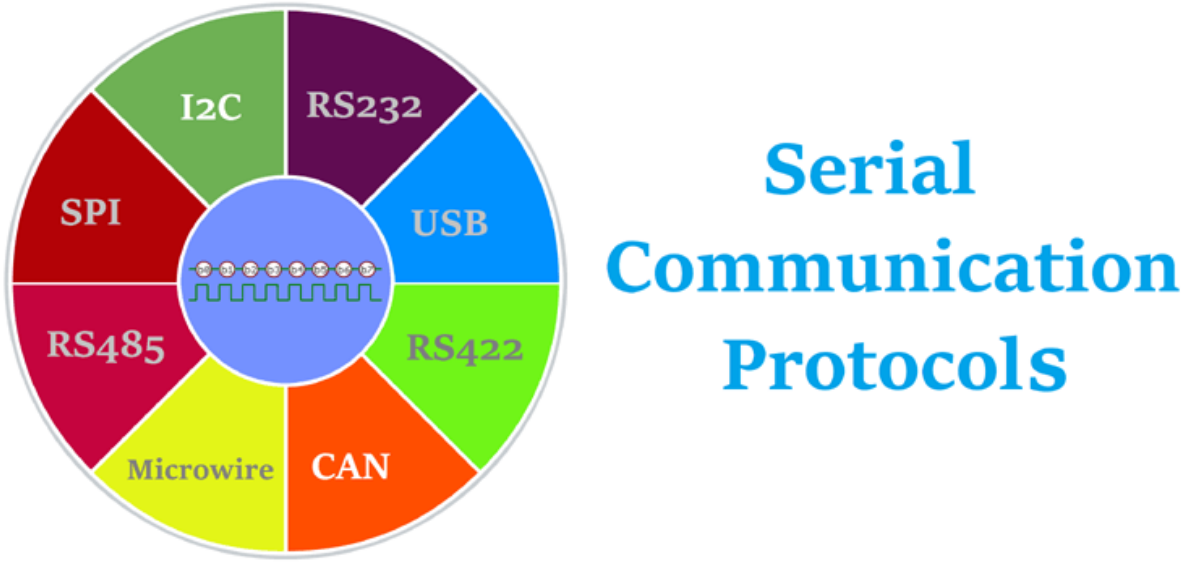
Serial Communication Protocols

By Abhiemanyu Pandit Apr 29, 2019

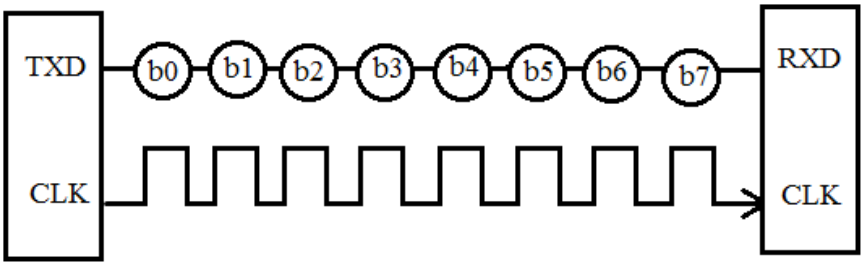


Serial Communication Protocols

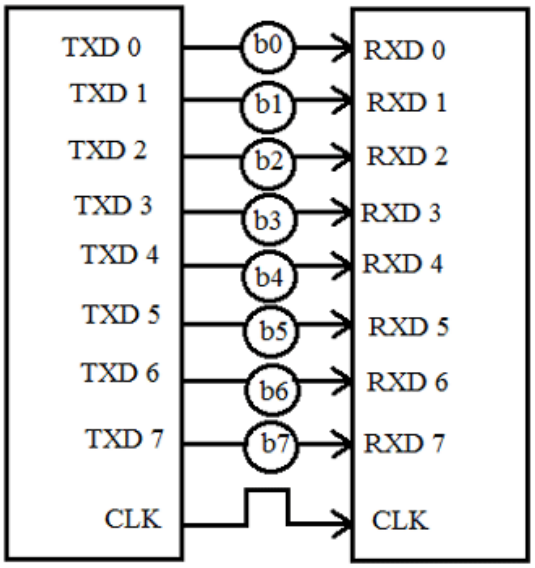
Before starting with Serial Communication Protocols, Let's break the terminology in three parts. The **communication** is very well known terminology which involves the exchange of information between two or more mediums. In embedded systems, the communication means the exchange of data between two microcontrollers in the form of bits. This exchange of data bits in microcontroller is done by some set of defined rules known as **communication protocols**. Now if the data is sent in **series** i.e. one after the other then the communication protocol is known as **Serial Communication Protocol**. More specifically, the data bits are transmitted one at a time in sequential manner over the data bus or communication channel in Serial Communication.

Types of Communication Protocols

There are different types of data transfer available in the digital electronics such as serial communication and parallel communication. Similarly the protocols are divided into two types such as **Serial Communication Protocol and Parallel Communication Protocols**. Examples of Parallel Communication Protocols are ISA, ATA, SCSI, PCI and IEEE-488. Similarly there are several examples of Serial Communication Protocols such as CAN, ETHERNET, I2C, SPI, RS232, USB, 1-Wire, and SATA etc.



Serial Communication Protocol



Parallel Communication Protocol

In this article, the **different types of Serial Communication Protocols** will be discussed. Serial communication is the most widely used approach to transfer information between data processing peripherals. Every electronics device whether it is Personal Computer (PC) or Mobile runs on serial communication. The protocol is the secure and reliable form of communication having a set of rules addressed by the source host (sender) and destination host (receiver) similar to parallel communication.

Transmission Modes in Serial Communication

As already said above that in serial communication data is sent in the form of bits i.e. binary pulses and it is well known that, binary one represents the logic HIGH and zero represents the logic LOW. There are several types of serial communication depending on the type of transmission mode and data transfer. The transmission modes are classified as Simplex, Half Duplex and Full Duplex.

Simplex Method:

In simplex method either of the medium i.e sender or receiver can be active at a time. So if the sender is transmitting the data then receiver can only accept and vice versa. So simplex method is **one-way communication** technique. The well-known examples of simplex method are Television and Radio.

Half Duplex Method:

In half duplex method both sender and receiver can be active but not at the same time. So if the sender is transmitting then receiver can accept but cannot send and similarly vice versa. The well-known examples of the half duplex is the internet where the user sends a request for a data and the gets it from server.

Full Duplex Method:

In full duplex method, both receiver and transmitter can send data to each other at the same time. The well-known example is mobile phone.

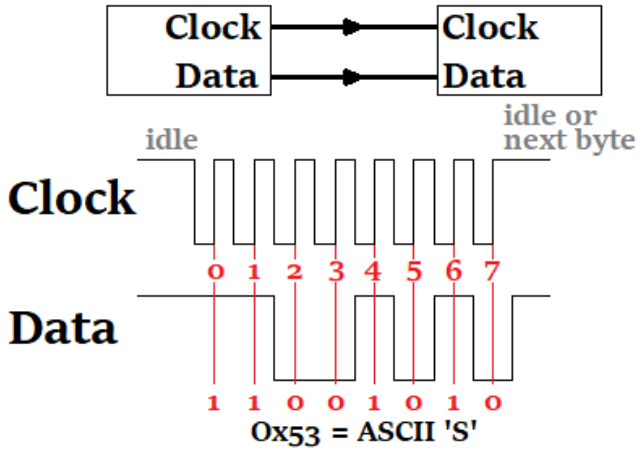
Apart from this, for appropriate data transmission, the clock plays important role and it is one of the primary source. Malfunction of the clock results in unexpected data transmission even sometimes data loss. So, the clock synchronisation becomes very important when using serial communication.

Clock Synchronization

The clock is different for serial devices and it is classified in two type viz. Synchronous Serial Interface and Asynchronous Serial Interface.

Synchronous Serial Interface:

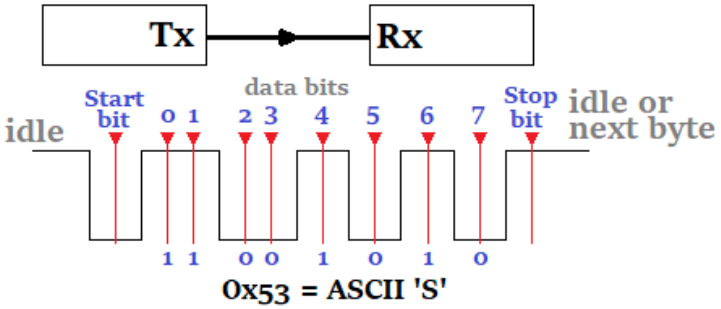
It is a point-to-point connection from a master to slave. In this type of interface, all the devices use single CPU bus to share data and clock. The data transmission becomes faster with same bus to share clock and data. Also there is no mismatch in baud rate in this interface. In transmitter side, there is a shift of the data onto serial line providing the clock as a separate signal as there is no start, stop and parity bits are added to data. In receiver side, the data is being extract using the clock provided by the transmitter and converts the serial data back to the parallel form. The well-known examples are I2C and SPI.



A Separate Clock Line is need in Synchronous Communication

Asynchronous Serial Interface:

In asynchronous Serial Interface, the external clock signal is absent. The Asynchronous Serial Interfaces can be seen in mostly in long distance applications and are a perfect fit for the stable communication. In asynchronous Serial Interface the absence of external Clock Source makes it rely on several parameters such as Data Flow Control, Error Control, Baud Rate Control, Transmission Control and Reception Control. On the **transmitter side**, there is a shifting of parallel data onto the serial line using its own clock. Also it adds the start, stop and parity check bits. On the receiver side, the receiver extracts the data using its own clock and convert the serial data back to the parallel form after stripping off the start, stop, and parity bits. The well-known examples are RS-232, RS-422 and RS-485.



No Separate Clock Line is need in Asynchronous Communication

Other Terms Related to Serial Communication

Apart from Clock Synchronization there are certain things to remember when transferring data serially such as Baud Rate, Data bit selection (Framing), Synchronisation and error checking. Let's discuss these terms in brief.

Baud Rate: Baud rate is rate at which the data is transferred between the transmitter and receiver in the form of bits per second (bps). The most commonly used baud rate is 9600. But there are other selection of baud rate such as 1200, 2400, 4800, 57600, 115200. The more the baud rate will be fats the data will be transferred at a time. Also for the data communication the baud rate has to be same for both transmitter and receiver.

Framing: Framing is referred to the number of data bits to be sent from transmitter to receiver. The number of data bits differs in case of application. Most of the application uses 8 bits as the standard data bits but it can be selected as 5, 6 or 7 bits also.

Synchronisation: Synchronization Bits are important to select a chunk of data. It tells the start and end of the data bits. The transmitter will set start and stop bits to the data frame and the receiver will identify it accordingly and do the further processing.

Error Control: The error control plays an important role while serial communication as there are many factors which affects and adds the noise in the serial communication. To get rid of this error the parity bits are used where parity will check for even and odd parity. So if the data frame contains the even number of 1's then it is known as even parity and the parity bit in the register is set to 1. Similarly if the data frame contains odd number of 1's then it is known as odd parity and clears the odd parity bit in the register.

Protocol is just like a common language that system uses to understand the data. As described above, the serial communication protocol is divided into types i.e. Synchronous and Asynchronous. Now the both will be discuss in detail.

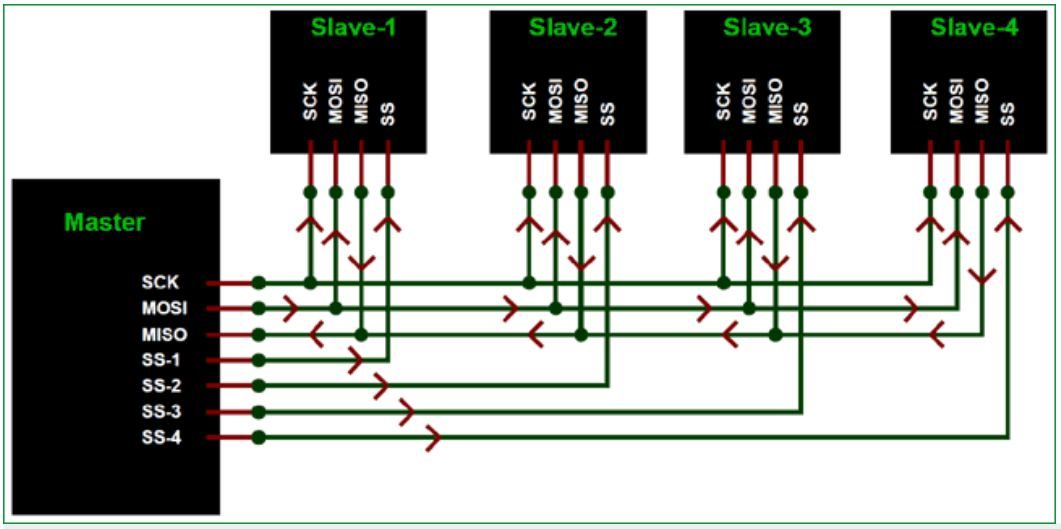
Synchronous Serial Protocols

The **synchronous type of serial protocols such as SPI, I2C, CAN and LIN** are used in different projects because it is one of the best resources for onboard peripherals. Also these are the widely used protocols in major applications.

SPI Protocol

The Serial Peripheral Interface (SPI) is a synchronous interface which allows several SPI microcontrollers to be interconnected. In SPI, separate wires are required for data and clock line. Also the clock is not included in the data stream and must be furnished as a separate signal. The SPI may be configured either as master or as a slave. The four basic SPI signals (MISO, MOSI, SCK and SS),

Vcc and Ground are the part of data communication. So it needs 6 wires to send and receive data from slave or master. Theoretically, the SPI can have unlimited number of slaves. The data communication is configured in SPI registers. The SPI can deliver up to 10Mbps of speed and is ideal for high speed data communication.

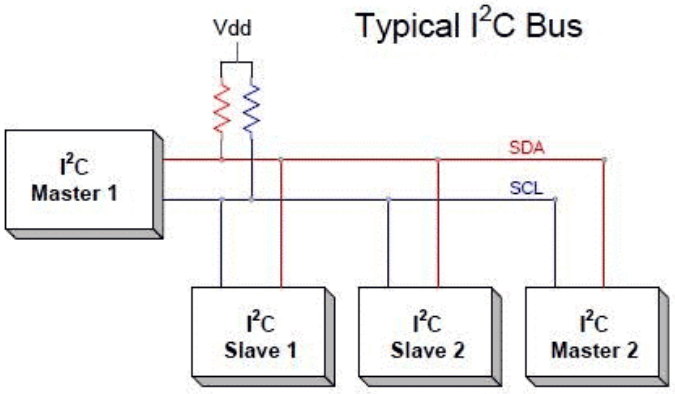


SPI Communication

Most of the microcontrollers have inbuilt support for SPI and can be directly connected SPI supported device:

- SPI Communication with PIC Microcontroller PIC16F877A
- How to Use SPI Communication in STM32 Microcontroller
- How to use SPI in Arduino: Communication between two Arduino Boards

I2C Serial Communication



I2C Communication

Inter integrated circuit (I2C) two-line communication between different ICs or modules where two lines are SDA (Serial Data Line) and SCL (Serial Clock Line). Both the lines must be connected to a positive supply using a pull up resistor. I2C can deliver speed up to 400Kbps and it uses 10 bit or 7 bit addressing system to target a specific device on the i2c bus so it can connect up to 1024 devices. It has limited length communication and is ideal for onboard communication. I2C networks are easy to setup since it uses only two wires and new devices can simply be connected to the two common I2C bus lines. Same like SPI, microcontroller generally have I2C pins to connect any I2C device:

- How to use I2C Communication in STM32 Microcontroller
- I2C Communication with PIC Microcontroller PIC16F877
- How to use I2C in Arduino: Communication between two Arduino Boards

USB

USB (Universal Serial Bus) is widely protocol with different versions and speeds. A maximum of 127 peripherals can be connected to a single USB host controller. USB acts as "plug and play" device. The USB are used in almost devices such as keyboards, printers, media devices, cameras, scanners and mouse. It is designed for easy installation, faster data rated, less cabling and hot swapping. It has replaced the bulkier and slower serial and parallel ports. USB uses differential signalling to reduce interference and allow high-speed transmission over a long distance.

A differential bus is built with two wire, one of represents the transmitted data and the other its complement. The idea is that the 'average' voltage on the wires does not carry any information, resulting in less interference. In USB, the devices are allowed to draw a certain amount of power without asking the host. USB uses only two wires to for data transfer and are faster than the serial and

parallel interface. USB versions supports different speeds such as 1.5Mbps (USB v1.0), 480 Mbps (USB2.0), 5Gbps (USB v3.0). Length of individual USB cable can reach up to 5 meters without a hub and 40 meters with hub.

CAN

The Controller Area Network (CAN) is used in e.g. automotive to allow communication between ECUs (Engine Control Units) and sensors. The CAN protocol is robust, low-cost and message based and covers in many applications - e.g. cars, trucks, tractors, industrial robots. The CAN bus system allows for central error diagnosis and configuration across all ECUs. CAN messages are prioritized via IDs so that the highest priority IDs are non-interrupted. Each ECU contains a chip for receiving all transmitted messages, decide relevance and act accordingly - this allows easy modification and inclusion of additional nodes (e.g. CAN bus data loggers). The applications include start/stop of vehicles, collision avoidance systems. The CAN bus systems can provide speed up to 1Mbps.

Microwire

MICROWIRE is a 3Mbps [full-duplex] serial 3-wire interface essentially a subset of the SPI interface. Microwire is a serial I/O port on microcontrollers, so the Microwire bus will also be found on EEPROMs and other Peripheral chips. The 3 lines are SI (Serial Input), SO (SerialOutput) and SK(Serial Clock). The Serial Input (SI) line to the microcontroller, SO is the serial output line, and SK is the serial clock line. Data is shifted out on the falling edge of SK, and is valued on the rising edge. SI is shifted in on the rising edge of SK. An additional bus enhancement to MICROWIRE is called MICROWIRE/Plus. The main difference between the two buses seems to be that MICROWIRE/Plus architecture within the microcontroller is more complex. It supports speeds up to 3Mbps.

Asynchronous Serial Protocols

The asynchronous type of serial protocols are very essential when it comes to longer distance reliable data transfer. **Asynchronous communication does not require a timing clock** that is common to both devices. Each device independently listens and sends digital pulses that represent bits of data at an agreed-upon rate. Asynchronous serial communication is sometimes referred to as Transistor-Transistor Logic (TTL) serial, where the high voltage level is logic 1, and the low voltage equates to logic 0. Almost every microcontroller on the market today has at least one Universal Asynchronous Receiver-Transmitter (UART) for serial communication. The examples are RS232, RS422, RS485 etc.

RS232

The RS232 (Recommended Standard 232) is very common protocol used to connect different peripherals such as Monitors, CNCs etc. The RS232 comes in male and female connectors. The RS232 is point-to-point topology with maximum one device connected and covers distance up to 15 meters at 9600 bps. Information on the RS-232 interface is transmitted digitally by logical 0 and 1. The logical "1" (MARK) corresponds to a voltage in the range from -3 to -15 V. The logical "0" (SPACE) corresponds to a voltage in the range from +3 to +15 V. It comes in DB9 connector which has 9 pinouts such as TxD, RxD, RTS, CTS, DTR, DSR, DCD, GND.

RS422

The RS422 is similar to RS232 which allows to simultaneously send and receive messages on separate lines but uses a differential signal for this. In the RS-422 network, there can only be one transmitting device and up to 10 receiving devices. The data transfer speed in RS-422 depends on the distance and can vary from 10 kbps (1200 meters) to 10 Mbps (10 meters). The RS-422 line is 4 wires for data transmission (2 twisted wires for transmission and 2 twisted wires for receiving) and one common GND ground wire. The voltage on the data lines can be in the range from -6 V to +6 V. The logical difference between A and B is greater than +0.2 V. Logical 1 corresponds to the difference between A and B less than -0.2 V. The RS-422 standard does not define a specific type of connector, usually it can be a terminal block or a DB9 connector.

RS485

Since RS485 uses multi-point topology, it is most used in the industries and are industry preferred protocol. RS422 can connect 32 line drivers and 32 receivers in a differential configurations but with the help of additional repeaters and signal amplifiers up to 256 devices. The RS-485 does not define a specific type of connector, but it is often a terminal block or a DB9 connector. The speed of operation also depends on the length of the line and can reach 10 Mbit / s at 10 meters. The voltage on the lines is in the range from -7 V to +12 V. There are two types of RS-485 such as half duplex mode RS-485 with 2 contacts and full duplex mode RS-485 with 4 contacts. To learn more about using RS485 with other microcontrollers, check the links:

- RS-485 MODBUS Serial Communication using Arduino UNO as Slave
- RS-485 Serial Communication between Raspberry Pi and Arduino Uno
- RS485 Serial Communication between Arduino Uno and Arduino Nano
- Serial Communication between STM32F103C8 and Arduino UNO using RS-485

Conclusion

Serial Communication is one of the widely used communication interface systems in electronics and embedded systems. The data rates can be different for different applications. The Serial Communication Protocols can play decisive role when dealing in this kind of applications. So choosing the right Serial protocol becomes very important.