IEE 2463
Programmable Electronic Systems

Standard Communication Protocols

Electrical Engineering Department
Pontificia Universidad Católica de Chile
peclab.ing.uc.cl

Lecture Goals

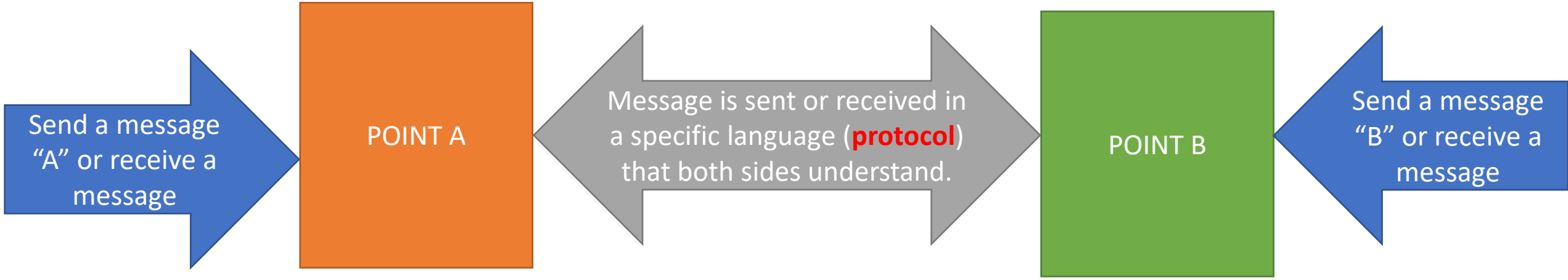
- To identify the hardware and software used in communication protocols.
- To identify the importance of hardware in the implementation of communication protocols.
- To understand the UART communication protocol.
- To understand the SPI communication protocol.
- To understand the I2C communication protocol.

Introduction
Communication Protocols

This section provides a brief overview of different communication protocols.

Communication Protocols

Introduction



Data: Information needs to be sent or received.

Driver: Takes the information and mask it in a specific Protocol to make it understandable for other drivers.

Bus: Is through where the information travels, it can be one or several wires, each communication protocol define its own number of wires, i.e. the bus architecture.

Communication Protocols

Introduction

Here is where the information is created (**software-operating system (OS)**). After thinking (or processing information), we are ready to send the data to our hardware. To do that, we need a **driver.**

This is our **hardware** to receive messages. It uses a **driver** (neural connections) to allow the software (brain) to see the data, expect the information based on the same protocol (lenguaje) and is connected to the same **bus.**

Hardware: A human uses the mouth and tonge as **hardware** to send a message.

Driver: This is an intermediate layer beween the brain (software OS) and the mouth (hardware) that allow to pass the information. In this case, represented by the neural connections between brain and mouth that make posible to move your hardware. Without driver, you can have a perfect data, and a perfect hardware, but information can not be passed.

Protocol: The protocol used is not only the lenguaje, it is also the rules involved, e.g only one person talk at a time. That represents the **protocol**.

Bus: Sound is a mechanical wave that travels through the air.

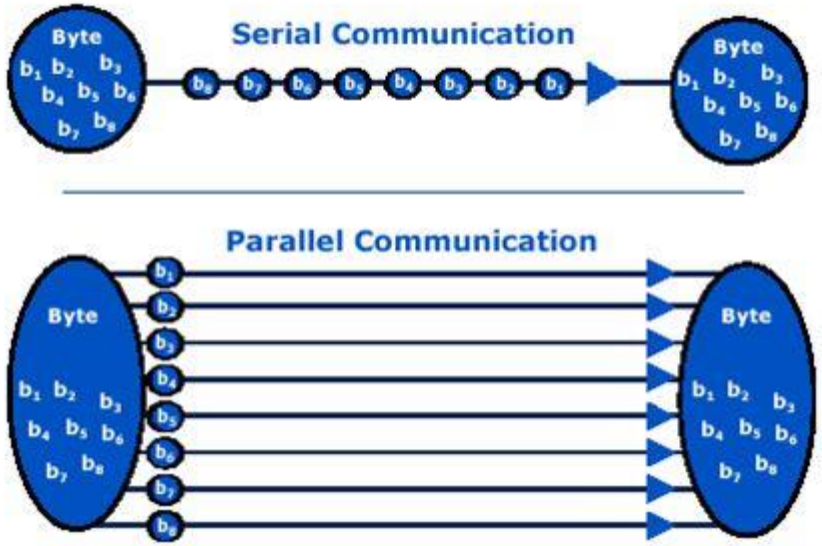
Communication Protocols

Introduction

We use several communication protocols in daily life. Like:

- USB (wired)
- Wifi (wireless)
- TCP/IP (wired)
- Bluetooth (wireless)

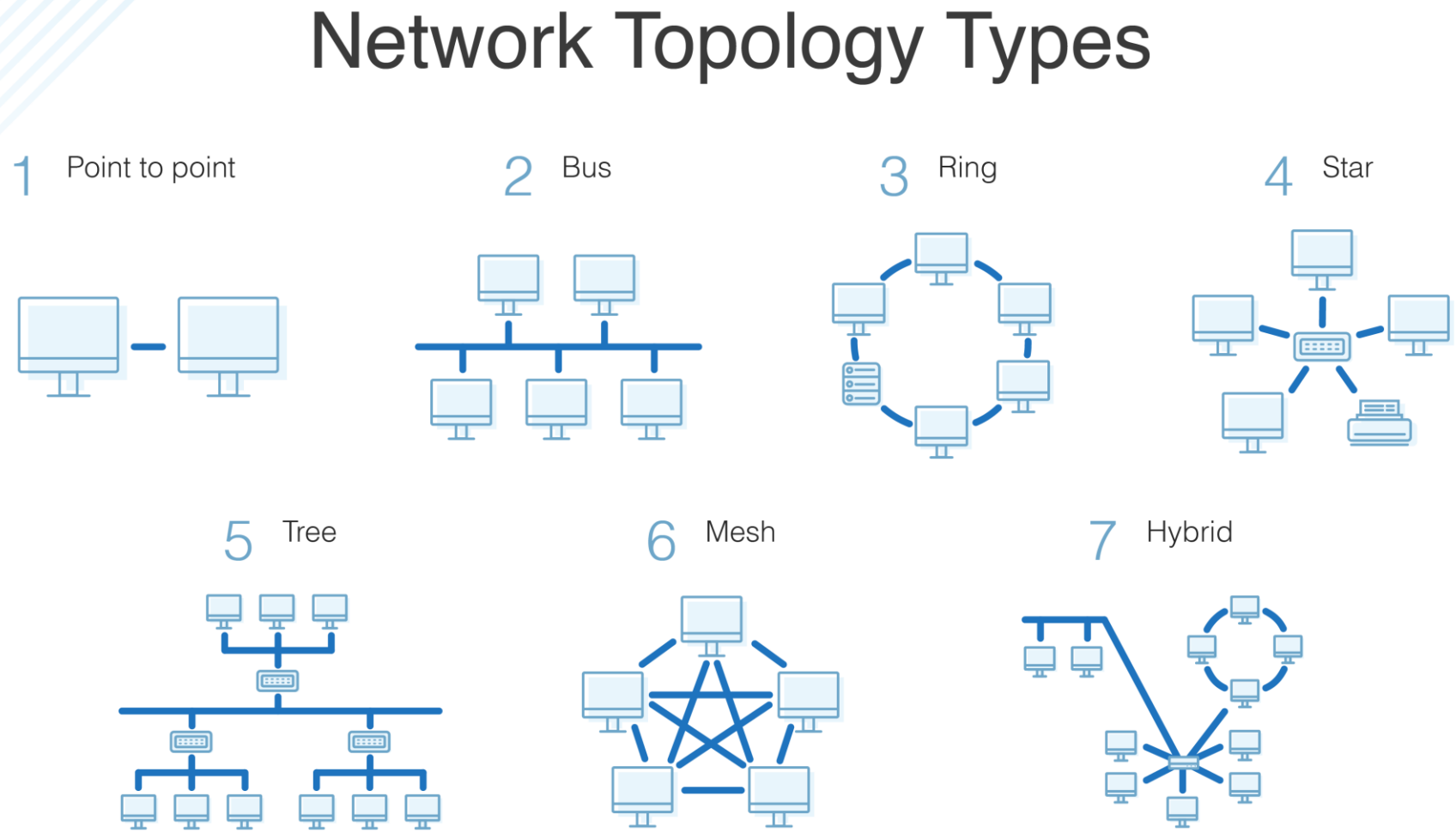
Serial Communication Protocols



(Reference: microcontrollerpicavr.blogspot.in)

Communication Protocols

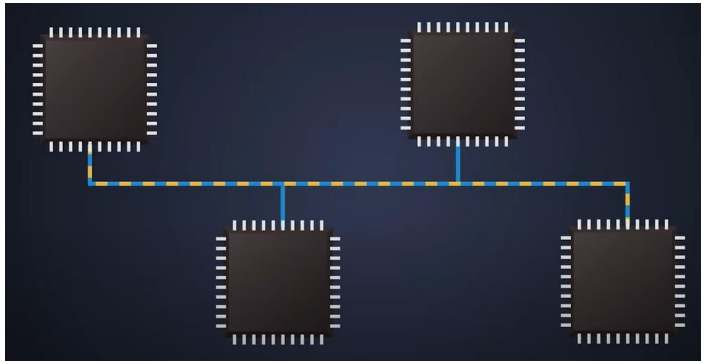
Topologies of Commuication



Communication Protocols

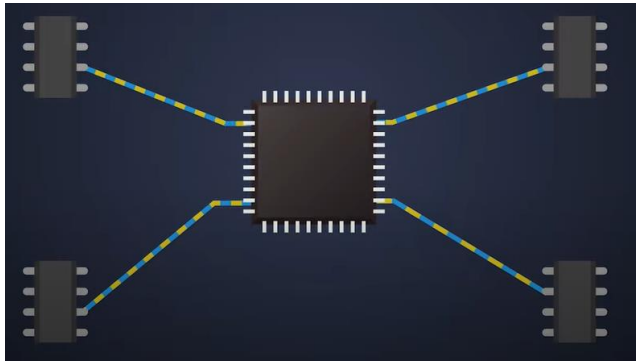
Topologies of Commuication

BUS TOPOLOGY



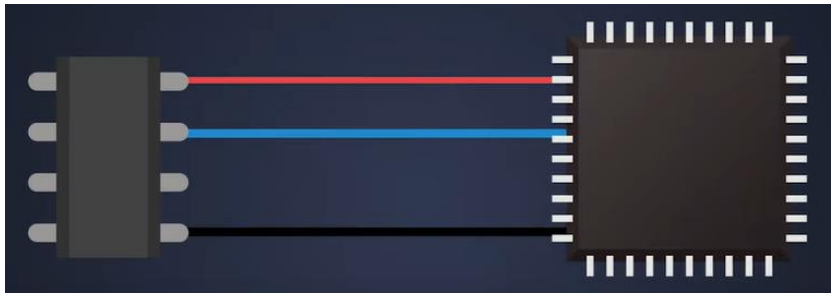
- One bus to connect them all
- If one device is out, nothing happened.
- Specific ID is for each device, so they can talk to each other. Or can also broadcast a message for all.
- Examples are **CAN**, **I2C**, ETHERNET

STAR TOPOLOGY



- One MASTER is required.
- Master read from or write to slaves.
- No master => No communication
- Example is : **SPI** (synchronous communication protocol)

Peer-to-Peer Topology



- Only two devices involved.
- No more devices can be part of the comunication
- Example is **UART**.

Communication Protocols

Synchronous and Asyncronous Communication

ASYNCHRONOUS COMMUNICATION

- Clocks for each device are separated.
- They need to agree the baud ratio. (e.g 9600 baud). This means the speed of the transfer is known for both sides (time between one bit and the next(, but their clocks are not necessarily the same or in same phase.
- Each data is sent to a specific rate.

We will study UART and CAN

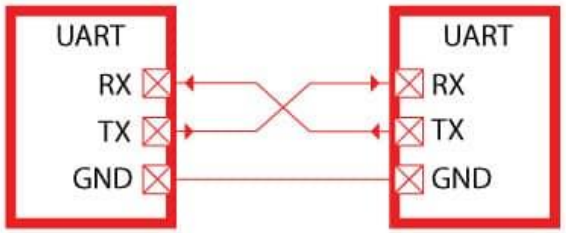
SYNCHRONOUS COMMUNICATION

- Clocks for each device is the same.
- Transmitter send a first data to say that will start.
- Receiver prepares for receiving data.
- Transmitier send the data
- Transmiter send a closing bit to say that all data is sent.
- Receiver send an acknowledge and transaction is finished.
- If transmitter does not receive acknowledge, it will send data again.

We will Study SPI and I2C

UART
Communication Protocols

Universal Asynchronous Receiver/Transmitter (UART) Communication Protocol

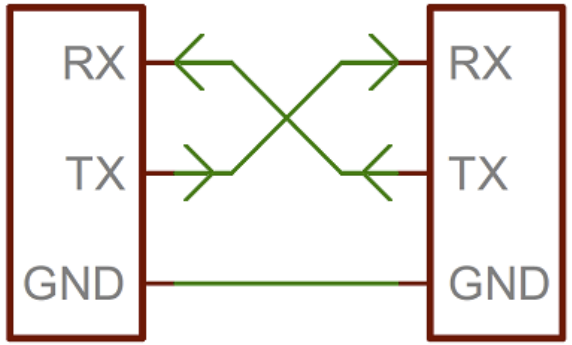


Communication Protocols

UART

UART stands for **Universal Asynchronous Receiver/Transmitter**.

- It is a hardware communication interface used for serial communication between devices.
- Only uses two communication lines: a transmit line (TX) and a receive line (RX).
- It operates asynchronously, meaning that data is sent without the need for a separate clock signal.
- The timing information is embedded within the data stream using start and stop bits.



Rx and Tx must be refered to same potential.

Communication Protocols

UART

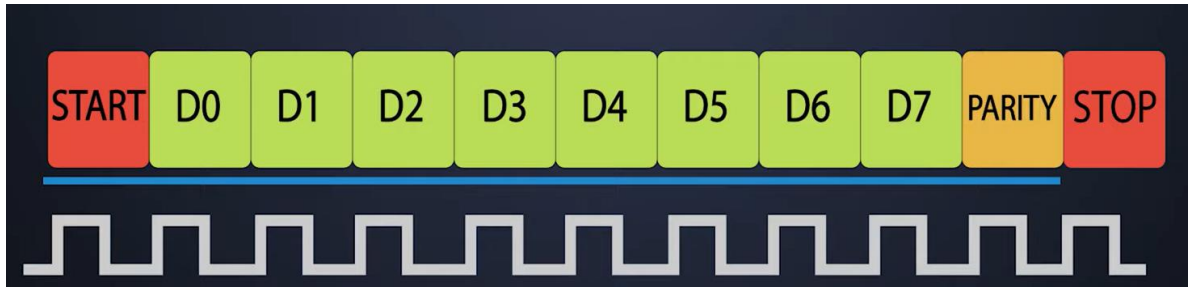
1. **Data Format:** UART uses a specific data format to transmit and receive information. It typically includes a start bit, a fixed number of data bits (usually 7 or 8), an optional parity bit for error detection, and one or more stop bits.
2. **Transmission:** When data needs to be sent, it is serialized bit by bit, starting with the start bit, followed by the data bits, optional parity bit, and stop bit(s). The start bit indicates the beginning of a data frame, while the stop bit(s) indicate the end.
3. **Reception:** On the receiving end, the UART continuously monitors the receive line for incoming data. It waits for the start bit and then samples the subsequent bits at regular intervals to reconstruct the transmitted data frame.
4. **Baud Rate (bit/s):** UART communication operates at a specific baud rate, which represents the number of bits transmitted per second. The baud rate needs to be configured identically on both the transmitting and receiving devices for proper communication.

UART is widely used for connecting devices within a system or for communication between different systems. It offers a simple and reliable way to exchange data over short distances and is compatible with a wide range of devices and microcontrollers.

Communication Protocols

UART

BIT FRAME

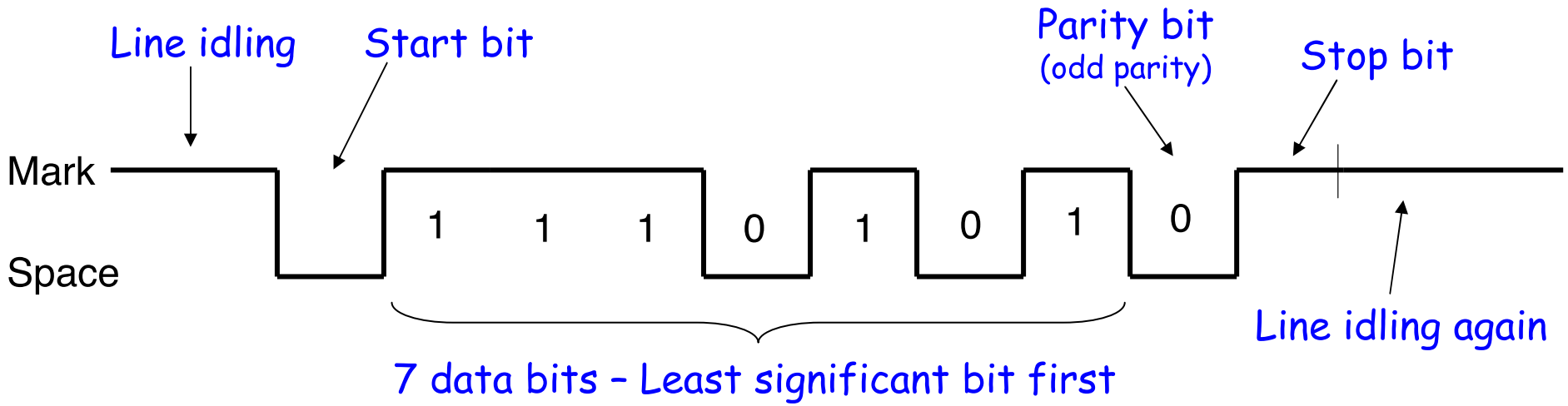


- IDLE state is HIGH.
- Starts bit is LOW and Stop bit is HIGH.
- Data is sent at he pre-set baud rate. For instance 9600 bit/s
- Parity bit is optional.
- Once data is sent. The communication stay in idle again (HIGH).

Communication Protocols

UART

Send the ASCII letter 'W' (1010111)

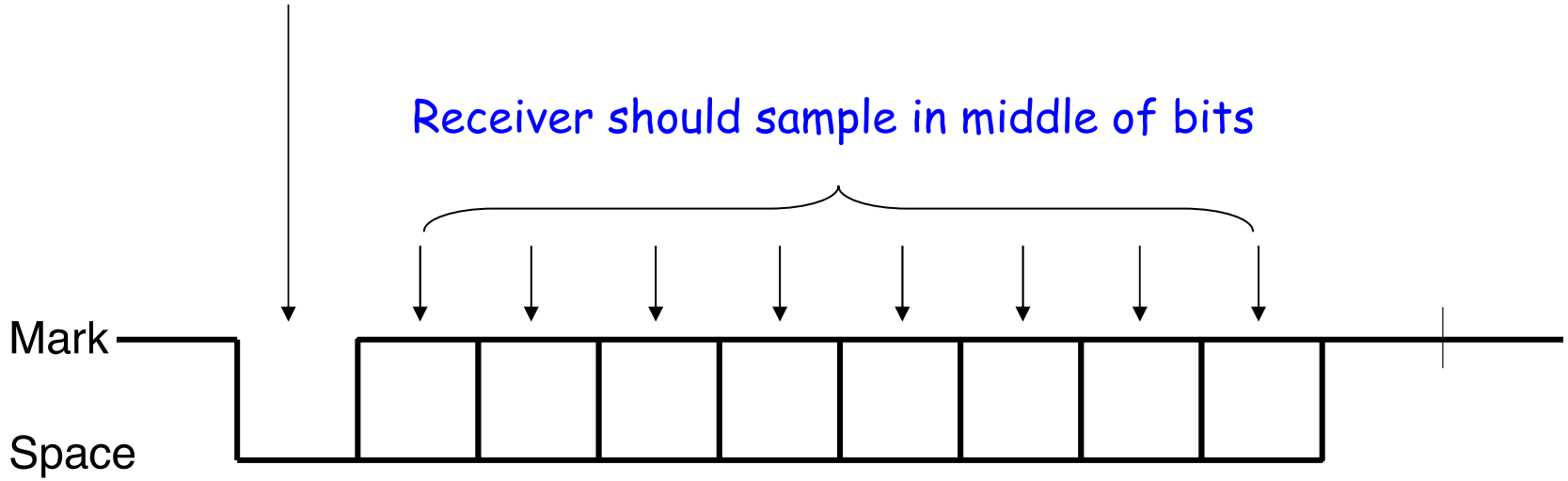


Communication Protocols

UART

Start bit says a character is coming,
receiver resets its timers

Receiver should sample in middle of bits

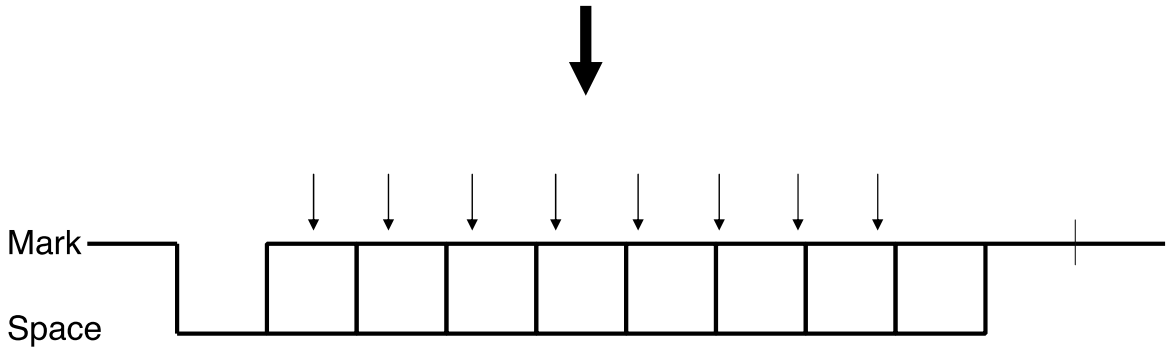


Receiver uses a timer (counter) to time when it samples.
Transmission rate (i.e., bit width) must be known!

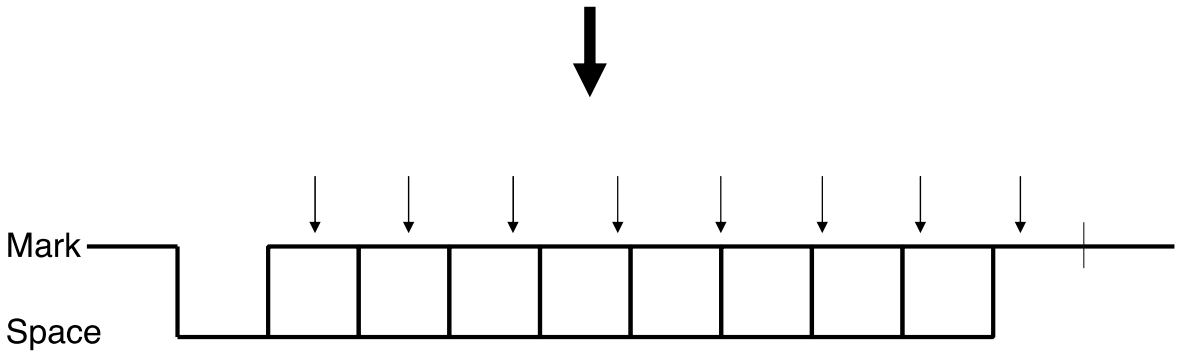
Communication Protocols

UART

If receiver samples too quickly, see what happens…



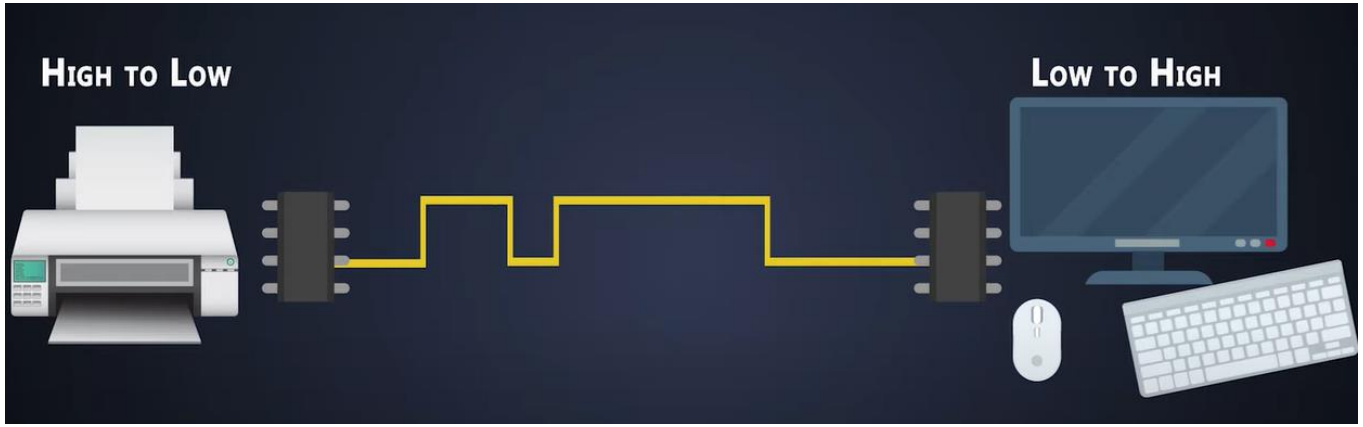
If receiver samples too slowly, see what happens…

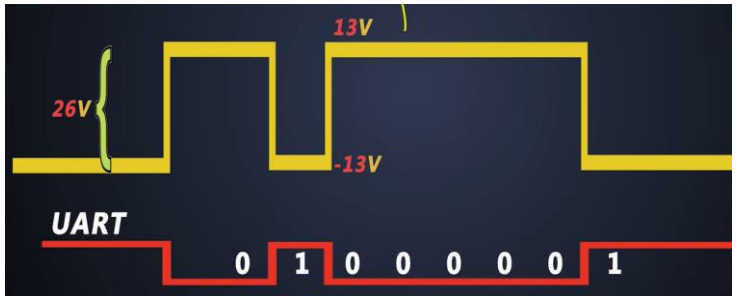


Receiver resynchronizes on every start bit.
Only has to be accurate enough to read 9 bits.

Communication Protocols

UART used as External Communication RS232

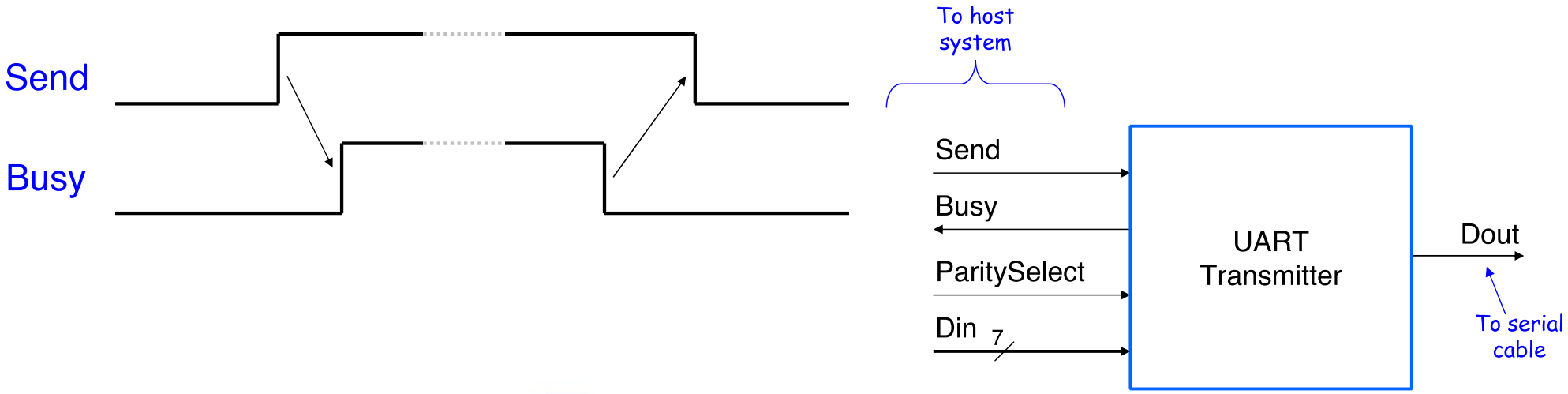




- In this case we increase the voltage of the communication signals to avoid Electromagnetic interference spoil the data.
- Some Hand-shake protocol is implemented.

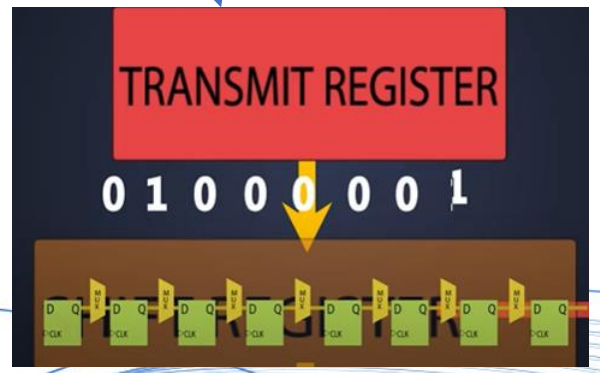
UART Transmitter Handshaking

- System asserts Send and holds it high when it wants to send a byte
- UART asserts Busy signal in response
- When UART has finished transfer, UART de-asserts Busy signal
- System de-asserts Send signal



Communication Protocols

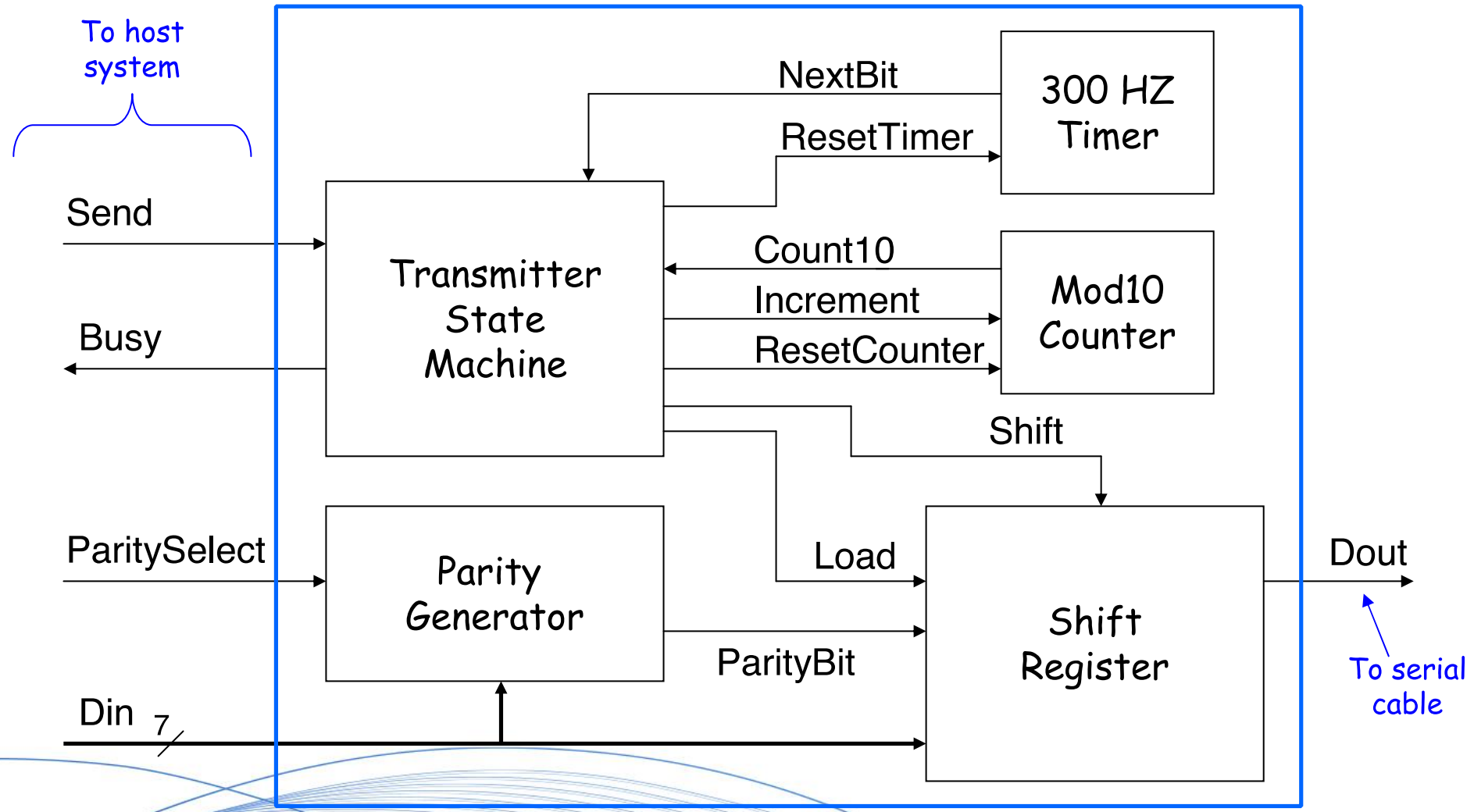
UART SHIFT Registers



Transmiter Register has the data and send it to the Shift register (**Parallel IN**). Then, the shift register send it in serial to the receiver (**SERIAL OUT**). Same process is for the receiver.

Communication Protocols

UART SHIFT Registers



Communication Protocols

UART

Advantages:

- Simple to use
- Use only two wires. Very simple Hardware

Disadvantages:

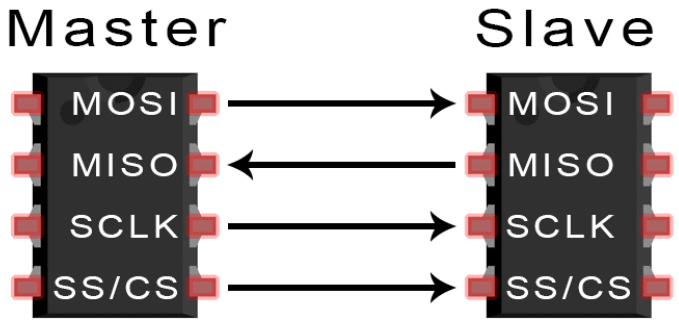
- Only two devices can be interconnected.
- The baudrate must be same, so the slower device limit the other.
- No acknowledge

Applications

- Very simple applications like interfaces for devices and printers.

I2C
Communication Protocols

Serial Peripheral Interface (SPI) Communication Protocol



Communication Protocols

I2C

The I2C (Inter-Integrated Circuit) communication protocol is a widely used serial communication protocol that allows multiple integrated circuits to communicate with each other using a two-wire interface.

It was developed by Philips Semiconductors (now NXP Semiconductors) in the early 1980s.

I2C

Key features of the I2C protocol include:

1.Master-Slave Relationship: In an I2C communication, there is one master device that initiates and controls the communication, and one or more slave devices that respond to the master's commands or requests.

2.Addressing: Each slave device on the I2C bus has a unique address assigned to it. The master device uses these addresses to select specific slave devices for communication.

3.Multi-Master Support: I2C supports a multi-master configuration, which means that multiple master devices can be connected to the same bus. However, to avoid conflicts, a bus arbitration mechanism is used to determine which master has control of the bus at a given time.

I2C

4. Acknowledgment: After receiving each byte of data, the receiving device (either master or slave) sends an acknowledgment (ACK) or non-acknowledgment (NACK) signal to indicate successful or unsuccessful data reception, respectively.
5.Data Transfer Modes: I2C supports two modes of data transfer: addressing mode and data transfer mode. In addressing mode, the master sends the slave address to specify the target device. In data transfer mode, the master and slave devices exchange data.

I2C is commonly used in various applications, such as connecting sensors, displays, memory chips, and other peripherals to microcontrollers or microprocessors. It provides a straightforward and efficient way for devices to communicate over short distances on the same PCB (Printed Circuit Board) or between separate boards.

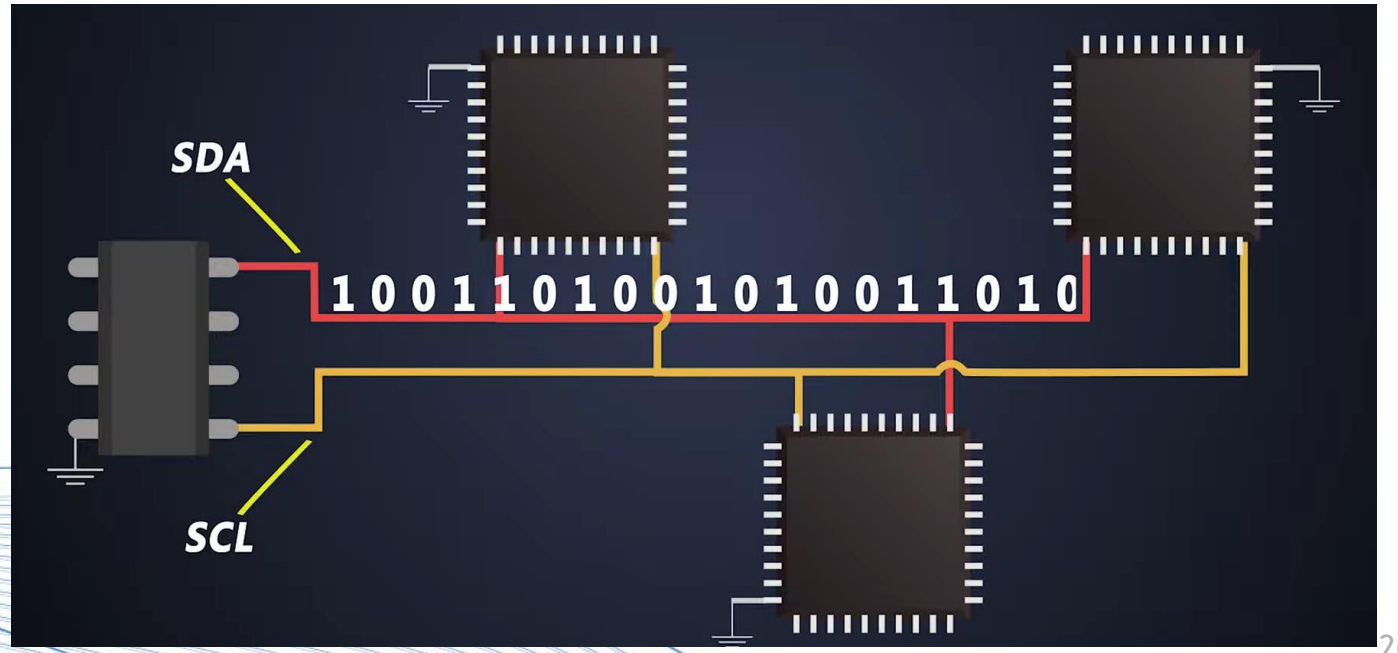
Communication Protocols

I2C

- Synchronous communication protocol
- Bus topology protocol
- Its has two main pins

1.SDA (Serial Data Line): This line carries the actual data being transmitted between devices. It is used for both transmitting and receiving data.
2.SCL (Serial Clock Line): This line provides the clock signal that synchronizes the data transfer between devices. It dictates the timing of data transmission.

- There is one (or more) Masters.
- There is one or more slaves.
- Each device has an ID.



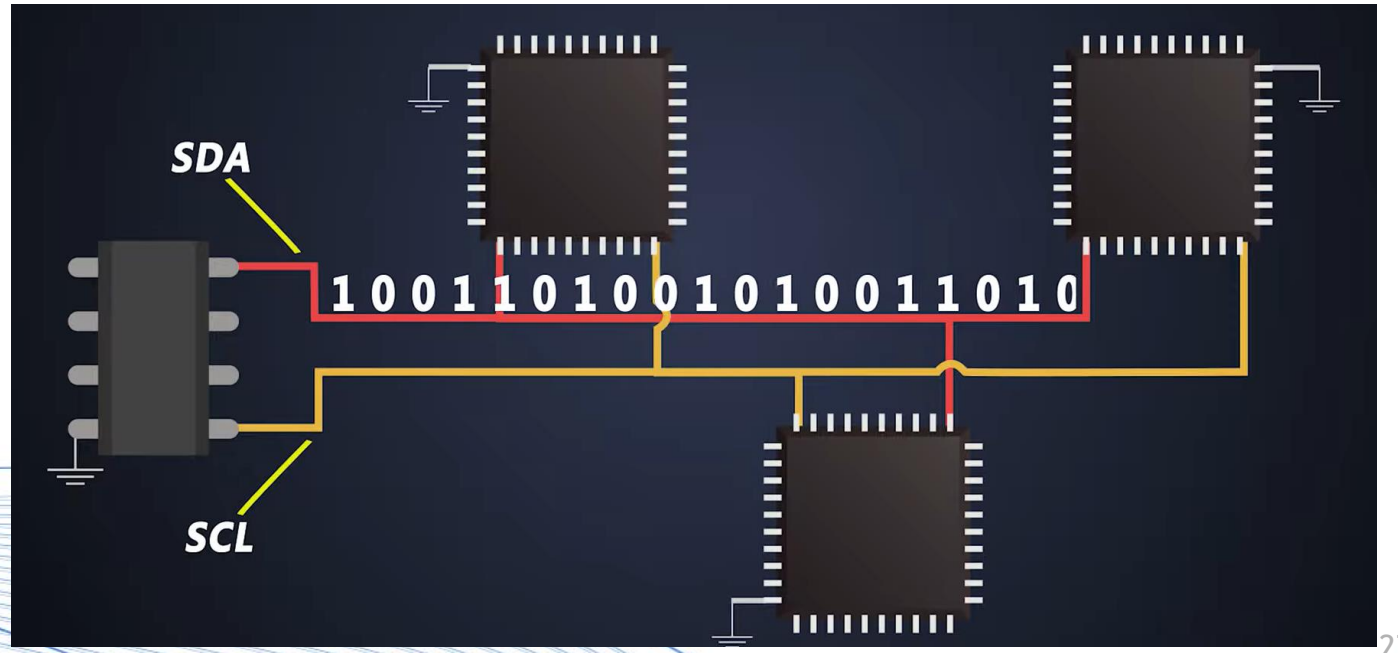
Communication Protocols

I2C

- Synchronous communication protocol
- Bus topology protocol
- Its has two main pins

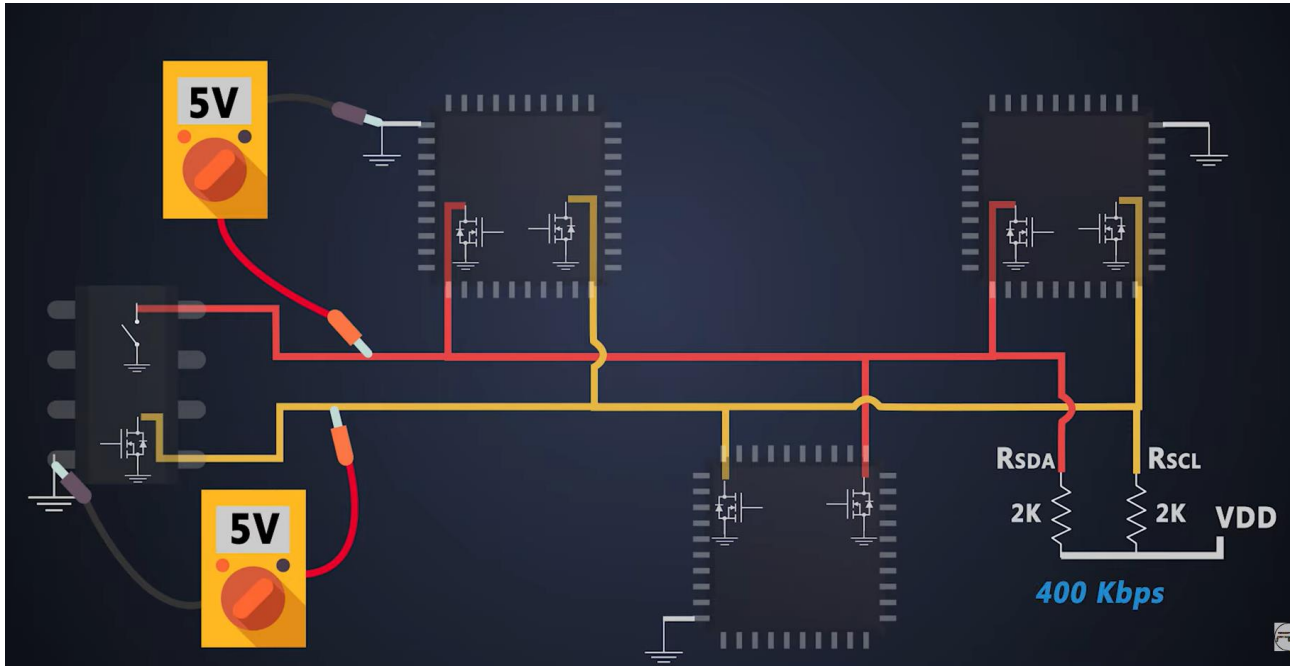
1.SDA (Serial Data Line): This line carries the actual data being transmitted between devices. It is used for both transmitting and receiving data.
2.SCL (Serial Clock Line): This line provides the clock signal that synchronizes the data transfer between devices. It dictates the timing of data transmission.

- There is one (or more) Masters.
- There is one or more slaves.
- Each device has an ID.



Communication Protocols

I2C



- SDA and SCL are configured as open collector, so they are all connected to pull-up resistor.
- This allow to read the same value in all master when MOSFET in the master closes or opens.
- The pull-up resistor is smaller when we use higher baud rate. But this also need more current, i.e. more power dissipation
- I2C can work in different modes:
  - Standard: upto 100kbps
  - Fast: upto 400kbps
  - Fast+: upto 1Mbps
  - High Speed: up to 3.4Mbps
- The slower device in a bus topology dfine the speed of the bus.

Communication Protocols

I2C

- I2C BIT FRAME:

- START: Pulling it from 1 to 0, indicates the start of the transaction.
- S0-S7: Slave Address. (can be extended to 11bits)
- R/W: Master sends a read (1) or write (0) signal.
- ACK: After the previous sequence. Slave send ACK.
- R0-R7: Is the address for the register into the slave, where the master wants to read or write.
- ACK: After the previous sequence. Slave send ACK again.
- D0-D7: Then, the 8bit data is sent or received.
- ACK: Final acknowledge. (if any acknowledge fails , receive NACK and communication fails).
- STOP: Stop bit to notify the slave that the communication has ended.

Communication Protocols

I2C Timing for sending an address.

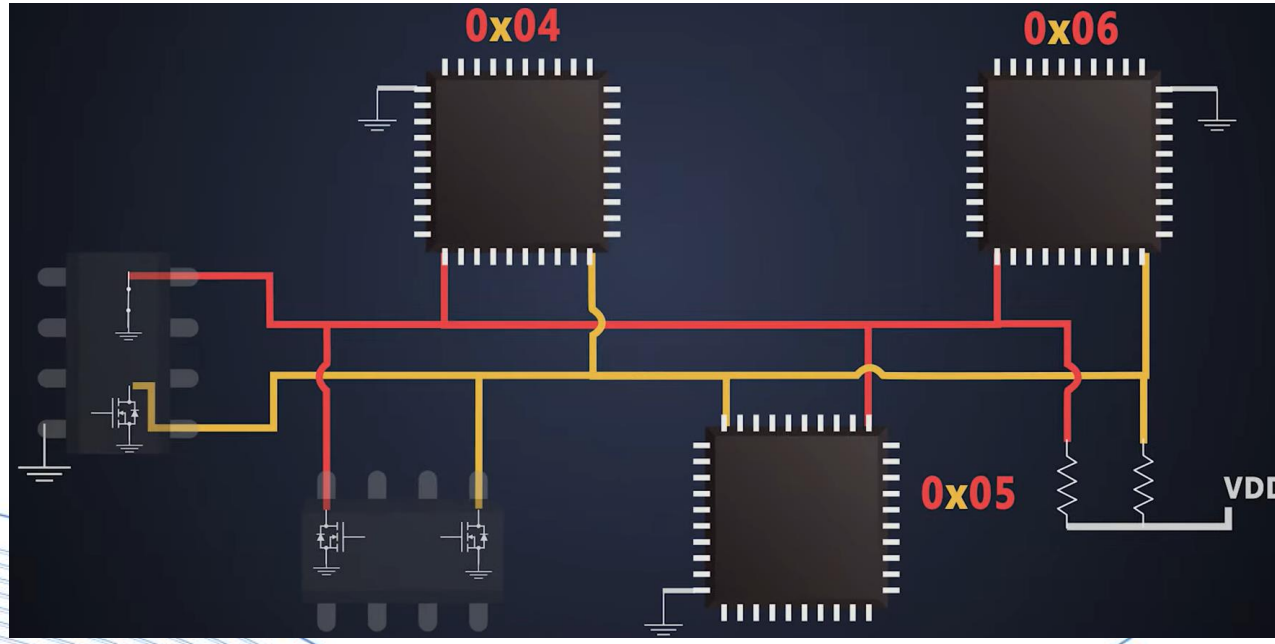
- Notice that the receiver reads the data only at the transition from low to high. So we must ensure that we have the correct data at thar time. For that we synchronize the data and SCL as showed in the picture.

Communication Protocols

I2C Multi-Master Arbitration

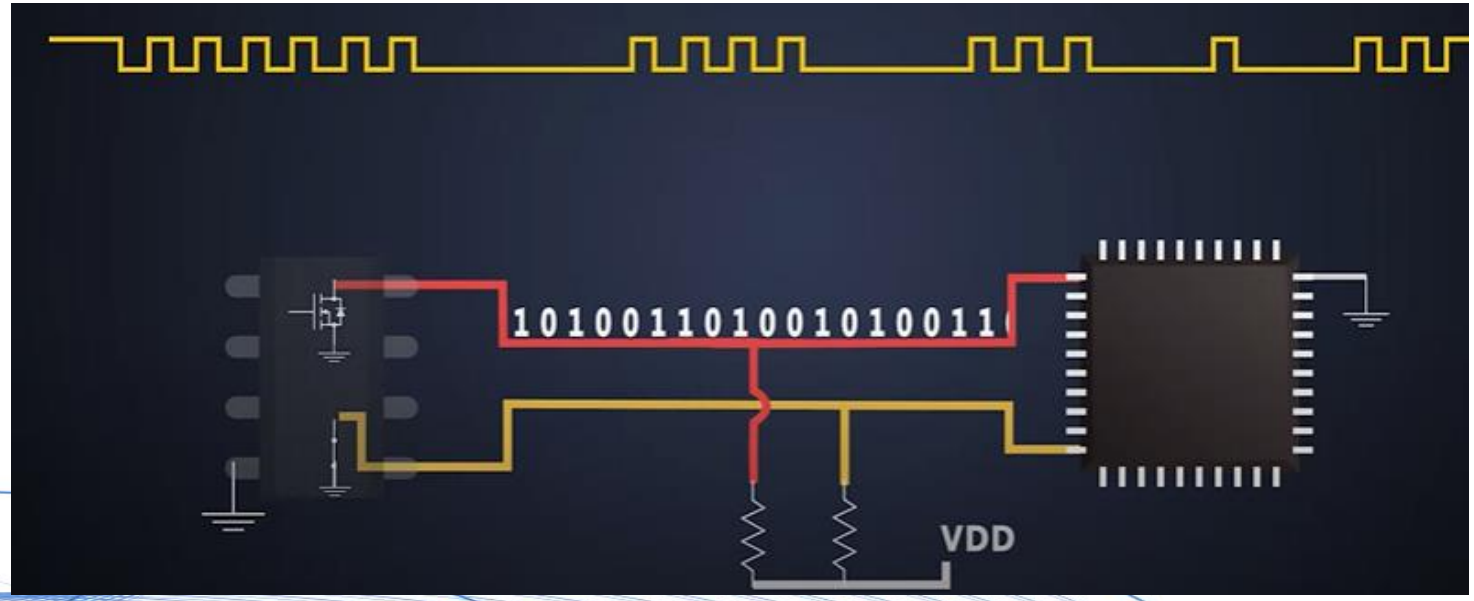
- Suppose two master address simultaneously:

- At the bit S1, both masters write different values on the same line. However, the master 1 send the line to ground, it forces the line to 0V, overwriting the information written master 2.
- After this, Master 2 stops sending data and master 1 send the date.
- This is known as arbitration. (same happens in CAN protocol)



I2C Clock Stretching

- It is a mechanism of the slave to tell the master that wait until it is ready for the next transaction.
- The slave forces to 0 the SCL pin. Stopping the Data Frame in the master.
- It is a way for the slave to say I'm BUSY!
- After slave release the SCL, then master continues with the data transmission.



Communication Protocols

I2C

Advantages:

- It needs only two lines.
- Simple self arbitration in Multi-master systems.
- Good error handling by its 3 ACK.
- If different speed slaves are connected to the bus, the master can adjust the speed for each slave while communicating to each one.

Disadvantages:

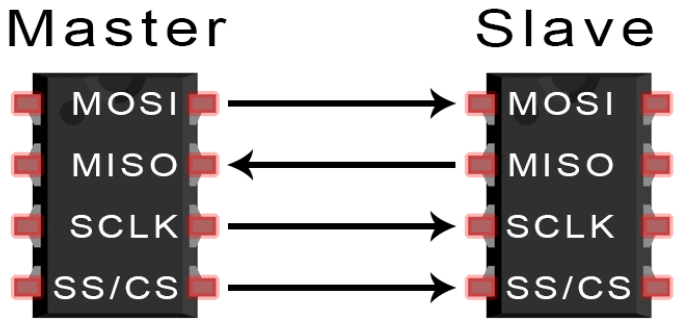
- Slave address is fixed by the manufacture. Only 7bits available. (E.g. We can not use two identical chips. This is being solved in modern I2C slaves chips by configurable address)
- Firmware is more complicated to handle Bus Arbitration and Clock stretching.
- Large amount of overhead data. This reduces data throughput. (start,stop, 3xACk, R/W, 7bit slaves Add)
- As any bus topology. If one slave is faulty to low, it blocks the bus.

Applications

- Many ADCs, DACs, EEPROM, Timers, sensors and peripheral.

SPI
Communication Protocols

Serial Peripheral Interface (SPI) Communication Protocol



SPI Features

- The Serial Peripheral Interface (SPI) is a **synchronous** serial communication protocol commonly used for **short-distance communication between microcontrollers**, sensors, and other peripheral devices.
- It was developed by **Motorola** in the 1980s and has since become widely adopted in the industry.
- The SPI protocol **typically involves a master device that controls one or more slave devices**.
- It uses **a full-duplex communication** method, which means that data can be simultaneously transmitted and received between devices.
- It supports **single Master** and multiple slaves.

SPI Features

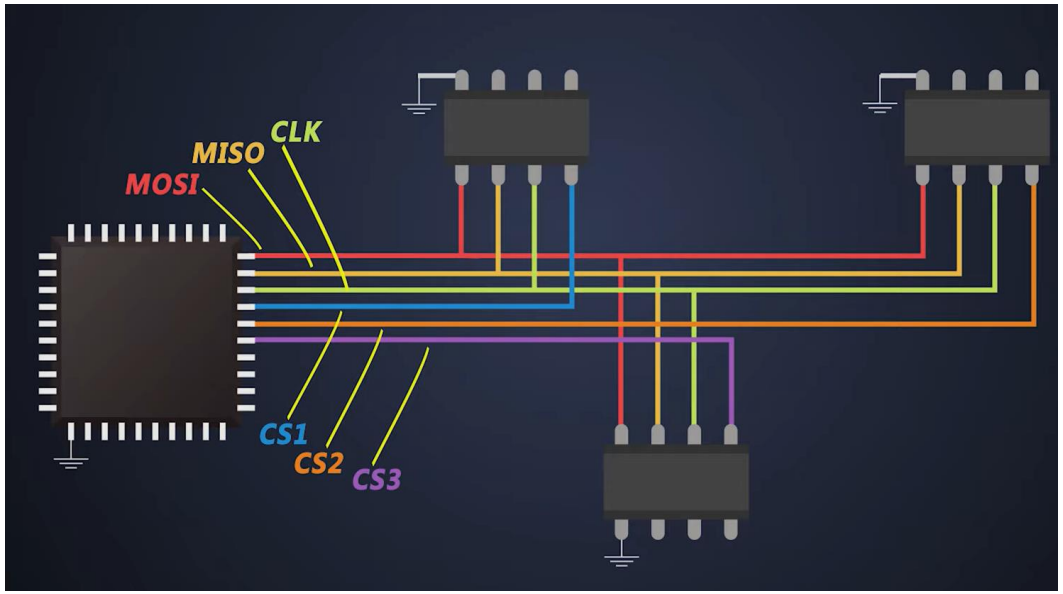
- The SPI protocol operates in a master-slave fashion, where the **master device initiates and controls the communication.**
- The master generates the clock signal on the **SCLK** line and sends data on the **MOSI** line. The slave device(s) receive the clock signal and data and respond by sending data back on the **MISO** line.
- SPI supports different **modes (0 to 3)** that define the clock polarity and phase, allowing flexibility in communication between devices.

Overall, the SPI protocol offers a simple and efficient means of communication, making it popular for connecting peripherals and components in embedded systems and microcontroller-based projects.

SPI Features

- SPI is a very Fast Synchronous communication protocol. Up to 60Mbps
- It is full-duplex.
- It is configured in bus topology. It requires 4 wires (MISO, MOSI, CLK, CS).

- The number of GPIOS restrict the number of slaves.
- CS is active low signal. Generally connected with a pull-up resistor at Vcc.



SPI Pins

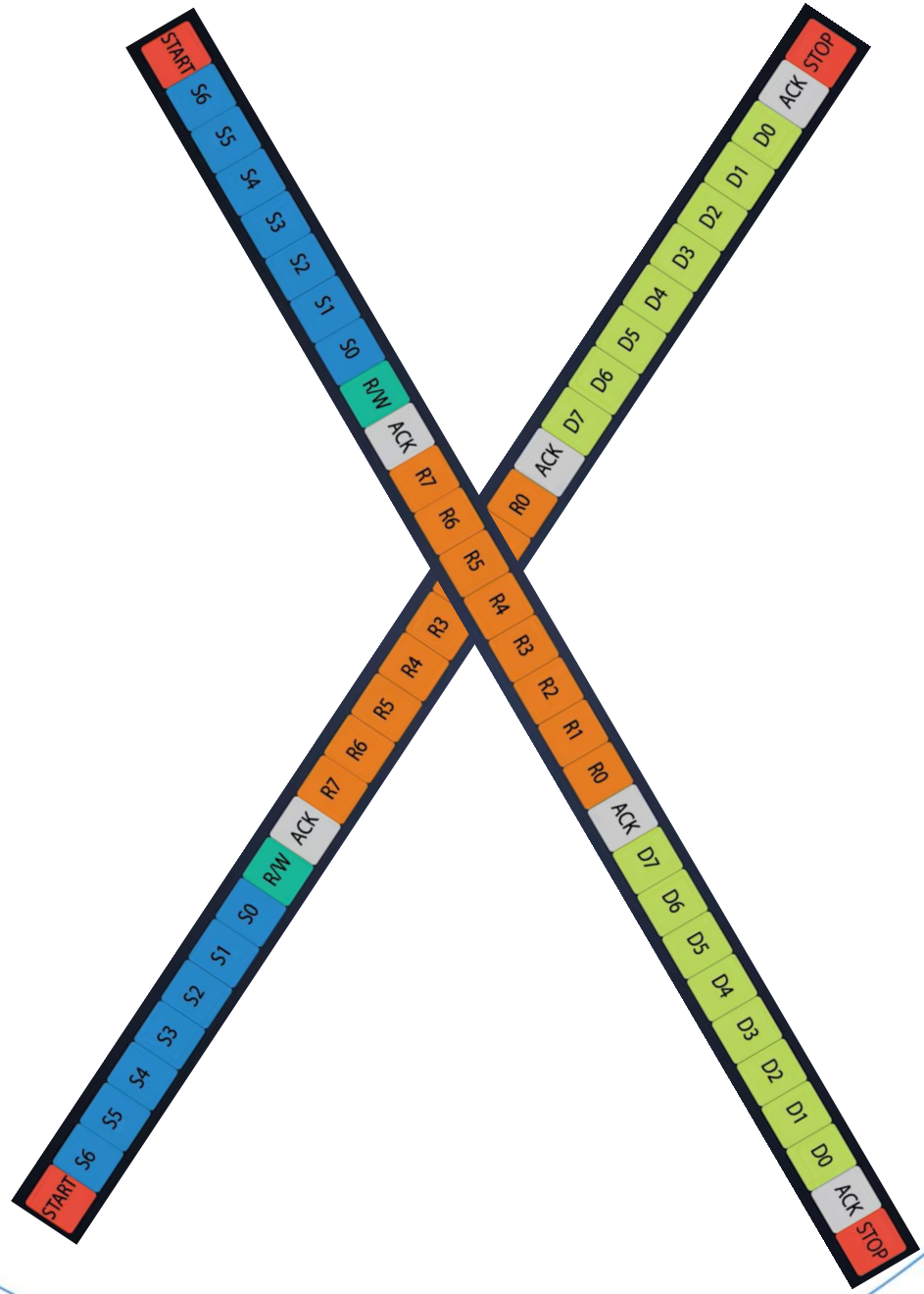
Protocol Structure:

1. **SCLK** (Serial Clock): This line provides the clock signal generated by the master to synchronize the data transfer. It is controlled by the master and is used to determine the timing of data transmission.
2. **MOSI** (Master Out, Slave In): This line carries data from the master to the slave. The master sends data to the slave on this line.
3. **MISO** (Master In, Slave Out): This line carries data from the slave to the master. The slave sends data to the master on this line.
4. **SS/CS** (Slave Select/Chip Select): This line is used by the master to select a specific slave device for communication. Each slave device typically has its own SS/CS line, allowing the master to communicate with multiple slaves on the same bus.

SPI

BIT FRAME.

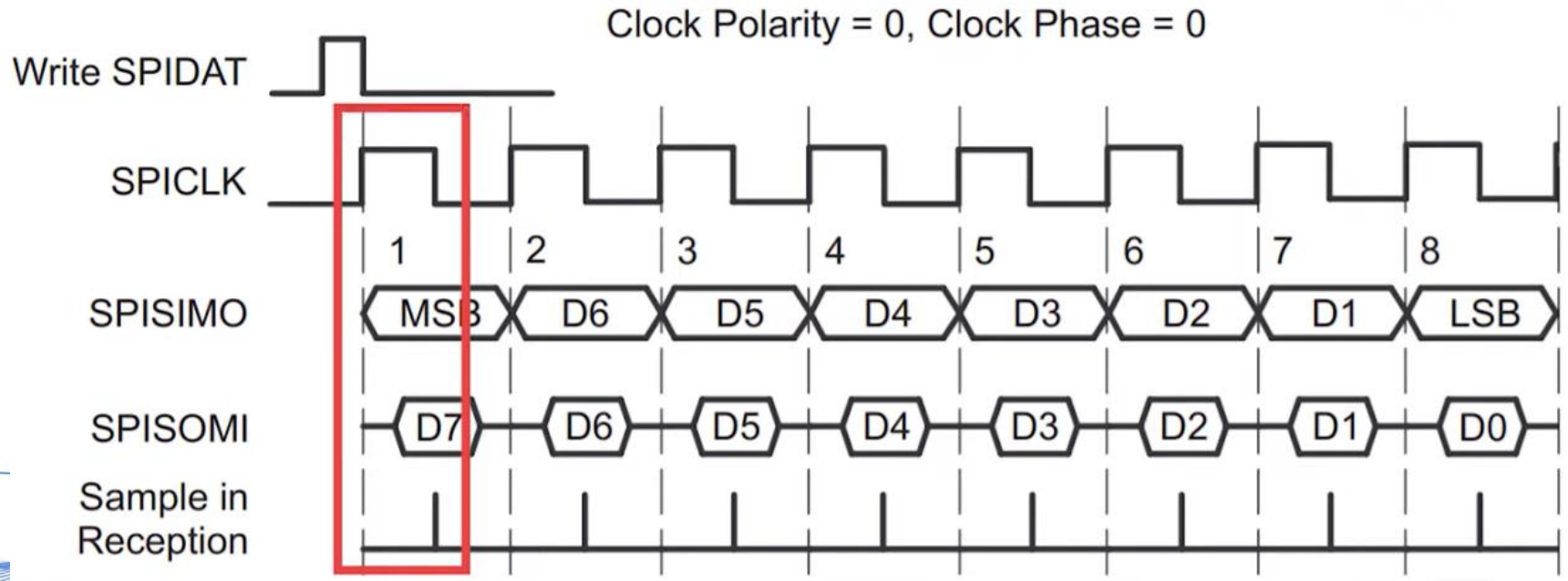
- SPI is a “de facto” standard protocol.
- It means, there is **no specific BIT FRAME.**
- **Is makes easy to send and receive data.**



Communication Protocols

SPI MODES

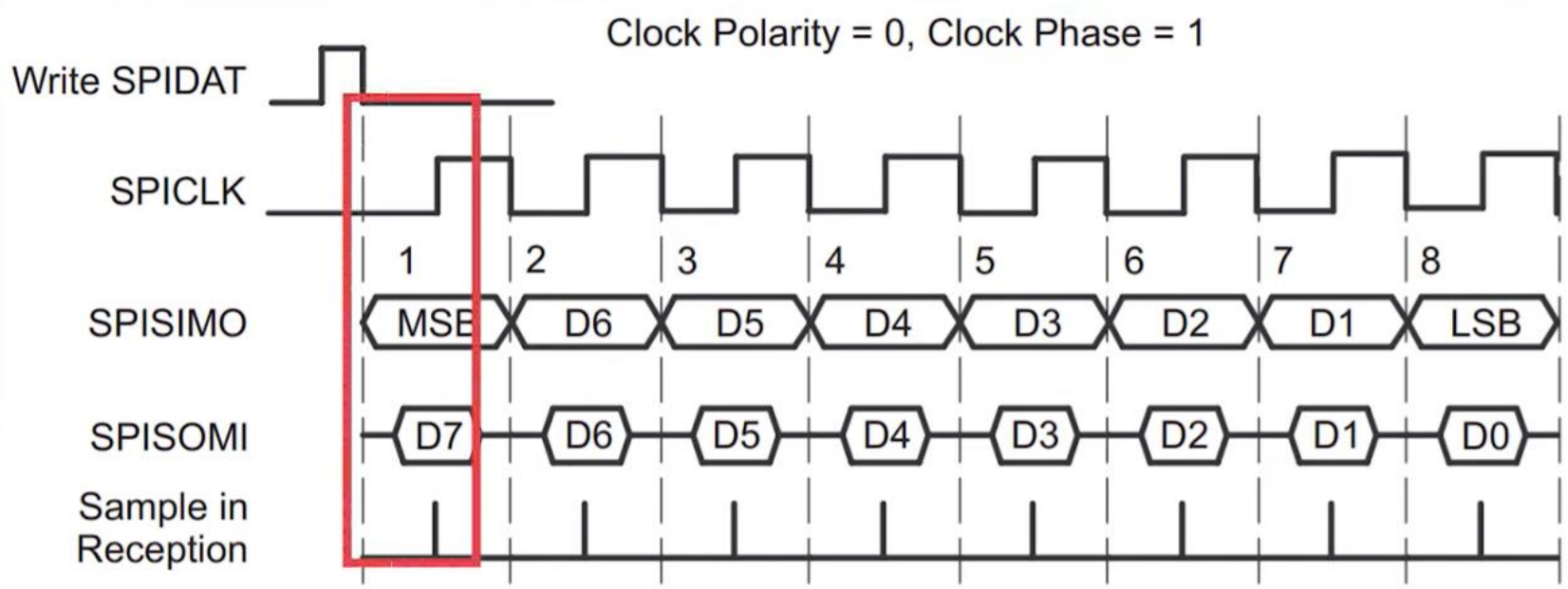
	CLK Polarity	CLK Phase	Comments
MODE 1	0	0	Data is output inmediatly after rising Edge of SPICLK Input Data is latched at the next falling-Edge of SPICLK



Communication Protocols

SPI MODES

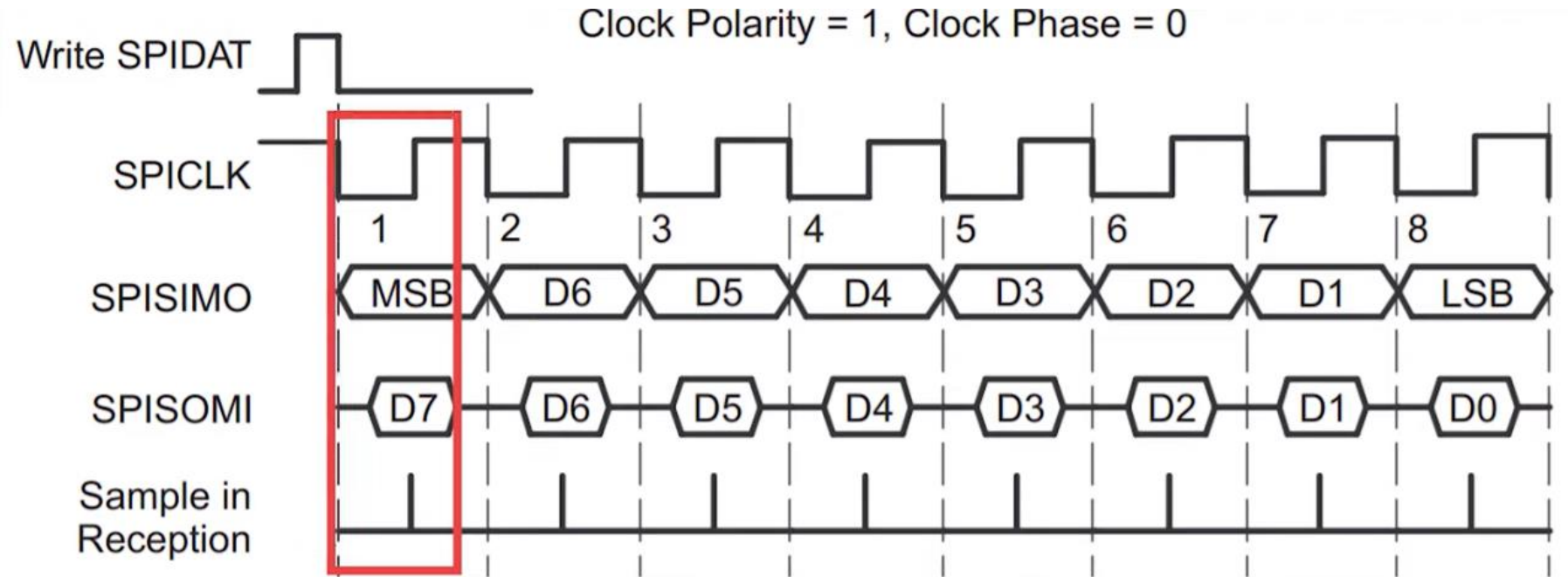
	CLK Polarity	CLK Phase	Comments
MODE 2	0	1	The SPICLK starts after some delay. Data is output one half cycle before first rising Edge of SPICLK. The input data is latched after the first rising-Edge of SPICLK.



Communication Protocols

SPI MODES

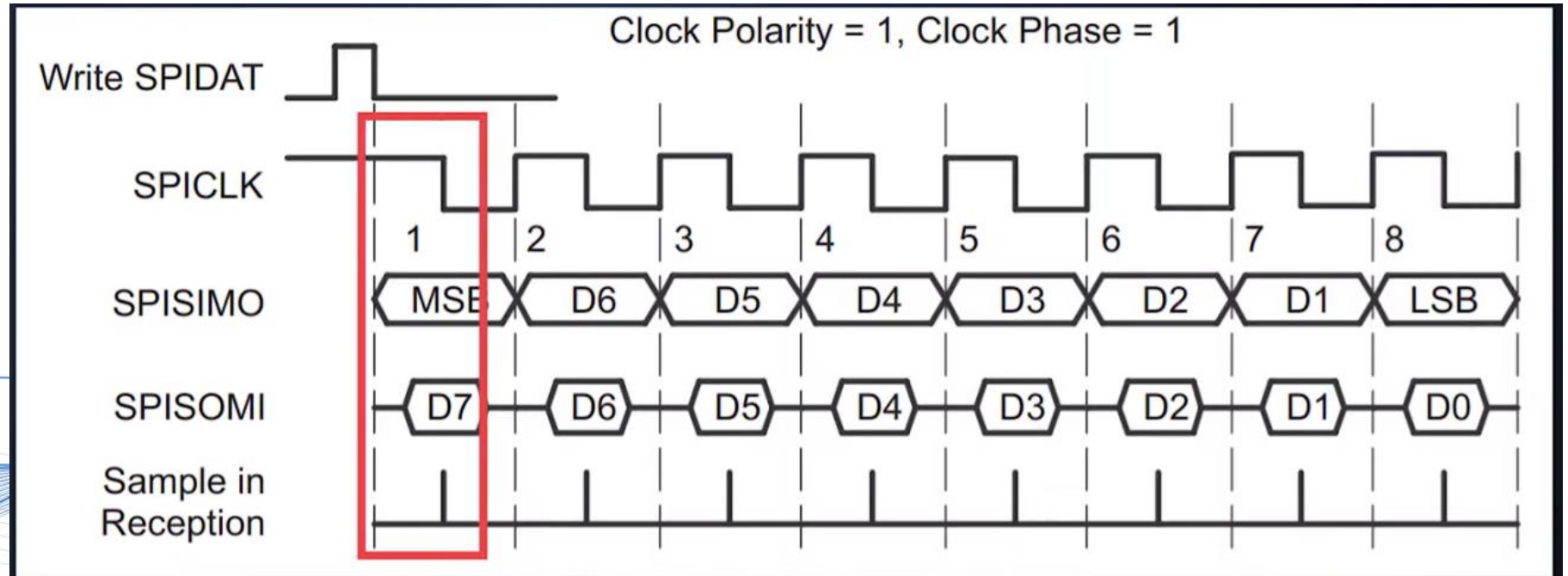
	CLK Polarity	CLK Phase	Comments
MODE 1	1	0	Data is output inmediatly after falling Edge of SPICLK Input Data is latched at hte rising-Edge of SPICLK



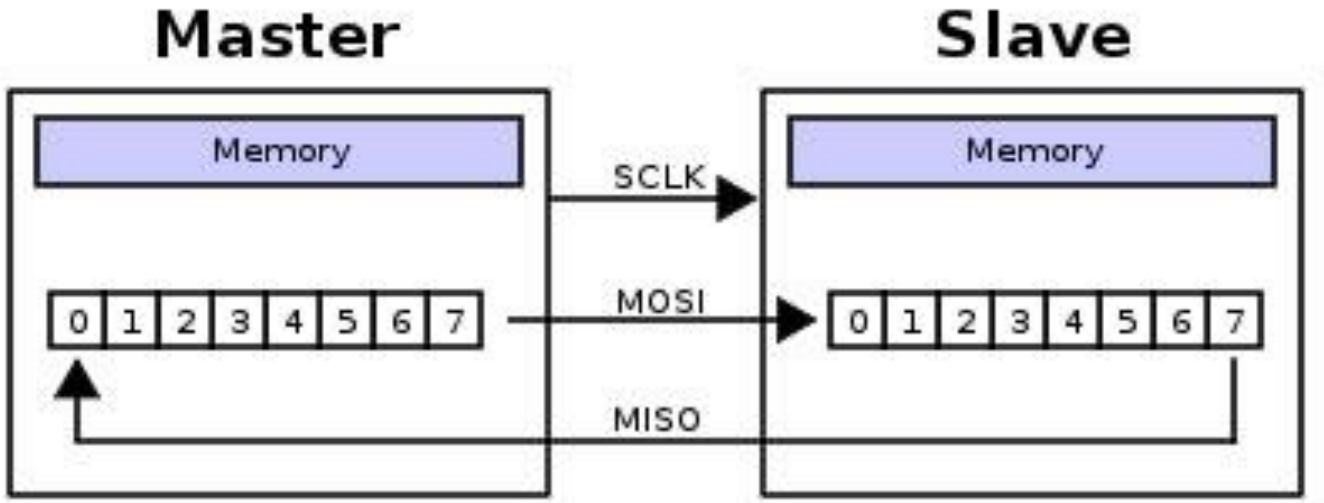
Communication Protocols

SPI MODES

	CLK Polarity	CLK Phase	Comments
MODE 4	1	1	The SPICLK starts after some delay. Data is output one half cycle before first falling Edge of SPICLK. The input data is latched after the first falling-Edge of SPICLK.



SPI Data Transmission and Reception



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

Output of the slave shift register is connected to the input of the master shift register.
Output of the master shift register is connected to the input of the slave shift register.
This makes them operate in a loop.
Master generates SCLK

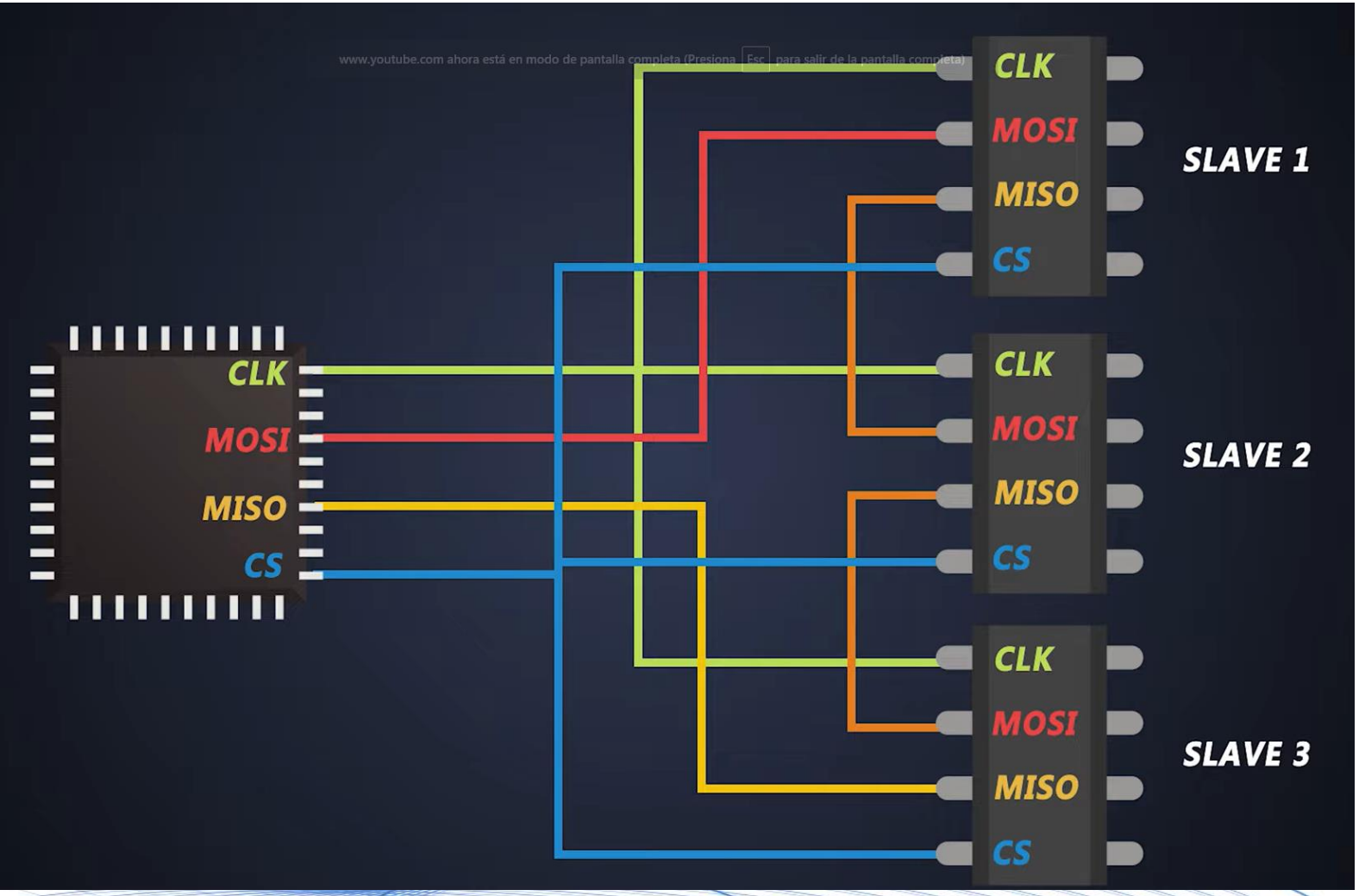
Communication Protocols

SPI Data Transmission and Reception

The data travels simultaneously from Master to slave. This process continues until the transaction finishes.

Communication Protocols

SPI Dasy Chain Connection



Connecting the MISO of one slave to the MOSI of other slave is known as Daysy chain.
This connection enables the use of only one CS.

In this case, to start communication, master set CS to low.
Then send the data as many times as we need to shift the information.

Finally Slave 3 send response to master through MISO.

Communication Protocols

SPI

Advantages:

- Low power consumption vs I2C. No need of Pull-up resistors.
- High Speed 60Mbps. Higher clock, no data frame, no overhead bits.
- Full Duplex. 2 lines for communication.
- Easy Interface. Less software and hardware complexity.
- The different MODES gives liberty to the manufactures to give any triggering logic to flipflops. Unlike other protocols, like I2C , where all logics must be same triggered.

Disadvantages:

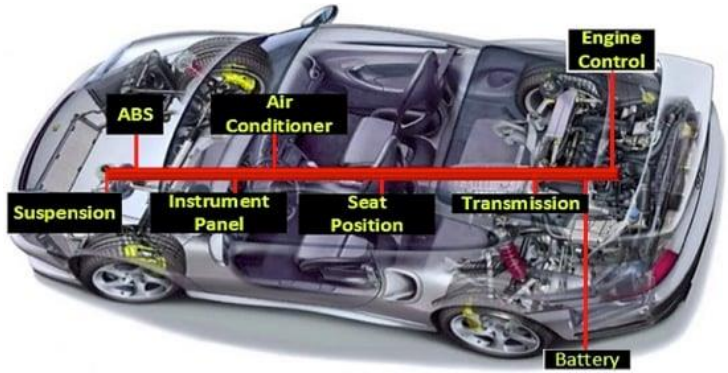
- Four signals are required (vs 2 for I2C or UART)
- Multi-master configuration is not supported.
- We need too many CS pins for several slaves (or we need to use MUXs).

Applications

- Many devices close to the processor, in same board, that require fast transfer of data. Usually memories or fast DAC and ADCs.

CAN
Communication Protocols

Control Area Network (CAN) Communication Protocol



CAN

CAN stands for **Control Area Network** Communication Protocol

- is a widely used communication protocol in the field of automotive and industrial applications.
- It was initially developed by Robert Bosch GmbH in the 1980s for use in automotive systems, and it has since become an international standard (ISO 11898) for high-speed serial communication.
- It is a message-based protocol. Data is sent serially over a two-wire bus, typically consisting of a CAN High (CANH) and a CAN Low (CANL) wire.

It mainly used in automotive industry , but also in industrial automation, medical devices, and other domains that require reliable and efficient communication between distributed nodes.

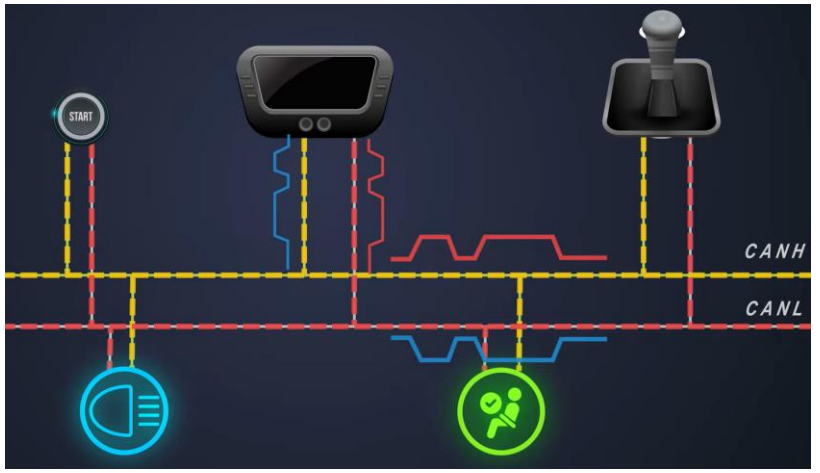
Communication Protocols

CAN

1. **Broadcast Communication:** Messages sent on the CAN bus are received by all connected nodes, allowing for efficient broadcasting of information to multiple devices simultaneously.
2. **Deterministic Communication:** CAN uses a prioritization scheme based on message identifiers, known as the Arbitration ID (CAN ID). Lower ID values indicate higher priority, enabling real-time communication and deterministic behavior.
3. **Error Detection and Fault Tolerance:** CAN incorporates a robust error detection mechanism using a cyclic redundancy check (CRC). If an error is detected during transmission, the receiving node can request retransmission, ensuring data integrity.
4. **Multi-Master Architecture:** CAN allows multiple Electronics Control Units (ECUs) to act as message transmitters and receivers. This decentralized architecture promotes flexibility and scalability in system design.
5. **Extensibility**: The CAN protocol supports different data lengths (CAN 2.0A and CAN 2.0B) and various data rates, allowing adaptation to different application requirements.

Communication Protocols

CAN



- Very useful bus topology communication protocol
- CAN HIGH and CAN LOW are the data transfer signals
- If one element is disconnected, nothing happens.
- Much reliable than star configuration.
- It does not require a master.
- Each device put a message in the bus, any other device can take the message.
- It is a half-duplex and asynchronous communication protocol
- Data speed is around 500kbps to 1MBps. Data can be transmitted up to 40m.

Communication Protocols

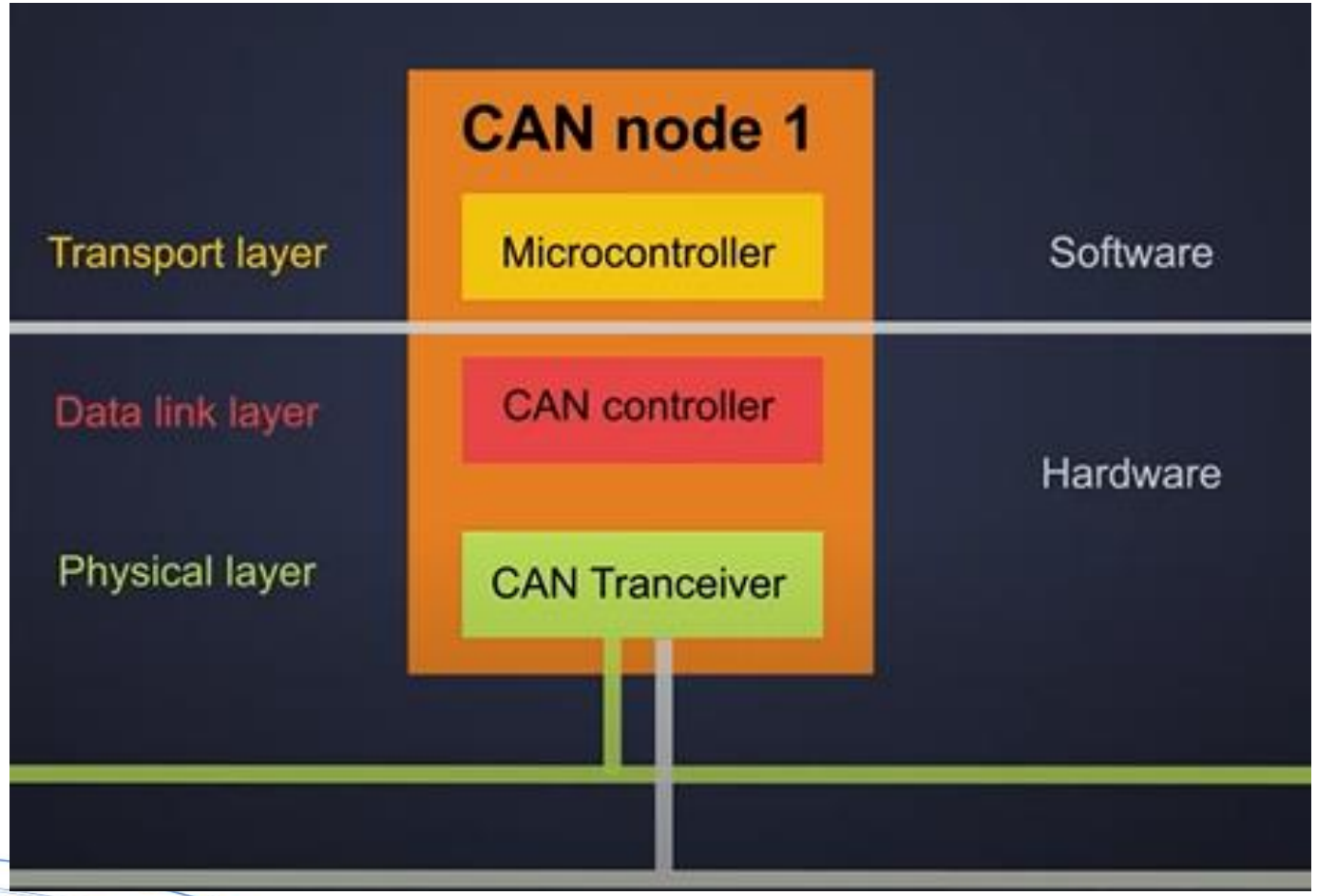
CAN

- A car is a very NOISE (electromagnetically speaking) environment.
- Therefore, we use differential signals (CAN HIGH – CAN LOW)
- Also, CAN uses Cycling Redundancy Check (CRC). This makes the receiver send the data back to the transmitter , so it can know if the receiver got the right data.

Communication Protocols

CAN

- Each device in a CAN bus is known as CAN NODE.
- Each node is composed of a Microcontroller, which generate or receive the data.
- A CAN controller, which has the protocol
- A CAN transceiver, which set it as input or output

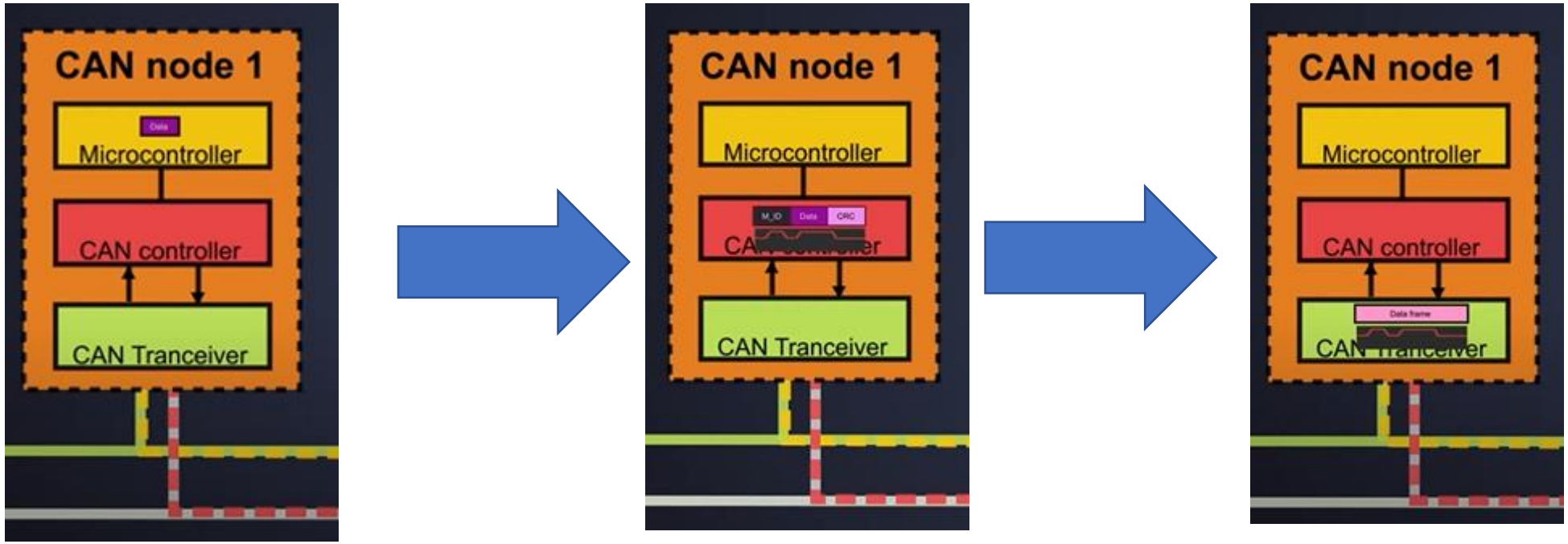


CAN Data Flow

1. Microcontroller initiates the can controller with a certain baud rate.
    1. All nodes must have same baud rate
2. Microcontroller prepare the data
3. Data is sent to CAN controller. It prepare a data in a particular form that contains the CAN node ID formulate the CAN frame.
4. The CAN controller check if there is no data in the bus. If no, it send to the CAN transceiver that transform the data in differential signal.
5. All NODES Transceivers receive the data. It extract the data and transform it from differential signals to single ended, then send it to the microcontroller.

Communication Protocols

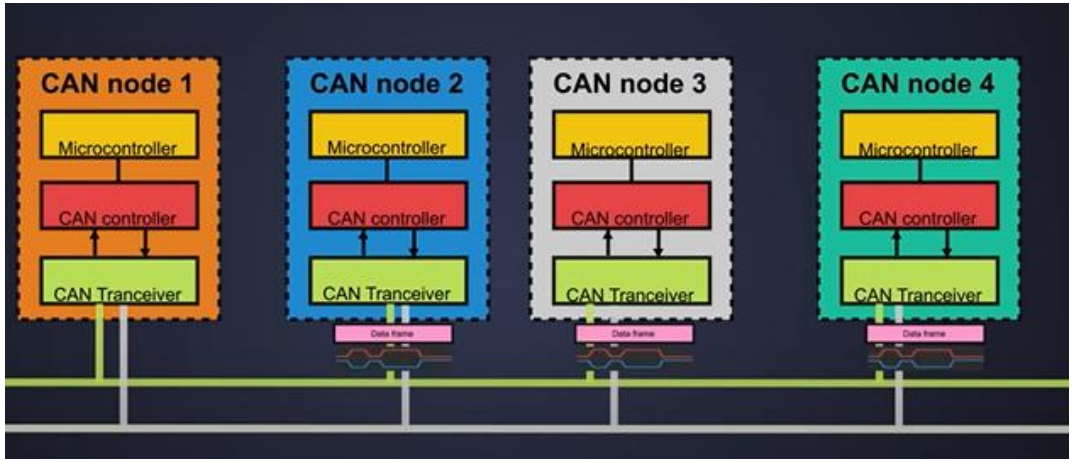
CAN

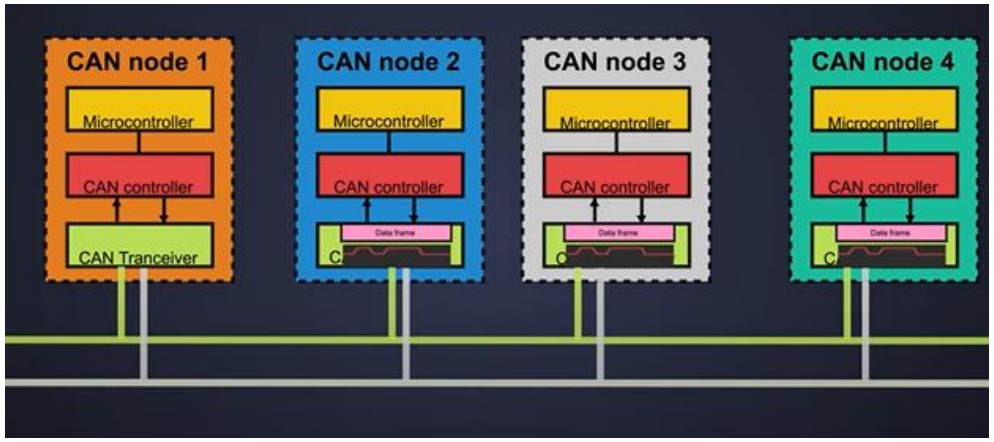


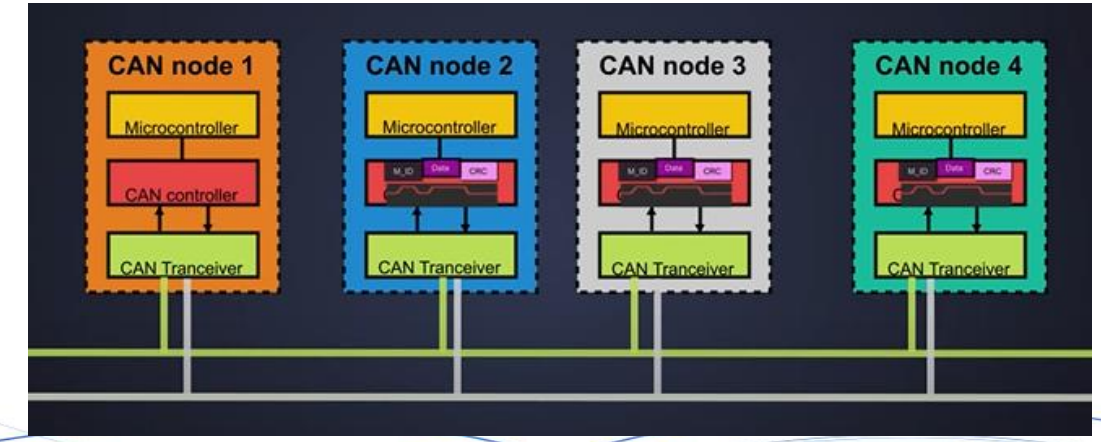
Step 1 Step 2 Step 3

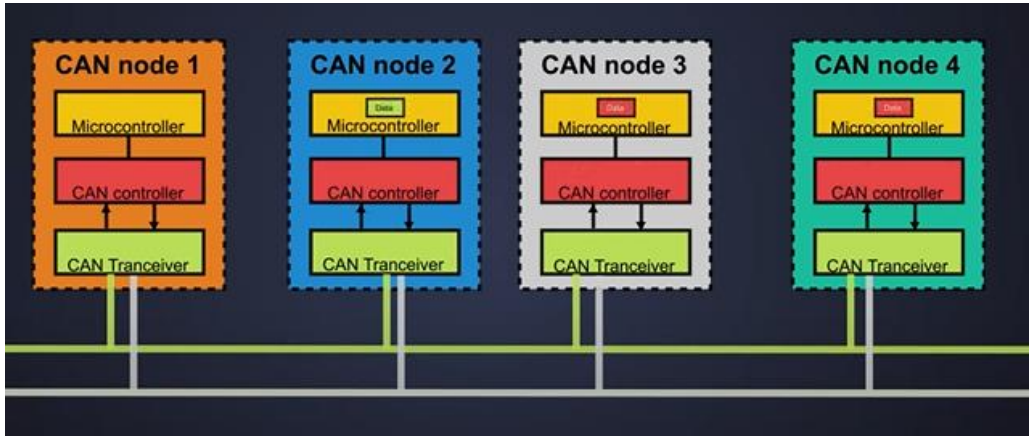
Communication Protocols

CAN









Communication Protocols

CAN

- DATA FRAME

SF: Start Field. Indicates the beginning of a message with a dominant bit

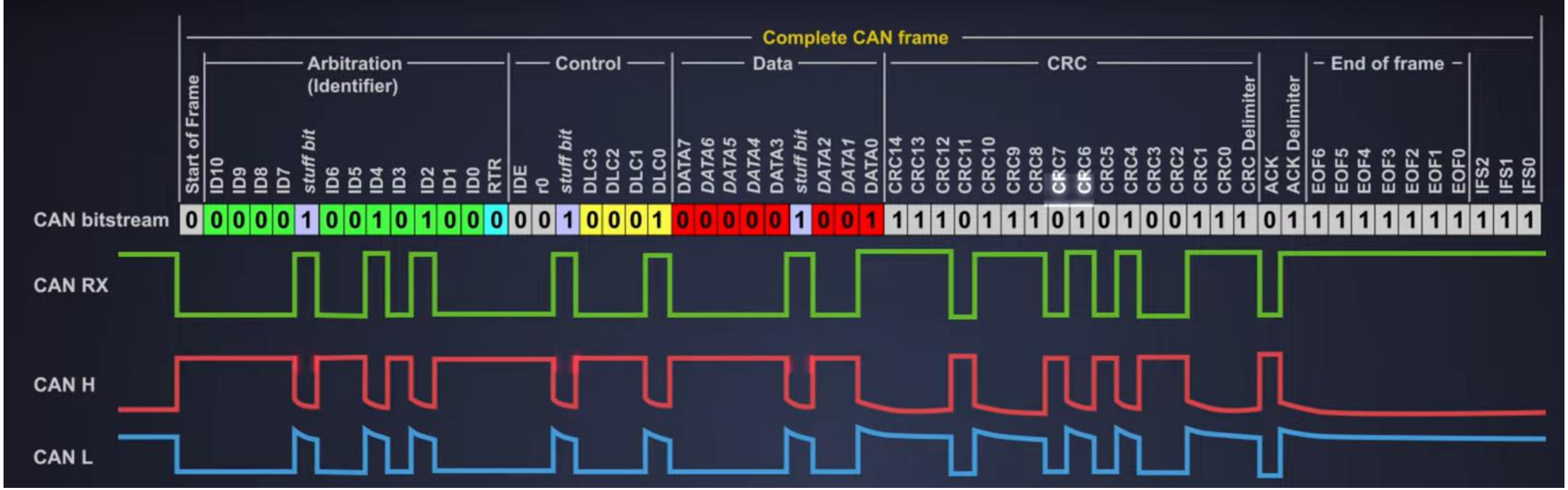
Message Identifier: Defines the level of priority of the Node. (11bits for standard ; 29bit for extended CAN). 0x0 is the higher priority.

DATA: Data

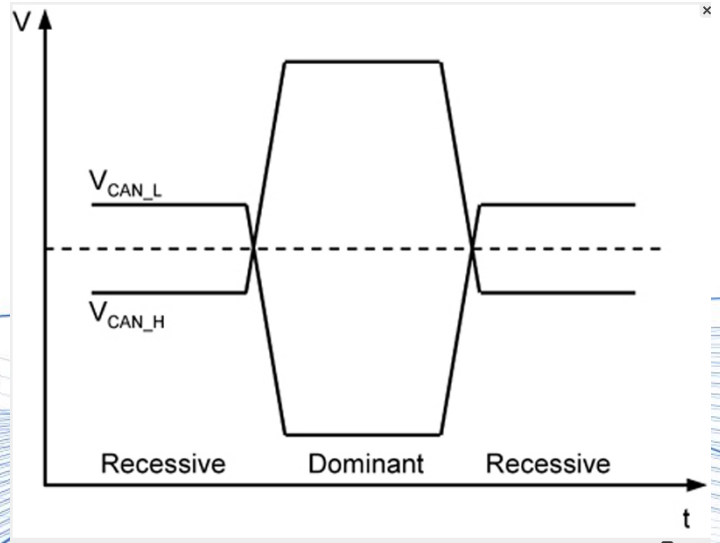
CRC (Cyclic Redundancy Field): Fault detection

ACK: Acknowledge

EF: End of Data protocol

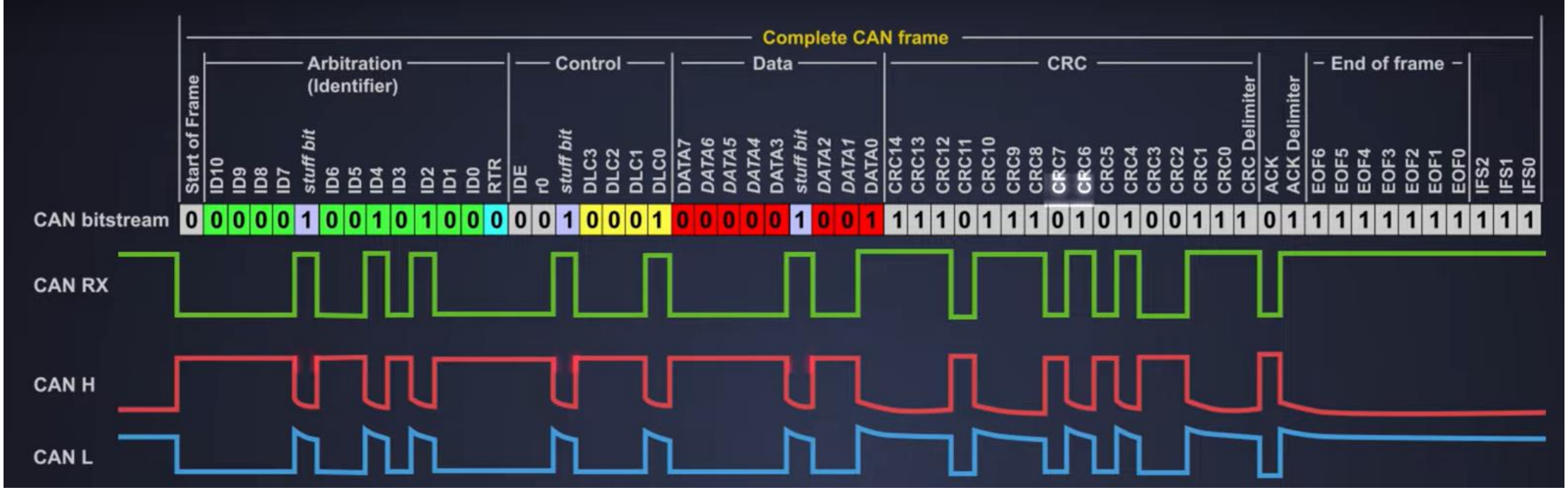


Voltages on CAN HIGH and CAN LOW are between 3.75V and 5.25V, with a common mode voltage.



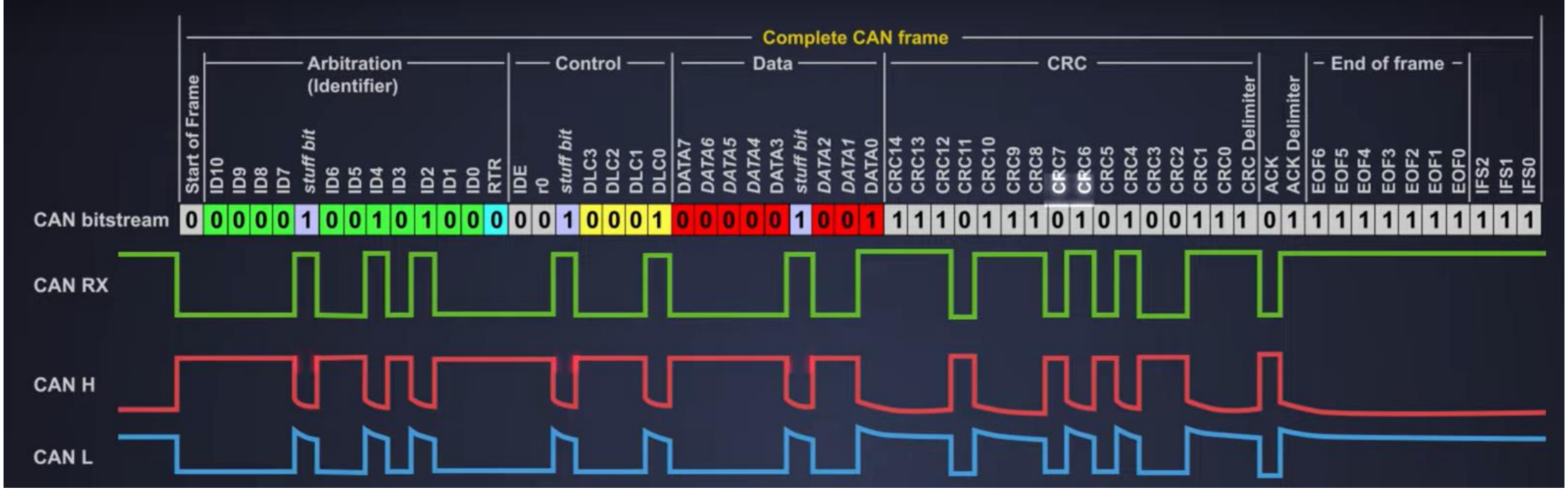
DOMINANT state is interpreted as 0
RECESSIVE state is interpreted as 1

see



SF (Start Frame): Is dominant bit (0) and indicates the start of the frame. If it is 0, means that some NODE has started a transmission on the bus or the bus is set to idle.

ID0-ID10: Indicates the ID of the ECU to be communicated to.

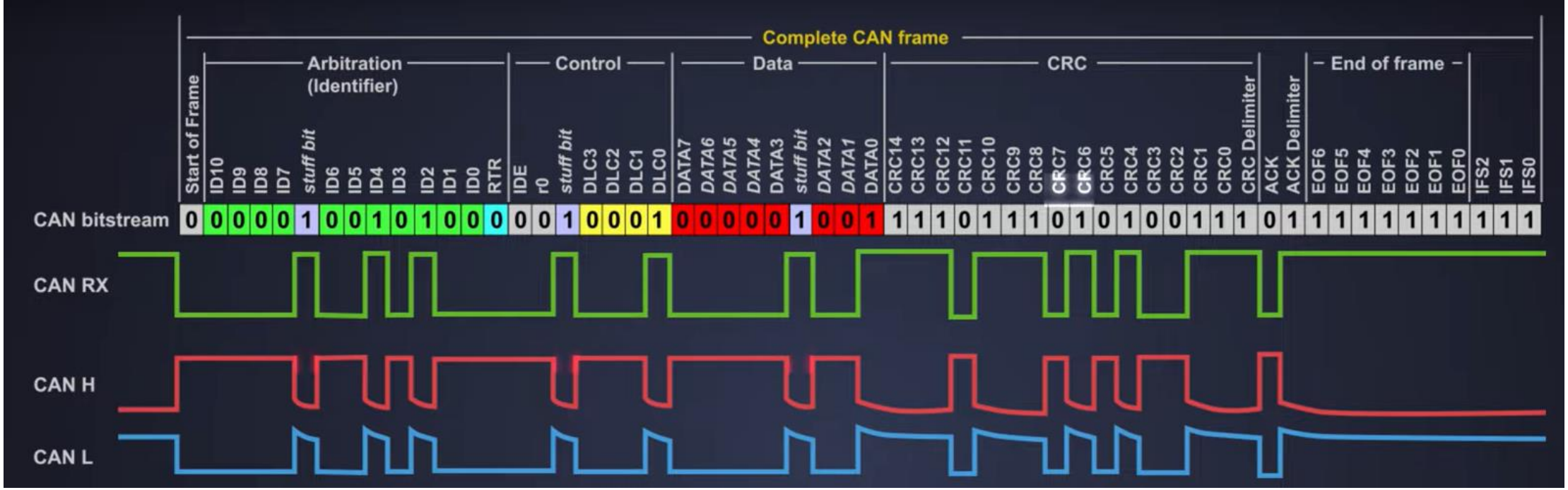


RTR: If 0, it is a normal DATA drama. If 1, it is requesting data to a specific node. Instead of waiting that the node send the data.

IDE (Identifier extension bit): Specify if we are used standard (0) or extended (1) CAN. (allow use the same Bus for both CAN versions)

r0: Reserve for future purpose.

DLC (data length code): 4bit code that specify the number of bytes transferred 8, 16, 32 or 64 bits.



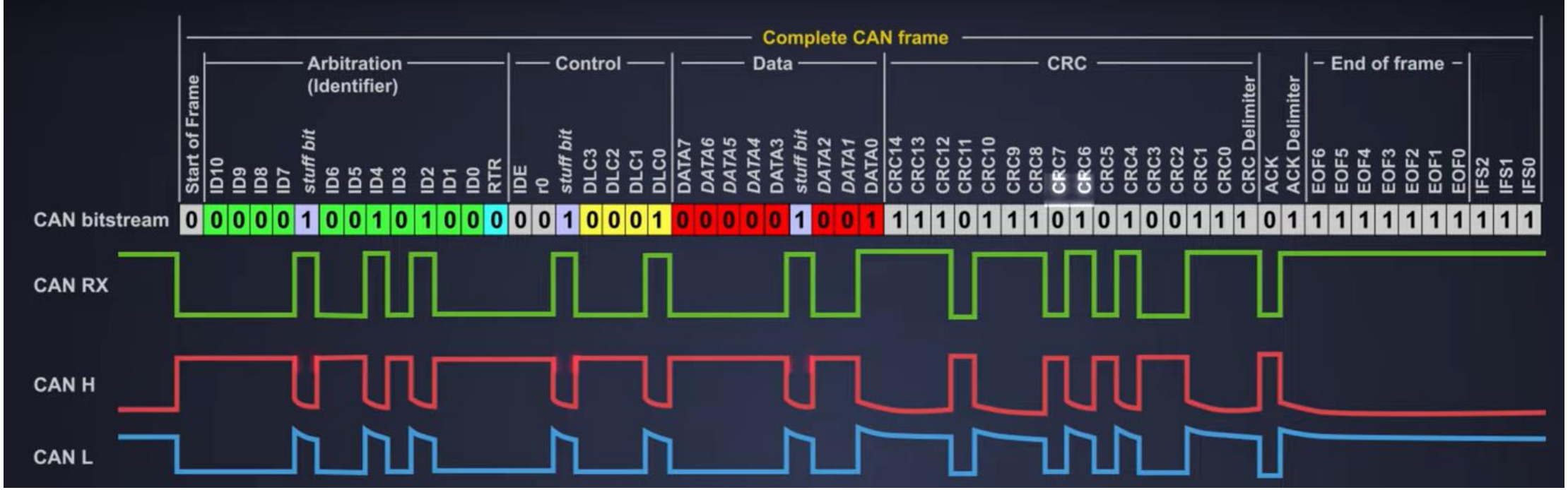
DATA: Are the 8 bits of data sent by each frame. (only 8 bit per frame)

CRC (Cyclic Redundancy Field): also known as safety field.

a) CRC contains 15bit code for check if data was well sent. (algorithm of sum)

b) CRC Delimiter: Checks the form of the CAN frame is correct (FORM check). The receiver check this bit, and if it OK, It interprets the data. Otherwise, it send an error frame! (FORM error). Invalid frame is ignored by the controller.

c) If an error is detected, the receiver notifies immediately to the transmitter, and it sends data again.



ACK: There are two acknowledge bits. When the receiver get the message properly, it must rewrite the ACK bit to indicates that the bit was received successfully.
If the receiver keeps ACK high, the transmitter resend the data.
EOF (End of Frame): It indicates the end of message. Usually all are recessive (1).
IFS (inter frame Space): Mark a space between frames. Indicate the time that the controller has to move the received frame to the message buffer area. It is a time given by the controller to process the package.

Communication Protocols

CAN

Advantages:

- Very robust against EMI (Electromagnetic Interference) (due to differential signals)
- Very robust for error detection (stuff bit, CRC, Delimiter, EOF)
- Less cable required due it its BUS topology.
- Several devices can be connected.
- Arbitration due to Dominant and Recessive lines.

Disadvantages:

- Speed is limited to 1Mbps.
- Require more hardware for generation of its special voltages lines.
- Half-duplex (only one direction data flow at the same time)

Applications

- Mostly automotive, , but also in industrial automation, medical devices, and other domains that require reliable and efficient communication between distributed nodes.

Electrical Engineering Department
Pontificia Universidad Católica de Chile
peclab.ing.uc.cl