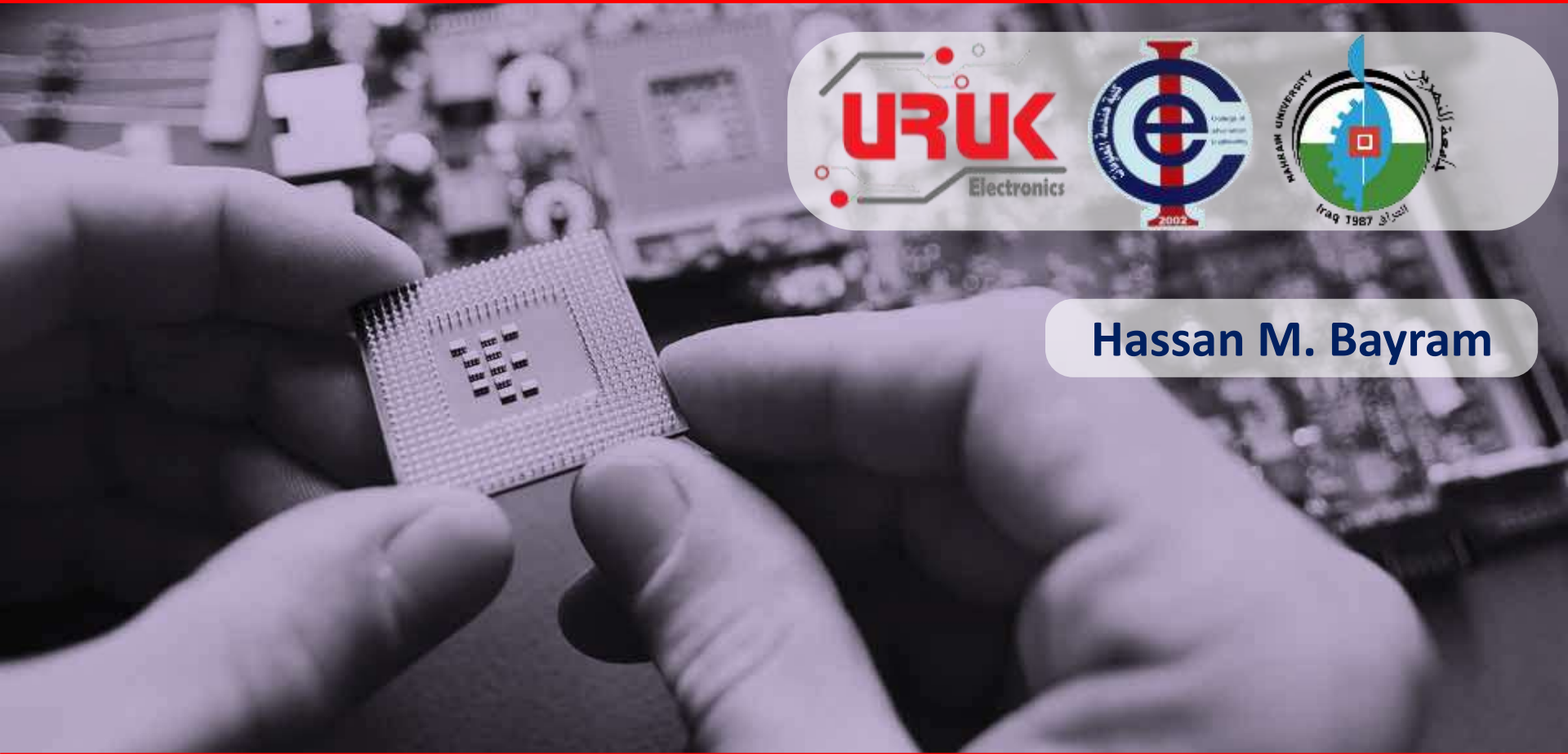


# Interfacing Techniques in Embedded Systems



Hassan M. Bayram

# Lecture Contents

- Introduction
- Serial and Parallel Communication
- Serial Vs. Parallel
- Asynchronous and Synchronous Serial Communication
- UART
- SPI
- I2C
- USB
- Talking with Arduino Project.

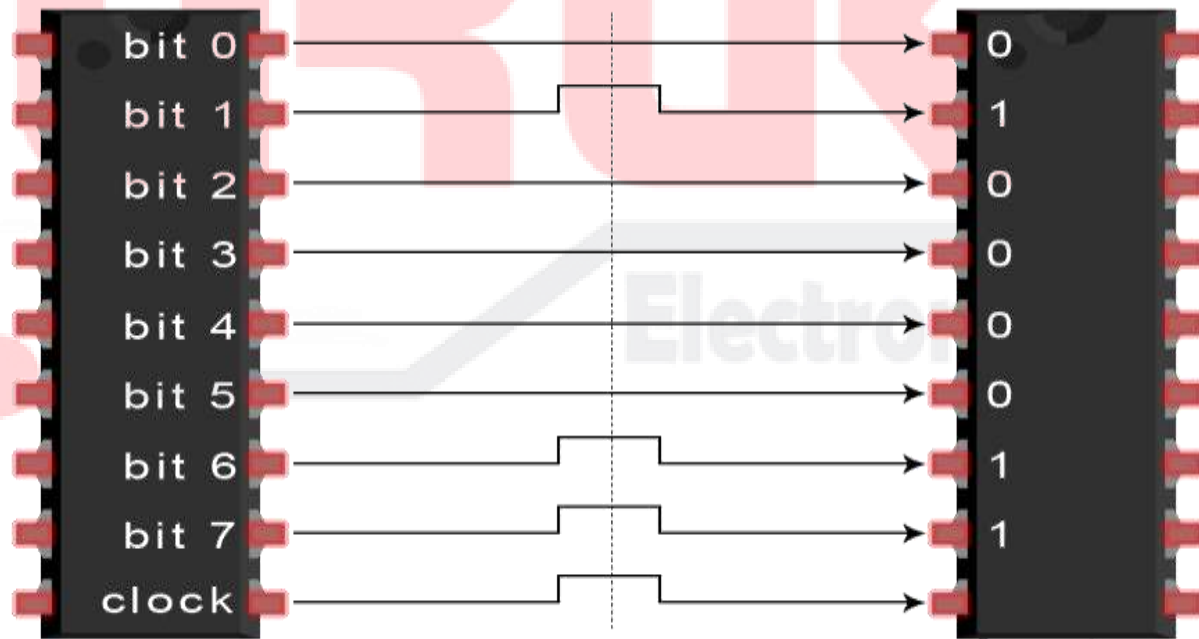
# Introduction to Communication

- Communication is the act of **giving, receiving or exchanging information, ideas and opinions** so that the message is completely understood by both parties.
- In **Microcontrollers**, communication is simply **exchanging data between two or more microcontrollers**.
- There are **many protocols** for communication, but all of those are based on either **serial communication** or **parallel communication**.



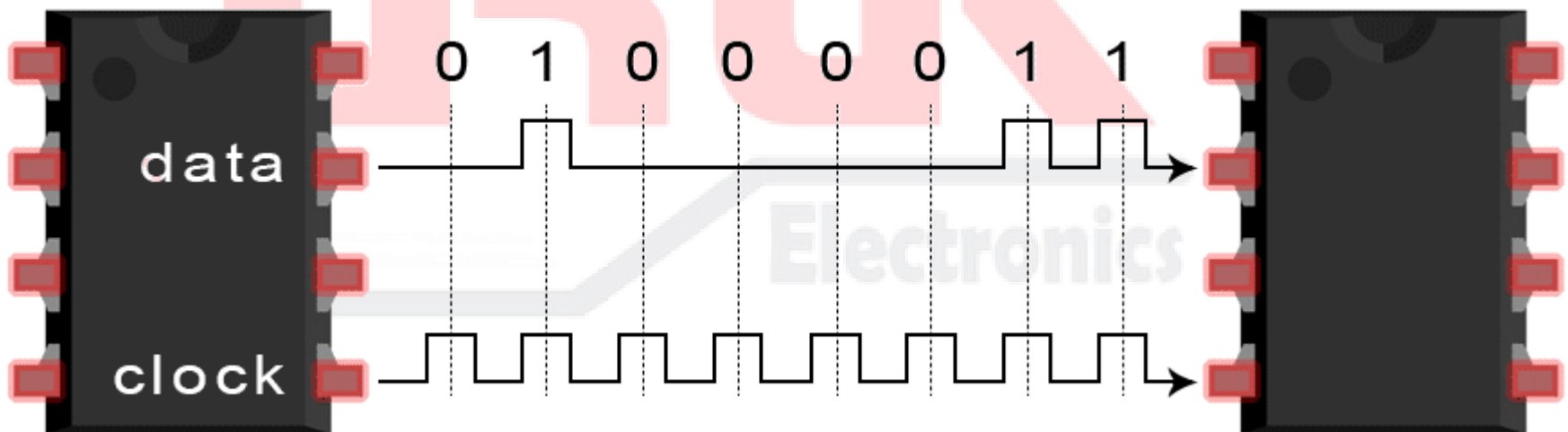
# Parallel Communication

- Parallel communication is the process of sending or receiving multiple bits at the same time through parallel channels.



# Serial Communication

- Serial communication is the process of sending or receiving data in one bit at a time pattern.



# Serial Vs. Parallel

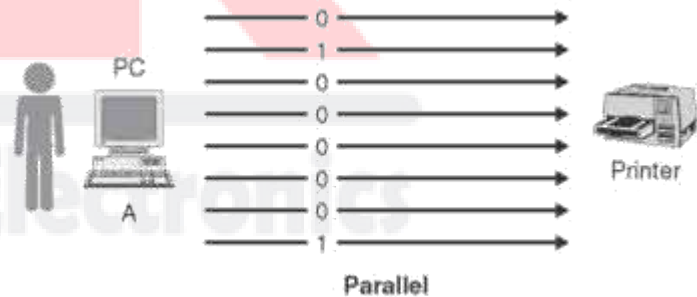
- Serial Communication:

- **One data** bit is transferred at a time.
- **Slower.**
- **Less number of cables** are required to transmit data.



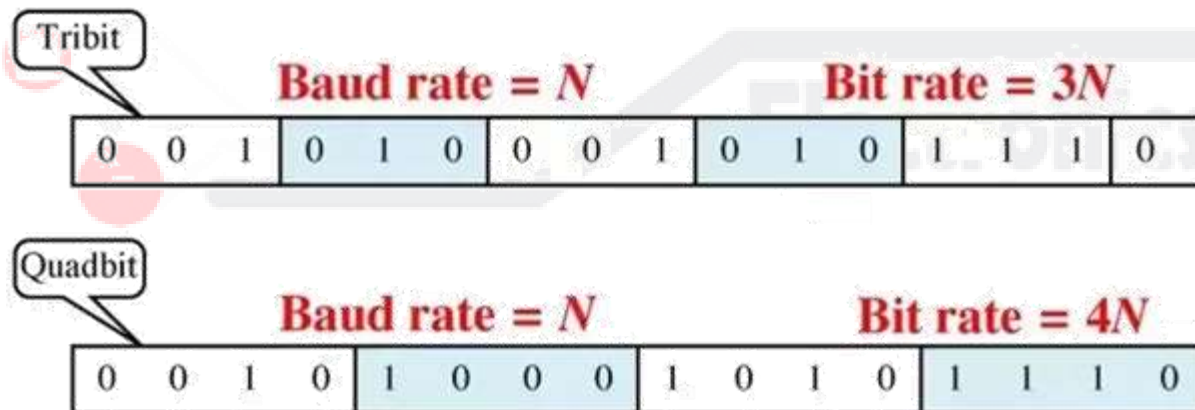
- Parallel Communication:

- **Multiple data** bits are transferred at a time.
- **Faster.**
- **Higher number of cables** are required to transmit data.



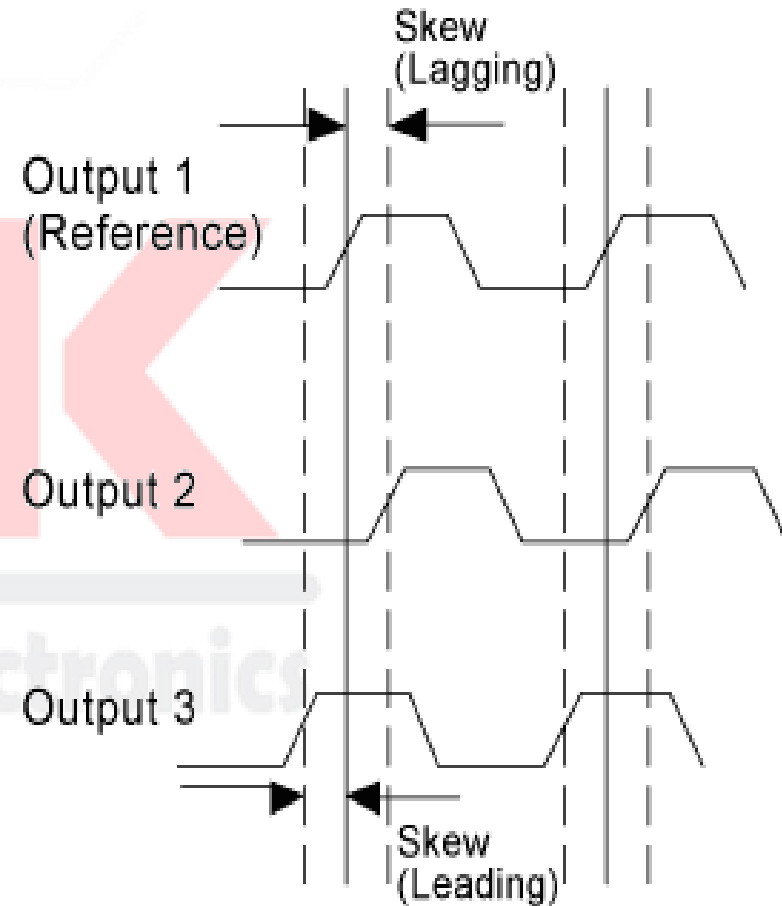
# Baud Rate Vs. Bit Rate

- **Bit Rate:** Number of bits that are transmitted per unit time.
- **Baud rate:** number of signals or symbols changes that occur per second. Where, a symbol is one or several voltages, frequency, or phase setting.
  - For two microcontrollers to communicate serially they should have the same baud rate, else serial communication won't work.
  - Different baud rates are available for use, ex: 2400, 4800, 9600, 19200, 38400, etc.



# Clock Skew

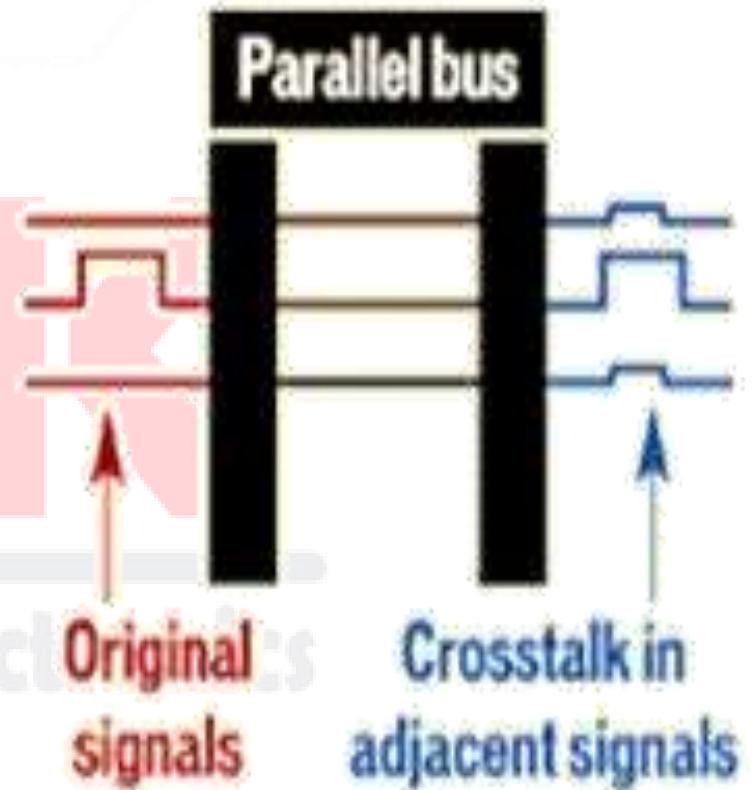
- In parallel circuit, a clock skew is the time difference in the arrival of two sequentially adjacent registers.
- time lag in the data bits through different channels of the same bus.
- It is inevitable due to difference in physical conditions of the channels, like temperature, resistance, path long, etc.





# Cross Talk

- Crosstalk is a phenomenon by which a signal transmitted on one channel of a transmission bus creates an undesired effect in another channel.
- An undesired capacitive, inductive, or conductive coupling is usually what is called crosstalk, from one circuit, part of circuit, or channel to another.



# What limits parallel communication?

- Speed:

Speed of Parallel Link = Bit Rate \* number of channel

But, **clock skew** reduces the speed of every link to the slowest of all of the links.

- Cable Length:

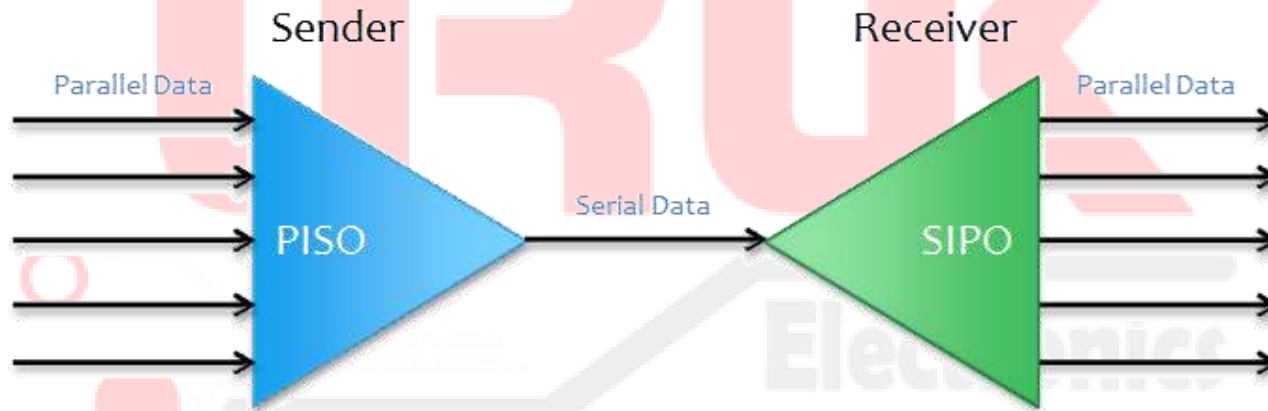
**Crosstalk** creates interference between the parallel lines, and the effect only magnifies with the length of the communication link. This limits the length of the communication cable that can be used.

# Advantages of Serial over Parallel

- Clock skew between different channels is not an issue.
- Fewer connection is required for serial communication, occupies less space, and allows better isolation of the channel from its surroundings.
- Cross talk is not an issue, because there are fewer conductors in proximity.
- Serial is cheaper to implement.
- Many devices and peripherals have serial interface.

# How is data sent serially?

- When a **particular data set is in the microcontroller**, it is **in parallel form**, and any bit can be accessed irrespective of its bit number.
- when **data set is transferred into the output buffer** to be transmitted, it **still in parallel form**.
- The **buffer converts the data into serial data** (Parallel In Serial Out).
- **Data is then transmitted in Serial mode**.
- When **data is received** by another microcontroller in its receiver buffer, **data is to be converted back to its parallel form** using (Serial In Parallel Out)
- Serial data can be transferred in two modes: **Asynchronous** and **Synchronous**.



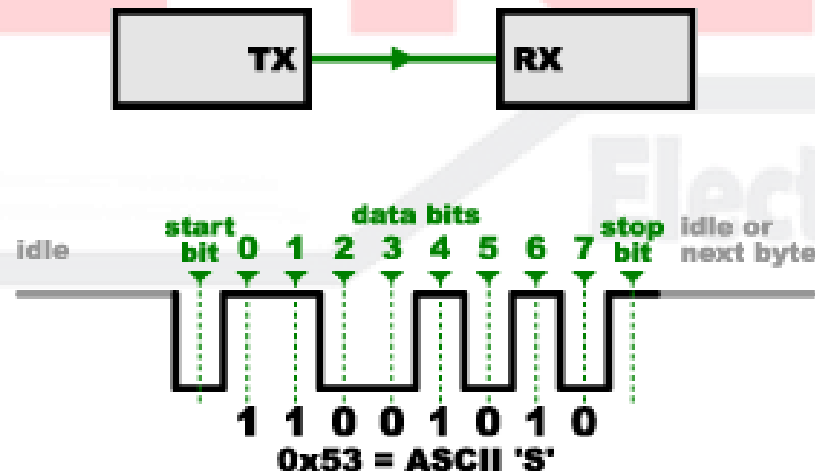
(c) Copyright, Mayank Prasad, 2012  
maxEmbedded.wordpress.com

Data Transfer in  
Serial Communication



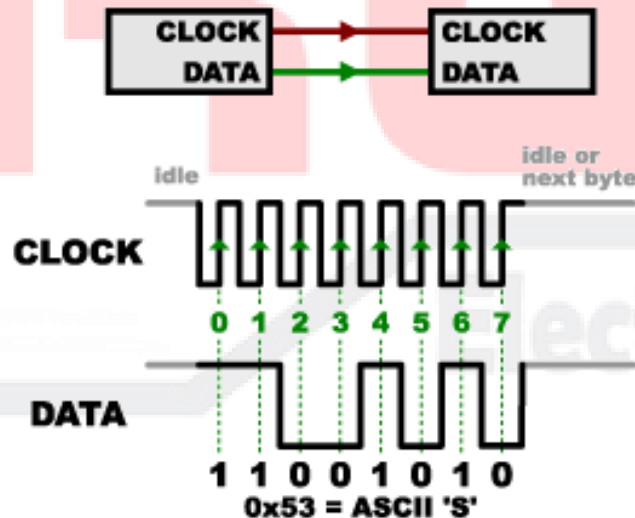
# Asynchronous Data Transfer

- Data transfer is called **Asynchronous** when data bits are “not Synchronized” with a **clock line** (there is **no clock line**).
- **Asynchronous** data transfer has a **protocol**, which is usually as follow:
  - The **first bit is always the START bit**, which signifies the start of communication on the serial line.
  - **DATA bits** (usually **8-bits**).
  - **STOP** bit or bits, which **signals the end of data packet**.
  - **PARITY** bit may be used **just before the STOP bit**



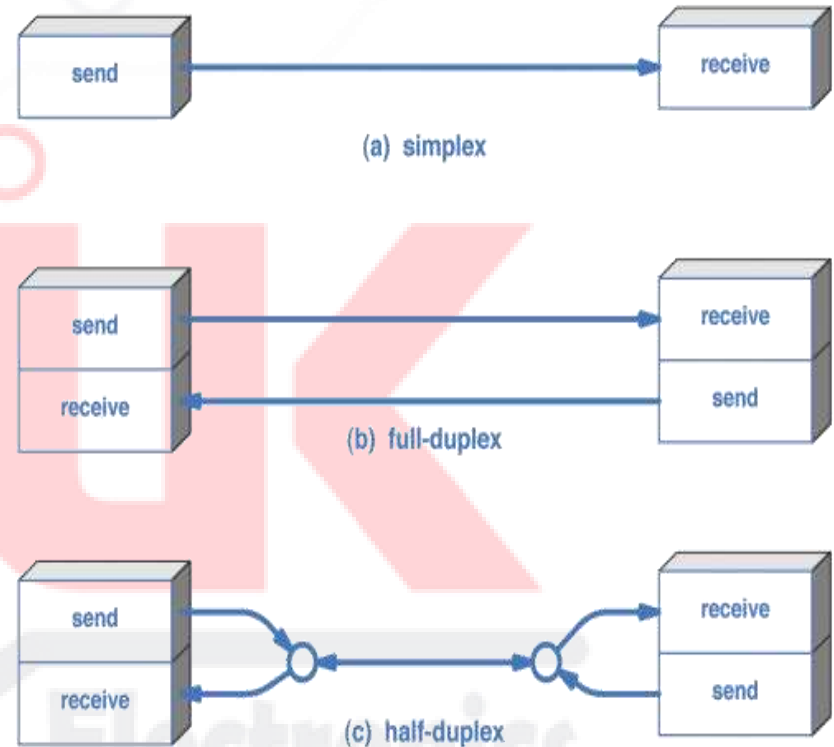
# Synchronous data transfer

- **Synchronous** data transfer is when the **data bits** are “**synchronized**” with **clock pulse**.
- The concept for Synchronous data transfer is as follow:
  - **Data bit sampling** is **done** with respect with **clock pulses**.
  - Since data are sampled depending upon clock pulses, and since the clock sources are very reliable, so there is **much less error in synchronous** as compared to asynchronous.



# Serial communication Terminologies

- **MSB/LSB**: since data is transferred bit-by-bit in serial communication, one needs to know **which bit is sent out first** MSB or LSB.
- **Simplex** Communication:
- **Half-Duplex** Communication:
- **Full-Duplex** Communication:



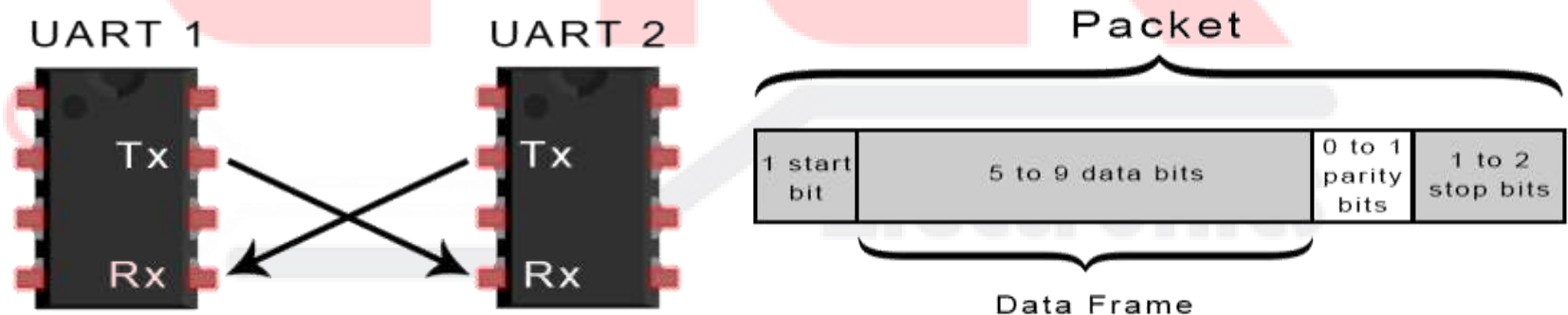
# Serial communication protocols

- A variety of communication protocols have been developed based on serial communication in the past few decades, some of which are:
  - **UART**
  - **SPI** – Serial Peripheral Interface.
  - **I2C** – Inter-integrated Circuit.
  - **USB** – Universal Serial Bus.
  - **RS-232** – Recommended Standard 232.

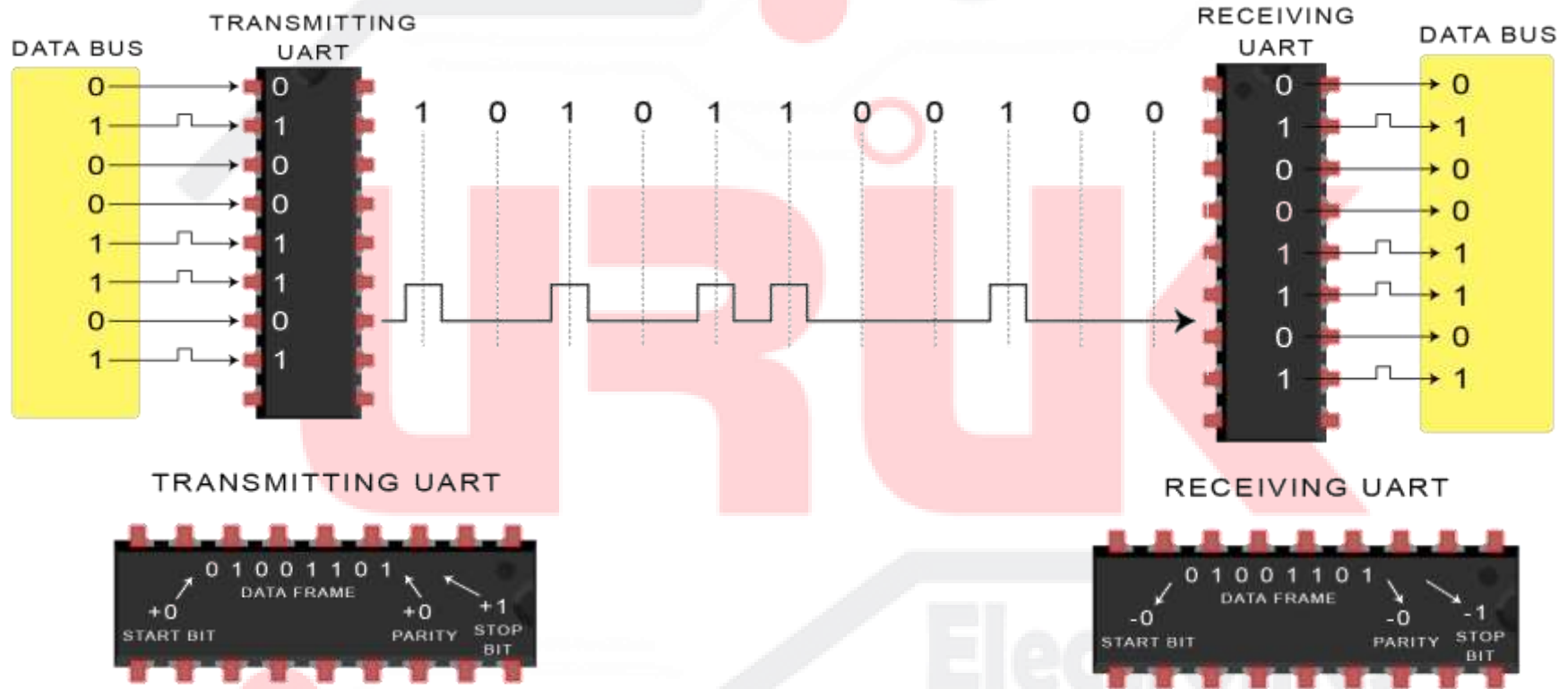


# UART

- **UART** stands for “**U**niversal **A**synchronous **R**eceiver **T**ransmitter”.
- It is basically a **hardware** that convert **parallel data** into **serial data**.
- UART adds specific bit to the data packet instead of the clock signal.

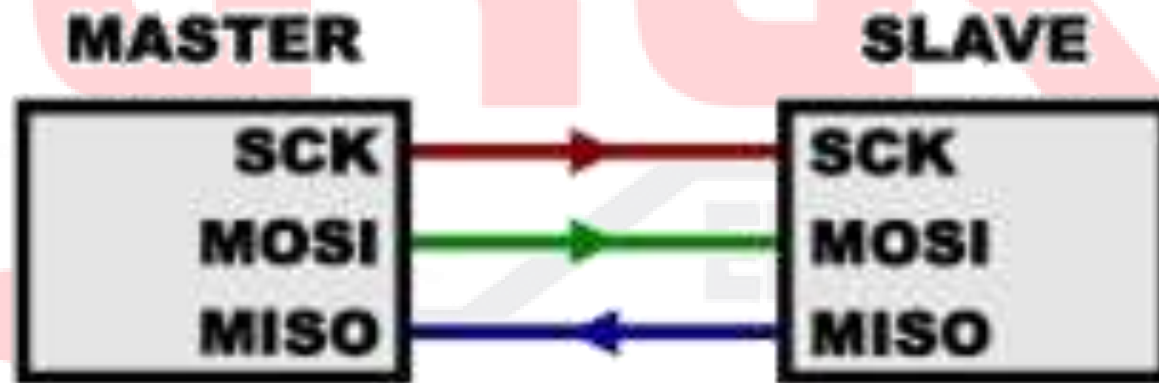


# How UART Works?



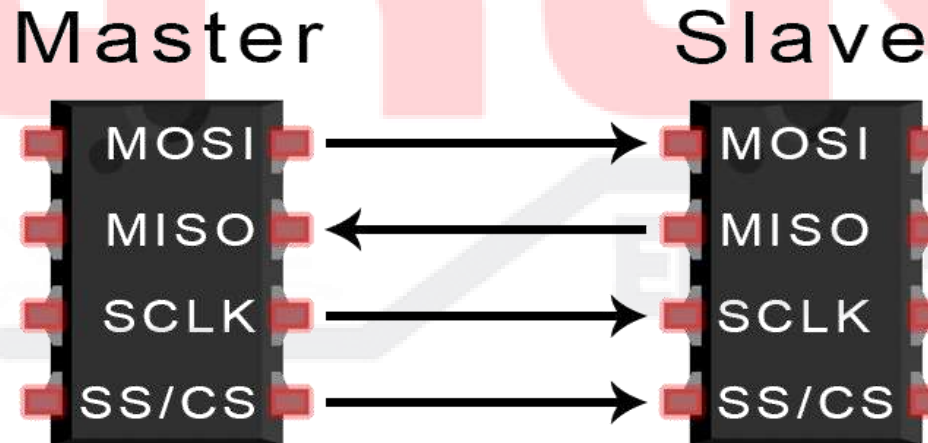
# SPI – Serial Peripheral interface

- Common **Synchronous** communication **protocol** used by many devices. For example: **SD** card, **RFID** card reader, and **2.4GHz** wireless transmitter/receivers.
- Sending data without interruptions.
- **Master/Slave** relationship, where master controls data transfer.

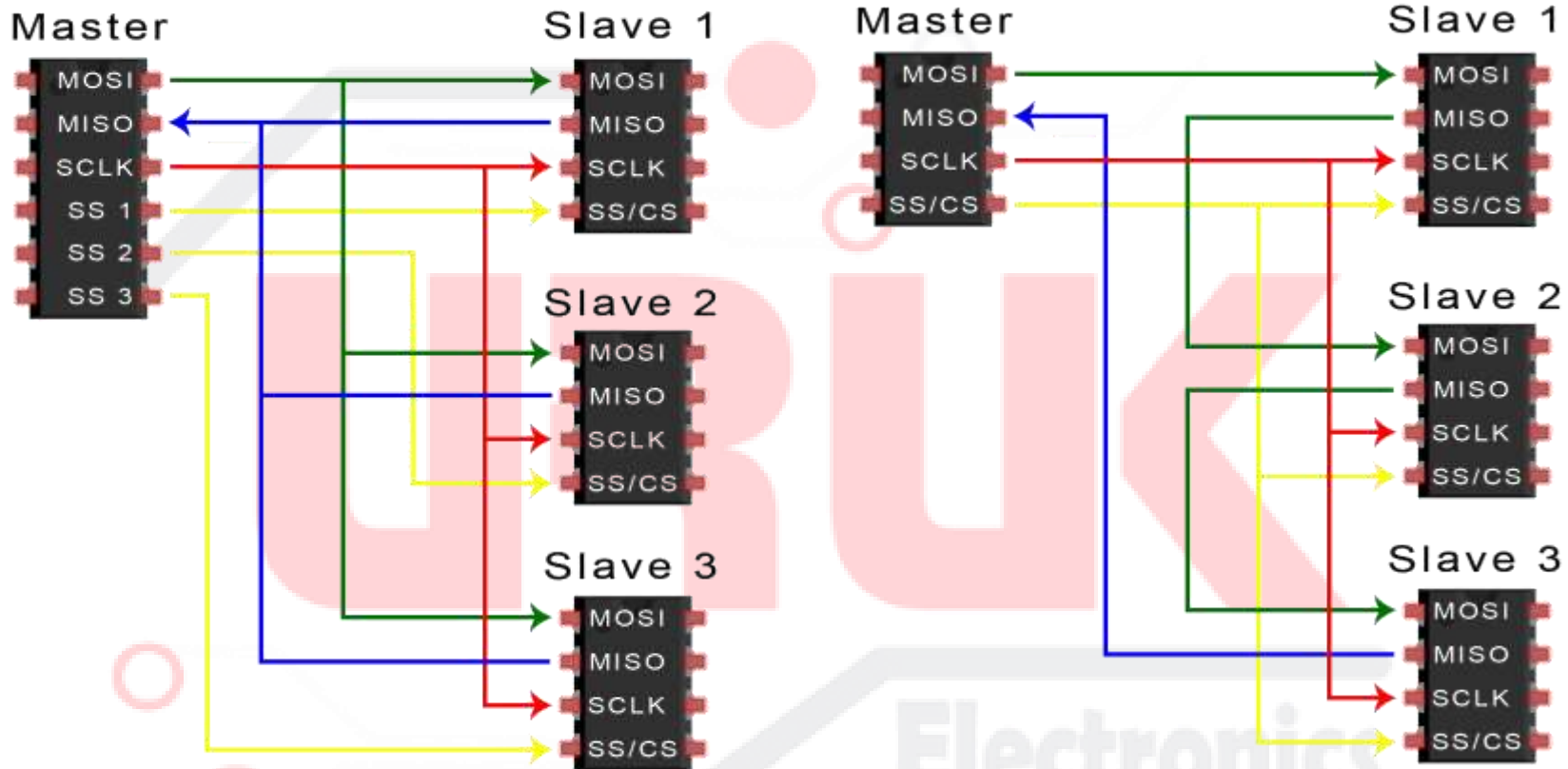


# SPI – Serial Peripheral interface

- Designed to transfer data between various IC chips, at very high speed, hence short bus connections is used to maintain stability.
- Data and control lines of the SPI are:
  - Master Out Slave In (MOSI) also called Serial Data In (SDI).
  - Master In Slave Out (MISO) also Called Serial Data Out (SDO).
  - Serial Clock (SCLK or SCK): generated by the master for Synchronization.
  - Slave Select (SS) on the master and Chip Select (CS) on the slave.



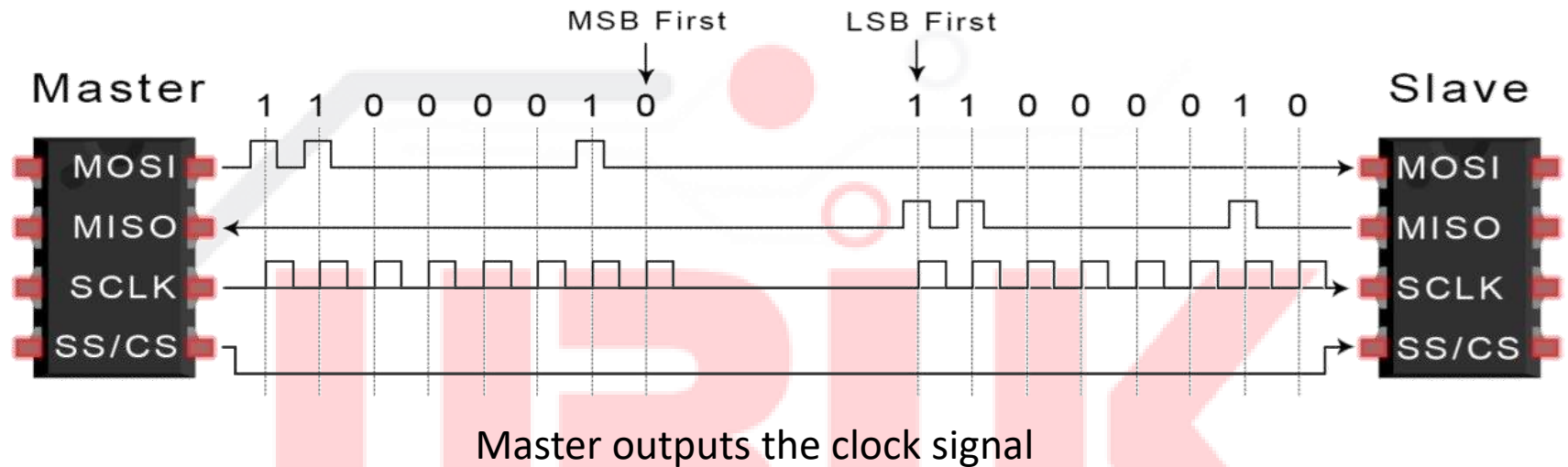
# SPI – Serial Peripheral interface



SPI bus in independent slave configuration

Daisy-Chained SPI Bus

# Steps of SPI Data Transmission



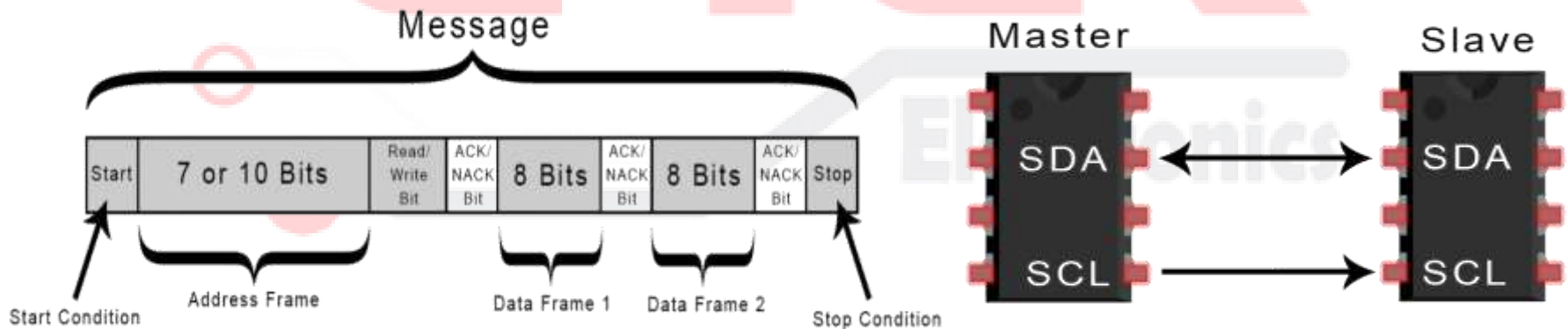
The master switches the SS/CS pin to a low voltage state, which activates the slave

The master sends the data one bit at a time to the slave along the MOSI line, the slave reads the bits as they are received

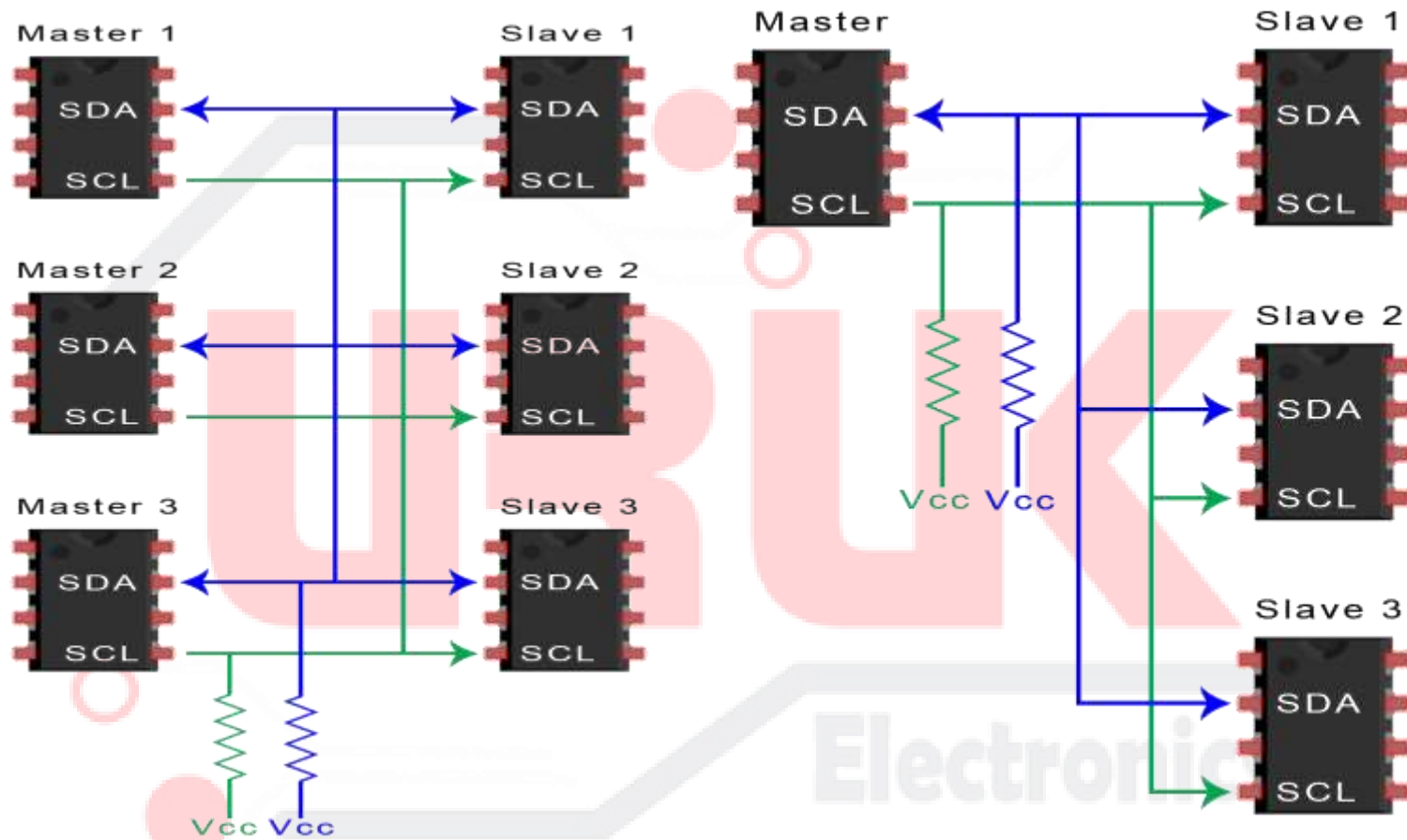
If a response is needed, the slave returns data one bit at a time to the master along the MISO line. The master reads bits as they are received.

# I2C Inter Integrated Circuits bus interface

- Bi-directional 2-wire, Multi Master, Low Bandwidth, and short distance serial communication bus protocol.
- Has a 7-bit address space with 16 reserved address, hence 112 is the maximum number of nodes that can communicate on the same bus.
- The two bi-directional lines are:
  - SDA: Serial Data line for data transfer
  - SCL: Serial Clock line for clock signal and synchronization.
- Typical voltage used are +5V and +3.3V

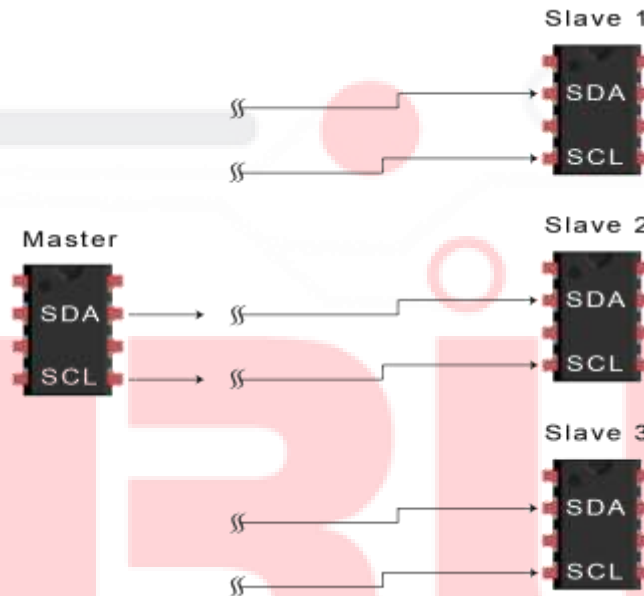


# Masters/Slaves Connections





# How I2C Works?



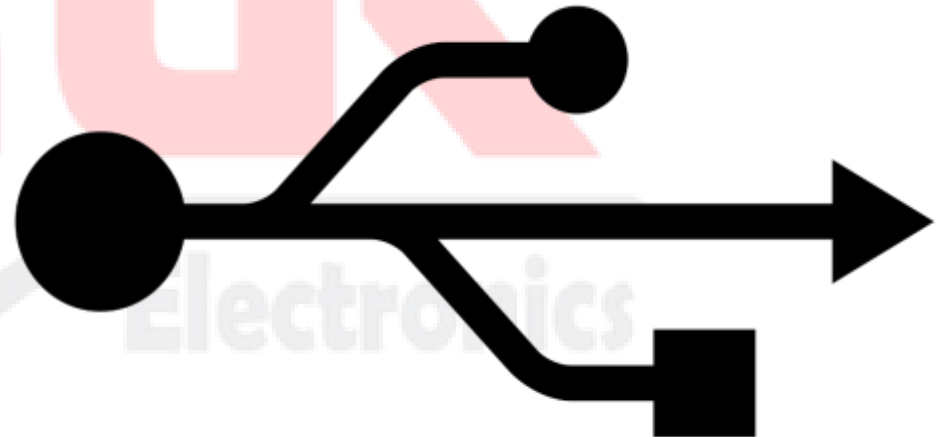
The master sends or receives the data frame:

The master sends the start condition to every connected slave by switching the SDA line from a high voltage level to a low voltage level *before* switching the SCL line from high to low









To stop the data transmission, the master sends a stop condition to the slave by switching SCL high before switching SDA high:

# USB Universal Serial Bus

- It is a set of interfacing specifications for high speed wired communication between electronic systems and peripherals and devices using single standardized interface socket.
- Advantages of USB:
  - Low cost.
  - Expandability.
  - Auto configuration.
  - Hot swapping.
  - Providing power to the bus.



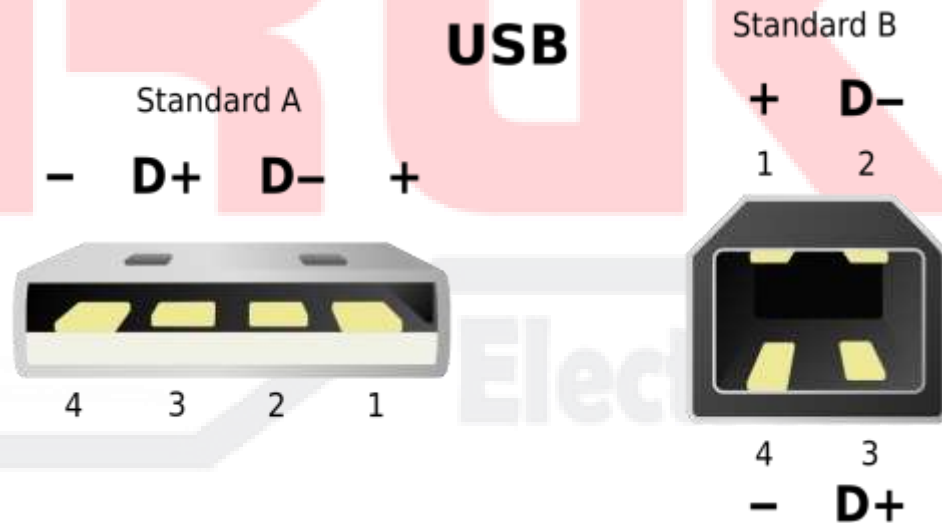
# USB Versions

LOGO	VERSION	SPEED NAME	MAX TRANSFER RATE IN BITS	DEFAULT POWER	CURRENT SYMBOL	TRANSFER TIME FOR SINGLE 25GB FILE
	USB 1.1	LOW SPEED FULL SPEED	1.5 Mbps 12 Mbps	5 Volt 500mA		~9.25 Hours
	USB 2.0	HIGH SPEED	480Mbps	5 Volt 500mA		~14 Minutes
	USB 3.0	SUPER SPEED	5Gbps	5 Volt 900 mA		~70 Seconds
	USB 3.1	SUPER SPEED PLUS	10Gbps	5 Volt 900 mA		~35 Seconds



# USB Connectors and cable lines

- Two types of **connectors** are used within USB:
  - “A” connectors to **upstream** toward the **computer**.
  - “B” connectors for **downstream** and connect to **individual devices**.
- Four **lines** within the USB Cable:
  - +5 volts.
  - GND.
  - +D.
  - -D.

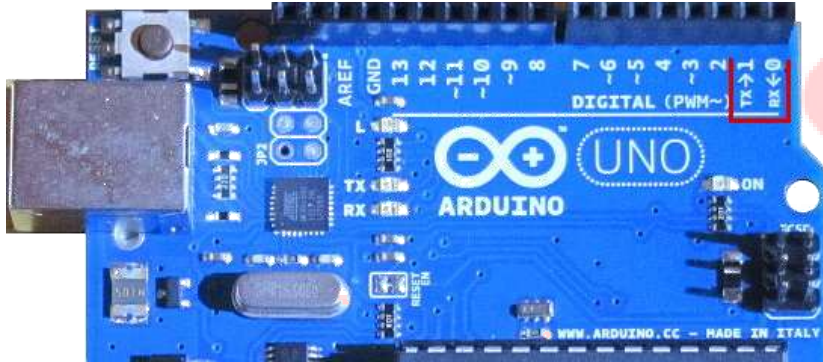


# Serial Communication In Arduino

- All Arduino boards have at least one hardware serial port.
- Some Arduino boards have more than one hardware Serial ports. Like Arduino Due and Arduino Mega.
- The hardware Serial port is connected to the USB port.
- Using the USB ports the Arduino board can exchange data, and indirectly accessing all available resources via a proxy program like Processing.

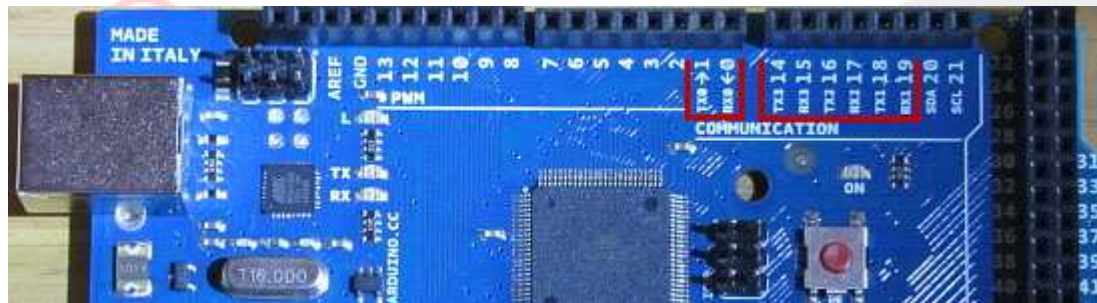
Electronics

# Serial Communication In Arduino



UNO

Due



Mega

# Serial Communication in Arduino

- An **Arduino** board **may** or **may not** contain an **on-board chip** to convert the **hardware serial port** to **USB**.
- A **standard Arduino** has a **single serial port**, but **serial communication** is also **possible** using **software libraries**.
- **Software serial** is **not** as **fast** as the **hardware serial**.

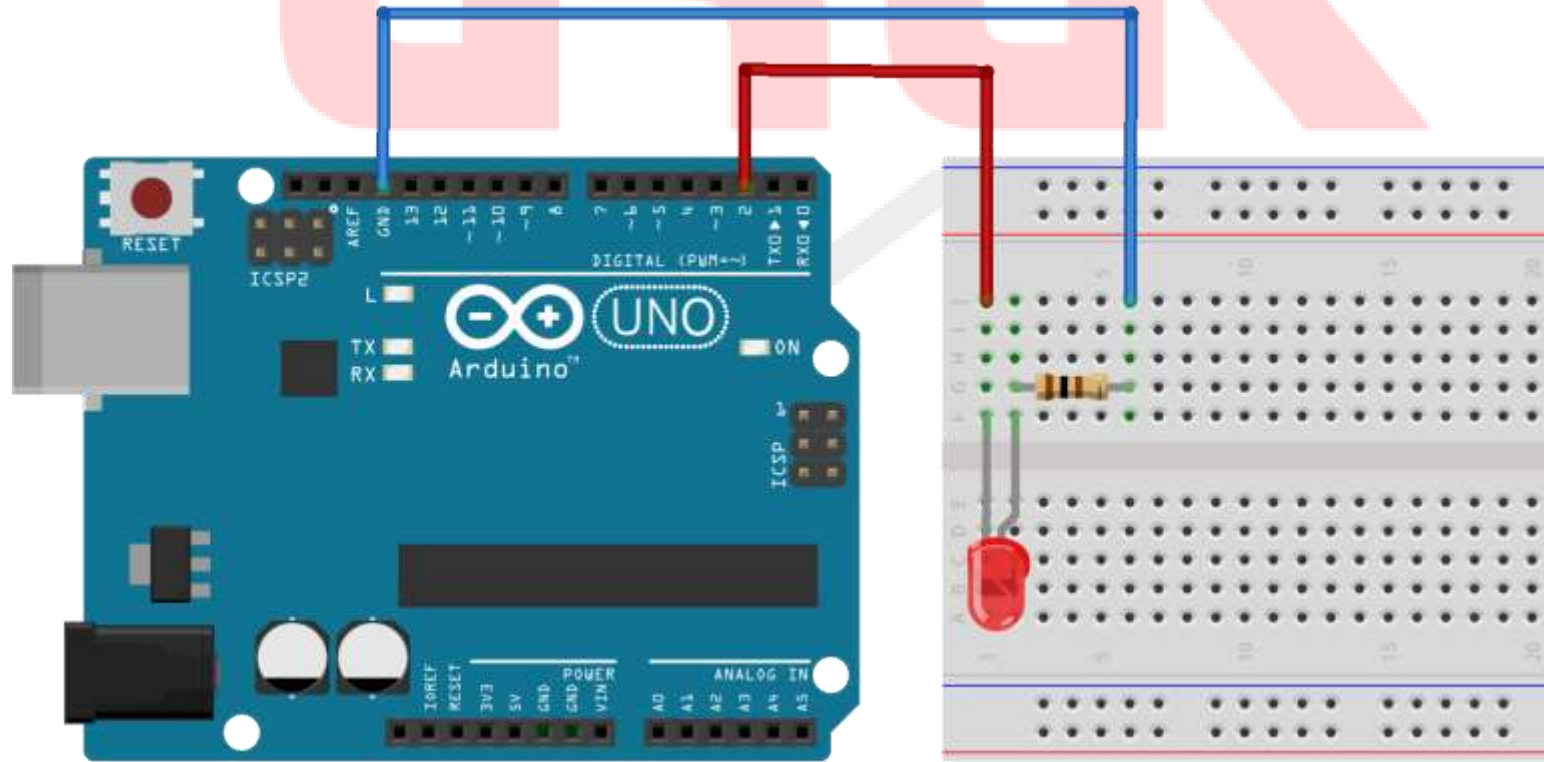
# Talk with Arduino

- Using **Serial Communication** via USB port to establish data exchange between an **Arduino Uno** platform and a **PC**.
- Arduino **Establishes a communication** by sending a **greeting** messages to the PC (**Serial Monitor**).
- Giving number of **options** to the user to choose from.
- **Selecting** an option by **sending data** to the **Arduino** to execute one of the following:
  - **Turning a LED (Pin 13) ON.**
  - **Turning a LED (Pin 13) OFF.**
  - **Welcoming the audience.**

Electronics



# Circuit Diagram



# Initialization Section

```
byte LED = 13;  
String input;
```

assigning digital port 13  
To LED variable

Defining a  
String Variable  
named input

Electronics

# Setup Function

```
void setup() {  
  pinMode(LED, OUTPUT);  
  Serial.begin(9600);  
  Serial.flush();  
}
```

Setting LED as an output port

Establishing a serial communication session with 9600 baud rate.

Emptying the Serial Buffer

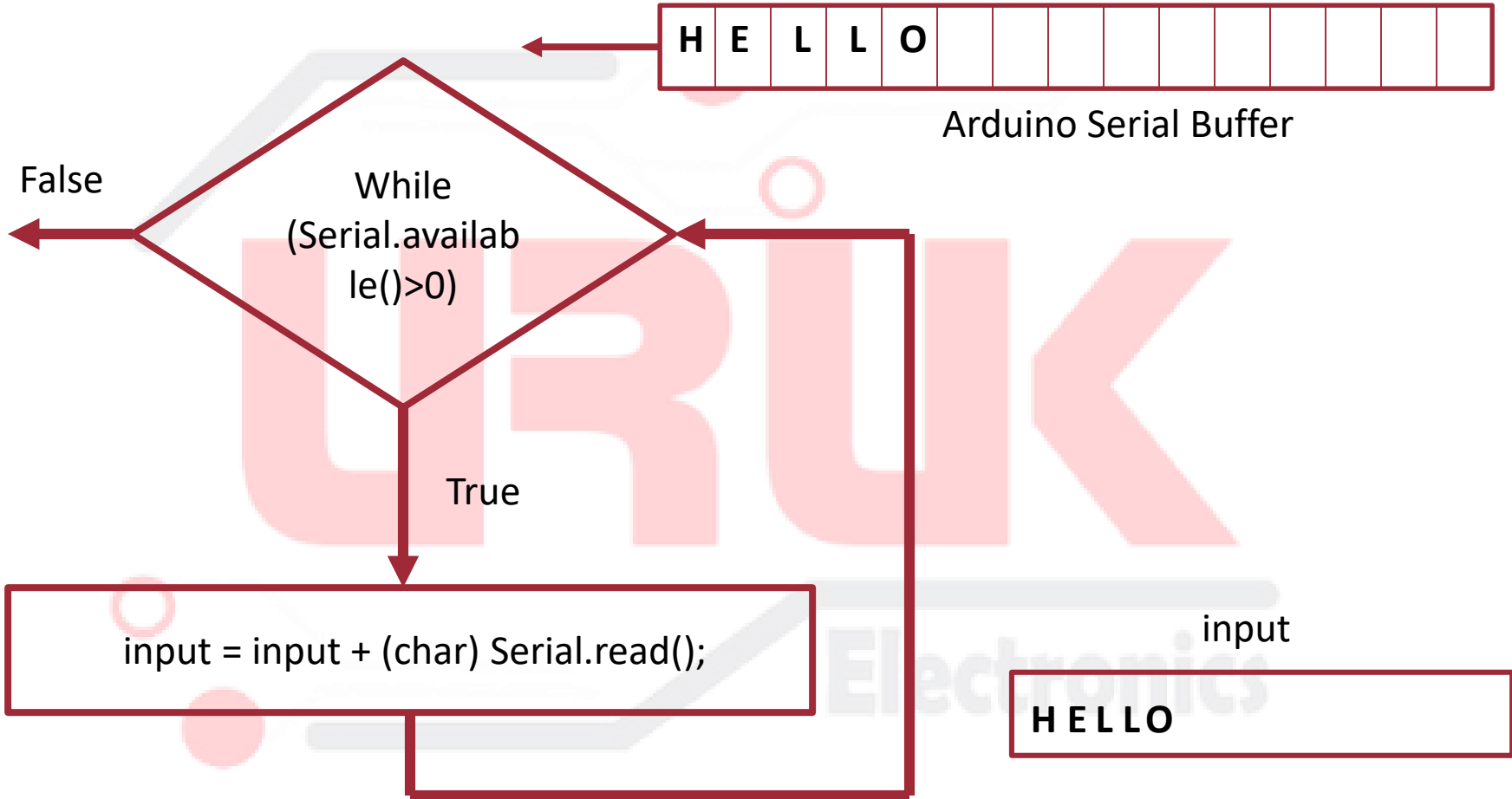
# Loop Function

```
void loop() {  
  
    Serial.println("Hello Sir,");  
    Serial.println("What can I do fro you, Sir?");  
    Serial.println("Write (Hello Arduino) to Start Short Conversation");  
    Serial.println("Write (LED ON) to turn on an LED");  
    Serial.println("Write (LED OFF) to turn off an LED");  
  
    while(Serial.available()>0)  
    {  
        input = input + (char) Serial.read();  
        delay(5);  
    }  
    if (input == "LED ON")  
    {  
        digitalWrite(LED, HIGH);  
    }  
    if (input == "LED OFF")  
    {  
        digitalWrite(LED, LOW);  
    }  
    if (input == "Hello Arduino")  
    {  
        talking();  
    }  
    input = "";  
}
```

Electronics



# Receiving a String procedure



**Thank you ...**

**Q&A**