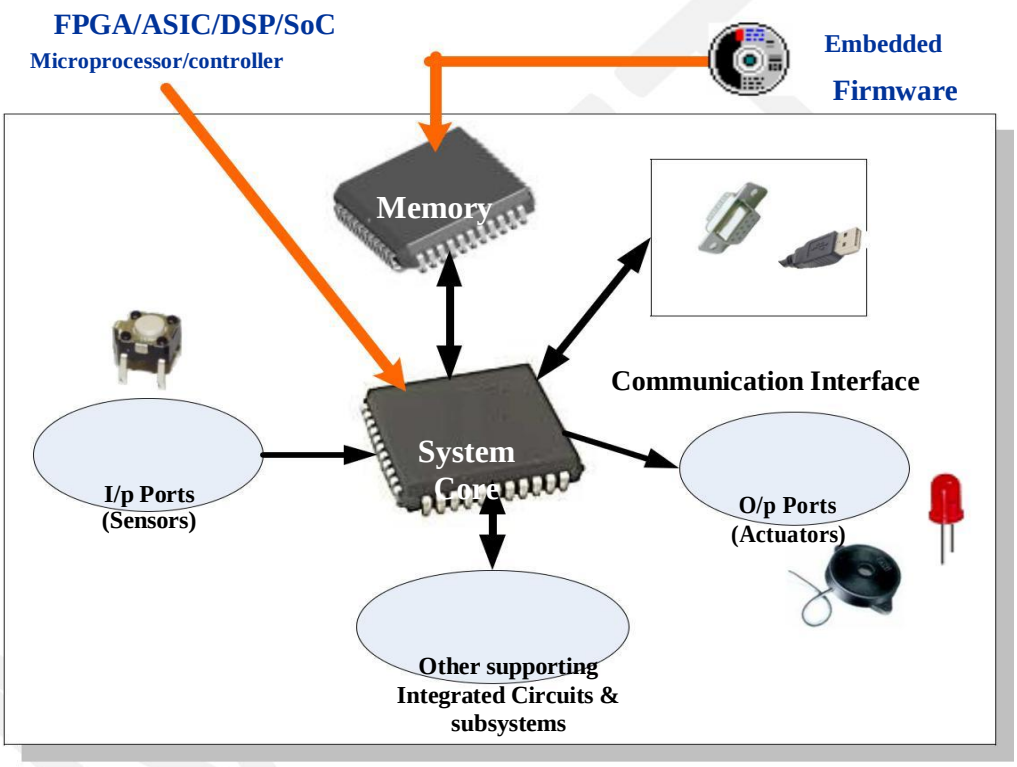
ELEMENTS OF EMBEDDED SYSTEMS:

An embedded system is a combination of 3 things, Hardware Software Mechanical Components and it is supposed to do one specific task only. A typical embedded system contains a single chip controller which acts as the master brain of the system. Diagrammatically an embedded system can be represented as follows:



Embedded System
Real World

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

The control is achieved by processing the information coming from the sensors and user interfaces and controlling some actuators that regulate the physical variable.

Keyboards, push button, switches, etc. are Examples of common user interface input devices and LEDs, LCDs, Piezoelectric buzzers, etc examples for common user interface output devices for a typical embedded system.The requirement of type of user interface changes from application to application based on domain.

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

The memory of the system is responsible for holding the code (control algorithm and other important configuration details). There are two types of memories are used in any embedded system. Fixed memory (ROM) is used for storing code or program. The user cannot change the firmware in this type of memory. The most common types of memories used in embedded systems for control algorithm storage are OTP,PROM,UVEPROM,EEPROM and FLASH

An embedded system without code (i.e. the control algorithm) implemented memory has all the peripherals but is not capable of making decisions depending on the situational as well as real world changes.

Memory for implementing the code may be present on the processor or may be implemented as a separate chip interfacing the processor

In a controller based embedded system, the controller may contain internal memory for storing code such controllers are called Micro-controllers with on-chip ROM, eg. Atmel AT89C51.

The Core of the Embedded Systems: The core of the embedded system falls into any one of the following categories.

- **General Purpose and Domain Specific Processors**
  - Microprocessors
  - Microcontrollers
  - Digital Signal Processors
- **Programmable Logic Devices (PLDs)**
- **Application Specific Integrated Circuits (ASICs)**
- **Commercial off the shelf Components (COTS)**

GENERAL PURPOSE AND DOMAIN SPECIFIC PROCESSOR:

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

Microprocessor:

- A silicon chip representing a Central Processing Unit (CPU), which is capable of performing arithmetic as well as logical operations according to a pre-defined set of Instructions, which is specific to the manufacturer
- In general the CPU contains the Arithmetic and Logic Unit (ALU), Control Unit and Working registers
- Microprocessor is a dependant unit and it requires the combination of other hardware like Memory, Timer Unit, and Interrupt Controller etc for proper functioning.
- Intel claims the credit for developing the first Microprocessor unit Intel 4004, a 4 bit processor which was released in Nov 1971

· Developers of microprocessors.

- Intel – Intel 4004 – November 1971(4-bit)
- Intel – Intel 4040.
- Intel – Intel 8008 – April 1972.
- Intel – Intel 8080 – April 1974(8-bit).
- Motorola – Motorola 6800.
- Intel – Intel 8085 – 1976.
- Zilog - Z80 – July 1976

Microcontroller:

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

Microprocessor Vs Microcontroller:

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

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

- General Purpose Processor or GPP is a processor designed for general computational tasks
- GPPs are produced in large volumes and targeting the general market. Due to the high volume production, the per unit cost for a chip is low compared to ASIC or other specific ICs
- A typical general purpose processor contains an Arithmetic and Logic Unit (ALU) and Control Unit (CU)
- Application Specific Instruction Set processors (ASIPs) are processors with architecture and instruction set optimized to specific domain/application requirements like Network processing, Automotive, Telecom, media applications, digital signal processing, control applications etc.
- ASIPs fill the architectural spectrum between General Purpose Processors and Application Specific Integrated Circuits (ASICs)
- The need for an ASIP arises when the traditional general purpose processor are unable to meet the increasing application needs
- Some Microcontrollers (like Automotive AVR, USB AVR from Atmel), System on Chips, Digital Signal Processors etc are examples of Application Specific Instruction Set Processors (ASIPs)
- ASIPs incorporate a processor and on-chip peripherals, demanded by the application requirement, program and data memory

Digital Signal Processors (DSPs):

- 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
- 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
- DSP can be viewed as a microchip designed for performing high speed computational operations for „addition“, „subtraction“, „multiplication“ and „division“

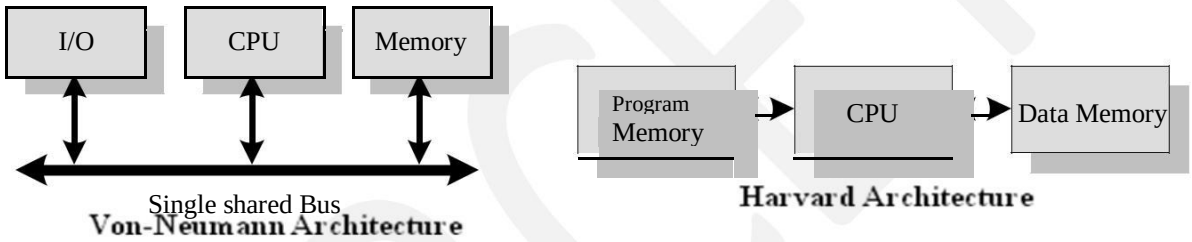
- A typical Digital Signal Processor incorporates the following key units
  - Program Memory
  - Data Memory
  - Computational Engine
  - I/O Unit
- Audio video signal processing, telecommunication and multimedia applications are typical examples where DSP is employed

RISC V/s CISC Processors/Controllers:

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

Harvard V/s Von-Neumann Processor/Controller Architecture

- The terms Harvard and Von-Neumann refers to the processor architecture design.
- Microprocessors/controllers based on the **Von-Neumann** architecture shares a single common bus for fetching both instructions and data. Program instructions and data are stored in a common main memory
- Microprocessors/controllers based on the **Harvard** architecture will have separate data bus and instruction bus. This allows the data transfer and program fetching to occur simultaneously on both buses
- With Harvard architecture, the data memory can be read and written while the program memory is being accessed. These separated data memory and code memory buses allow one instruction to execute while the next instruction is fetched ("Pre-fetching")

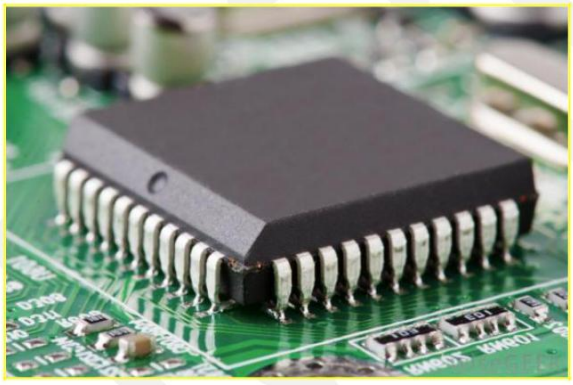


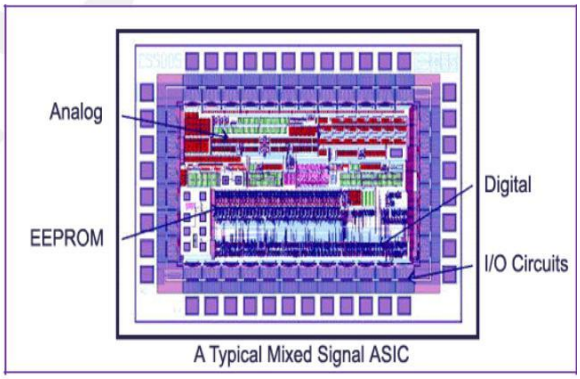
Harvard V/s Von-Neumann Processor/Controller Architecture:

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

Application Specific Integrated Circuit (ASIC):

- A microchip designed to perform a specific or unique application. It is used as replacement to conventional general purpose logic chips.
- ASIC integrates several functions into a single chip and thereby reduces the system development cost
- Most of the ASICs are proprietary products. As a single chip, ASIC consumes very small area in the total system and thereby helps in the design of smaller systems with high capabilities/functionalities.
- ASICs can be pre-fabricated for a special application or it can be custom fabricated by using the components from a re-usable „*building block*" library of components for a particular customer application





A Typical Mixed Signal ASIC

- Fabrication of ASICs requires a non-refundable initial investment (Non Recurring Engineering (NRE) charges) for the process technology and configuration expenses
- If the Non-Recurring Engineering Charges (NRE) is born by a third party and the Application Specific Integrated Circuit (ASIC) is made openly available in the market, the ASIC is referred as Application Specific Standard Product (ASSP)
- The ASSP is marketed to multiple customers just as a general-purpose product , but to a smaller number of customers since it is for a specific application.

- Some ASICs are proprietary products , the developers are not interested in revealing the internal details.

Programmable Logic Devices (PLDs):

- Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform.

- Logic devices can be classified into two broad categories - Fixed and Programmable. The circuits in a fixed logic device are permanent, they perform one function or set of functions - once manufactured, they cannot be changed

- Programmable logic devices (PLDs) offer customers a wide range of logic capacity, features, speed, and voltage characteristics - and these devices can be re-configured to perform any number of functions at any time

- Designers can use inexpensive software tools to quickly develop, simulate, and test their logic designs in PLD based design. The design can be quickly programmed into a device, and immediately tested in a live circuit

- PLDs are based on re-writable memory technology and the device is reprogrammed to change the design

Programmable Logic Devices (PLDs) – CPLDs and FPGA

- Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs) are the two major types of programmable logic devices

FPGA:

- FPGA is an IC designed to be configured by a designer after manufacturing.
- FPGAs offer the highest amount of logic density, the most features, and the highest performance.
- Logic gate is Medium to high density ranging from **1K to 500K** system gates

- These advanced FPGA devices also offer features such as built-in hardwired processors (such as the IBM Power PC), substantial amounts of memory, clock management systems, and support for many of the latest, very fast device-to-device signaling technologies

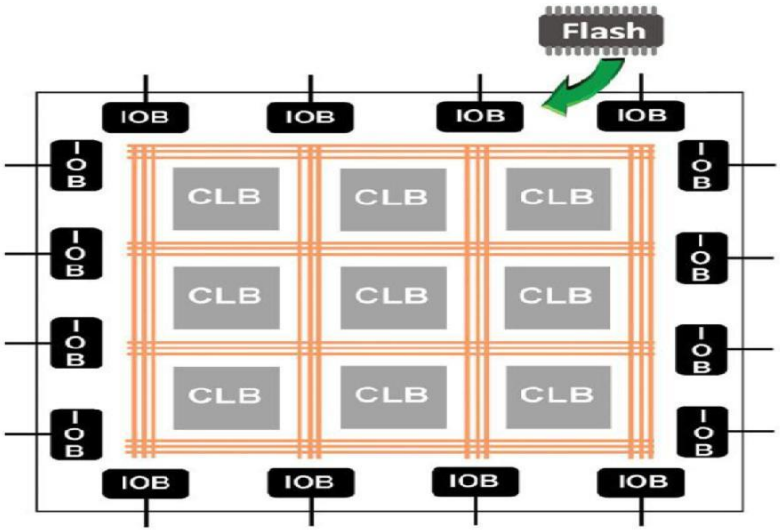


Figure: FPGA Architecture

- These advanced FPGA devices also offer features such as built-in hardwired processors, substantial amounts of memory, clock management systems, and support for many of the latest, very fast device-to-device signaling technologies.

- FPGAs are used in a wide variety of applications ranging from data processing and storage, to instrumentation, telecommunications, and digital signal processing

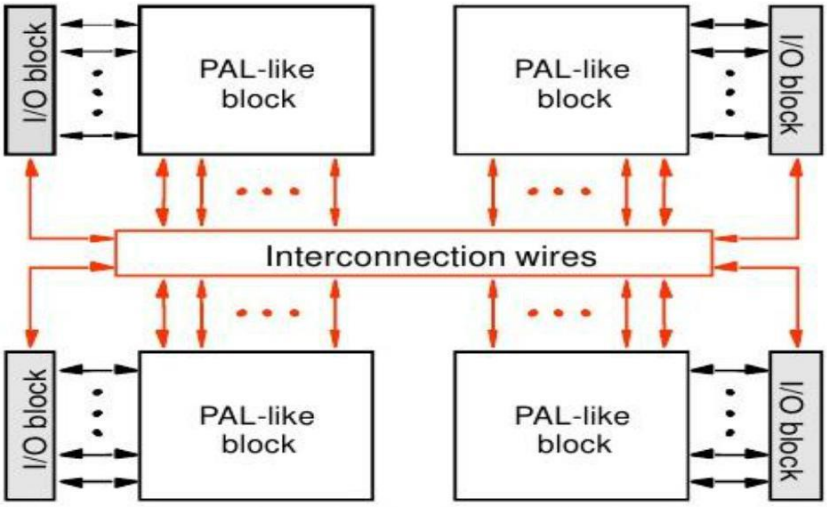
CPLD:

- A **complex programmable logic device** (**CPLD**) is a programmable logic device with complexity between that of PALs and FPGAs, and architectural features of both.

- CPLDs, by contrast, offer much smaller amounts of logic - up to about 10,000 gates.

- CPLDs offer very predictable timing characteristics and are therefore ideal for critical control applications.

Structure of a CPLD



- CPLDs such as the Xilinx **CoolRunner** series also require extremely low amounts of power and are very inexpensive, making them ideal for cost-sensitive, battery-operated, portable applications such as mobile phones and digital handheld assistants.

ADVANTAGES OF PLDs:

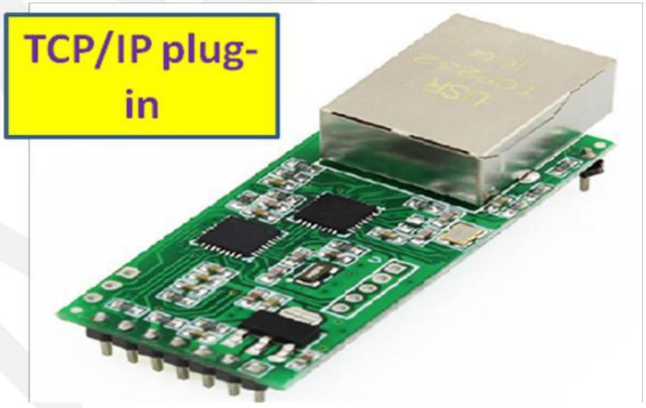
- PLDs offer customer much more flexibility during design cycle
- PLDSs do not require long lead times for prototype or production-the PLDs are already on a distributor‟s self and ready for shipment
- PLDs do not require customers to pay for large NRE costs and purchase expensive mask sets
- PLDs allow customers to order just the number of parts required when they need them. allowing them to control inventory.
- PLDs are reprogrammable even after a piece of equipment is shipped to a customer.
- The manufacturers able to add new features or upgrade the PLD based products that are in the field by uploading new programming file

Commercial off the Shelf Component (COTS):

- A Commercial off-the-shelf (COTS) product is one which is used „*as-is*‟
- COTS products are designed in such a way to provide easy integration and interoperability with existing system components

- Typical examples for the COTS hardware unit are Remote Controlled Toy Car control unit including the RF Circuitry part, High performance, high frequency microwave electronics (2 to 200 GHz), High bandwidth analog-to-digital converters, Devices and components for operation at very high temperatures, Electro-optic IR imaging arrays, UV/IR Detectors etc

- A COTS component in turn contains a General Purpose Processor (GPP) or Application Specific Instruction Set Processor (ASIP) or Application Specific Integrated Chip (ASIC)/Application Specific Standard Product (ASSP) or Programmable Logic Device (PLD)



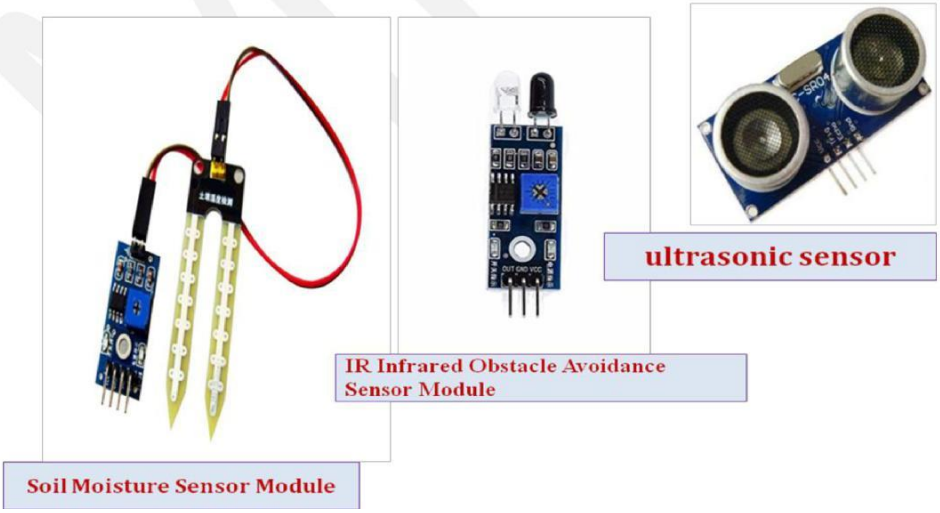
- The major advantage of using COTS is that they are readily available in the market, cheap and a developer can cut down his/her development time to a great extend.
- There is no need to design the module yourself and write the firmware .
- Everything will be readily supplied by the COTs manufacturer.

Sensors & Actuators:

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

Sensor:

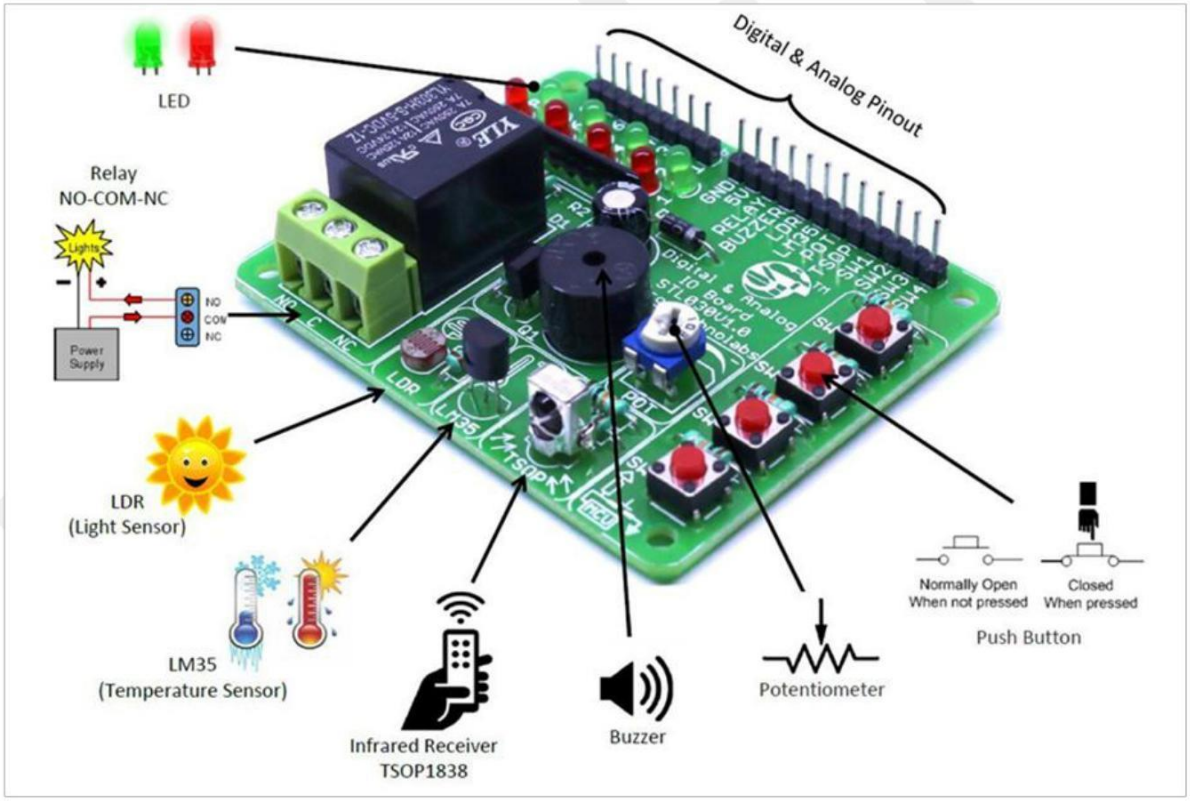
- A transducer device which converts energy from one form to another for any measurement or control purpose. Sensors acts as input device
- Eg. Hall Effect Sensor which measures the distance between the cushion and magnet in the Smart Running shoes from adidas
- **Example: IR, humidity , PIR(passive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



Actuator:

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

  Eg. Micro motor actuator which adjusts the position of
- the cushioning element in the Smart Running shoes from adidas



Silicon TechnoLabs Digital Analog Arduino Starter kit

Communication Interface:

- Communication interface is essential for communicating with various subsystems of the embedded system and with the external world

- The communication interface can be viewed in two different perspectives; namely;

1. Device/board level communication interface (Onboard Communication Interface)

2. Product level communication interface (External Communication Interface)

1. Device/board level communication interface (Onboard Communication Interface):

The communication channel which interconnects the various components within an embedded product is referred as Device/board level communication interface (Onboard Communication Interface)

- Examples: Serial interfaces like I2C, SPI, UART, 1-Wire etc and Parallel bus interface

2. Product level communication interface (External Communication Interface):

The „Product level communication interface" (External Communication Interface) is responsible for data transfer between the embedded system and other devices or modules. The external communication interface can be either wired media or wireless media and it can be a serial or parallel interface.

- Examples for wireless communication interface: Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves (RF), GPRS etc.
- Examples for wired interfaces: RS-232C/RS-422/RS 485, USB, Ethernet (TCP-IP), IEEE 1394 port, Parallel port etc.

1. Device/board level or On board communication interfaces: The

Communication channel which interconnects the various components within an embedded product is referred as Device/board level communication interface (Onboard Communication Interface)

These are classified into

1.1 I2C (Inter Integrated Circuit) Bus

1.2 SPI (Serial Peripheral Interface) Bus

1.3 UART (Universal Asynchronous Receiver Transmitter)

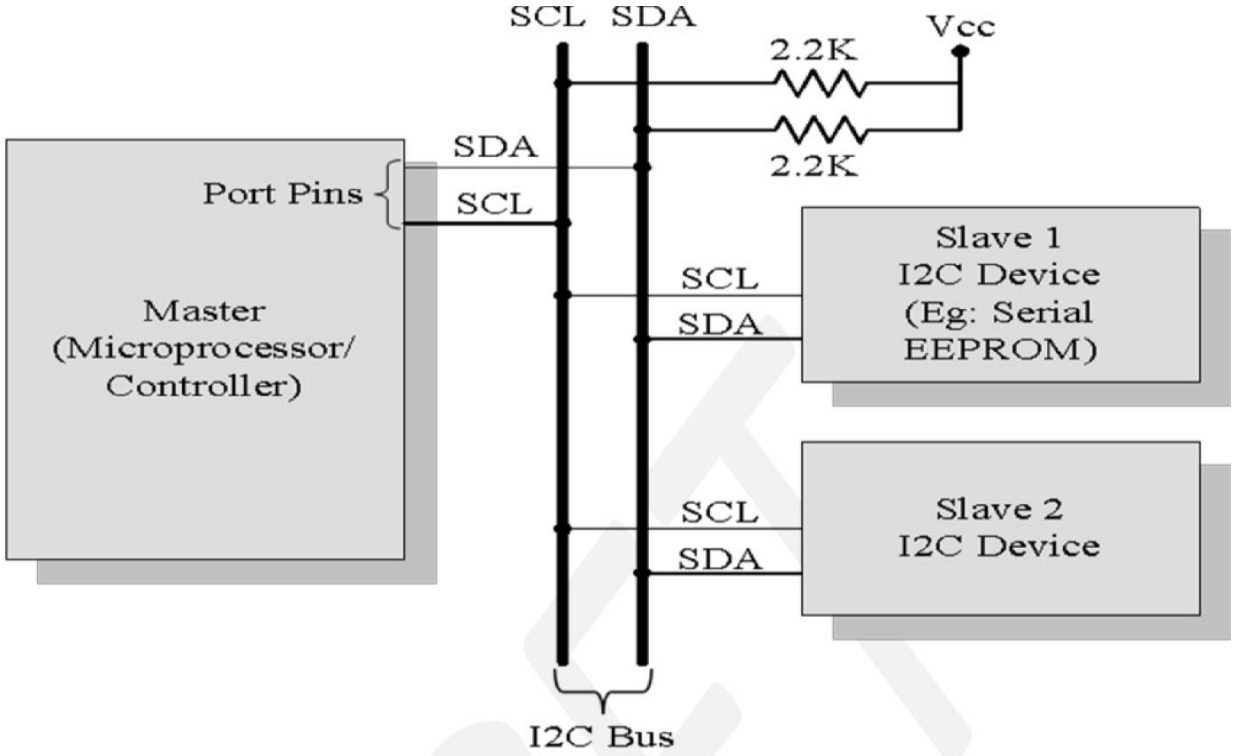
1.4 1-Wires Interface

1.5 Parallel Interface

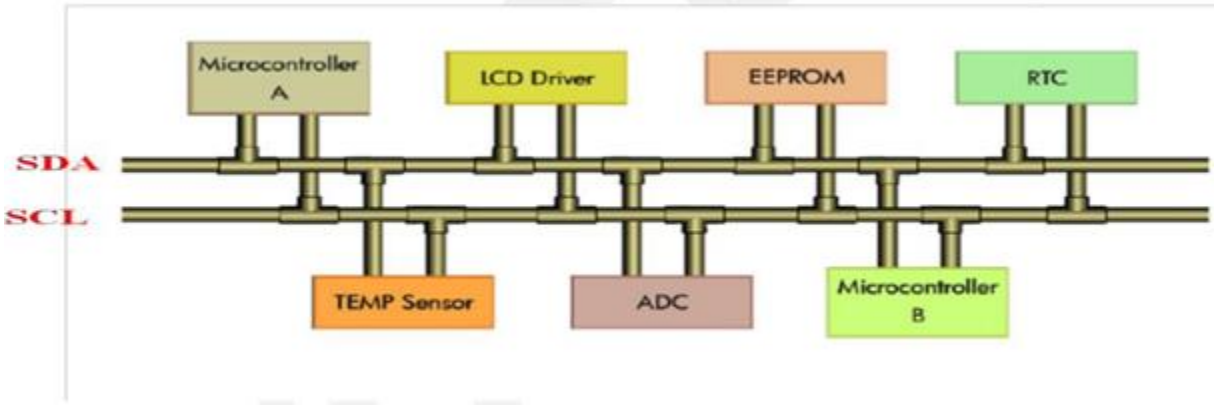
1 I2C (Inter Integrated Circuit) Bus:

Inter Integrated Circuit Bus (I2C - Pronounced „I square C“) is a synchronous bi-directional 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 1980“s. 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 is comprised of two bus lines, namely; Serial Clock – SCL and Serial Data – SDA.



I2C Bus Interfacing



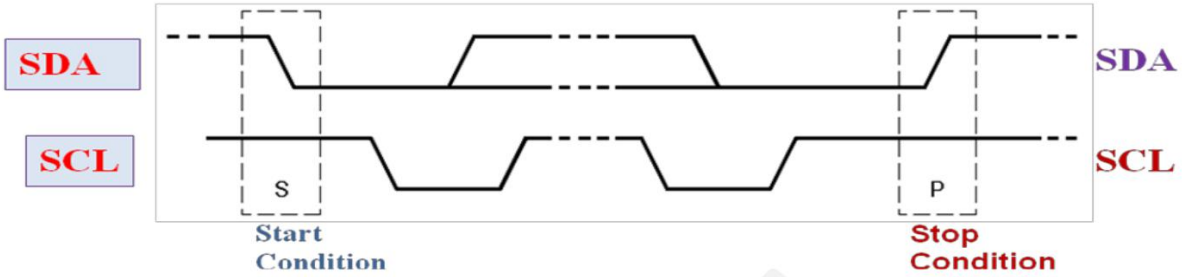
SCL line is responsible for generating synchronization clock pulses and SDA is responsible for transmitting the serial data across devices.I2C bus is a shared bus system to which many number of I2C devices can be connected. Devices connected to the I2C bus can act as either „Master" device or „Slave" device.

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

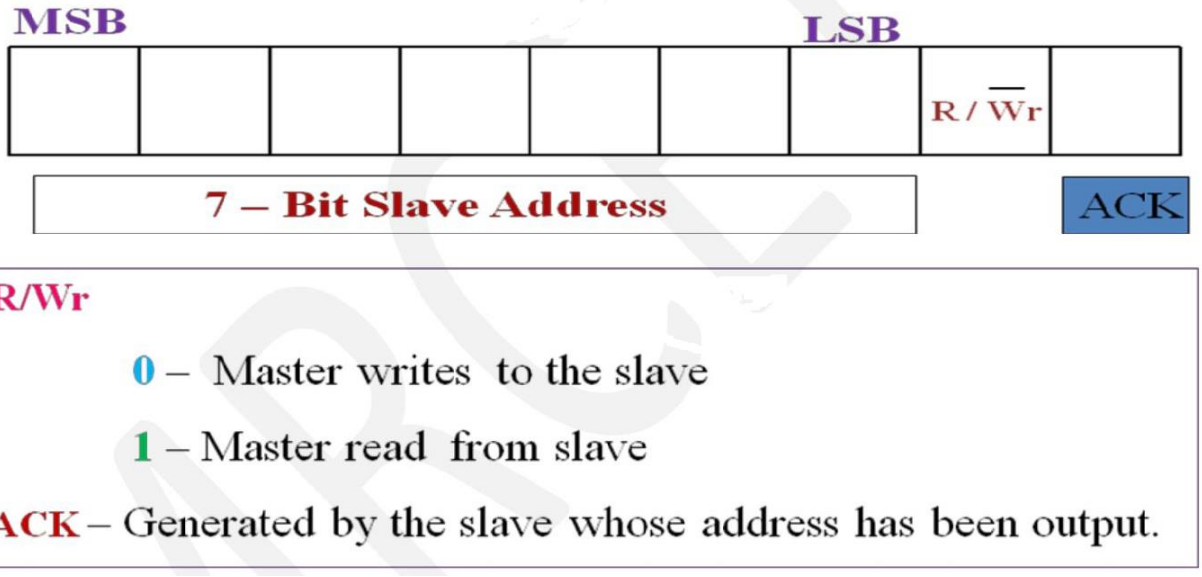
Slave devices wait for the commands from the master and respond upon receiving the commands. Master and „Slave" devices can act as either transmitter or receiver. Regardless whether a master is acting as transmitter or receiver, the synchronization clock signal is generated by the „Master" device only.I2C supports multi masters on the same bus.

The sequence of operation for communicating with an I2C slave device is:

1. Master device pulls the clock line (SCL) of the bus to „HIGH"
2. Master device pulls the data line (SDA) „LOW", when the SCL line is at logic „HIGH" (This is the „Start" condition for data transfer)

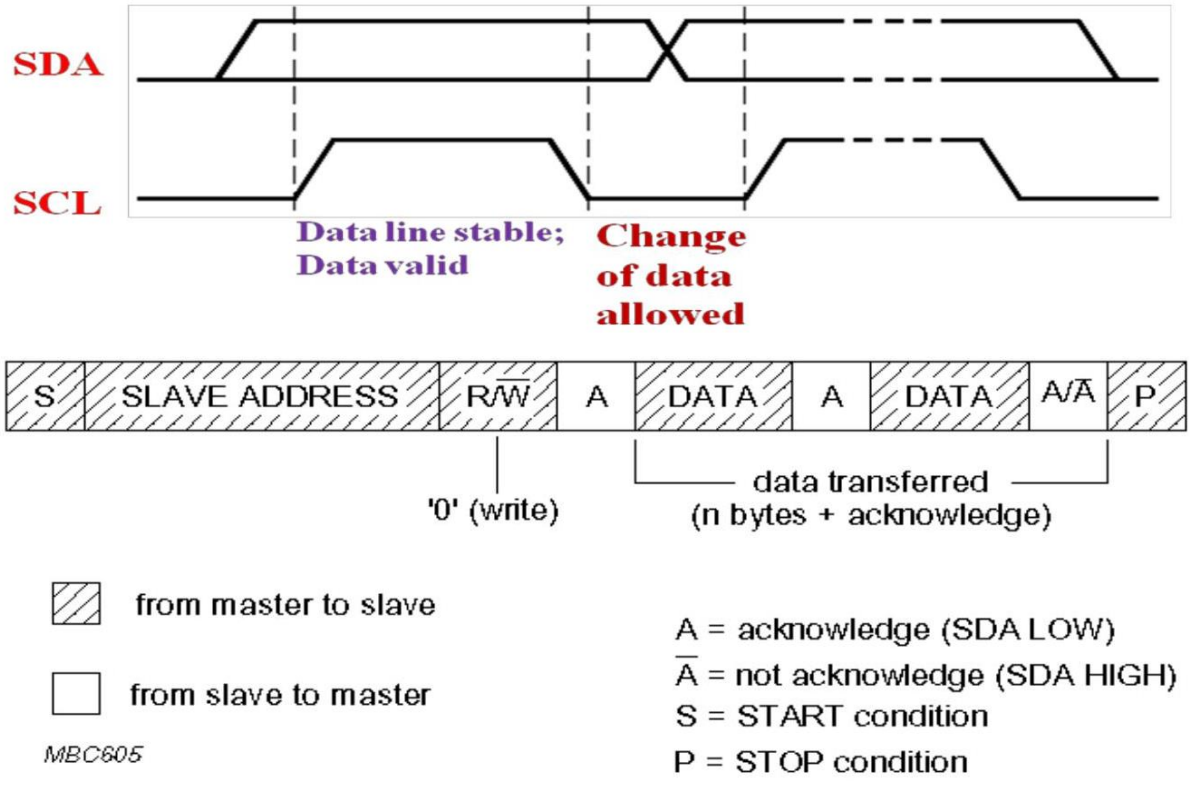


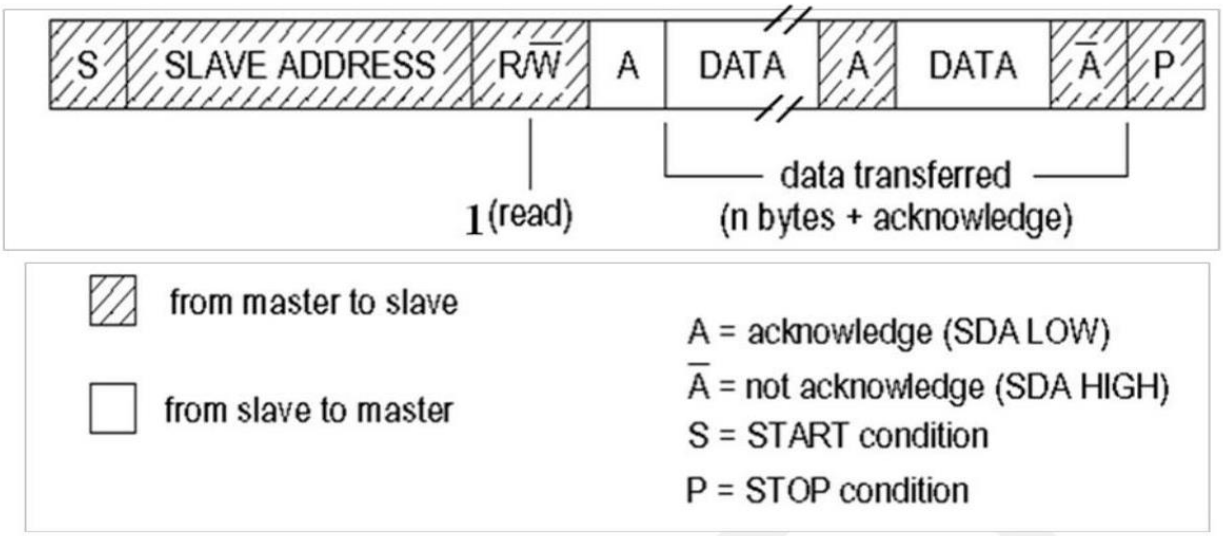
3. Master sends the address (7 bit or 10 bit wide) of the „Slave" device to which it wants to communicate, over the SDA line.
4. Clock pulses are generated at the SCL line for synchronizing the bit reception by the slave device.
5. The MSB of the data is always transmitted first.
6. The data in the bus is valid during the „HIGH" period of the clock signal
7. In normal data transfer, the data line only changes state when the clock is low.



8. Master waits for the acknowledgement bit from the slave device whose address is sent on the bus along with the Read/Write operation command.

9. Slave devices connected to the bus compares the address received with the address assigned to them
10. The Slave device with the address requested by the master device responds by sending an acknowledge bit (Bit value =1) over the SDA line
11. Upon receiving the acknowledge bit, master sends the 8bit data to the slave device over SDA line, if the requested operation is „Write to device".
12. If the requested operation is „Read from device", the slave device sends data to the master over the SDA line.
13. Master 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
14. Master terminates the transfer by pulling the SDA line „HIGH" when the clock line SCL is at logic „HIGH" (Indicating the „STOP" condition).

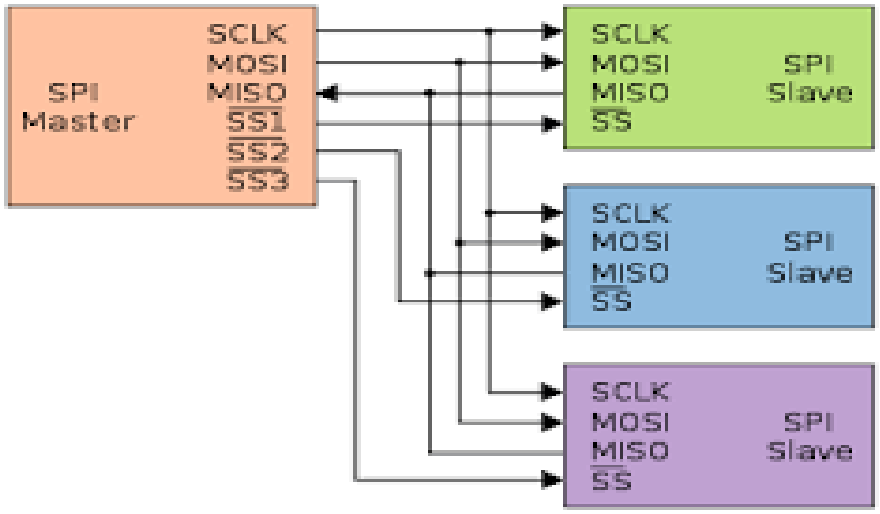




1.2 Serial Peripheral Interface (SPI) Bus:

The Serial Peripheral Interface Bus (SPI) is a synchronous bi-directional full duplex four wire serial interface bus. The concept of SPI is introduced by Motorola.SPI is a single master multi-slave system.

- It is possible to have a system where more than one SPI device can be master, provided the condition only one master device is active at any given point of time, is satisfied.
- SPI is used to send data between Microcontrollers and small peripherals such as shift registers, sensors, and SD cards.



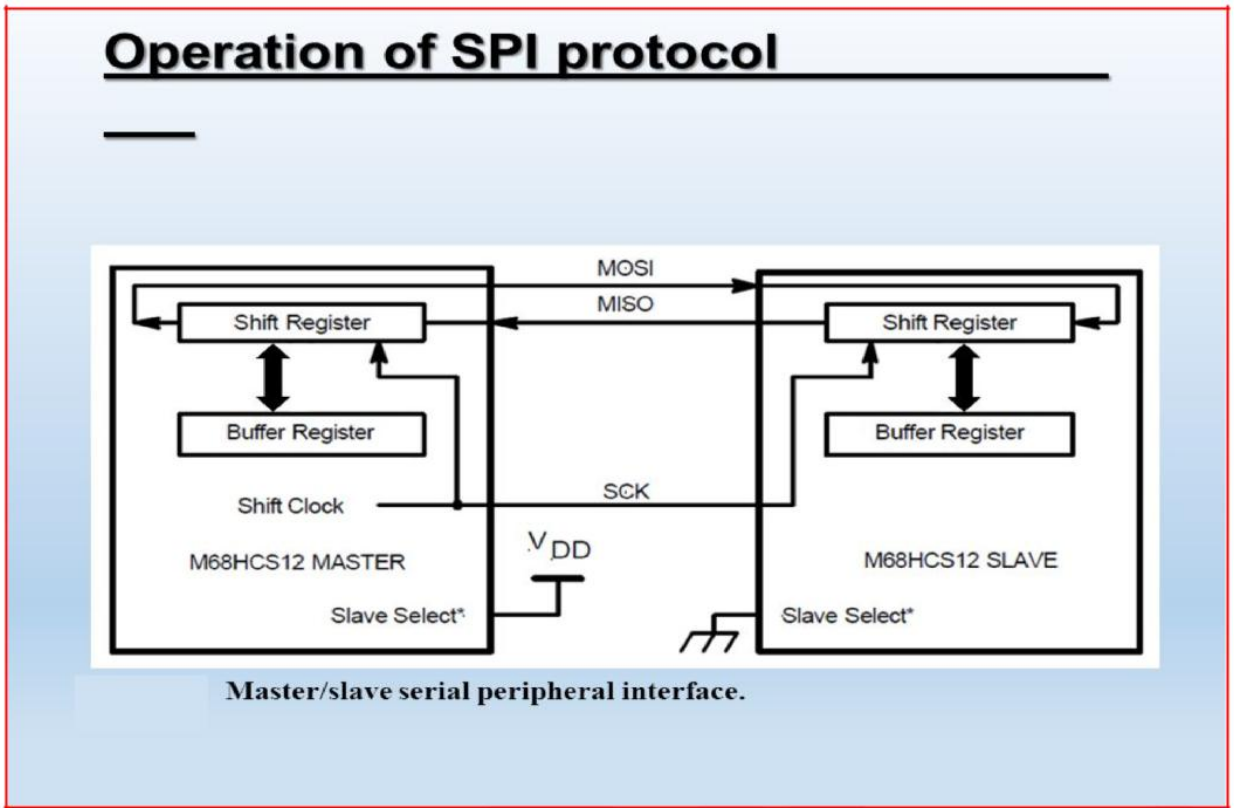
SPI requires four signal lines for communication. They are:

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

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

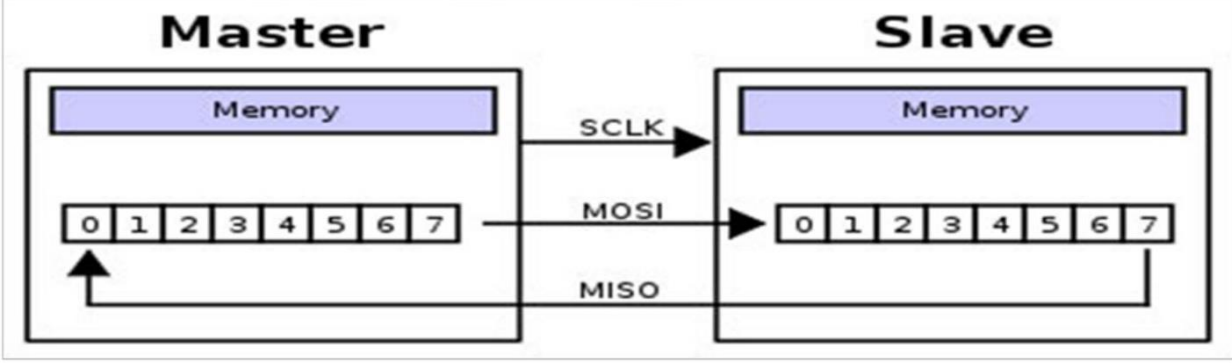
Serial Clock (SCLK): Signal line carrying the clock signals
Slave Select (SS): Signal line for slave device select. It is an active low signal.
The master device is responsible for generating the clock signal.
Master device selects the required slave device by asserting the corresponding slave devices slave select signal „LOW".

- The data out line (MISO) of all the slave devices when not selected floats at high impedance state
- The serial data transmission through SPI Bus is fully configurable.
- SPI devices contain certain set of registers for holding these configurations.
- The Serial Peripheral Control Register holds the various configuration parameters like master/slave selection for the device, baudrate selection for communication, clock signal control etc.
- The status register holds the status of various conditions for transmission and reception.SPI works on the principle of „Shift Register".
- The master and slave devices contain a special shift register for the data to transmit or receive.
- The size of the shift register is device dependent. Normally it is a multiple of 8.



- During transmission from the master to slave, the data in the master"s shift register is shifted out to the MOSI pin and it enters the shift register of the slave device through the MOSI pin of the slave device.

- At the same time the shifted out data bit from the slave device's shift register enters the shift register of the master device through MISO pin



Master shifts out data to Slave, and shift in data from Slave

I2C V/S SPI:

I2C	SPI
Speed limit varies from 100kbps, 400kbps, 1mbps, 3.4mbps depending on i2c version.	More than 1mbps, 10mbps till 100mbps can be achieved.
Half duplex synchronous protocol	Full Duplex synchronous protocol
Support Multi master configuration	Multi master configuration is not possible
Acknowledgement at each transfer	No Acknowledgement
Require Two Pins only SDA, SCL	Require separate MISO, MOSI, CLK & CS signal for each slave.
Addition of new device on the bus is easy	Addition of new device on the bus is not much easy a I2C
More Overhead (due to acknowledgement, start, stop)	Less Overhead
Noise sensitivity is high	Less noise sensitivity