends of the USB cable for connecting the USB peripheral device and host device.

  o 'Type A' connector is used for upstream connection (connection with host) and Type B connector is used for downstream connection (connection with slave device).

  o The USB connector present in desktop PCs or laptops are examples for 'Type A' USB connector.

o Both Type A and Type B connectors contain 4 pins for communication.

• The Pin details for the connectors are listed in the table given below.

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

USB uses differential signals for data transmission. It improves the noise immunity. USB interface has the ability to supply power to the connecting devices. Two connection lines (Ground and Power) of the USB interface are dedicated for carrying power. It can supply power up to 500 rnA at 5 V. USB supports or different types of data transfers, namely; Control, Bulk, Isochronous and Interrupt.

• Control transfer is used by USB system software to query, configure and issue commands to the USB device.

• Bulk transfer is used for sending a block of data to a device. Bulk transfer supports error checking and correction.

o Transferring data to a printer is an example for bulk transfer.

• Isochronous data transfer is used for real-time data communication. In Isochronous transfer, data is transmitted as streams in real-time. Isochronous transfer doesn't support error checking and retransmission of data in case of any transmission loss.

o All streaming devices like audio devices and medical equipment for data collection make use of the isochronous transfer.

• Interrupt transfer is used for transferring small amount of data. Interrupt transfer mechanism makes use of polling technique to see whether the USB device has any data to send. The frequency of polling is determined by the USB device and it varies from 1 to 255 milliseconds.

o 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 it is available from various players with different names.

o Apple Inc's implementation of 1394 protocol is popularly known as Firewire.

o i.LINK is the 1394 implementation from Sony Corporation

o 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.

• 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.

• Unlike USB interface (except USB OTG), IEEE 1394 doesn't require a host for communicating between devices.

o For example, you can directly connect a scanner with a printer for printing.

• The data- rate supported by 1394 is far higher than the one supported by USB2.0 interface.

• The 1394 hardware implementation is much costlier than USB implementation.

Explain the different external communication interfaces in brief

RS-232 C & RS-485
Universal Serial Bus (USB)
IEEE 1394 (Firewire)
Infrared (IrDA)
Bluetooth (BT)
Wi-Fi: Wi-Fi or Wireless Fidelity
ZigBee
General Packet Radio Service (GPRS), 3G, 4G, LTE

EMBEDDED FIRMWARE:

What is Embedded Firmware? What are the different approaches available for Embedded Firmware development?

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. They are listed below:

1. Write the program in high level languages like Embedded C/ C++ using an Integrated Development Environment

    o The IDE will contain a editor, compiler, linker, debugger, simulator, etc.

    o IDEs are different for different family of processors/ controllers.

    o For example, Keil microvision3 IDE is used for all family member 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 instruction set for each family of processor/ controller is different and the program written in either of the methods given above 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.
    - o 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'.
    - o If you are following the Assembly language based programming technique, you can use the utilities supplied by the processor/ controller vendors to convert the source code into 'HEX File'.
    - o 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. The reasons for this being:
    - o Writing codes in a high level language is easy, the code written in high level language is highly portable which means you can use the same code 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.

o Also 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 s/he should carry an instruction set reference manual with her/ him. A programmer using assembly language technique writes the program according to his/ her view and taste. Often he/ she 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/ her 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.

OTHER SYSTEM COMPONENTS:

Explain the role of Reset circuit in Embedded System

https://hardwarebee.com/introduction-to-power-on-reset-circuit/

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 (The processor undergoes reset when the reset pin of the processor is at logic high) or active low (The processor undergoes reset when the reset pin of the processor is at logic low).

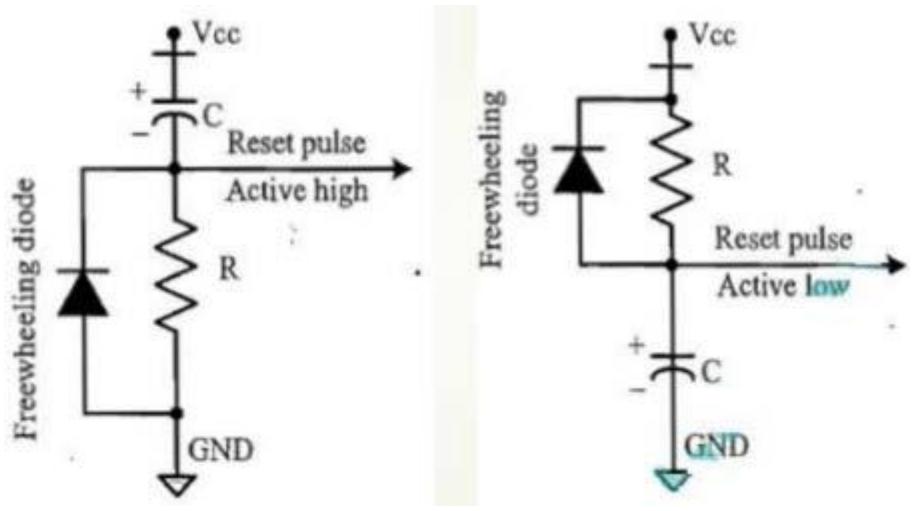
• Since the processor operation is synchronized to a clock signal, the reset pulse should be wide enough to give time for the clock oscillator to stabilize before the internal reset state starts.

• 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.

The following 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.



Explain the role of Brown-out circuit in Embedded System.

https://www.youtube.com/watch?v=5ZrjtKx2Xmg

https://iitestudent.blogspot.com/2014/03/brownout-protection.html

Brown-out Protection Circuit: Brown-out protection circuit prevents the processor/ controller from unexpected program execution behavior when the supply voltage to the processor/ controller falls below a specified voltage.

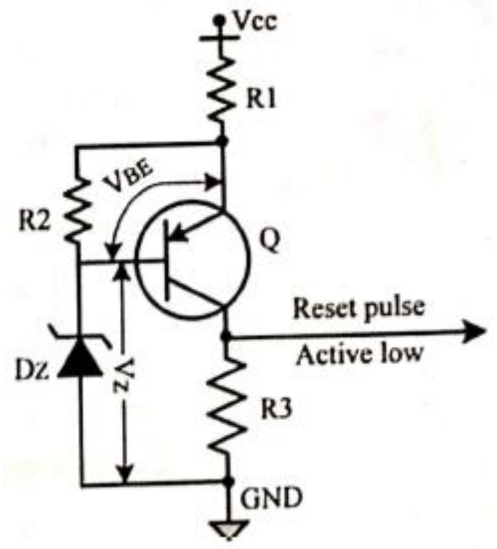
• It is essential for battery powered devices since there are greater chances for the battery voltage to drop below the required threshold. The processor behavior may not be predictable if the supply voltage falls below the recommended operating voltage. It may lead to situations like data corruption.

• A brown-out protection circuit holds the processor/ controller in reset state, when 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 don't integrate a built-in brown-out protection circuit, the same can be implemented using external passive circuits or supervisor ICs.

The following Figure illustrates a brown-out circuit implementation using Zener diode and transistor for processor/ controller with active low Reset logic.



The Zener diode, Dz, and transistor, Q, forms the heart of this circuit. The transistor conducts always when the supply voltage VCC is greater than that of the sum of VBE and VZ (Zener voltage). The transistor stops conducting when the supply voltage falls below the sum of VBE and VZ. Select the Zener diode with required voltage for setting the low threshold value for VCC. The values of Rl, R2, and R3 can be selected based on the electrical characteristics of the transistor in use.

Explain the role of Real Time Clock in Embedded System.

https://www.electronics-tutorials.ws/connectivity/real-time-clocks.html

https://www.youtube.com/watch?v=AabPY89319s

Real-Time Clock (RTC): Real-Time Clock is a system component responsible for keeping track of time. RTC holds information like current time (In hours, minutes and seconds) in 12-hour/ 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 updating, 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).

Explain the role of Watchdog Timer in Embedded System.

https://www.embedded.com/introduction-to-watchdog-timers/

https://www.youtube.com/watch?v=XOpH6N6567A

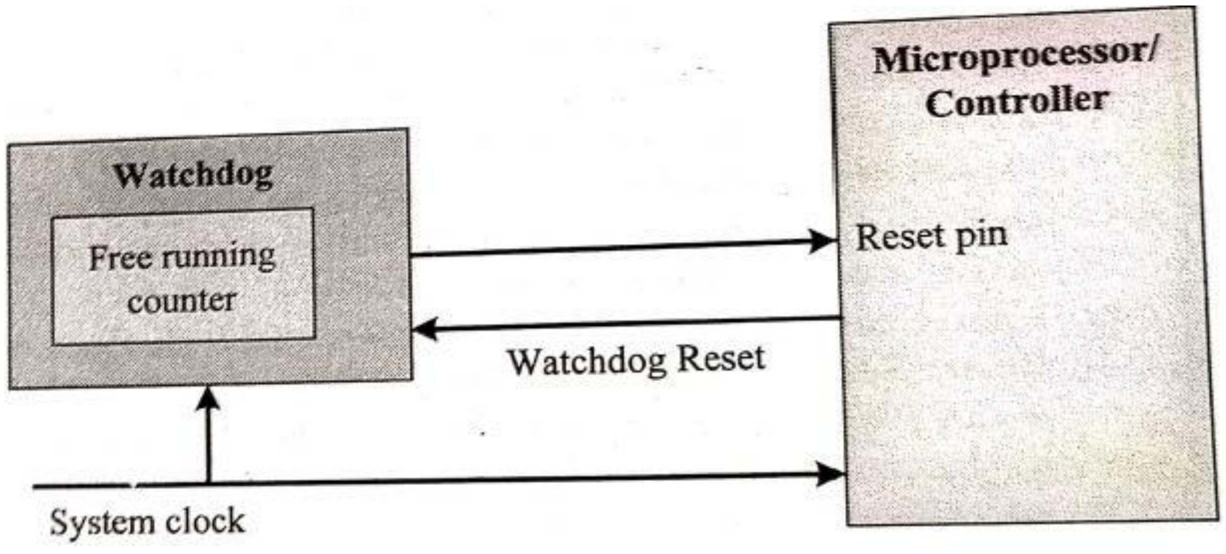
Watchdog Timer: In desktop Windows systems, if we feel our application is behaving in an abnormally or if the system hangs up, we have the 'Ctrl + Alt + Del' to come out of the situation. What it happens to embedded system?

• We have a watchdog to monitor the firmware execution and reset the system processor/ microcontroller when the program execution hangs up. A watchdog timer, or simply a watchdog, is a hardware timer for monitoring the firmware execution. 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 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, you can reset the count by writing a 0 (for an up counting watchdog timer) to the watchdog timer register.

• 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 DS 1232 integrates a hardware watchdog timer in it.

The following Figure illustrates the implementation of the external watchdog timer based microprocessor based supervisor circuit for a small embedded system.


Microprocessor/ Controller
Watchdog
Free running counter
Reset pin
Watchdog Reset
System clock

*** *** *** *** ***