BECE403E-EMBEDDED SYSTEM DESIGN

MODULE-6

EMERGING BUS STANDARDS AND COMMUNICATION

MODULE-6

Emerging bus standards and communication

- **UART, SPI, I2C**
- **NFC, CAN**
- **Bluetooth, Zigbee, Wi-Fi**

UART

IMPORTANCE OF COMMUNICATION PROTOCOL

- Embedded systems often require communication between different ICs or different devices, for example data gather by one of the device need to be shared to another device through wired or wireless medium.
- The communication between any two devices can take place only if they are compatible in their connectivity, in terms of hardware and software which can be achieved through the use of a protocol.
- Communication protocols are in short, a set of rules that communication entities must follow in order to exchange information.
- Why communication protocols and bus standards?
  - Formal format for message and rules for communication between devices
  - They cover authentication, error detection and correction, handling of signals
  - Standard bus are used for sending data, address and control signals between controllers and peripherals

UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER

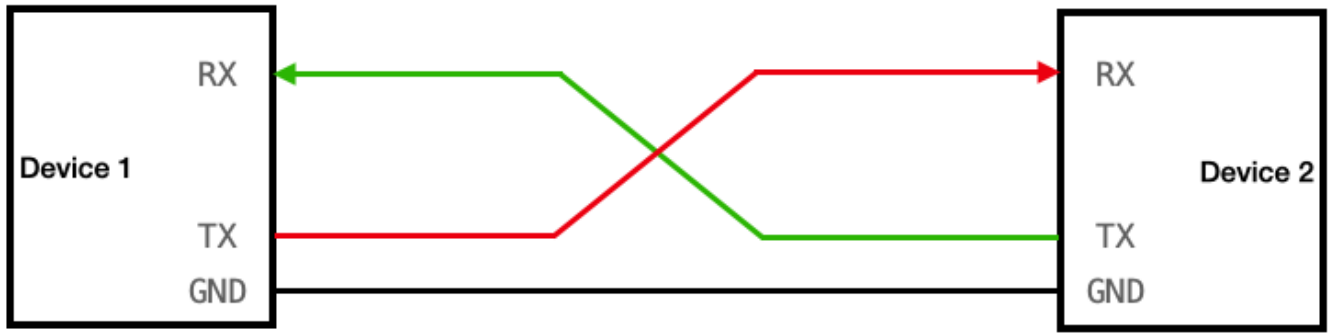
INTRODUCTION

- UART stands for Universal Asynchronous Receiver/Transmitter
- It is one of the simplest and oldest forms of device-to-device digital communication used to transmit and receive serially
- Serial communication refers to the use of a transmission line to sequentially transmit data bit by bit
- Two wires are required to transmit and receive data Tx and Rx Pin
- UARTs has no clock signal and follows asynchronous data communication
- Start and stop bits are used instead of clock signal to data packet for identifying beginning and end of data packet

UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER

INTRODUCTION

- Since there is no clock signal in asynchronous communication, the two communication devices need to agree on a baud rate.
- Different baud rates are 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1000000, 1500000.
- Both UARTs must operate at about the same baud rate, commonly used baud rates are 4800, 9600, 115200, etc.



UART Connection diagram

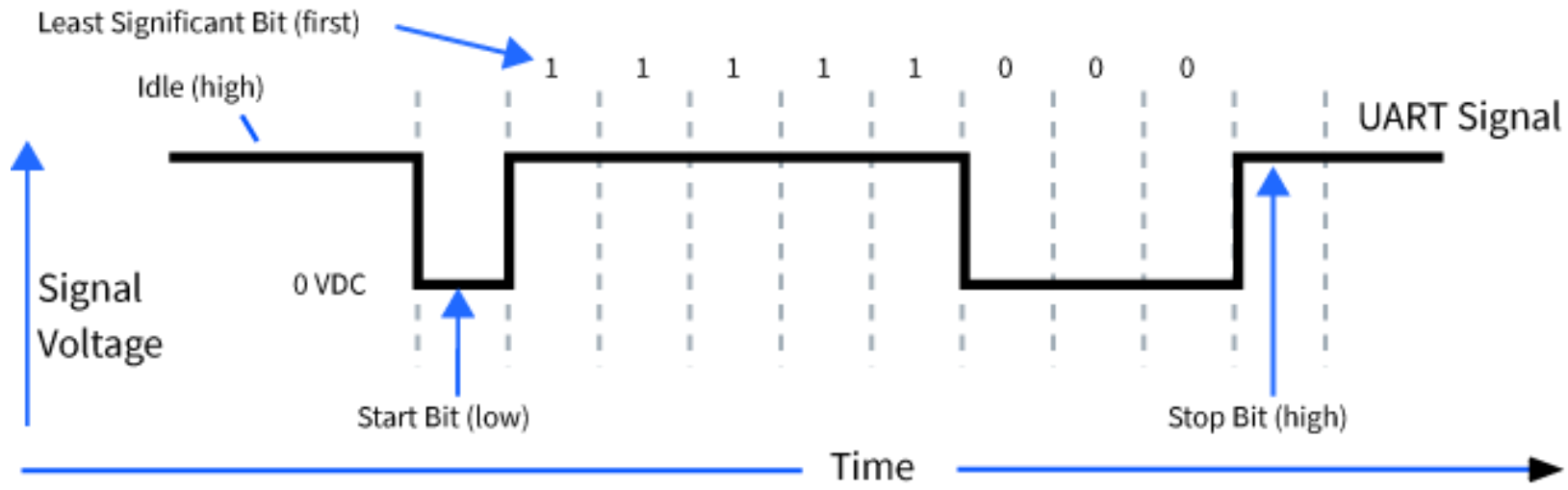
Parameters	Specification
Wires used	2
Maximum speed	115200 baud rate
Synchronous ?	No
Serial?	Yes
Max number of master	1
Max number of slave	1

UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER

INTRODUCTION

Start Bit (1 bit)	Data Frame (5 to 9 Data Bits)	Parity Bits (0 to 1 bit)	Stop Bits (1 to 2 bits)

UART data packet



A common asynchronous serial data format

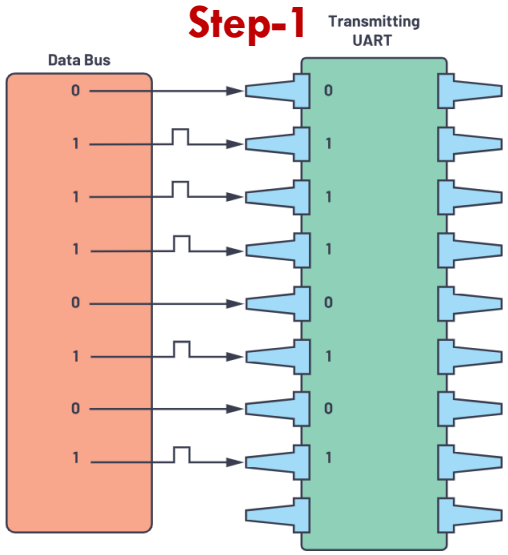
UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER

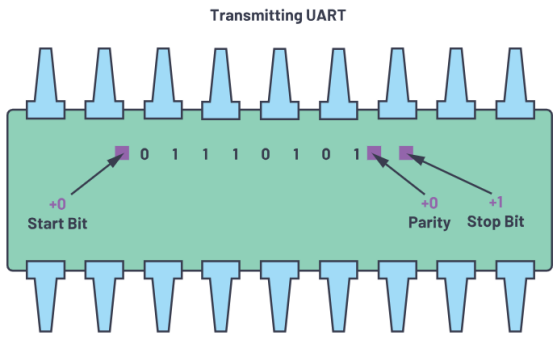
STEPS OF UART DATA COMMUNICATION

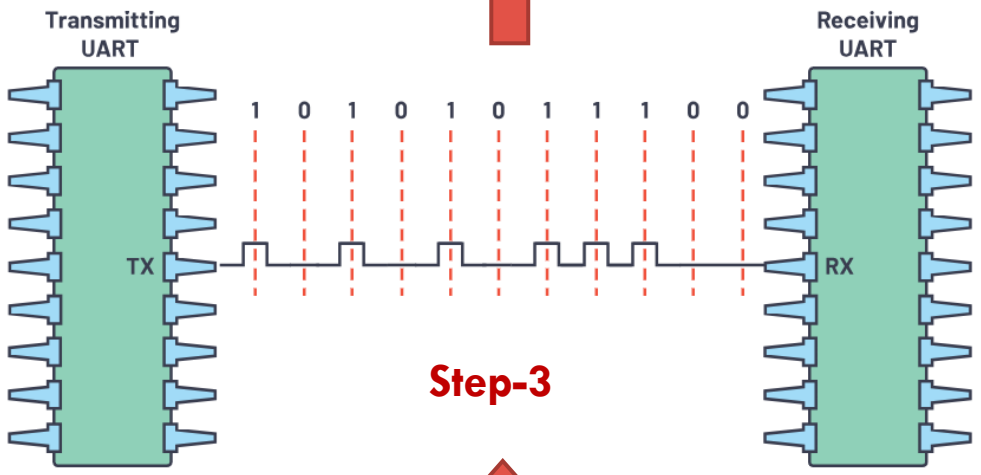
- Step-1: The transmitting UART receives data in parallel from the data bus.
- Step-2: The transmitting UART adds the start bit, parity bit, and the stop bit(s) to the data frame.
- Step-3: The entire packet is sent serially starting from start bit to stop bit from the transmitting UART to the receiving UART. The receiving UART samples the data line at the preconfigured baud rate.
- Step-4: The receiving UART discards the start bit, parity bit, and stop bit from the data frame.
- Step-5: The receiving UART converts the serial data back into parallel and transfers it to the data bus on the receiving end.

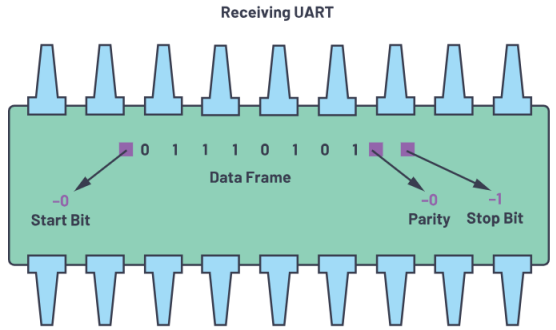
UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER

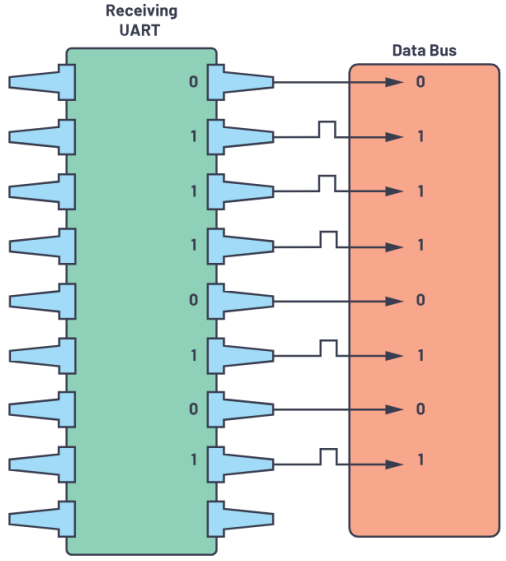
HOW UART WORKS











UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER

ADVANTAGES

- Only uses two wires
- No clock signal is necessary
- Has a parity bit to allow for error checking
- Structure of the data packet can be changed as long as both sides are set up for it
- Well documented and widely used protocol

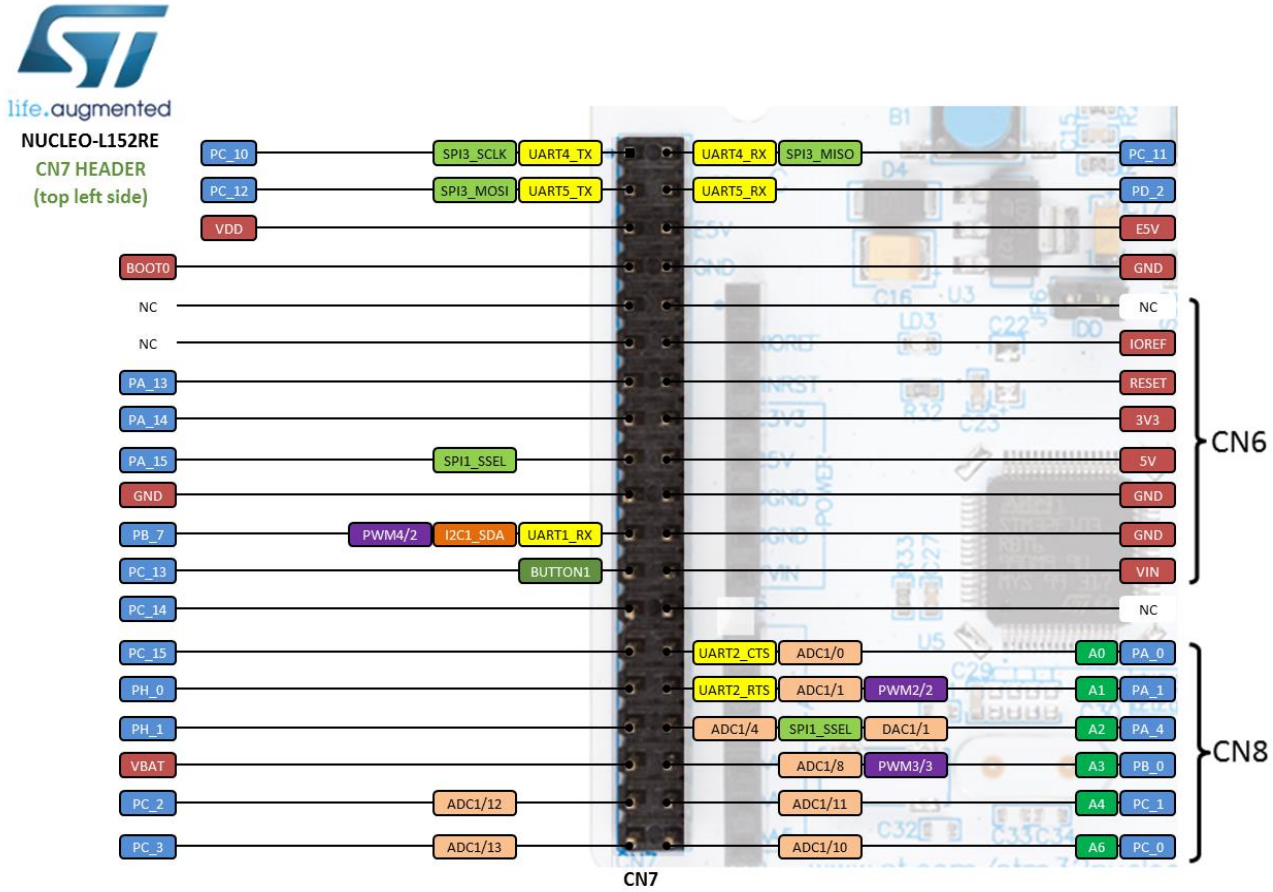
DISADVANTAGES

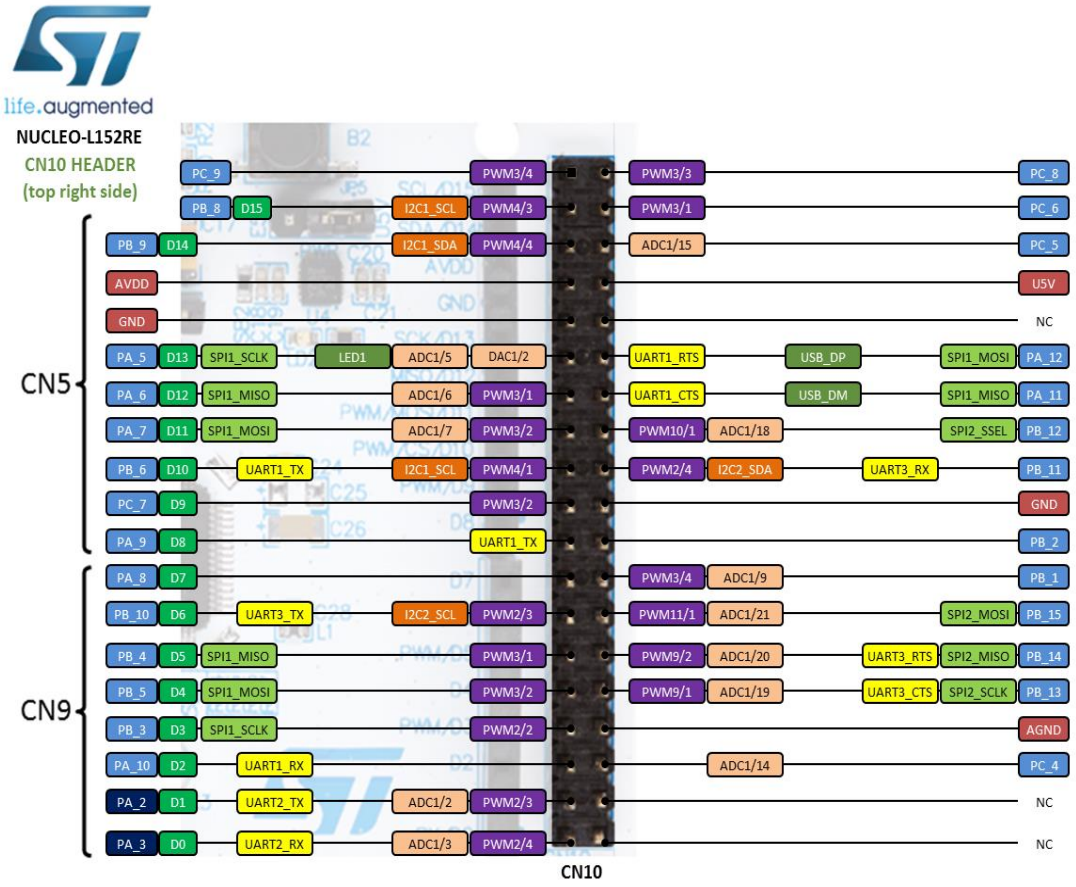
- The size of the data frame is limited to a maximum of 9 bits
- Doesn't support multiple slave or multiple master systems
- The baud rates of each UART must be within 10% of each other

UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER

UART IN NUCLEO BOARD

- STM32 Nucleo board has five UART interfaces and three USART interfaces





UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER

UART IN NUCLEO BOARD

Functions	Usage
Serial	Create a Serial port, connected to the specified transmit and receive pins
baud	Set the baud rate of the serial port
format	Set the transmission format used by the Serial port
putc	Write a character
getc	Read a character
printf	Write a formatted string
scanf	Read a formatted string
readable	Determine if there is a character available to read
writeable	Determine if there is space available to write a character
attach	Attach a function to call whenever a serial interrupt is generated

Summary of Serial API

UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER

EXAMPLE-1 Write a message to a UART device at a 19200 baud rate

```
#include "mbed.h"

Serial device(PC_10, PC_11);  // tx, rx

int main() {

    device.baud(19200);

    device.printf("Hello World\n");

}
```

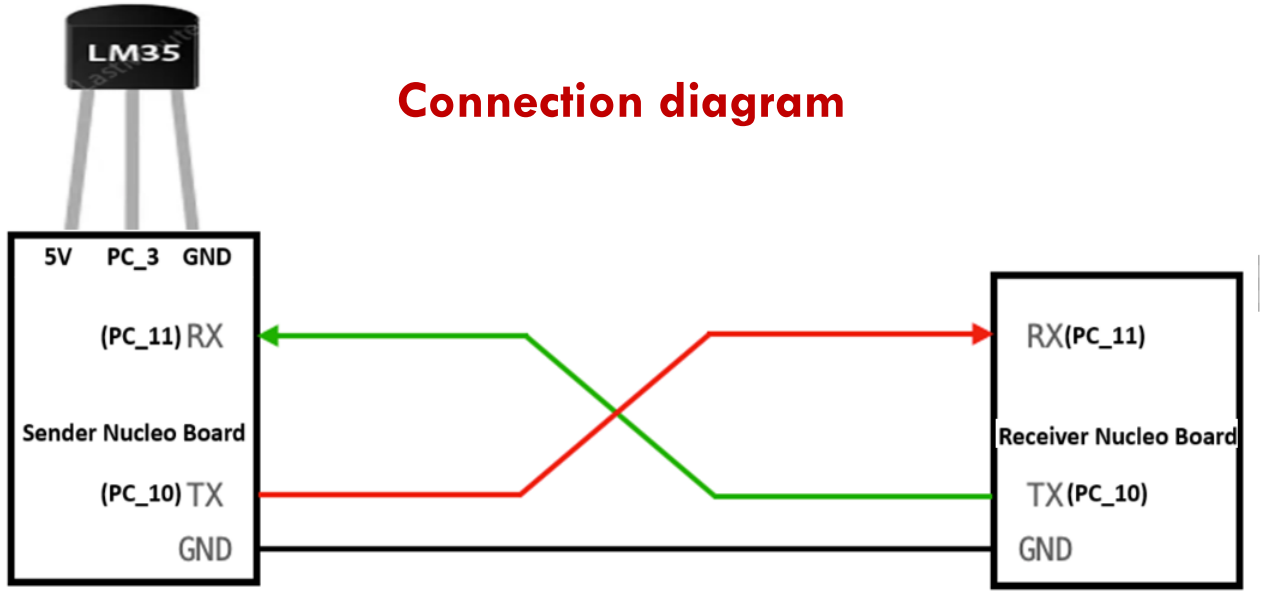
EXAMPLE-2 Provide a serial pass-through between the PC and an external UART

```
#include "mbed.h"
Serial pc(USBTX, USBRX); // tx, rx
Serial device(PC_10, PC_11);  // tx, rx
int main() {
    while(1) {
        if(pc.readable()) {
            device.putc(pc.getc());
        }
        if(device.readable()) {
            pc.putc(device.getc());
        }
    }
}
```

PROGRAMMING GPIO PINS

Example-3 - Transfer temperature value using UART

Write a C++ program with mbed APIs using UART to transmit temperature value from one Nucleo to another. Assume LM35 temperature sensor is connected to Sender Nucleo board which read the temperature value and send it to receiver Nucleo board. The receiver Nucleo board accepts the temperature value and print it on the serial monitor. Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.



Connection diagram

PROGRAMMING GPIO PINS

Example-3 - Transfer temperature value using UART

Sender Program

```
#include "mbed.h"
Serial Sender(PC_10, PC_11);  // tx, rx
AnalogIn ain(PC_3);     ////API for analog input to temp value
int main()
{
Sender.baud(9600);
while(1)
{
float vout = ain*5;          //CONVERTING RECEIVED ANALOG VALUE TO 5V RANGE
float tempc = vout*100;      //CONVERTING VOLTAGE TO TEMPERTURE
Sender.printf("%.2f\n\r",tempc);
wait(5);
}
}
```

PROGRAMMING GPIO PINS

Example-3 - Transfer temperature value using UART

Receiver Program

```
#include "mbed.h"
Serial PC(USBTX,USBRX);
Serial Receiver(PC_10, PC_11);  // tx, rx
int main()
{
float tempc;
Receiver.baud(9600);
while(1)
{
if(Receiver.readable())
{
Receiver.sacnf("%.2f",&tempc);
PC.printf("The current temperature is :%.2f\n\r",tempc);
wait(5);
}
}
```

SPI

SERIAL PERIPHERAL INTERFACE

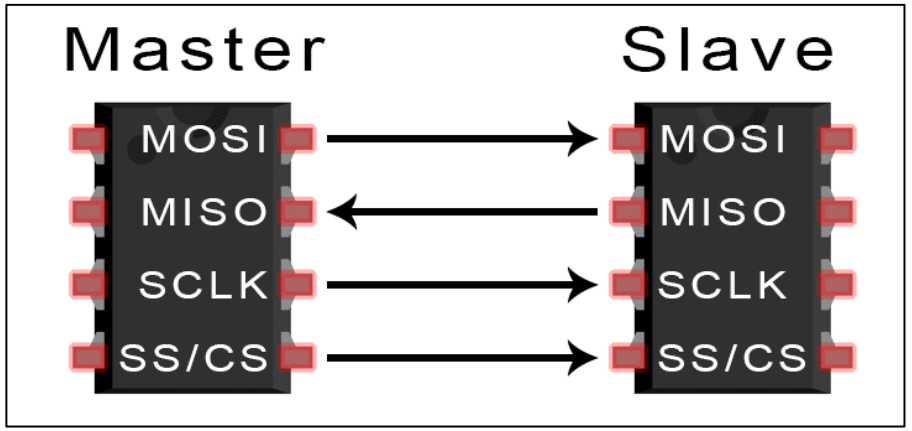
INTRODUCTION

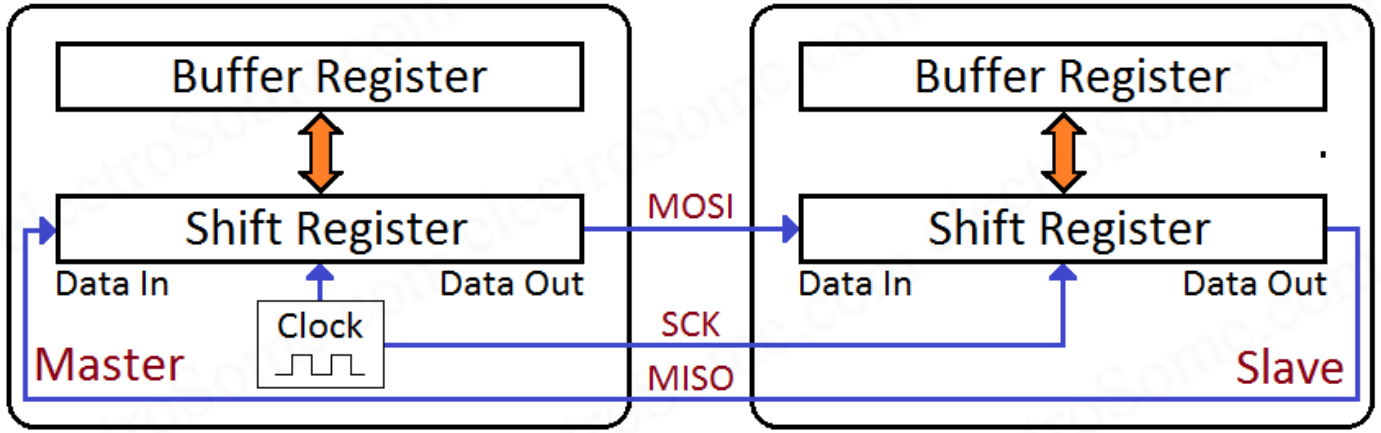
- SPI or Serial Peripheral Interface is a synchronous full-duplex interface serial communication interface used for short-distance communication.
  - Full-duplex means that both devices can send data at the same time.
  - Synchronous means a separate clock line used to specify the rate at which the data bits are transmitted.
- It is mainly used in embedded systems to transfer data from one device to another and it is available in most of the microcontrollers like PIC, AVR, ARM and so on.
- Devices communicating via SPI are in a master-slave relationship, the master is the controlling device (usually a microcontroller), while the slave (usually a sensor, display, or memory chip) takes instruction from the master.
- The simplest configuration of SPI is a single master, single slave system, but one master can control more than one slave. But it does not support multi-master communication.

SERIAL PERIPHERAL INTERFACE

INTRODUCTION

- The SPI bus consists of 4 signals or pins. They are
  - **Master – Out / Slave – In (MOSI):** Master generates data and slave receives
  - **Master – In / Slave – Out (MISO):** Slave generate data and transmit to Master.
  - **Serial Clock (SCLK):** The Master generates clock signal and supplies to the clock input of slave.
  - **Chip Select (CS) or Slave Select (SS):** Master selects a particular slave using chip select.





SERIAL PERIPHERAL INTERFACE

INTRODUCTION

- CLOCK:
  - It synchronizes the output of data bits from master to the sampling of bits by the slave.
  - One bit of data is transferred in each clock cycle, so the speed of data transfer is determined by the frequency of the clock signal.
  - SPI communication is always initiated by the master since the master configures and generates the clock.
- SLAVE SELECT (SS):
  - Ideally slave select pin is kept at high voltage level
  - The master decides which slave it wants to talk by setting particular salves SS as low
  - Master has multiple SS/CS pin to enable multiple slaves in parallel
  - If only single SS pin available, then multiple slave can be connected to master by daisy chain configuration

SERIAL PERIPHERAL INTERFACE

INTRODUCTION

- MOSI AND MISO:
  - The master sends data serially through MOSI line and slaves receives at MOSI pin
  - Slave communicates serially back to master using MISO pin
  - In master to slave communication MSB bit is first sent
  - In slave to master communication LSB bit is first sent

Parameters	Specification
Wires used	4
Maximum speed	10 Mbps
Synchronous ?	Yes
Serial?	Yes
Max number of master	1
Max number of slave	Theoretically unlimited

SERIAL PERIPHERAL INTERFACE

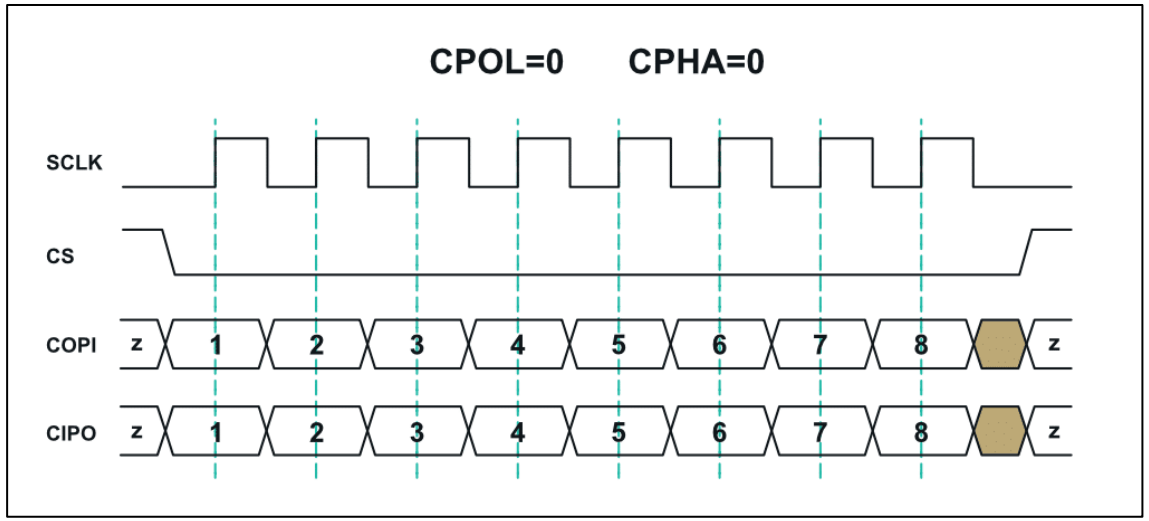
MODE OF OPERATION

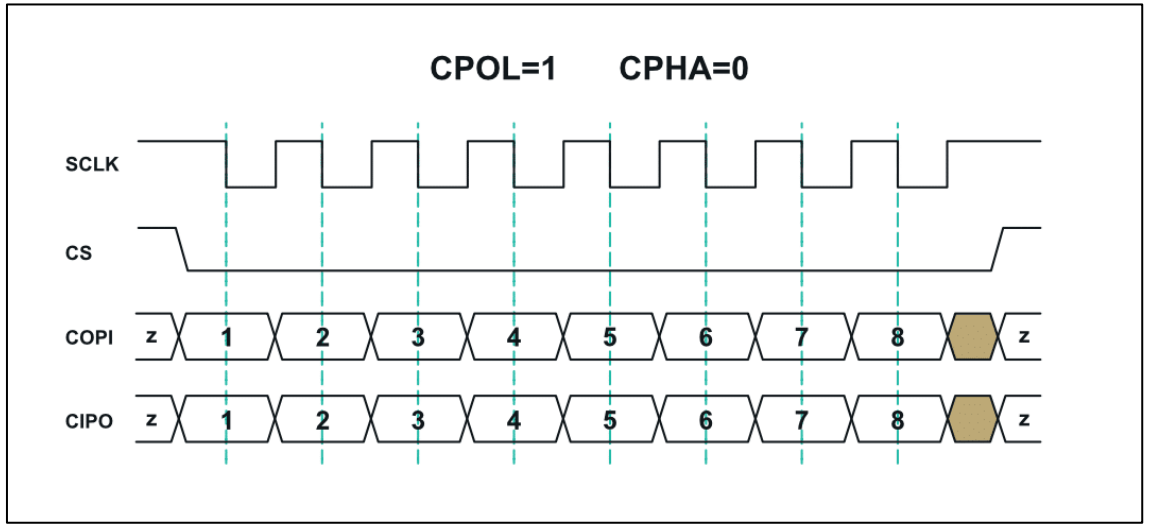
- The clock signal in SPI can be modified using the properties of Clock polarity (CPOL) and Clock phase(CPHA) which define when the bits are output and when they are sampled.
- CPOL refers to the logic at which the clock is set when no data is being transferred.
- CPHA represents whether the data will be read by the devices on the rising or falling edges of the clock
- Four SPI modes are available depending on the states of CPOL and CPHA bits (either 0 or 1).

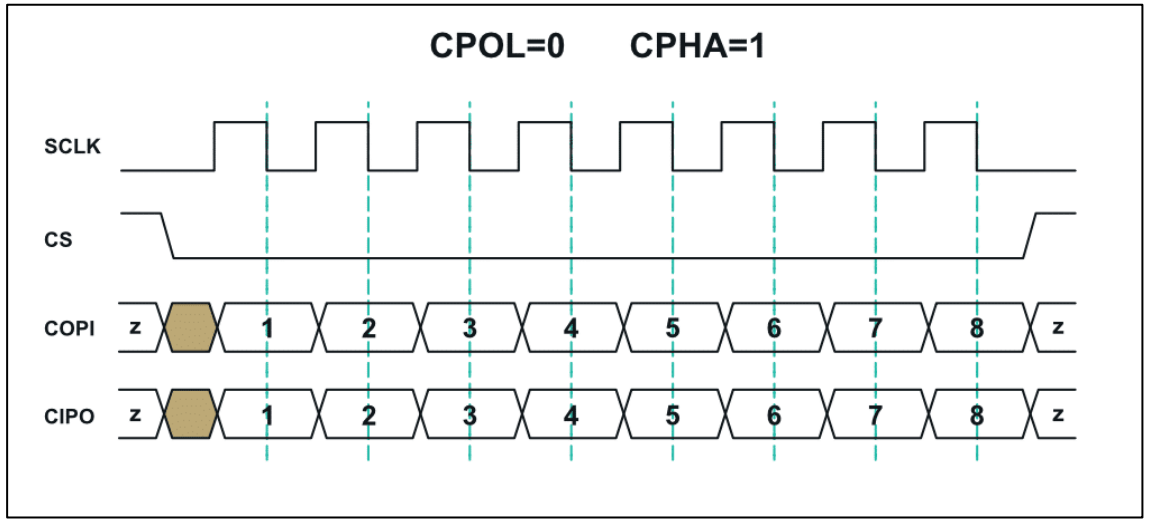
SPI Modes with CPOL and CPHA				
SPI Mode	**CPOL**	**CPHA**	**Clock Polarity in Idle State**	**Clock Phase Used to Sample and/or Shift the Data**
0	0	0	Logic low	Data sampled on rising edge and shifted out on the falling edge
1	0	1	Logic low	Data sampled on the falling edge and shifted out on the rising edge
2	1	0	Logic high	Data sampled on the falling edge and shifted out on the rising edge
3	1	1	Logic high	Data sampled on the rising edge and shifted out on the falling edge

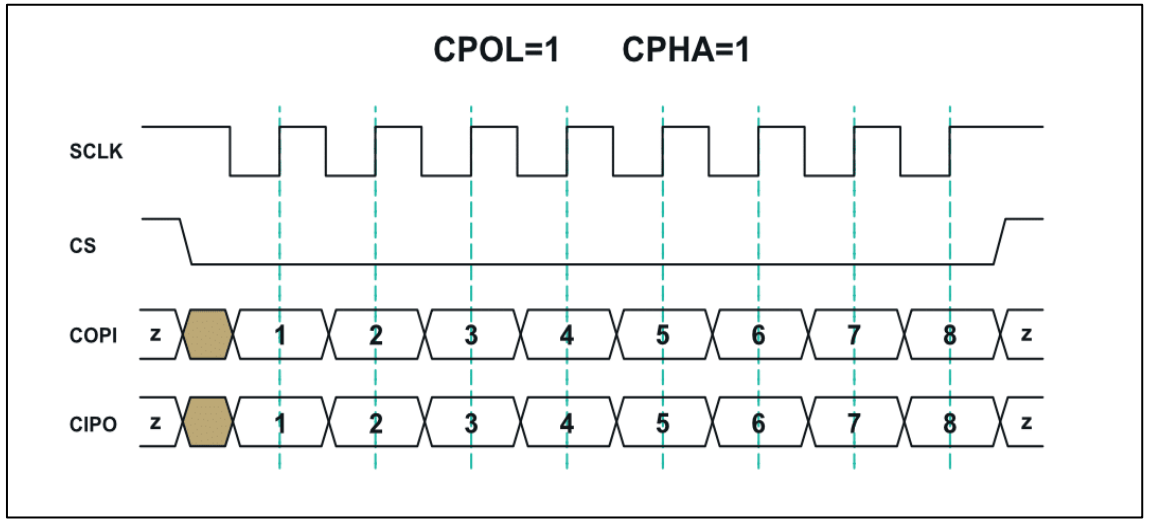
SERIAL PERIPHERAL INTERFACE

MODE OF OPERATION





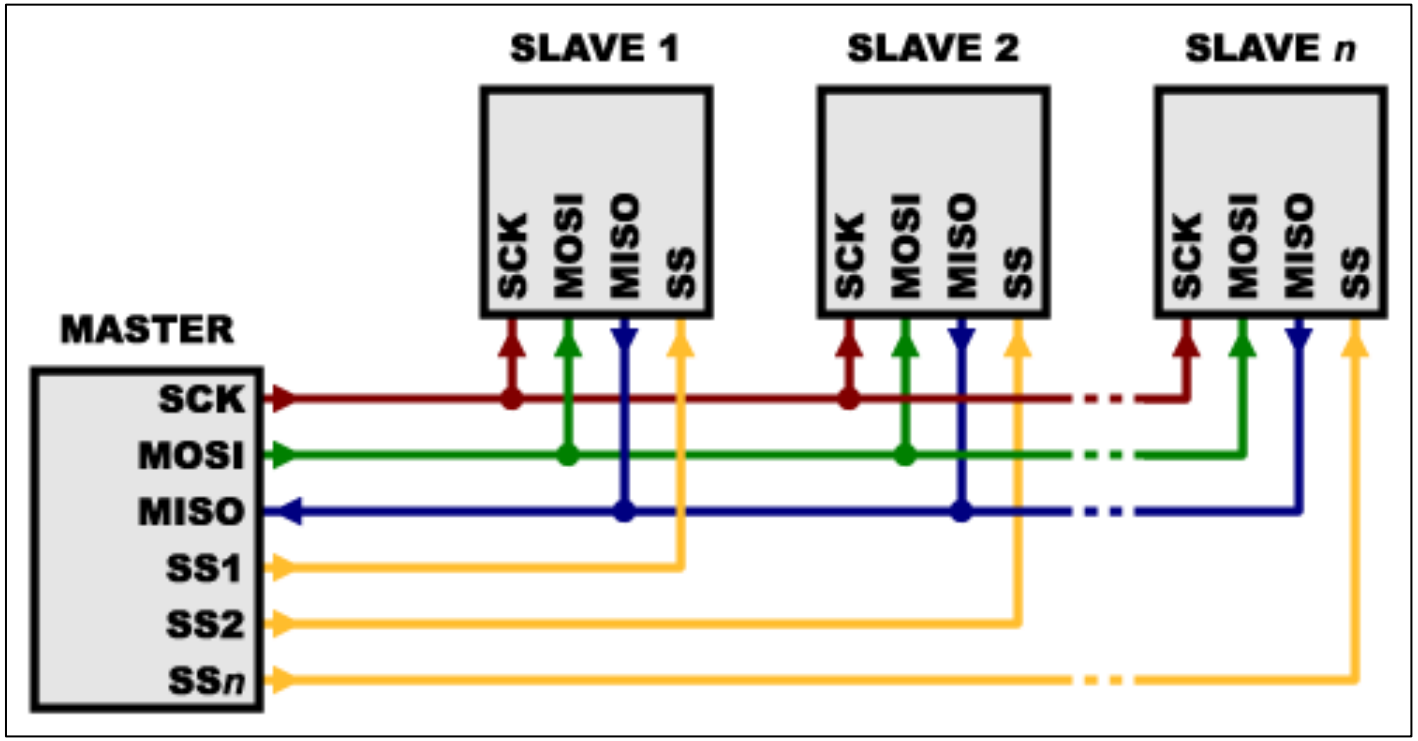




SERIAL PERIPHERAL INTERFACE

SPI CONFIGURATIONS - Independent Slave Configuration

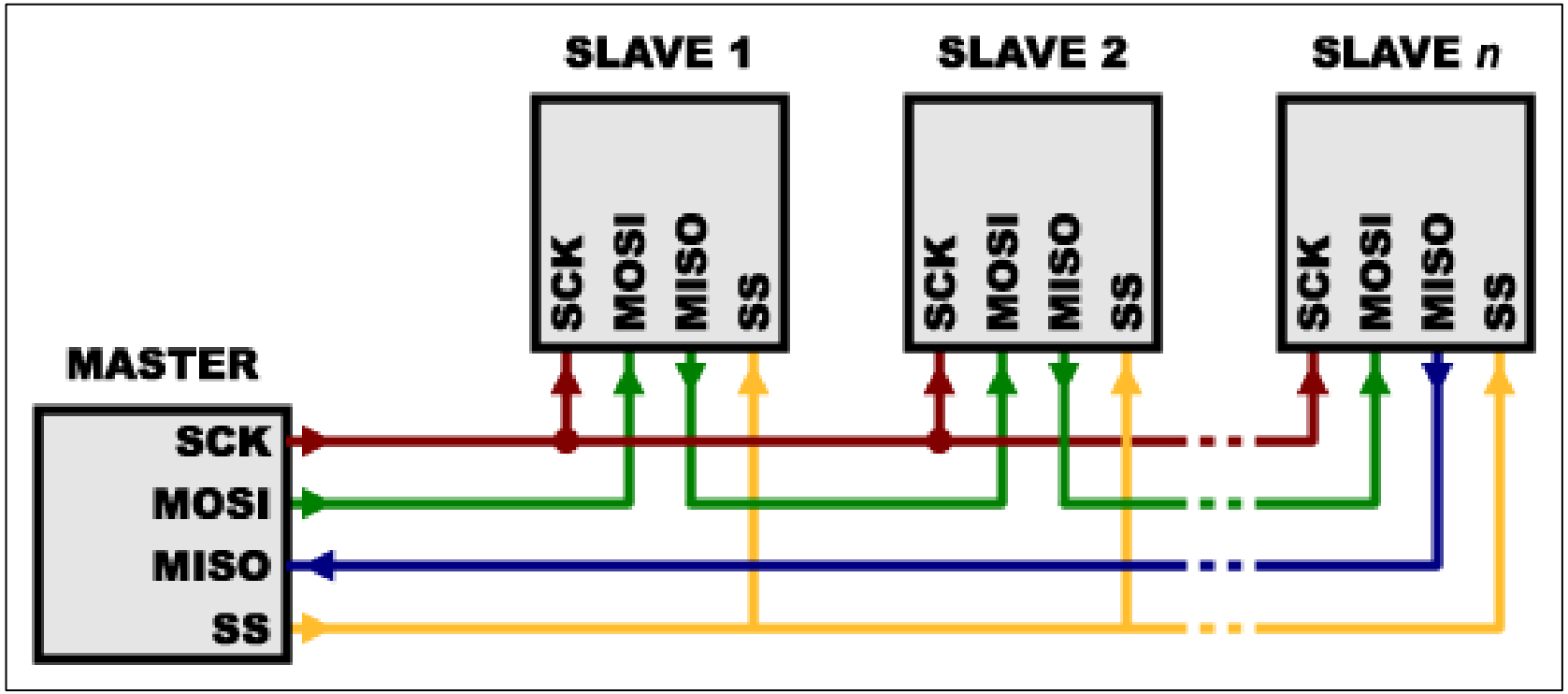
- In independent configuration, all slaves has dedicated select lines and connected individually to master
- Master SCK is connected with all the slave select lines
- MOSI and MISO pins of both master and slaves are connected to each other



SERIAL PERIPHERAL INTERFACE

SPI CONFIGURATIONS - Daisy Chain Configuration

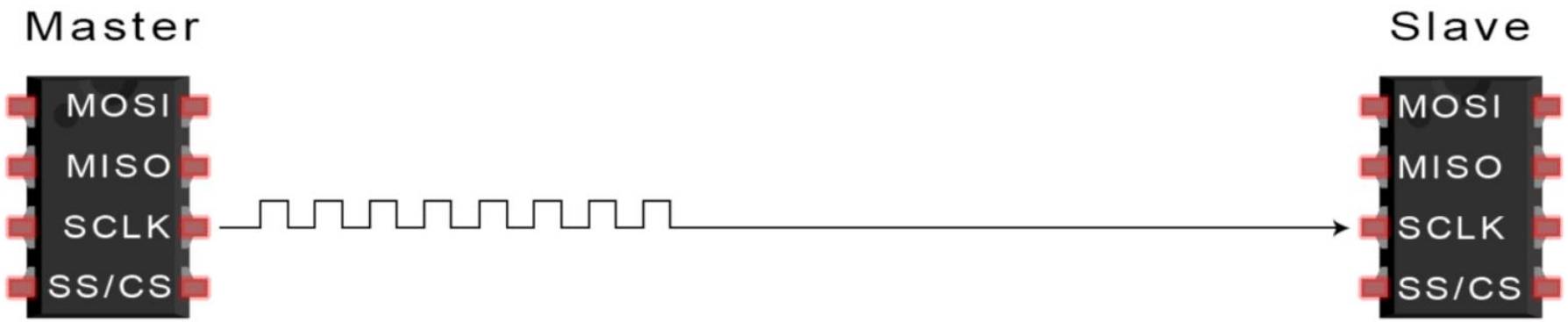
- In Daisy Chain Configuration, only a single Slave Select line is connected to all the slaves
- The MOSI of master is connected to MOSI of slave 1 and MISO of slave 1 is connected to MOSI of salve 2
- The MISO of final slave is connected to MISO of master



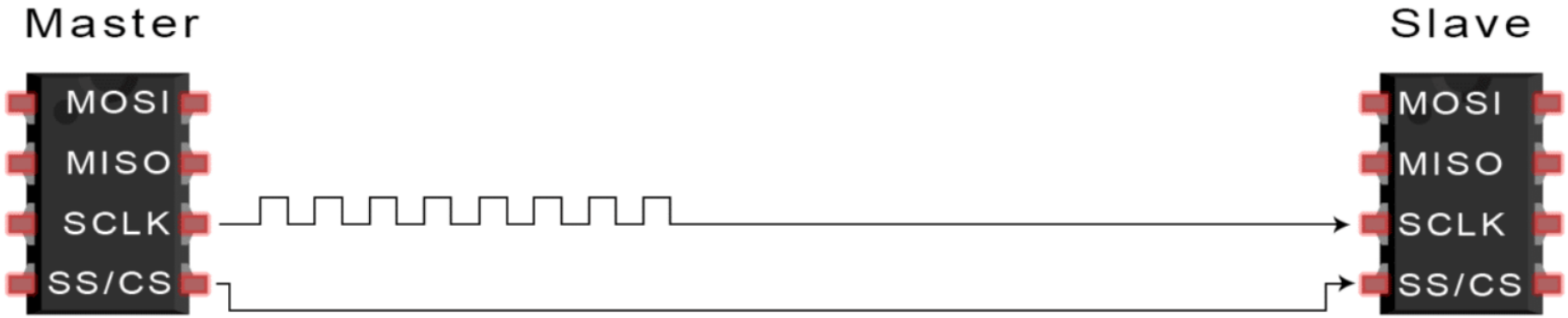
SERIAL PERIPHERAL INTERFACE

STEPS OF SPI DATA COMMUNICATION

- Step-1: The master outputs the clock signal



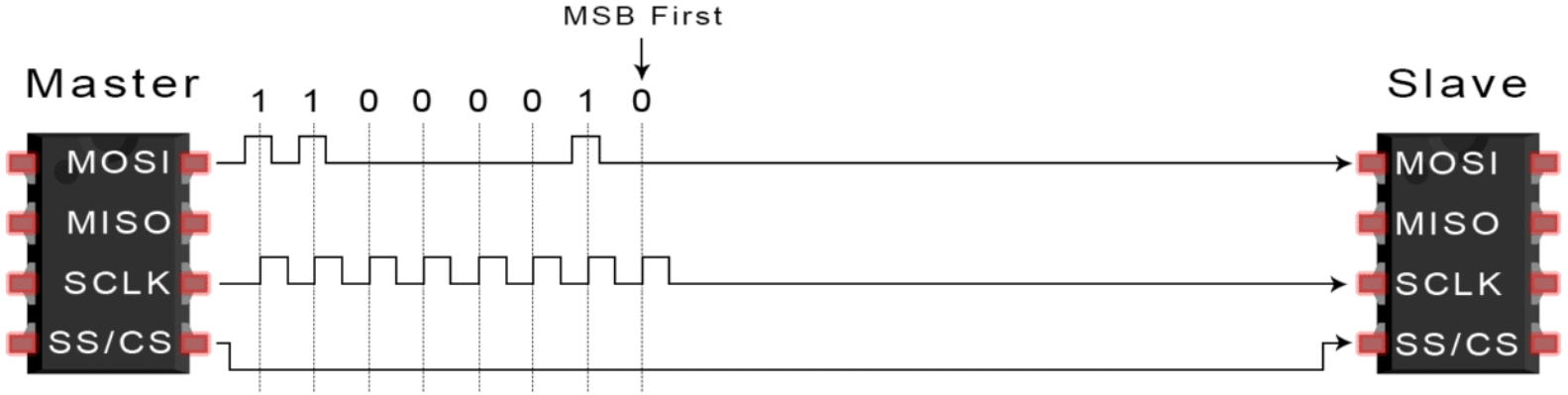
- Step-2: The master make the SS/CS pin to a low voltage state, which activates the slave



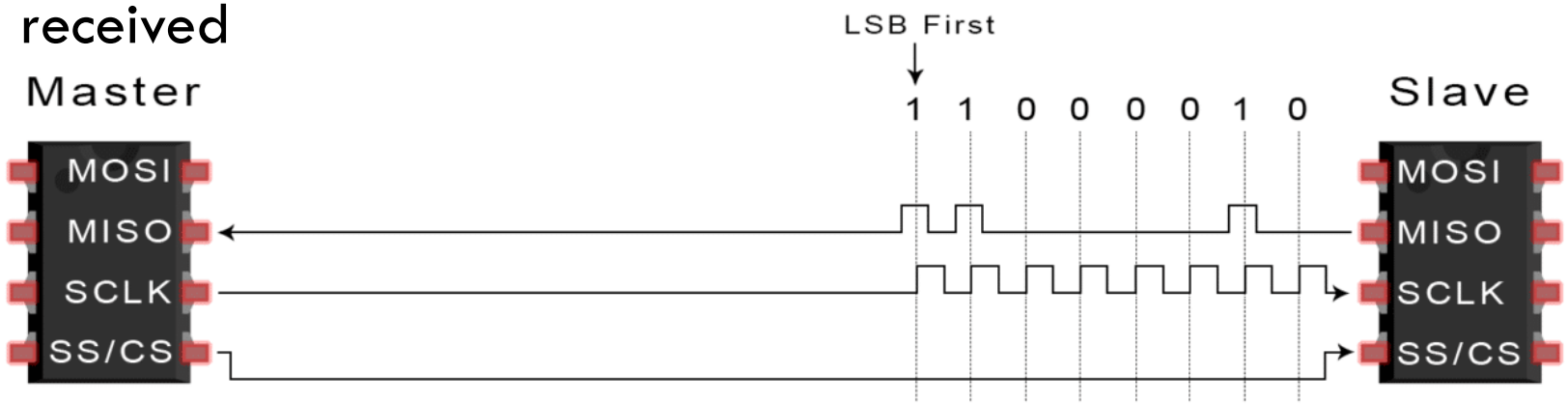
SERIAL PERIPHERAL INTERFACE

STEPS OF SPI DATA COMMUNICATION

- Step-3: In MOSI pin master sends data serially one bit at a time to slave and slaves receives the data in receiving sequence



- Step-4: Slave responds to the master by sending one bit at a time and master reads as the message received



SERIAL PERIPHERAL INTERFACE

ADVANTAGES

- Support full duplex communication
- High-speed of data communication compared to UART and I2C
- Simple slave addressing system
- Master devices handles all the Slaves, hence no chance of conflict
- Continuous streaming of data without start and stop bits

DISADVANTAGES

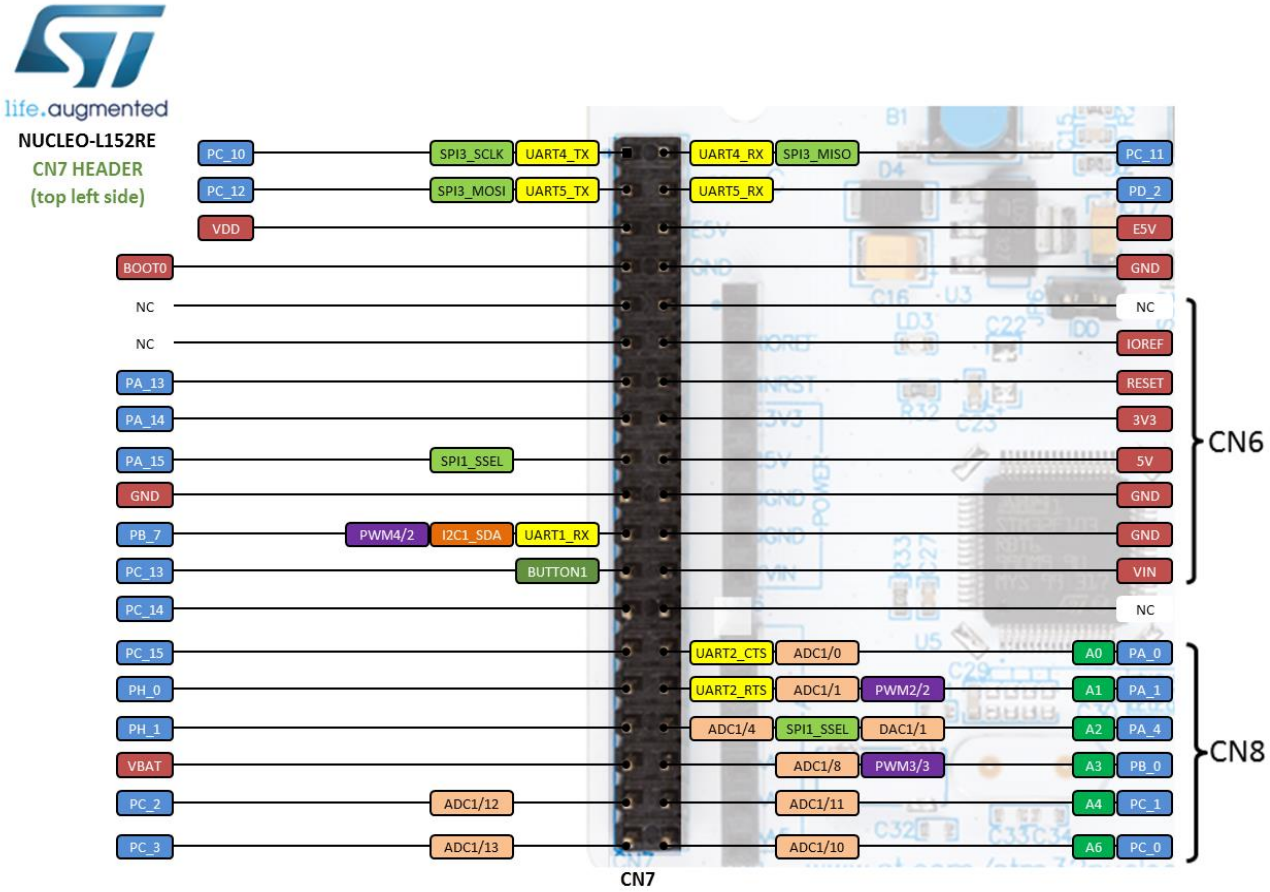
- Uses four pins (I2C and UARTs use two)
- Only single master is allowed in SPI
- Dedicated pin is required for slave on master (CS/SS)
- Acknowledgement mechanism is not provided

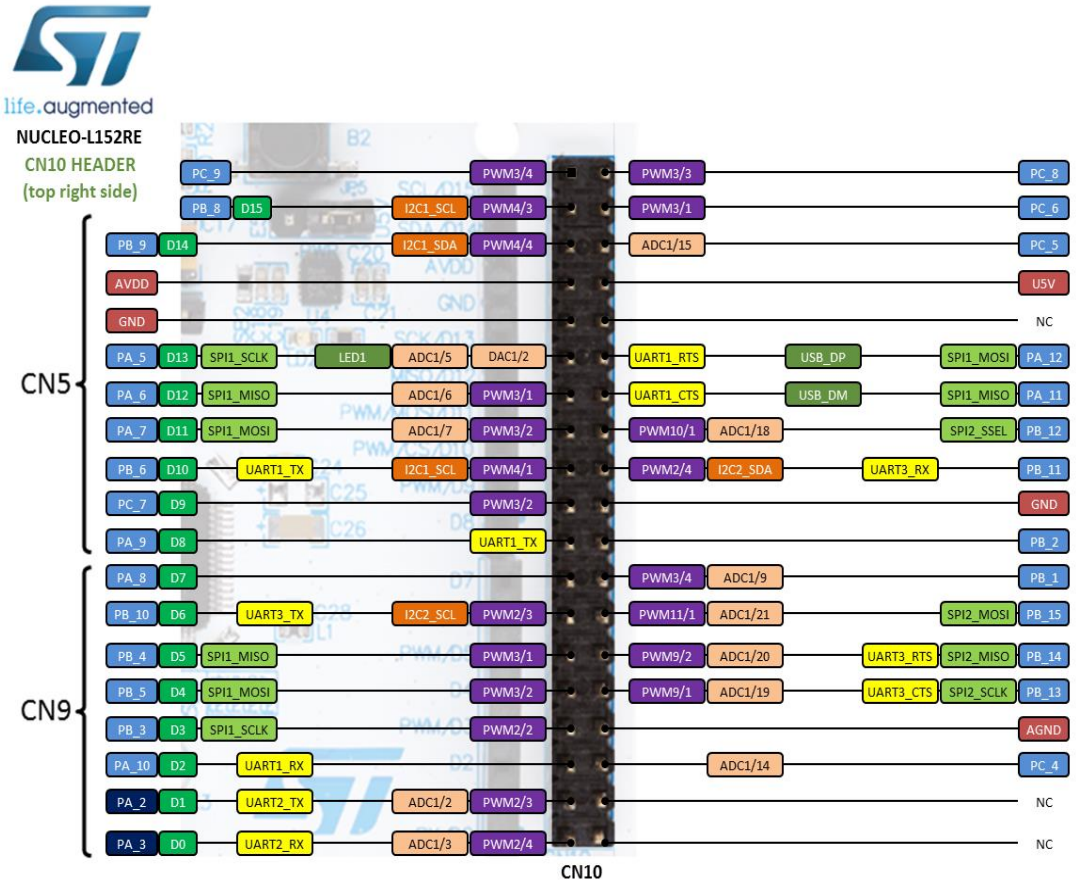
SERIAL PERIPHERAL INTERFACE

SPI IN NUCLEO BOARD

- NucleoL152RE has three SPI Interface and default settings of SPI interface is:

  Default clock frequency of 1 MHz, Default data length of 8 bits, Default mode of 0





SERIAL PERIPHERAL INTERFACE

SPI IN NUCLEO BOARD

- The mbed SPI master library functions are shown in the table below:

Function	Usage
SPI	Create a SPI master connected to the specified pins
format	Configure the data transmission mode and data length
frequency	Set the SPI bus clock frequency
write	Write to the SPI slave and return the response

- The mbed SPI slave library functions are shown in the table below:

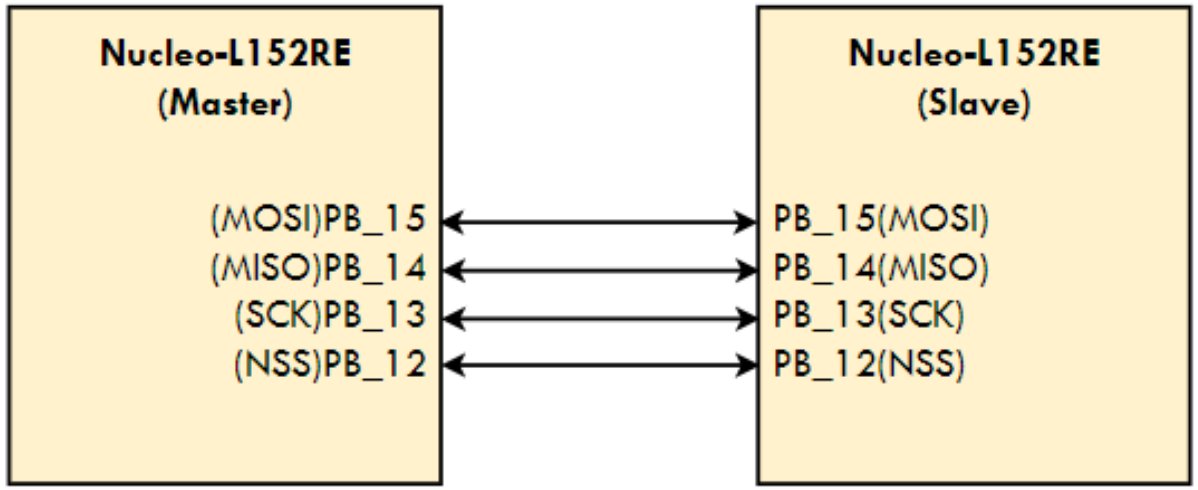
Function	Usage
SPISlave	Create a SPI slave connected to the specified pins
format	Configure the data transmission format
frequency	Set the SPI bus clock frequency
receive	Poll the SPI to see if data has been received
read	Retrieve data from receive buffer as slave
reply	Fill the transmission buffer with the value to be written out as slave on the next received message from the master

SERIAL PERIPHERAL INTERFACE

EXAMPLE-1 Read the input from Master and display on Slave

Write a program to implement a SPI communication between two Nucleo boards. Configure one of the Nucleo as master and other as slave. Establish a SPI communication between master and slave display each key press on the master's Teraterm to the slave Teraterm terminal. Design and implement this logic on the STM 32 Nucleo L152RE board using Keil studio could platform.

Connection Diagram



SERIAL PERIPHERAL INTERFACE

EXAMPLE-1 Read the input from Master and display on Slave

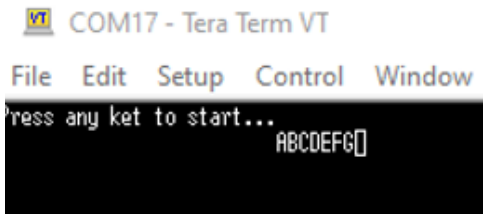
Program (Master)

```
1   //*****Master*********
2   #include "mbed.h"
3   SPI spi(PB_15,PB_14,PB_13);
4   DigitalOut cs(PB_12);
5   Serial pc(USBTX,USBRX);
6   int main(){
7   char send_val;
8   pc.printf("Press any ket to start...\n");
9   while(1){
10  send_val = pc.getc();
11  pc.printf("%c",send_val);
12  cs=0;
13  spi.write(send_val);
14  cs=1;
15  wait(0.01);
16  }
17  }
```

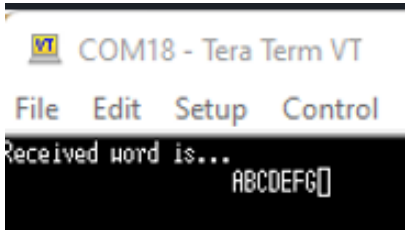
Program (Slave)

```
1   //*********SLAVE**************
2   #include "mbed.h"
3   SPISlave spi(PB_15,PB_14,PB_13, PB_12);
4   Serial pc(USBTX,USBRX);
5   char recd_val;
6   int main(){
7   pc.printf("Received word is...\n");
8   while(1){
9   if(spi.receive()){
10  recd_val=spi.read();
11  pc.printf("%c",recd_val);
12  }
13  }
14  }
```

Output (Master)



Output (Slave)

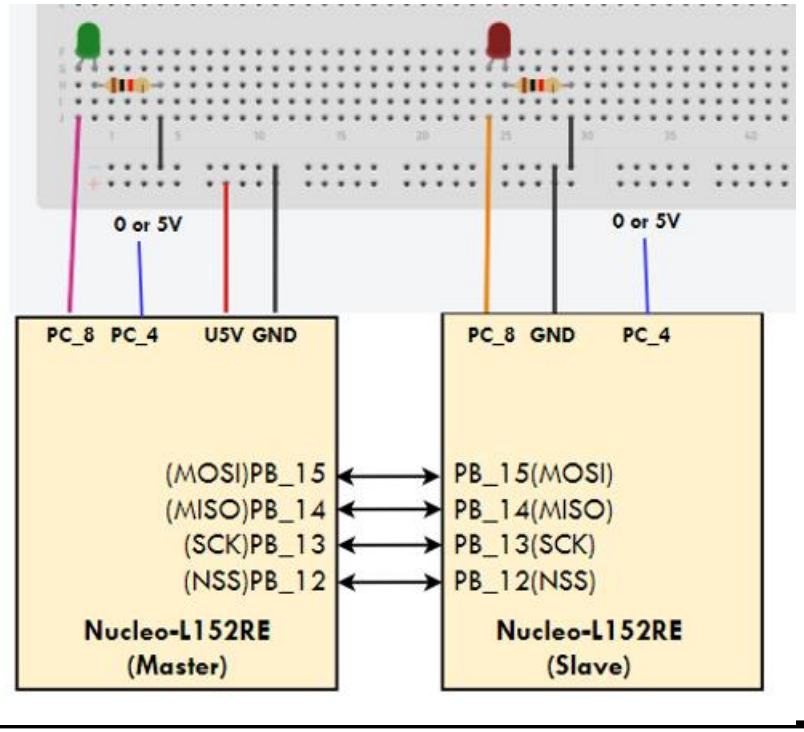


SERIAL PERIPHERAL INTERFACE

EXAMPLE-2 Exchange of button press state between Master and Slave

Write a program to implement a SPI communication between two Nucleo boards. Configure one of the Nucleo as master and other as slave. Both Nucleo are attached with a LED & a push button separately. Master LED can be controlled by using slave Nucleo's push button and slave Nucleo's LED can be controlled by master Nucleo's push button using SPI communication protocol. Design and implement this logic on the STM 32 Nucleo L152RE board using Keil studio could platform.

Connection Diagram



SERIAL PERIPHERAL INTERFACE

EXAMPLE-2 Exchange of button press state between Master and Slave

Program (Master)

```
#include "mbed.h"
SPI ser_port(PB_15,PB_14,PB_13); //MOSI,MISO,SCK
DigitalOut cs(PB_12);            // CS or SS
Serial pc(USBTX,USBRX);
DigitalOut led(PC_8);
DigitalIn switch_ip(PC_4);
int switch_word_tx;
char tx_val;
int main()
{
while(1)
{
switch_word_tx=0x00;
if(switch_ip==1)
switch_word_tx=switch_word_tx|0x01;
//pc.printf("Switch status tx %x\r\n",switch_word_tx);
cs=0;
tx_val=ser_port.write(switch_word_tx);
cs=1;
//pc.printf("during tx %x\r\n",tx_val);
wait(1);
led=0;
tx_val=tx_val&0x01;
//pc.printf("after tx %x\r\n",tx_val);
if(tx_val==1)
led=1;
}
}
```

Program (Slave)

```
//********SLAVE***********
#include "mbed.h"
SPISlave ser_port(PB_15,PB_14,PB_13,PB_12); //MOSI,MISO,SCK,CS
Serial pc(USBTX,USBRX);
DigitalOut led(PC_8);
DigitalIn switch_ip(PC_4);
int switch_word_rx;
char rx_val;
int main()
{
while(1)
{
switch_word_rx=0x00;
if(switch_ip==1)
switch_word_rx=switch_word_rx|0x01;
//pc.printf("Switch status rx %d\r\n",switch_word_rx);
if(ser_port.receive())
{
rx_val=ser_port.read();
//pc.printf("before reply %x\r\n",rx_val);
wait(1);
ser_port.reply(switch_word_rx);
//pc.printf("after reply %x\r\n",rx_val);
}
led=0;
rx_val=rx_val&0x01;
if(rx_val==1)
led=1;
}
}
```

SERIAL PERIPHERAL INTERFACE

Exercise-1 Read temperature sensor value of slave and display on master LCD

Write a program to implement a SPI communication between two STM32 Nucleo boards. Configure one of the Nucleo board as master and other as slave. Master Nucleo board connected with LCD and Nucleo board connected with temperature sensor LM35. Transfer temperature value read by slave to master and display it on LCD using SPI communication protocol. Design and implement this logic on the STM 32 Nucleo L152RE board using Keil studio could platform.

I2C

INTER-INTEGRATED CIRCUIT

NEED FOR I2C PROTOCOL

- Limitations of UART
  - × Not suited for multiple devices communication, only two devices can communicate with each other.
  - × Highest data rate for most of the UART device is115.2kbps which is still a low speed.
  - × There are more chances of losing the data over UART protocol because UART doesn't use any clock source for synchronizing the data
- Limitations of SPI
  - × It cannot support multiple masters communicating with multiple slave.
  - × Require minimum of 4 wire interface which require more space in IC.
- Because of the drawbacks of UART and SPI, there was need of a protocol which can decrease the number of wires required for communication, have flexible data rates along with multiple master and multiple slave communication.
- This brought I2C to the picture which have all the desired features of a multi-bus communication.

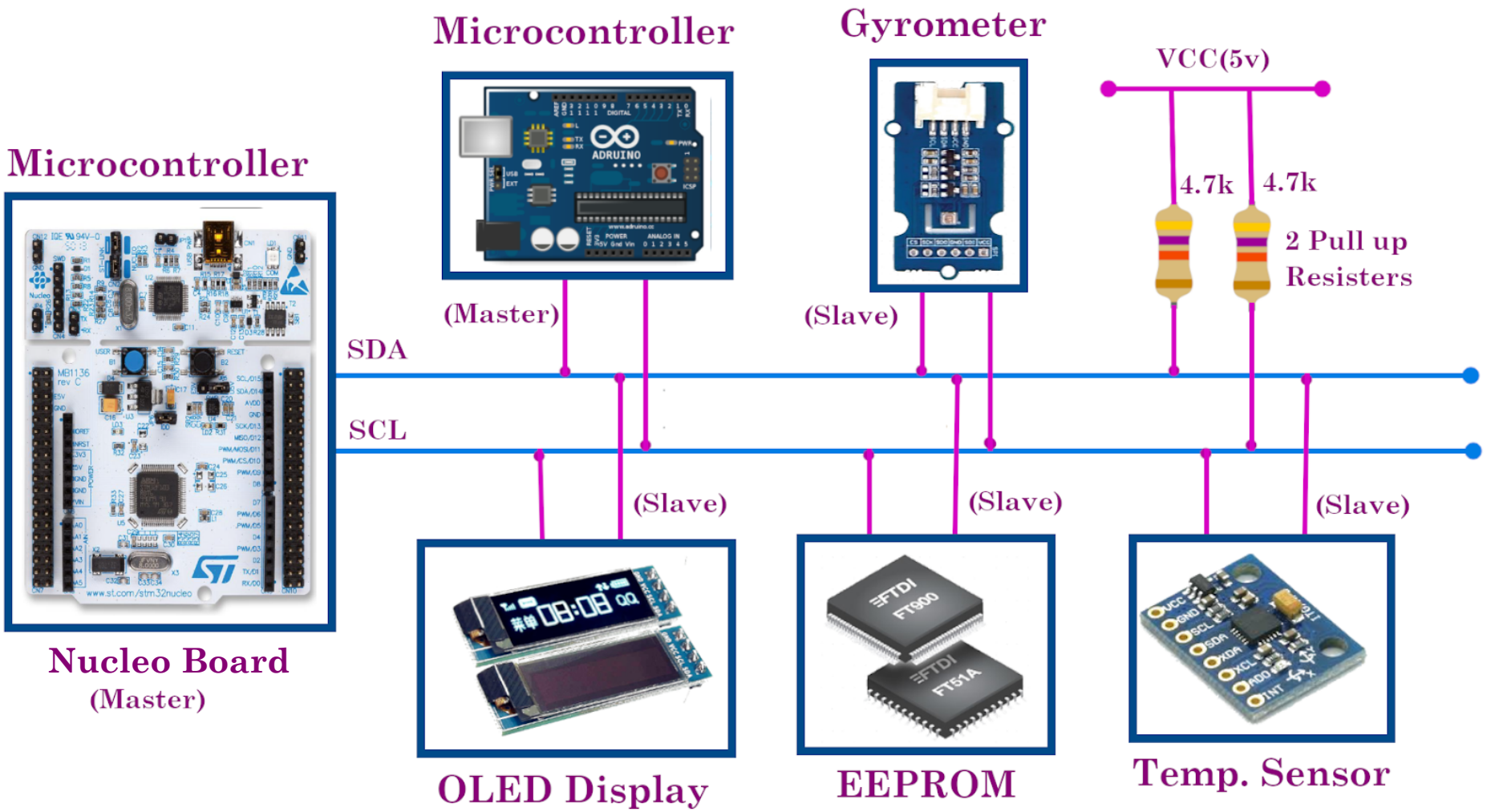
INTER-INTEGRATED CIRCUIT

INTRODUCTION

- I2C, stands for Inter-Integrated Circuit (also known as Two Wired Interface(TWI)), is a synchronous serial half-duplex communication protocol that enables digital devices to communicate with each other over short distances.
- It was originally designed by Philips Semiconductor in 1982 to facilitates the exchange of data between a master device (usually a microcontroller or a processor) and multiple slave devices (such as sensors, memory chips, or peripheral devices) on a shared bus.
- I2C has the best features of both SPI and UART protocols and it can support multi-master and multi-slave configuration.
- I2C protocol use only two pins (SDC and SCL) and data is send bit by bit on SDA line
  - SDA (Serial Data) – Bidirectional connection between master and slave
  - SCL (Serial Clock) – clock signal

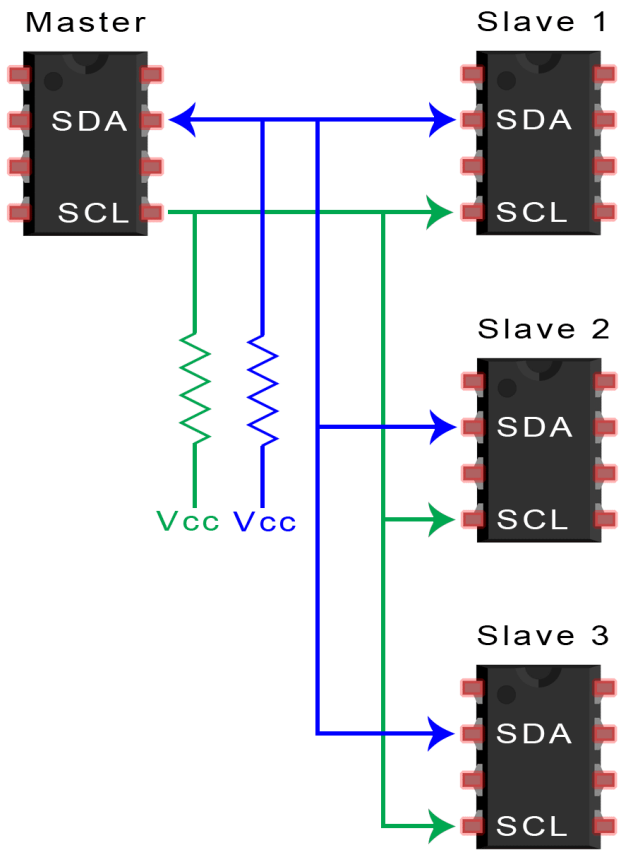
INTER-INTEGRATED CIRCUIT

I2C CONNECTION DIAGRAM

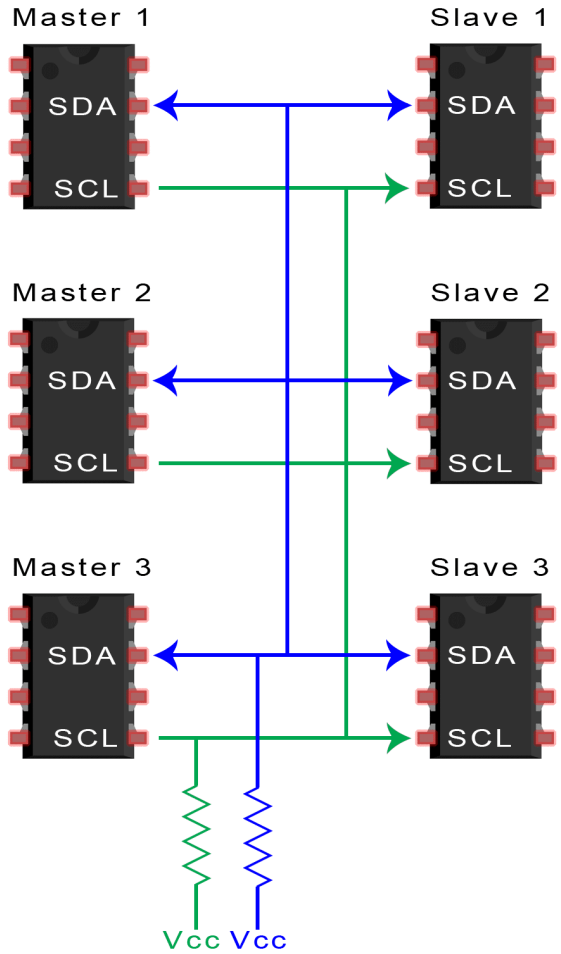


INTER-INTEGRATED CIRCUIT

I2C CONFIGURATIONS



Single Master with Multiple Slaves



Multiple Master with Multiple Slaves

INTER-INTEGRATED CIRCUIT

INTRODUCTION

- I2C is a synchronous protocol, hence the output is synchronized using clock signal, a Common clock signal is shares between master and slave.
- Master always controls the clock signal and master addresses the slave devices via SCL
- Data is transferred as messages in I2C and messages are broken to frames of data. Each message contains a binary address frame of the slave and data frames
- One or many data frames with data can be transferred in one message. Data on the I2C-bus can be transferred at different speed as shown in below table.

Type of Mode	Data Rate
Standard-mode (SM)	100 kbit/s
Fast-mode (FM)	400 kbit/s
Fast-mode Plus (FM+)	1 Mbit/s
High-speed mode (HS)	3.4 Mbit/s

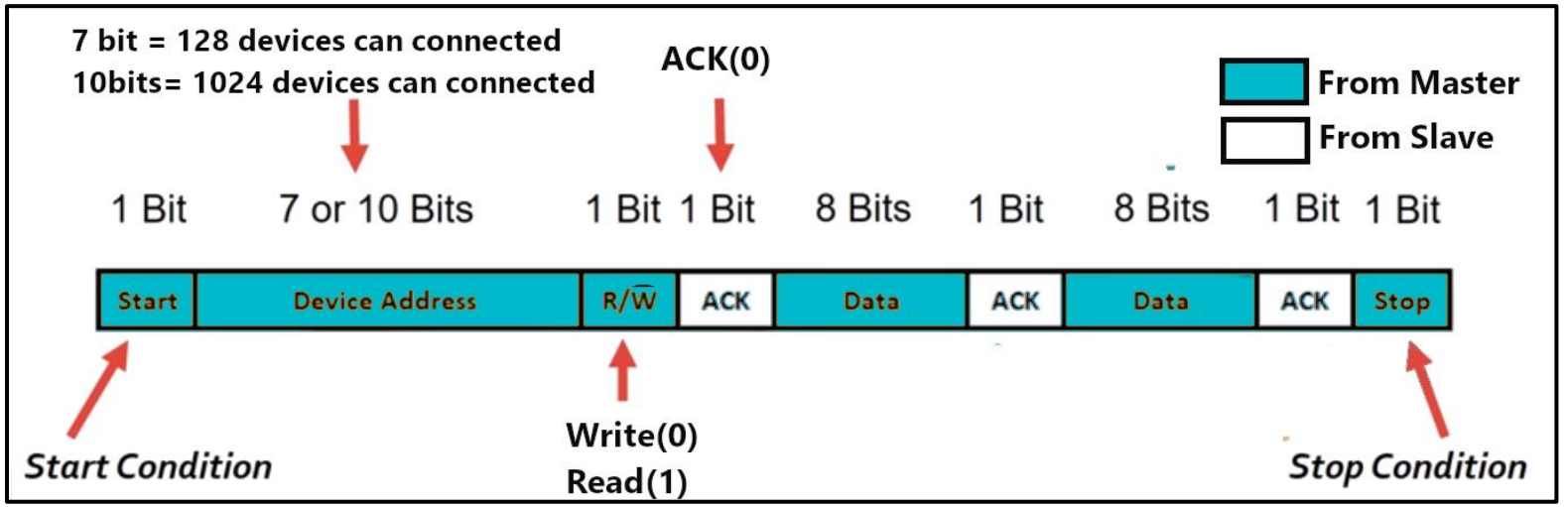
INTER-INTEGRATED CIRCUIT

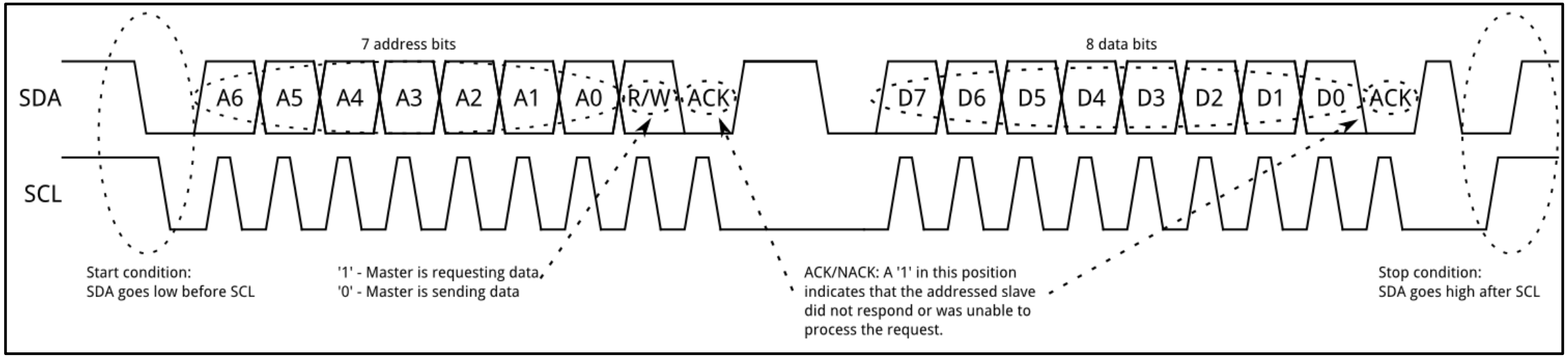
INTRODUCTION

- Use of a pull-up resistor in I2C:
  - ✓ They are used to ensure that the signal lines (SDA and SCL) remain in a known state when they are not actively being driven low by the bus master or slave devices.
  - ✓ The pull-up resistor helps to prevent the signal lines from floating, which can cause undefined behaviour in the communication.
  - ✓ When no device is actively pulling the signal line low, the pull-up resistor ensures that the line is pulled high to the logic high level.
  - ✓ This prevents electrical noise or other interference from causing the line to be misinterpreted as a logic low level.
  - ✓ A common value for these resistors is between 4.7kΩ and 10kΩ.

INTER-INTEGRATED CIRCUIT

I2C FRAME FORMAT





Data Transfer on the I2C bus

INTER-INTEGRATED CIRCUIT

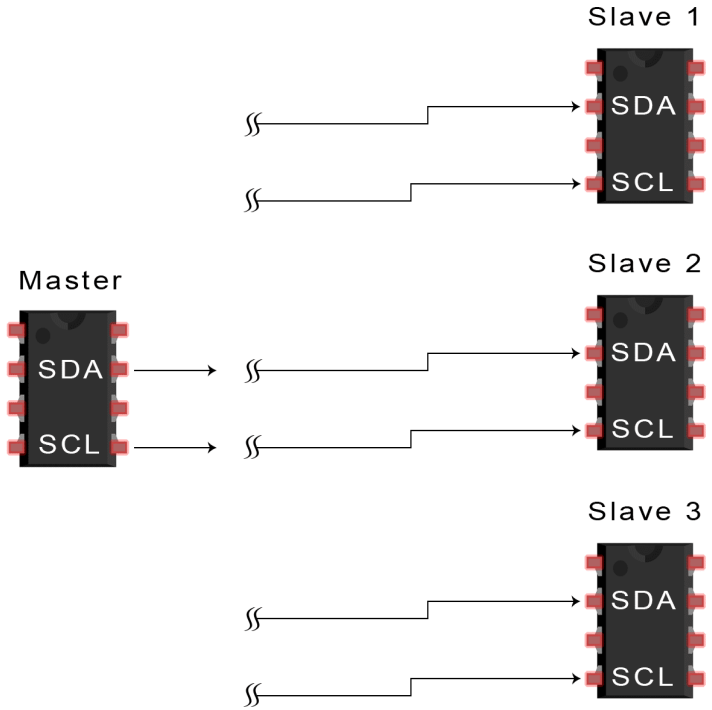
I2C DATA FORMAT

- **Start Condition:** The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low.
- **Address Frame:** A 7 or 10-bit sequence unique to each slave that identifies the slave when the master wants to communicate to it.
- **Read/Write Bit:** A single bit specifying whether the master is sending data to the slave (logic-0) or requesting data from it (logic-1).
- **ACK/NACK Bit:** Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.
- **Stop Condition:** The SDA line switches from a low voltage level to a high voltage level after the SCL line switches from low to high.

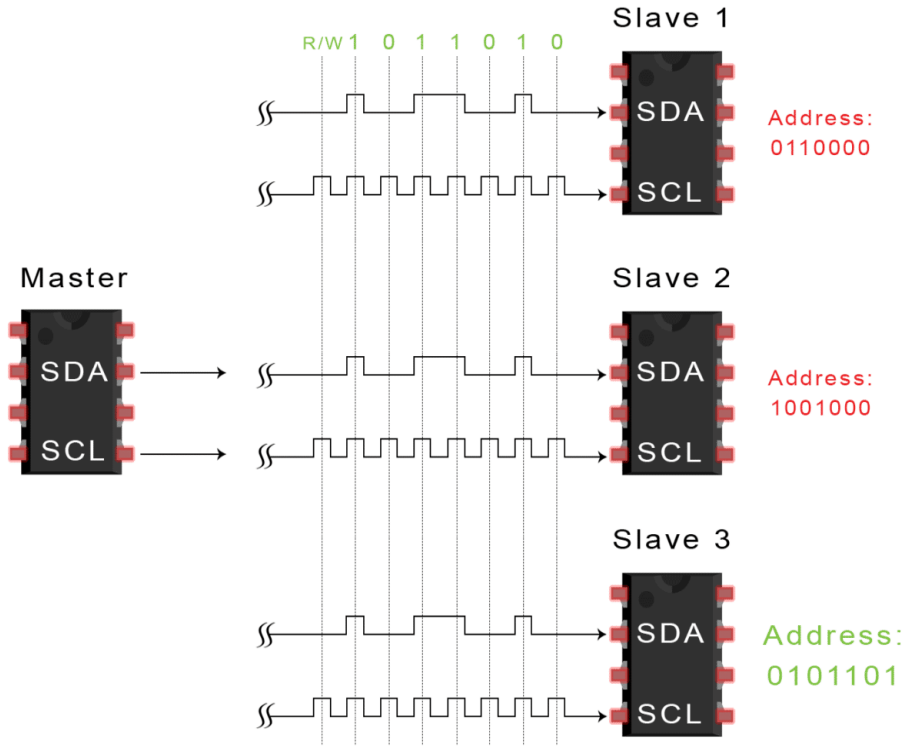
INTER-INTEGRATED CIRCUIT

STEPS OF I2C DATA COMMUNICATION

- STEP-1: The master sets the start condition by setting SDA pin to low before setting SCL pin
- STEP-2: The master sends 7 or 10 bit address to each slave with which it wants to communicate with, along with the read(1)/write(0) bit



STEP-1



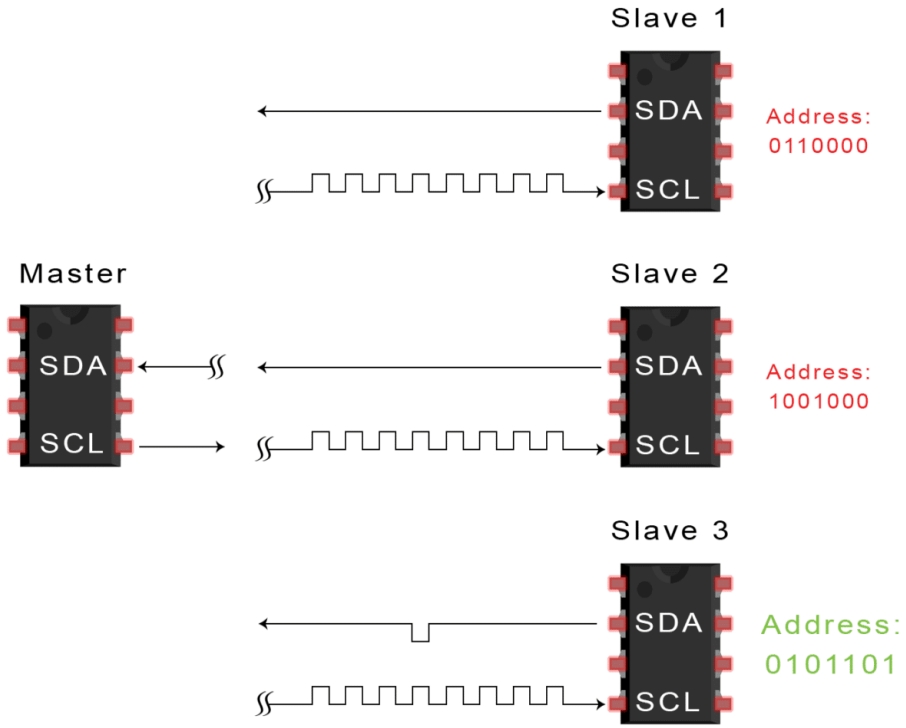
STEP-2

INTER-INTEGRATED CIRCUIT

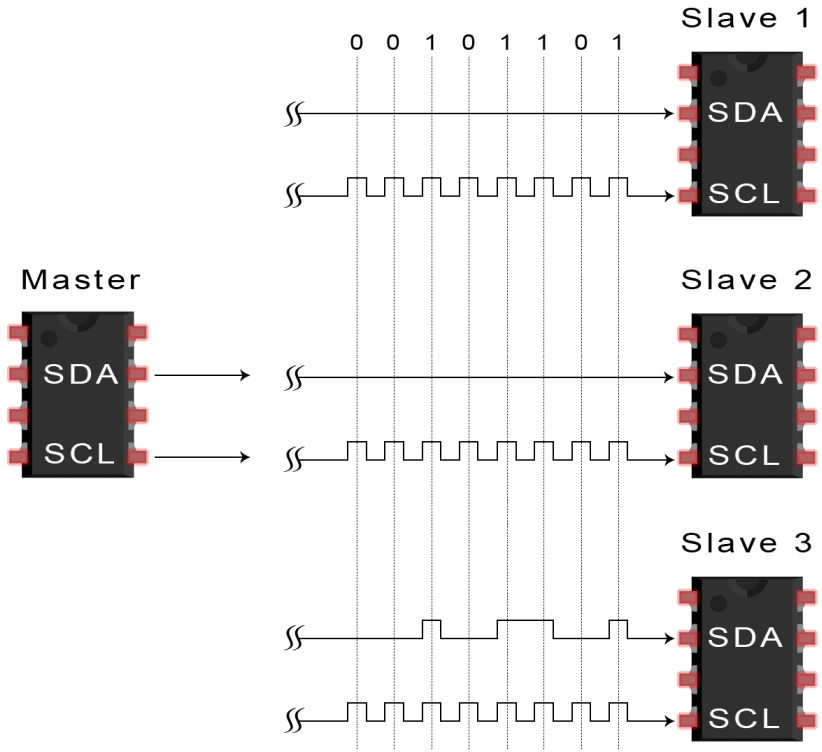
STEPS OF I2C DATA COMMUNICATION

- **STEP-3:** After receiving address from master, slave compare the address with its own address. If the address matches, ACK bit is sent and SDA pin is switched to low. If address does not matches slave keeps the SDA pin high.

- **STEP-4:** The master sends or receives the data frame



STEP-3

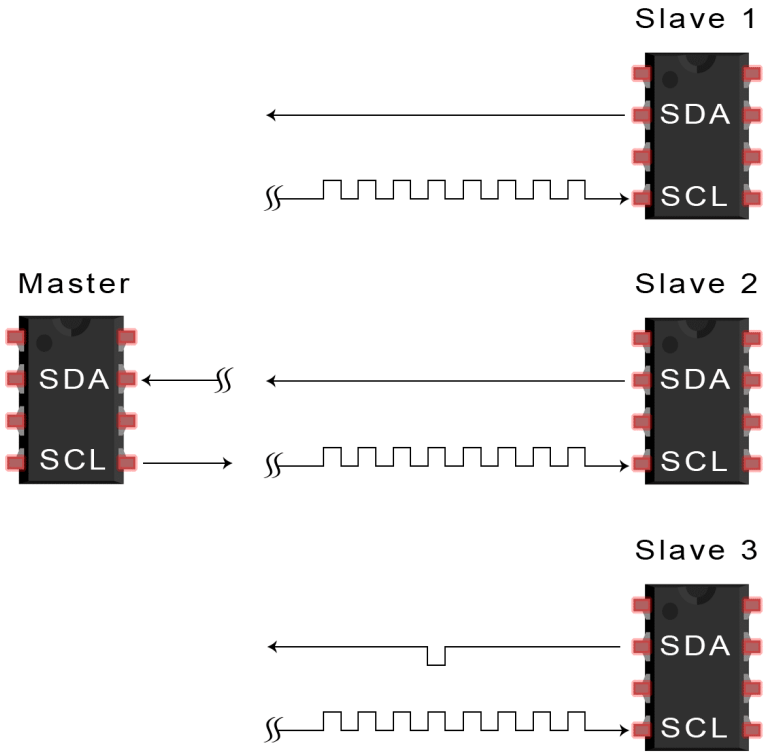


STEP-4

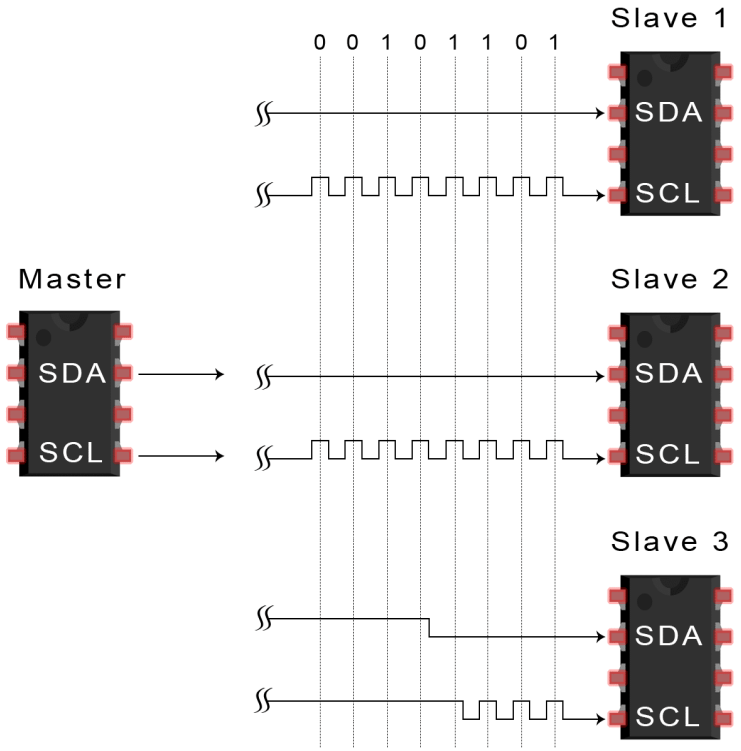
INTER-INTEGRATED CIRCUIT

STEPS OF I2C DATA COMMUNICATION

- STEP-5: After each successful data reception the receiving device sent a ACK bit
- STEP-6: To end the data transmission, the master sets the SCL pin high before setting SDA pin high



STEP-5

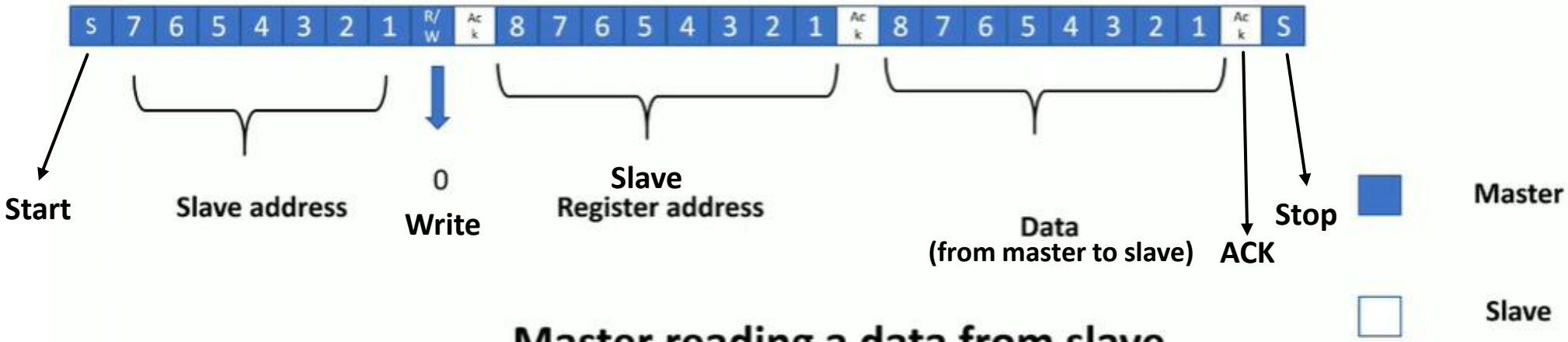


STEP-6

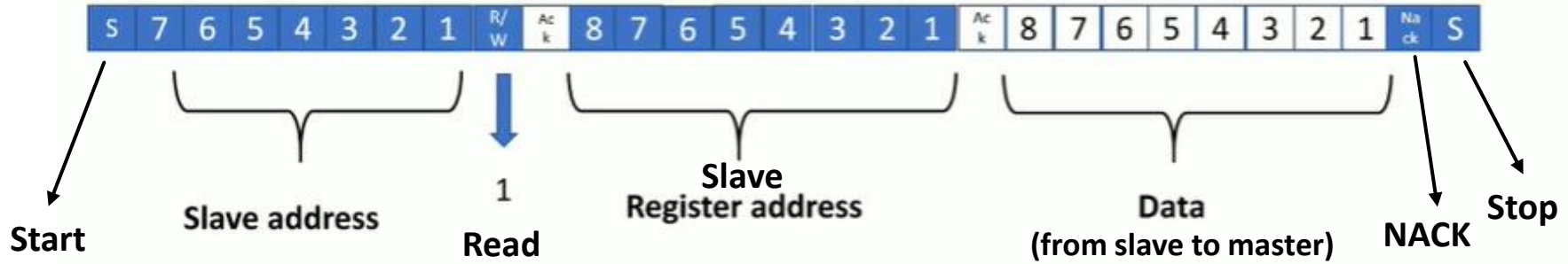
INTER-INTEGRATED CIRCUIT

I2C DATA COMMUNICATION METHODS

Master writting a data to slave



Master reading a data from slave



INTER-INTEGRATED CIRCUIT

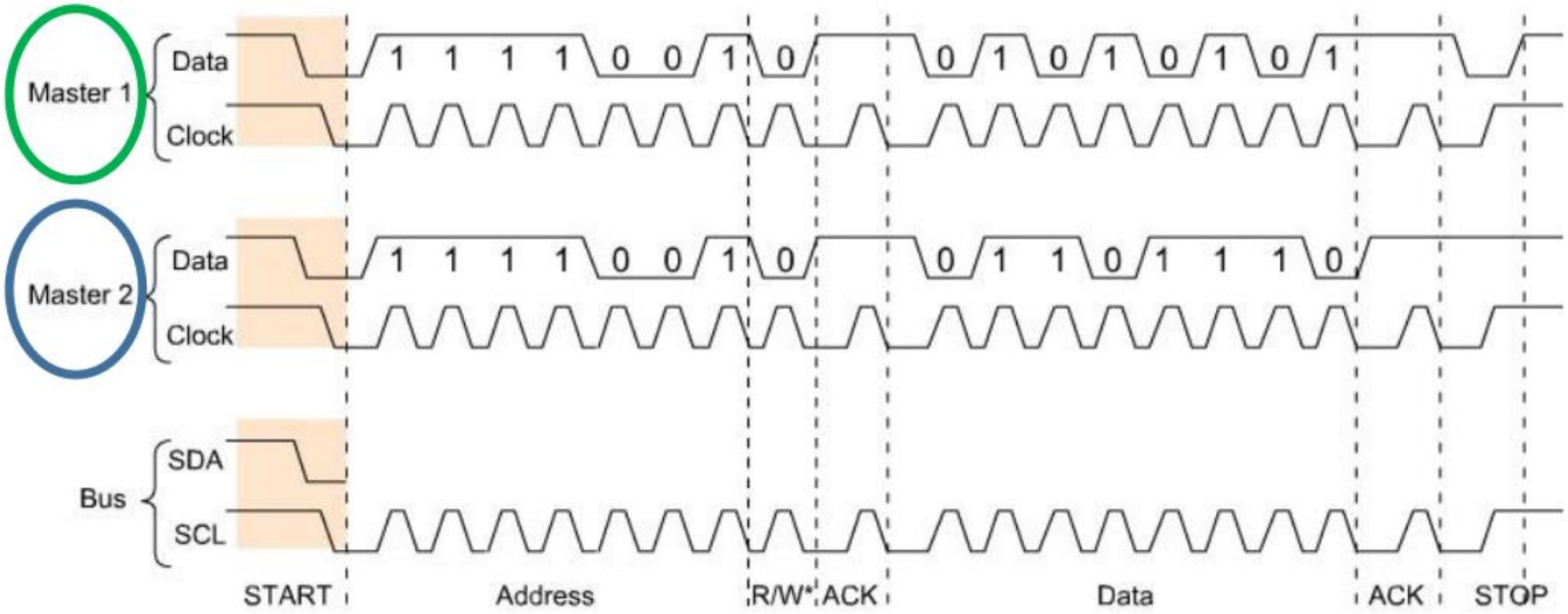
I2C BUS ARBITRATION

- Since I2C supports multi-master configuration, when two or more master devices attempt to try to communicate on the I2C bus at the same time, an arbitration process occurs to determine which device has priority to transmit data on the bus.
- This is known as I2C bus arbitration and it ensures that only one device has control of the bus at any given time, and that data is transmitted without any errors.
- The arbitration process in I2C is based on a wired-AND logic system. In this system, each device on the bus can pull the bus lines low (logic 0), but cannot pull them high (logic 1).
- Therefore, if two or more devices try to transmit data at the same time, they will both try to pull the bus lines low.
- The device that pulls the bus lines low first will win the arbitration and will be able to transmit its data on the bus.

INTER-INTEGRATED CIRCUIT

I2C BUS ARBITRATION

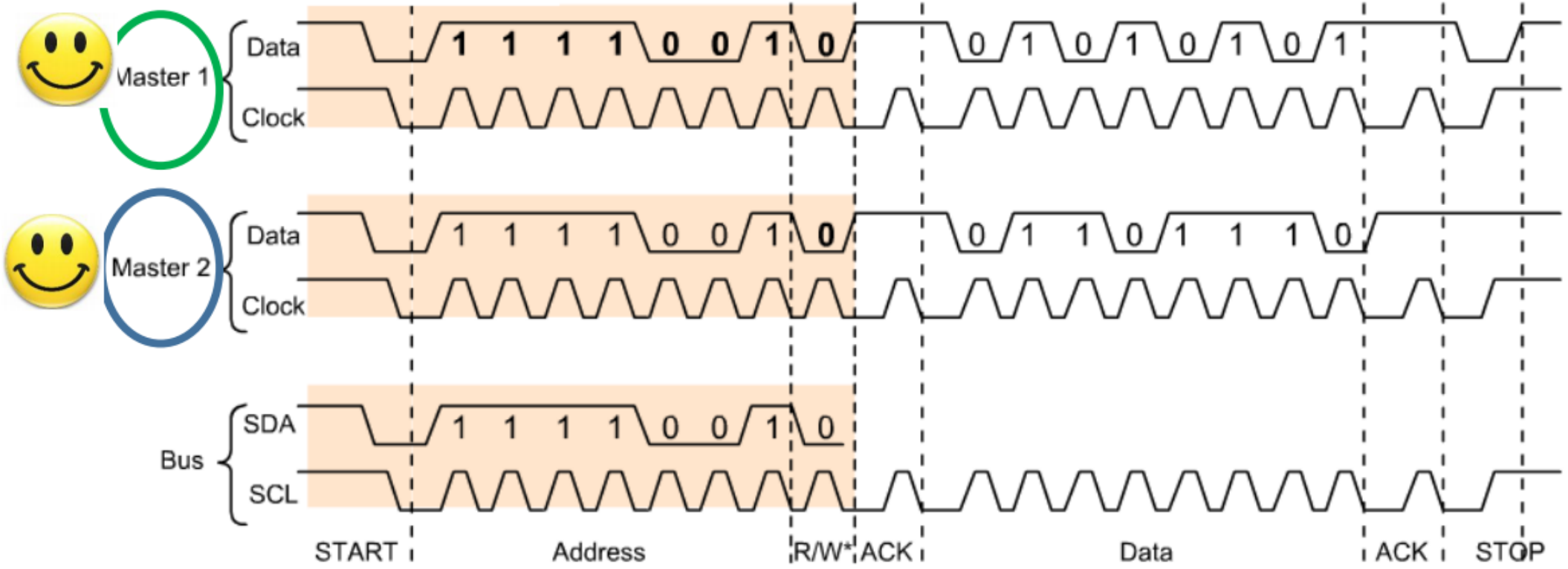
- In this example the two masters who have taken the control of the bus line have same speed and both are in the write mode and want to address the same Slave. So both started to make SDL line low to send the slave address (0x79)



INTER-INTEGRATED CIRCUIT

I2C BUS ARBITRATION

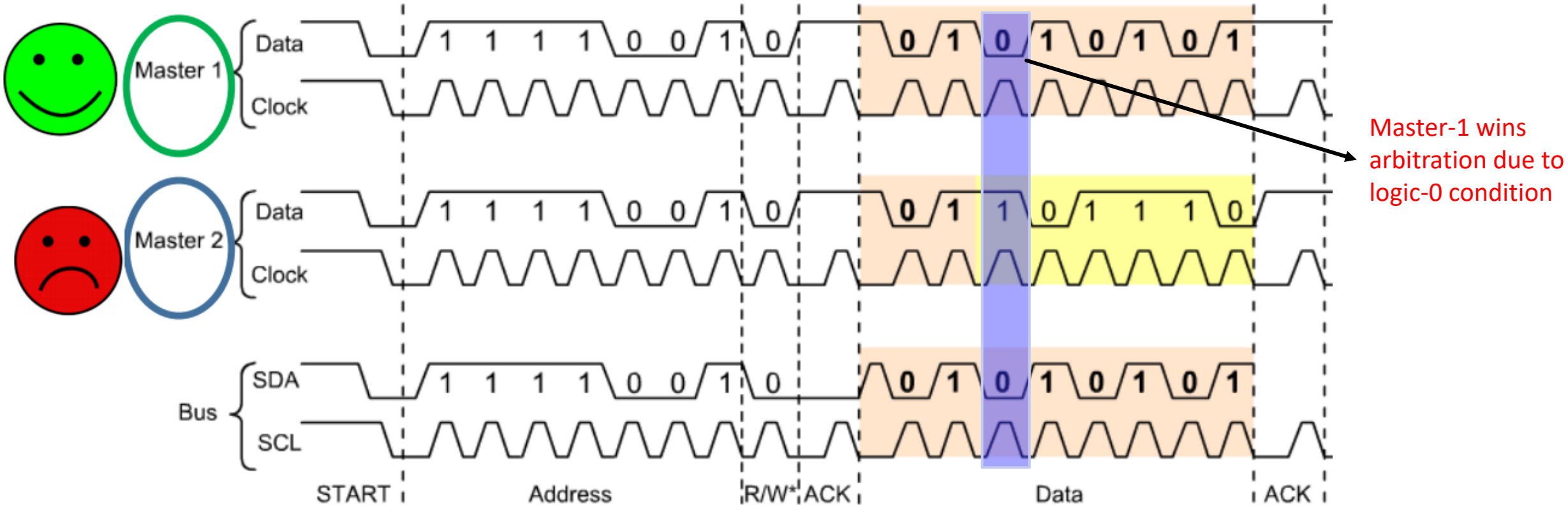
- After successful transmission of the slave address, the slave device acknowledges it and so far both the master are under the impression that it owns the bus because so far they have transmitted same data on the bus.



INTER-INTEGRATED CIRCUIT

I2C BUS ARBITRATION

- Now each master wants to transmit its own data to the slave. The moment their data bits do not match any more then the Master 2 loses arbitration and It must backs Off because when Master 2 tries to move the SDA line high the data on the bus remains low due to wired-AND configuration.



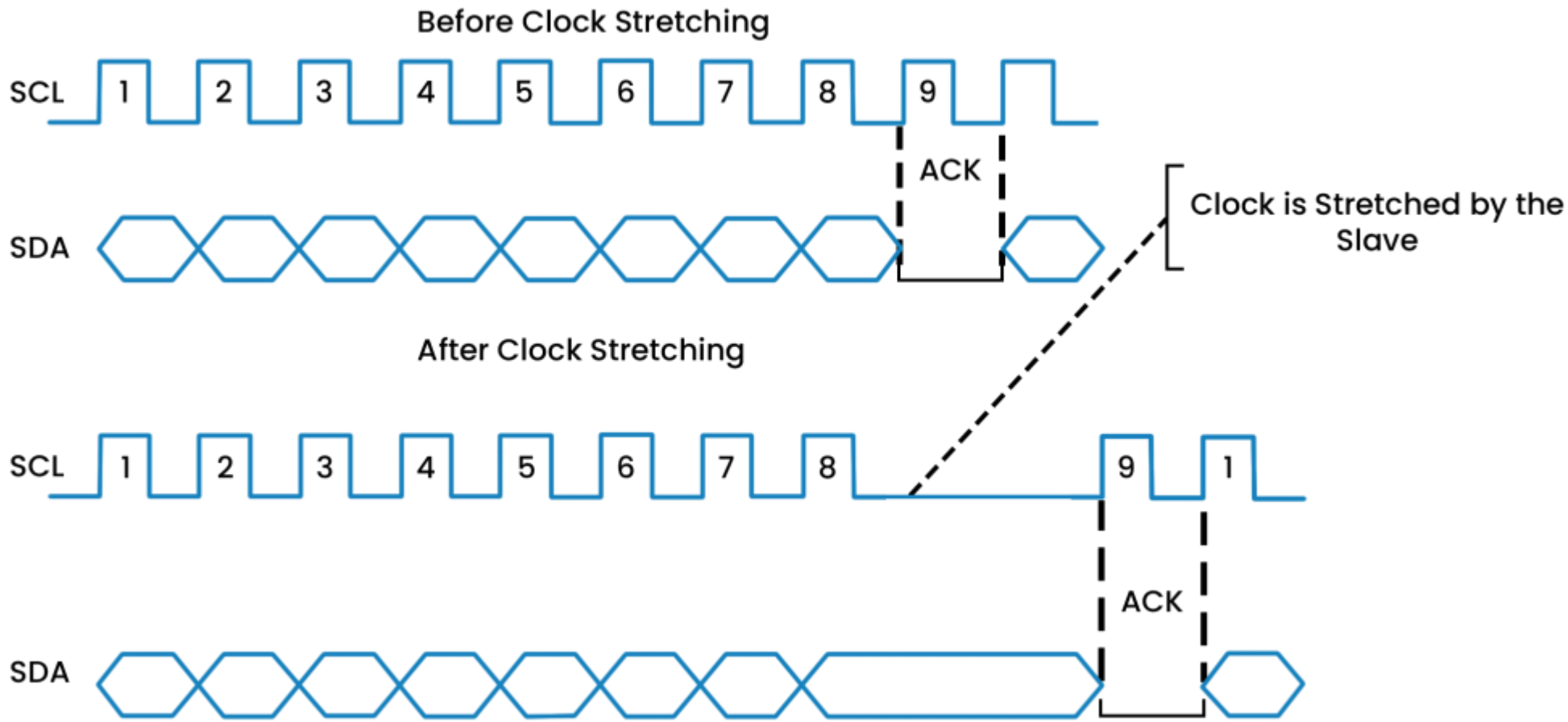
INTER-INTEGRATED CIRCUIT

I2C CLOCK STRETCHING

- An important features of I2C protocol is clock stretching, which is used to temporarily halt the master's clock signal by a slave device until it's ready to continue with the transmission.
- Clock stretching required especially in scenarios where different devices have varying processing speeds or when one device is slower than the other.
- It allows an I2C slave device to force the master device into a wait state by holding the SCL line LOW. The data transaction cannot be completed until SCL line held HIGH.
- Clock stretching can be useful in scenarios where the slave device needs more time to complete its operation such as store received data, or if there is a delay in the response.
- This typically occurs after the slave device has received and acknowledged a byte of data.

INTER-INTEGRATED CIRCUIT

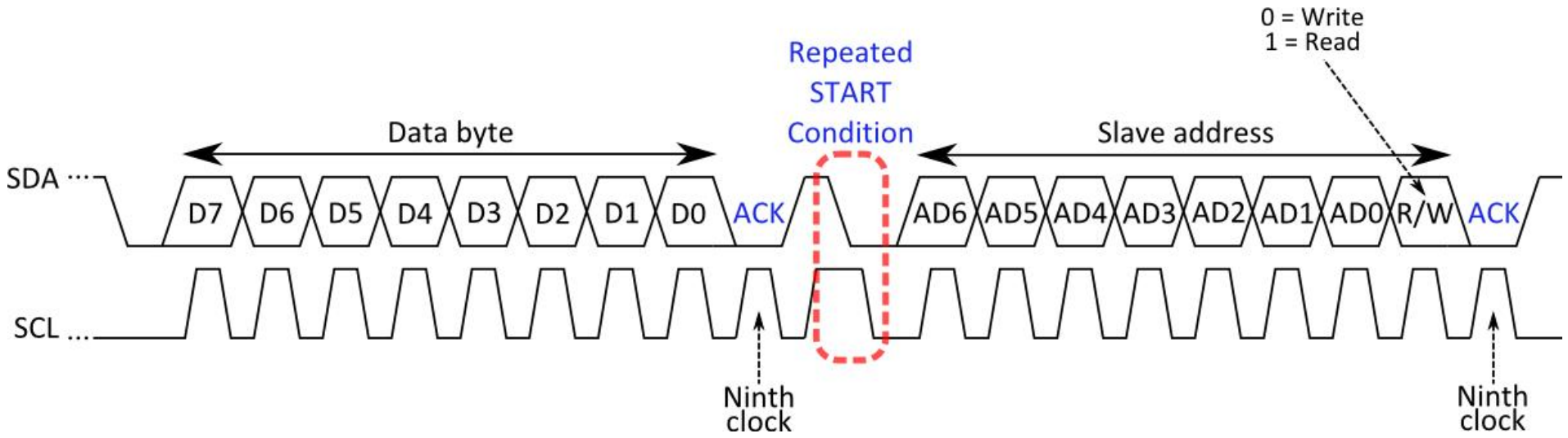
I2C CLOCK STRETCHING



INTER-INTEGRATED CIRCUIT

I2C REPEATED START

- A repeated START condition is useful when the master wishes to start a new communication but does not wish to lose control of the bus to another master by sending a STOP bit.
- Hence, a repeated start condition is used, instead of sending the STOP condition, the master sends another START condition followed by the address, read/write bit, and data.
- For generating a repeated START, the master changes the SDA line from high to low while the SCL line is high.



INTER-INTEGRATED CIRCUIT

ADVANTAGES

- Only uses two wires
- Supports multiple masters and slaves
- Unique device addressing
- ACK/NACK bit for successful transmission
- It can work well with both slow and fast ICs.

DISADVANTAGES

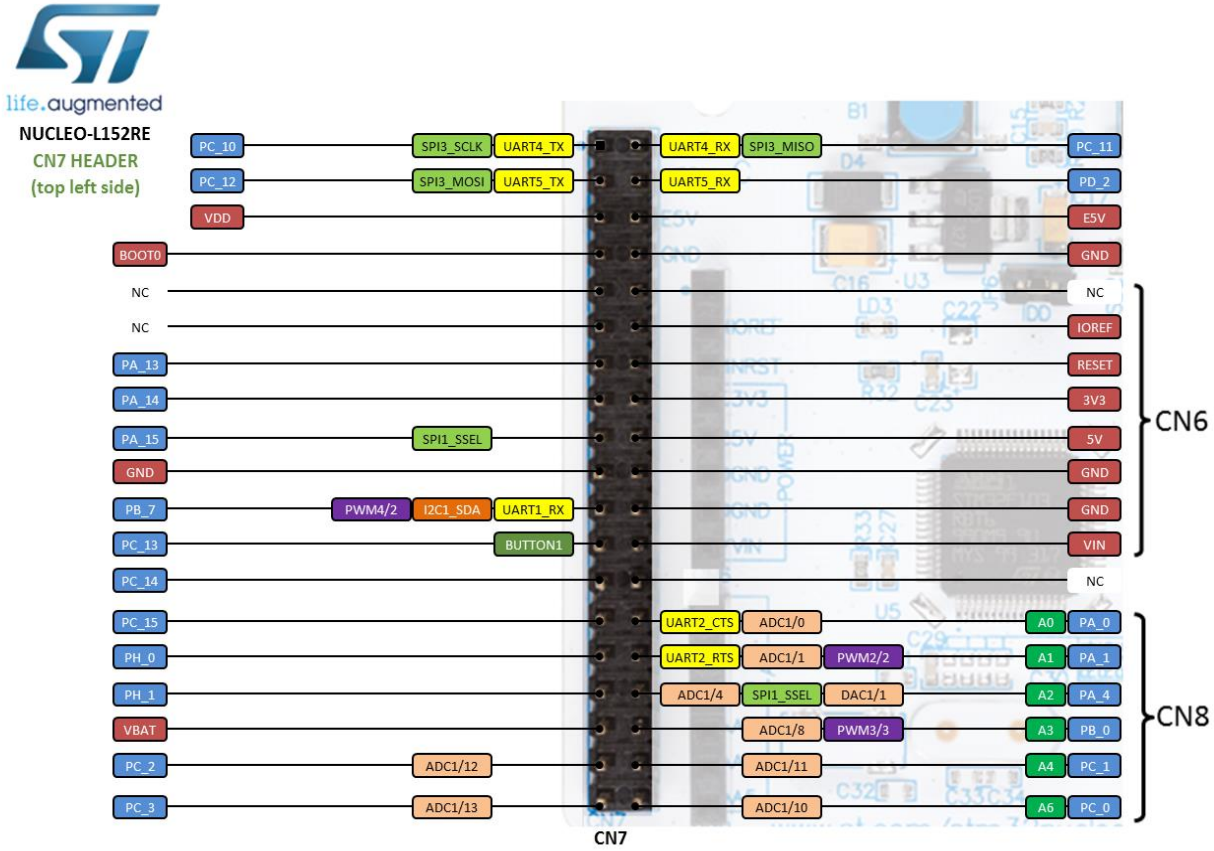
- Slower data transfer rate than SPI
- Power dissipation and require more space because of pull-up resistor
- Hardware complexity increases with multi master/slave configuration
- Frame overhead due to ack bits and device address bits

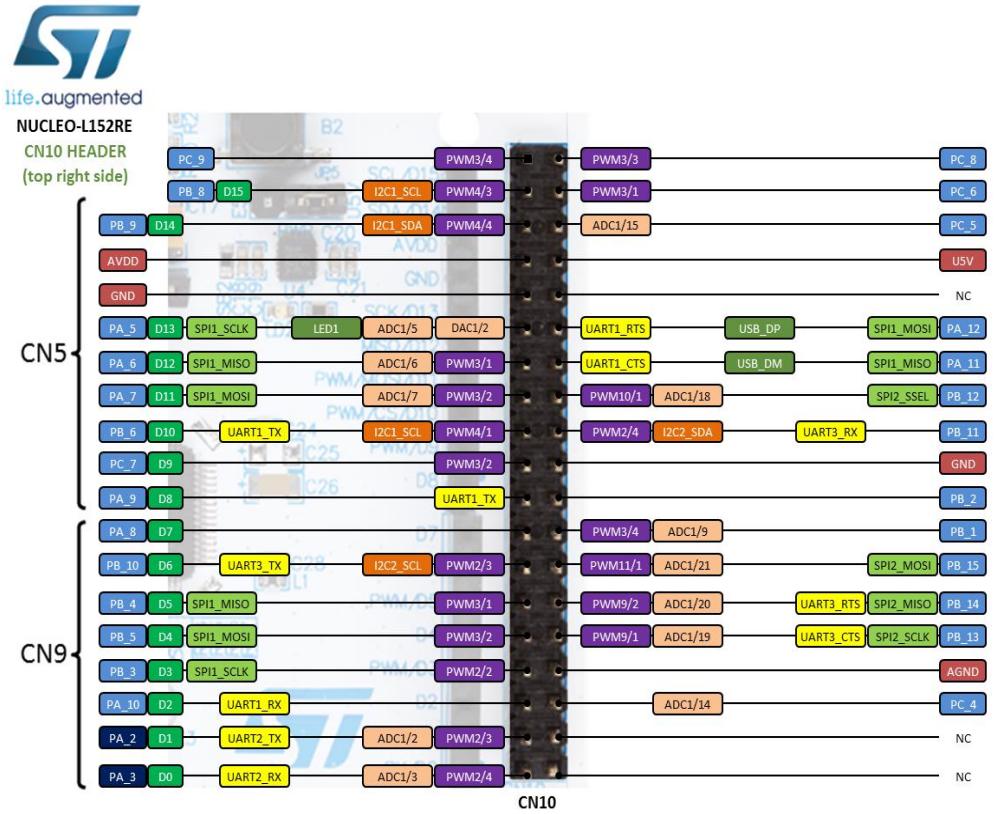
INTER-INTEGRATED CIRCUIT

I2C IN NUCLEO BOARD

- NucleoL152RE has TWO I2C Interface and default settings of SPI interface is:

  Default clock frequency of 100KHz.





INTER-INTEGRATED CIRCUIT

API Required – MASTER

Functions	Usage
I2C	Create an I²C Master interface, connected to the specified pins
frequency	Set the frequency of the I²C interface
read	Read from an I²C Slave
write	Write to an I²C Slave
start	Creates a start condition on the I²C bus
stop	Creates a stop condition on the I²C bus

I²C, inter-integrated circuit.

Examples:

- I2C master(PB_9,PB_8); // Creating I2C master interface in the order SDA, SCL pins of Nucleo board
- master.frequency(10000); // Set the frequency of I2C master interface as 100KHz
- master.start(); // start the I2C communication
- master.stop(); // stop the I2C communication
- char ch = master.read(); // Read a single byte from the I2C bus and store it in a variable ch
- master.write(0x43); // write a single byte data 0x43 to slave device
- master.read(0x52, rec_val, 10); // Read from I2C slave address 0x52 and store it in variable rec_val
- // with the data length of 10 byte
- master.write(0x52, send_val, 10); // Write to I2C slave address 0x52 with the value stored in a variable send_val
  // of length 10 byte

INTER-INTEGRATED CIRCUIT

API Required - SLAVE:

Function	Usage
I2CSlave	Create an I²C Slave interface, connected to the specified pins.
frequency	Set the frequency of the I²C interface.
receive	Checks to see if this I²C Slave has been addressed.
read	Read from an I²C Master.
write	Write to an I²C Master.
address	Sets the I²C Slave address.
stop	Reset the I²C Slave back into the known ready receiving state.

I²C, inter-integrated circuit.

int receive ( ) - A status indicating if the device has been addressed and how. Return values are,

0 - NoData - The slave has not been addressed.
1 - ReadAddressed - The master wants to read from this slave.
2 - WriteAddressed - The master is writing to this slave.
3 - WriteGeneral - The master is writing to all slave.

Examples:

- I2CSlave slave(PB_9,PB_8); // Creating I2C slave interface in the order SDA, SCL pins of Nucleo board
- slave.frequency(10000); // Set the frequency of I2C master interface as 100KHz
- slave.address(0x52) // Assign a address 0x52 to slave device
- slave.stop(); // stop the I2C communication
- char ch = slave.read(); // Read a single byte from the I2C master and store it in a variable ch
- slave.write(0x28); // write a single byte data 0x28 to master device
- slave.read(rec_val, 10); // Read from I2C master and store it in rec_val with the data length of 10 byte
- slave.write(send_val, 10); // Write to I2C master with the value stored in send_val of data length 10 byte

https://os.mbed.com/docs/mbed-os/v6.16/mbed-os-api-doxy/classmbed_1_1_i2_c_slave.html

INTER-INTEGRATED CIRCUIT

Example-1: Read the input from Master and display on Slave

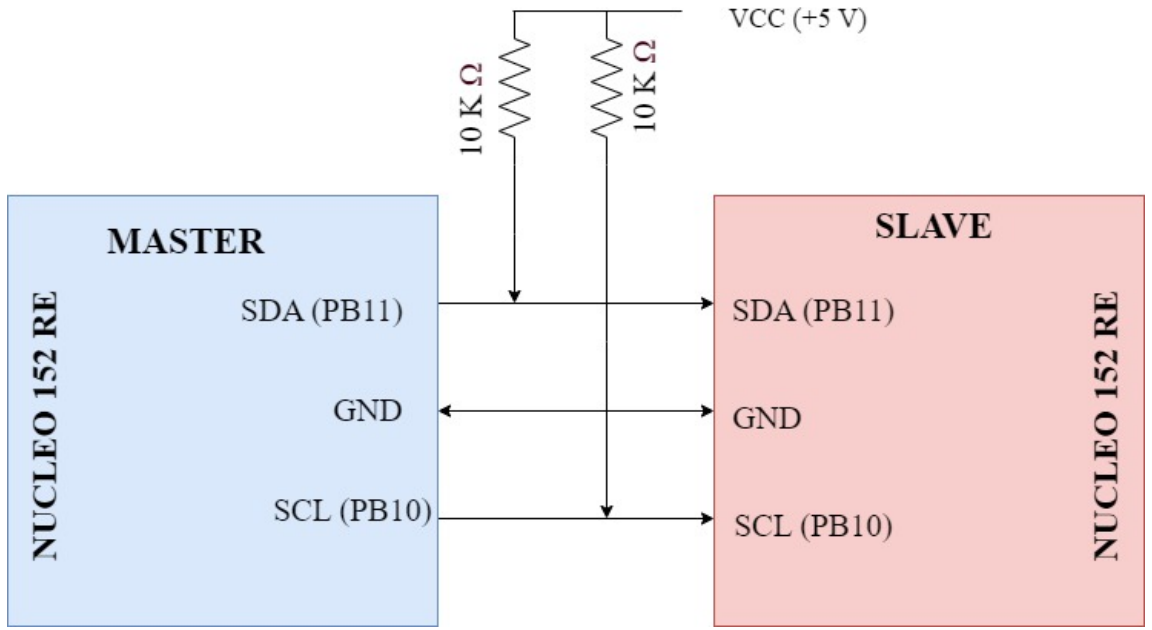
Write a mbed C++ program to implement a I2C communication between two Nucleo boards. Configure one of the Nucleo as master and other as slave. Establish a I2C communication between master and slave, display the message send from master's Teraterm to the slave Teraterm terminal. Design and implement this logic on the STM 32 Nucleo L152RE board using Keil studio could platform.

INTER-INTEGRATED CIRCUIT

Program (Master)

```
//Lab Task-1 Master
#include "mbed.h"
Serial pc (USBTX,USBRX);
I2C i2c(PB_11,PB_10); //SDA,SCL
int main()
{
int address = 0x50;
char data[20];
pc.printf("enter data to be sent\r\n");
pc.scanf("%s",&data);
pc.printf("%s",data);
int l=strlen(data);
i2c.write(address, data,l);
wait(2);
}
```

Connection diagram



INTER-INTEGRATED CIRCUIT

Program (Slave)

```
//Lab Task-1 Slave
#include <mbed.h>
Serial pc(USBTX, USBRX);
I2CSlave slave(PB_11, PB_10);  //SDA,SCL

int main() {
char buf[20];
char msg[] = "Slave!";  //Message to be sent to Master
slave.address(0x50);

while (1) {
int i = slave.receive();
switch (i) {
case I2CSlave::ReadAddressed:
slave.write(msg, strlen(msg) + 1); // Includes null char
break;
```

```
case I2CSlave::WriteGeneral:
slave.read(buf, 20);
pc.printf("Read1 : %s\r\n", buf);
wait(1);
break;

case I2CSlave::WriteAddressed:
slave.read(buf, 20);
pc.printf("Read2 : %s\r\n", buf);
wait(2);
break;
}
}
}
```

INTER-INTEGRATED CIRCUIT

Example-2: Read Accelerometer axis values and print it in serial monitor

Write a mbed C++ program to implement a I2C communication between Nucleo board and ADXL 345 Accelerometer. After initialization of the accelerometer, read the X,Y and Z-axis values and print it on the serial monitor. Design and implement this logic on the STM 32 Nucleo L152RE board using Keil studio could platform.

Important Note:

- Add ADXL345 mbed library files ADXL345_I2C.cpp and ADXL345_I2C.h into your project folder before build the project.

  ADXL345 mbed library Link: https://os.mbed.com/users/peterswanson87/code/ADXL345_I2C/

- Refer the ADXL345 accelerometer datasheet given in the below link to know more details

  ADXL345 datasheet link: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf

INTER-INTEGRATED CIRCUIT

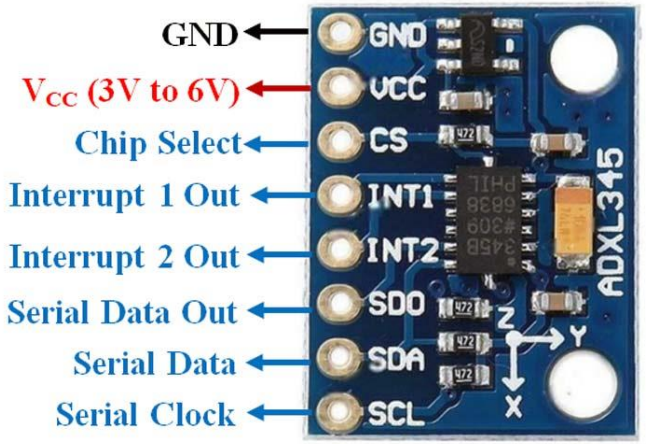
Introduction to ADXL345 Accelerometer

- An accelerometer is a device that measures acceleration forces, these forces can be static (such as gravity) or dynamic (caused by movement or vibration).
- Accelerometers are commonly found in smartphones, tablets, fitness trackers, cars, airplanes, and various other electronic devices to detect changes in velocity, orientation, and vibration.
- Accelerometers can measure acceleration along one, two, or three axes(X, Y, Z), depending on their design.
- The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g.
- ADXL345 measures static acceleration due to gravity as well as dynamic acceleration resulting from motion or shock.

INTER-INTEGRATED CIRCUIT

Introduction to ADXL345 Accelerometer

- The sensor outputs data formatted as 16-bit two's complement that is accessible via SPI or I2C interfaces.
- ADXL345 has user-selectable resolution and measurement ranges that can be selected by passing serial commands to it.



- ADXL345 Module Features & Specifications
  - ✓ Supply Voltage range: 2.0 V to 3.6 V DC
  - ✓ It can be interface with 3.3V or 5V Microcontroller.
  - ✓ Tap/Double Tap Detection, Free-Fall Detection
  - ✓ Support both SPI and I2C interfaces
  - ✓ Measuring Range: ±16g
  - ✓ Measuring Values (-16g to +16g): X: -235 to +270, Y: -240 to +260, Z: -240 to +270
  - ✓ Ultra-Low Power: 23uA in measurement mode, 0.1uA in standby@ 2.5V

INTER-INTEGRATED CIRCUIT

Configuring and reading acceleration values using ADXL345 mbed library APIs

Register 0x00—DEVID (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	0	1	0	1

- accelerometer.getDeviceID() - return the fixed ADXL345 device ID as 0xE5

Register 0x2D—POWER_CTL (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	Link	AUTO_SLEEP	Measure	Sleep	Wakeup	

- accelerometer.setPowerControl(0x00) - Set into standby mode to configure the device by setting D7 to D0 bits to 0.
- accelerometer.setPowerControl(MeasurementMode) - Set the device into measurement mode to read X,Y,Z values by setting D3 as 1

Register 0x31—DATA_FORMAT (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	

- accelerometer.setDataFormatControl(0x0B) – Set device into Full resolution (4mg/LSB) & range as +/-16g by assigning D3 as 1, D1 and D0 to 1.

Register 0x2C—BW_RATE (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	LOW_POWER	Rate			

- accelerometer.setDataRate(ADXL345_3200HZ) - set device output data rate to 3200 by setting D3 to D0 as 1111

Register 0x32 to Register 0x37—DATAX0, DATAX1, DATAY0, DATAY1, DATAZ0, DATAZ1 (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0

- accelerometer.getOutput(readings) – read 16-bit x, y, z axis values from registers 0x32 to 0x37 and convert into 8-bit fomat then store them in variable named readings

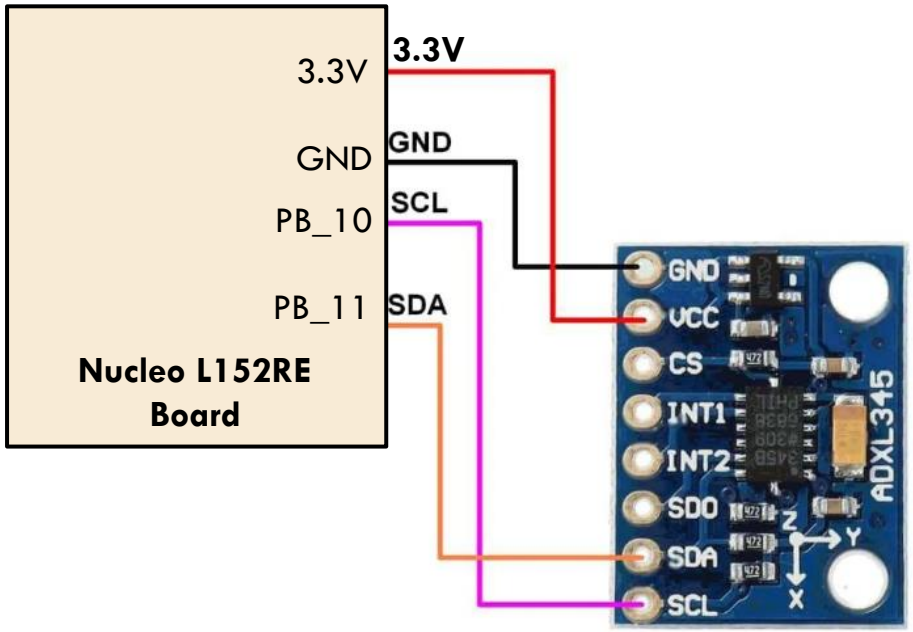
INTER-INTEGRATED CIRCUIT

Program

```
#include "mbed.h"
#include "ADXL345_I2C.h"

 ADXL345_I2C accelerometer(PB_11,PB_10);
 Serial pc(USBTX, USBRX);
 int X_out_raw, Y_out_raw, Z_out_raw;  // Outputs
int X_out, Y_out, Z_out;  // Outputs
 int main() {
    int readings[3] = {0, 0, 0};
    pc.printf("Starting ADXL345 test...\n");
    wait(.001);
    pc.printf("Device ID is: 0x%02x\n", accelerometer.getDeviceID());
    wait(.001);
        // Test whether any of the initialization fails.
    if (accelerometer.setPowerControl(0x00)){
        pc.printf("didn't intitialize power control\n");
        return 0;  }
         wait(.001);
     //Full resolution, +/-16g, 4mg/LSB.
     if(accelerometer.setDataFormatControl(0x0B)){
        pc.printf("didn't set data format\n");
        return 0;  }
        wait(.001);
```

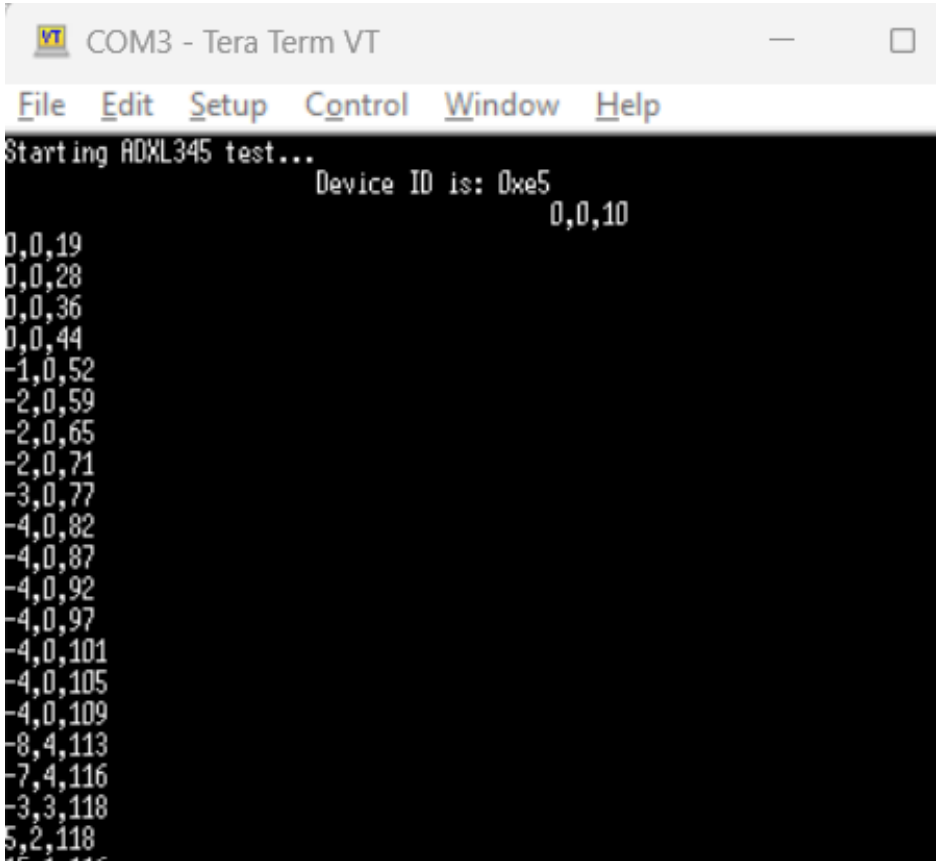
Connection Diagram



INTER-INTEGRATED CIRCUIT

```
    //3.2kHz data rate.
    if(accelerometer.setDataRate(ADXL345_3200HZ)){
        pc.printf("didn't set data rate\n");
        return 0;    }
        wait(.001);
    //Measurement mode.
   if(accelerometer.setPowerControl(MeasurementMode)) {
        pc.printf("didn't set the power control to measurement\n");
        return 0;   }
   while (1) {
        wait(0.1);
        accelerometer.getOutput(readings);
        wait(.1);
 X_out_raw = (int16_t)readings[0];
 Y_out_raw = (int16_t)readings[1];
 Z_out_raw = (int16_t)readings[2];
  // Low-pass filter for stable X,Y,Z values
 X_out = 0.94 * X_out + 0.06 * X_out_raw;
 Y_out = 0.94 * Y_out + 0.06 * Y_out_raw;
 Z_out = 0.94 * Z_out + 0.06 * Z_out_raw;
 pc.printf("%i,%i,%i\r\n",X_out,Y_out,Z_out);
   }
}
```

Output



INTER-INTEGRATED CIRCUIT

Exercise: Fall detection system using Accelerometer

Write a mbed C++ program to implement a fall detection system using ADXL345 accelerometer sensor configured in I2C interface. Read the X and Y-axis values from THE accelerometer, if the x and y-axis value exceeds threshold value of 100 activate the buzzer to indicate the fall detection condition. Design and implement this logic on the STM 32 Nucleo L152RE board using Keil studio could platform.

UART vs SPI vs I2C

Features	UART	SPI	I2C
Number of wires	4 (MOSI, MISO, SCK, and SS)	4 (MOSI, MISO, SCK, and SS)	2 (SDA and SCL)
Communication type	Full-duplex	Full-duplex	Half-duplex
Synchronous?	No	Yes	Yes
Maximum number of devices	Limited by number of Tx, Rx	Limited by number of SS	Limited by addressing scheme (7-bit or 10-bit)
Data transfer speed	115.2kbps	10Mbps	3.4Mbps
Distance	from 15 to 1000m	from 20 to 100m	more than 10m
Error handling	Simple (Parity bits)	No error handling	ACK/NACK feature
Complexity	Moderate	Moderate to high	Moderate
Multi-master configuration	No	No	Yes
Arbitration	No	No	Yes
Use case	Suitable for simple and long distance serial communications	Perfect for fast data transfers	Ideal for short-distance communications with multiple devices

CAN

CONTROLLER AREA NETWORK

INTRODUCTION

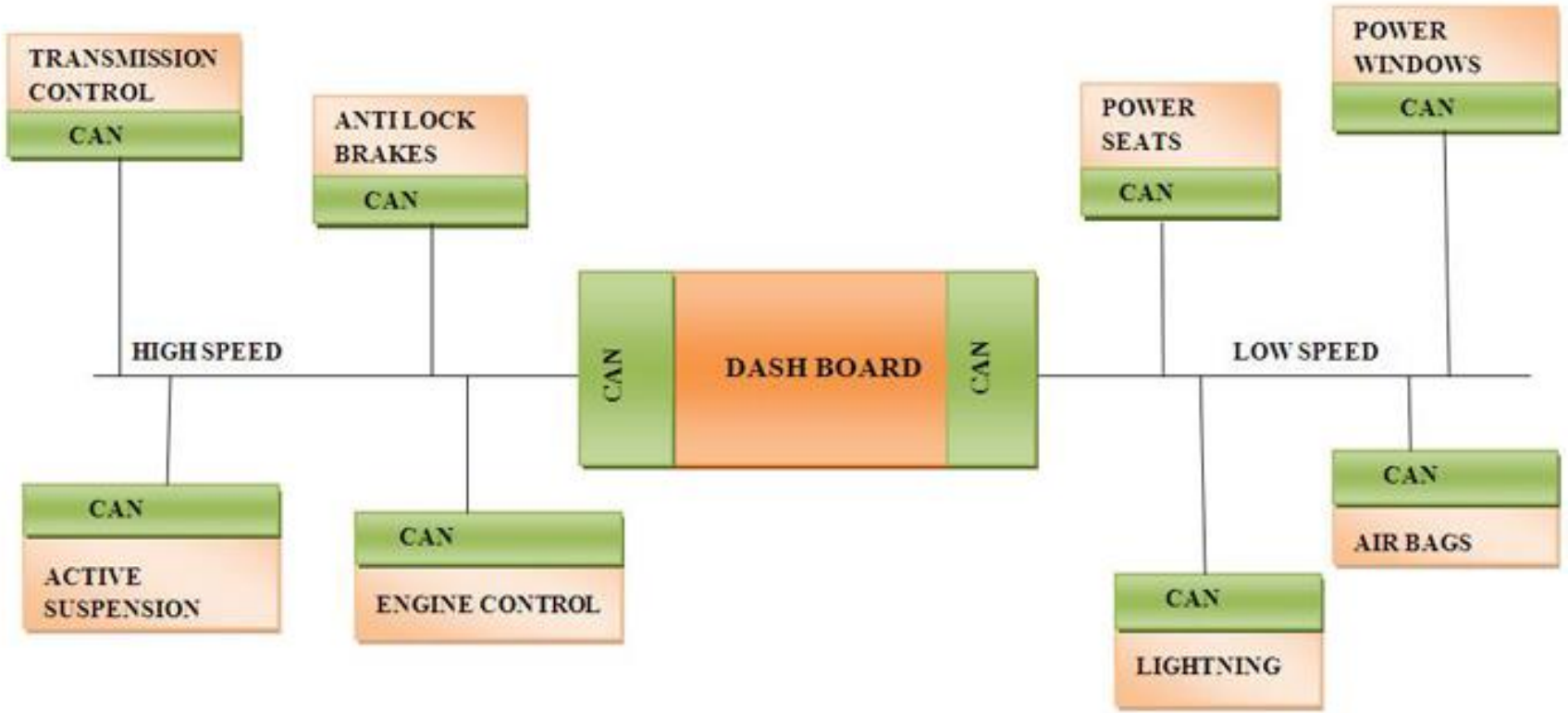
- Controller Area Network (CAN) protocol is Serial half-duplex asynchronous communication developed by Robert Bosch in 1986.
- In automobile system, there are many ECUs for various subsystems like steering, suspension, ADAS etc. interconnection between each other is essential which is costly and complex through conventional wiring while CAN provide an economic and appropriate solution.
- CAN was developed to use in automotive industry to address several challenges but due to its enrich features it is used in others including industrial automation, medical equipment etc.,
- CAN protocol uses a collision detection and arbitration method to prevent multiple nodes from transmitting at the same time and ensure that only one node can transmit at a time.

CONTROLLER AREA NETWORK

FEATURES

- Flexibility in Configuration - Any node can be connected or disconnected without disturbing other nodes on the network.
- Message-Based Protocol - CAN is a message based protocol not address based means transmitted data is available for all nodes and its receiver's choice to receive data or not.
- Multi-master Communication - Any device on the CAN network, called a node, can initiate communication and other nodes on the network can participate in the communication.
- Message Prioritization - CAN protocol do it by assigning lower identifier value (Lower the Identifier higher the Priority)
- CSMA/CA and CSMA/CD – CSMA/CA is used to check bus status and if it is free then the node will transmit data, CSMA/CD is used to stop transmission if data is corrupted in bus
- Good speed with noise reduction feature - offers maximum speed up to 1 Mbps for 40 m CAN-Bus length also reduces noise and EMI due to the use of a differential cable

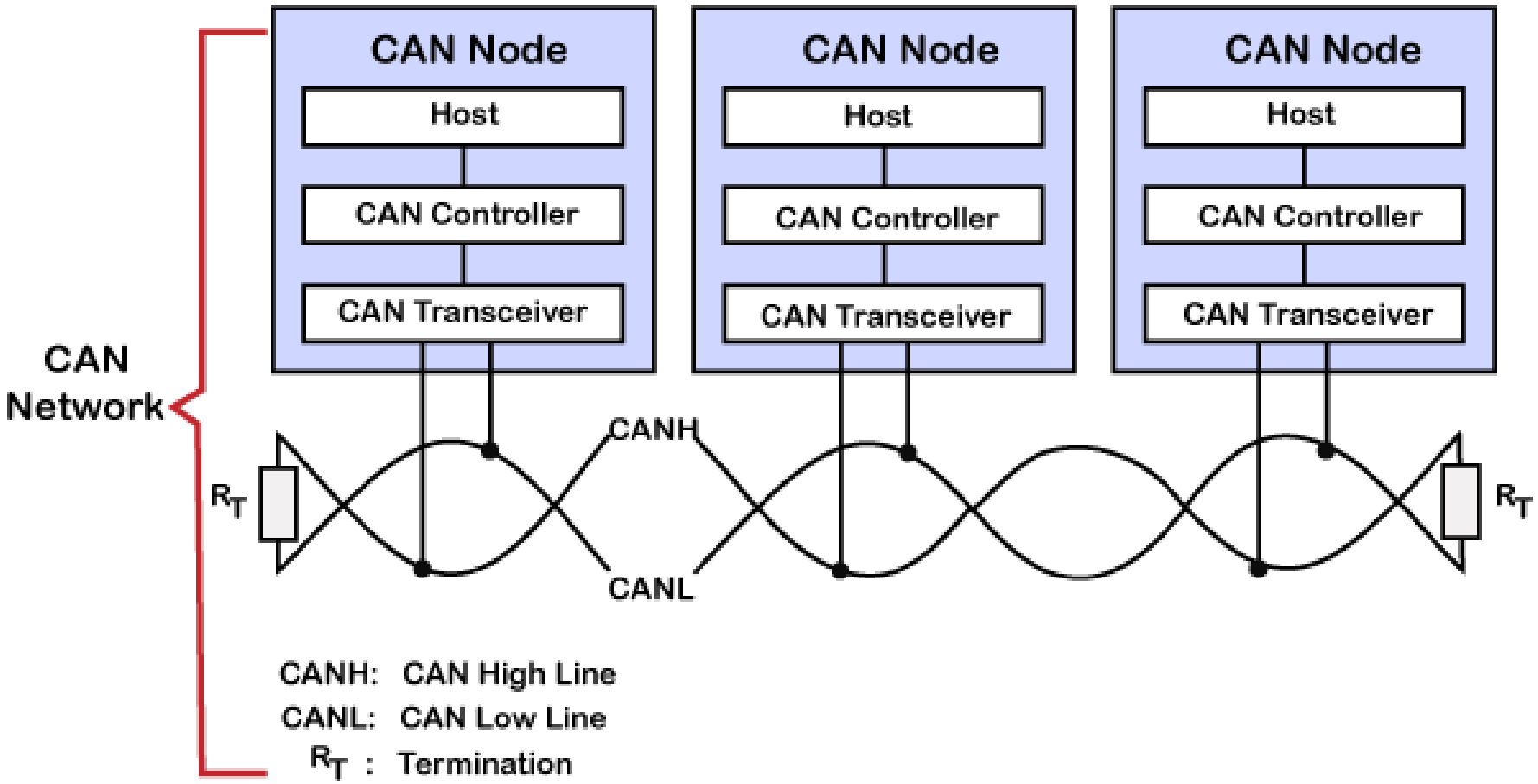
CONTROLLER AREA NETWORK



Pictorial view of CAN protocol wiring connection

CONTROLLER AREA NETWORK

CAN NETWORK



CONTROLLER AREA NETWORK

CAN NETWORK

- Each CAN nodes in CAN Network are connected on unshielded twisted pair cable i. e CAN low line (CAN_L) and CAN high line (CAN_H) to transmit or receive the data and they are terminated with 120Ω resistor to prevent the reflection.

- Every Node in CAN network have three components to accomplish Data transmission :

1. Host Controller: MCU which is host controller decides what messages it wants to transmit and what the received messages mean.

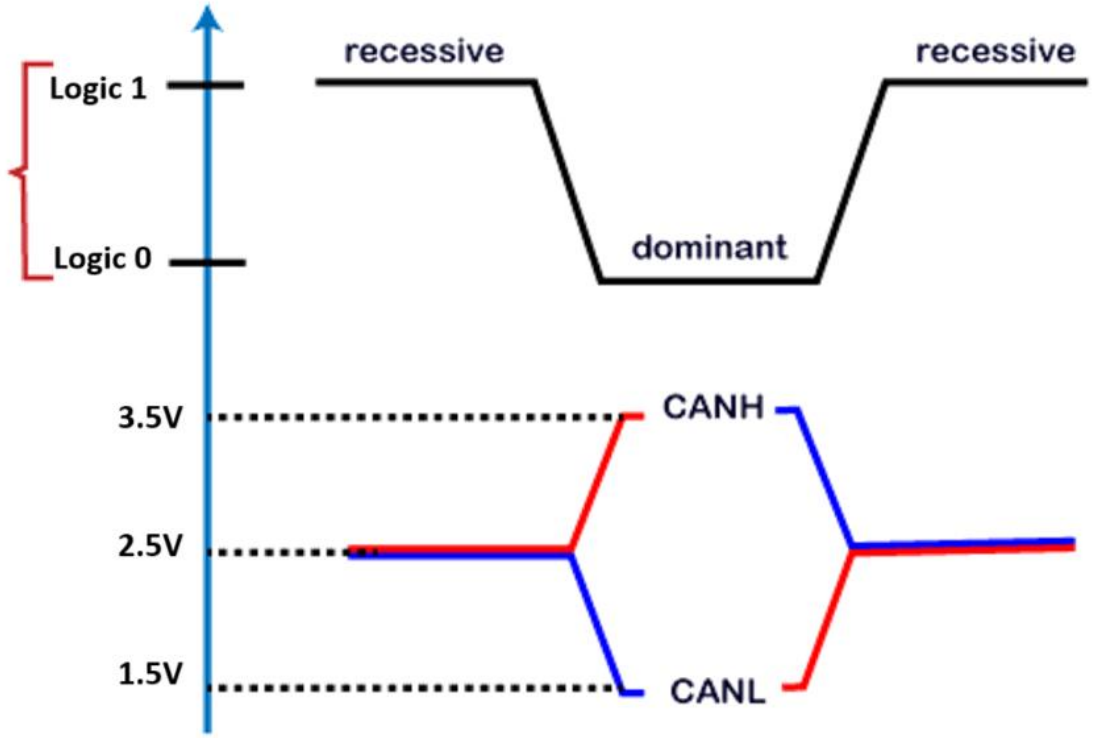
2. CAN Controller: It is responsible for initiating the transmission and reception of CAN messages, manages the data framing, error detection and handling etc.

3. CAN Transceiver: It converts the data from the CAN controller to CAN bus levels and also converts the data from CAN bus levels to suitable level that the CAN controller uses.

CONTROLLER AREA NETWORK

CAN BUS SIGNAL LEVEL

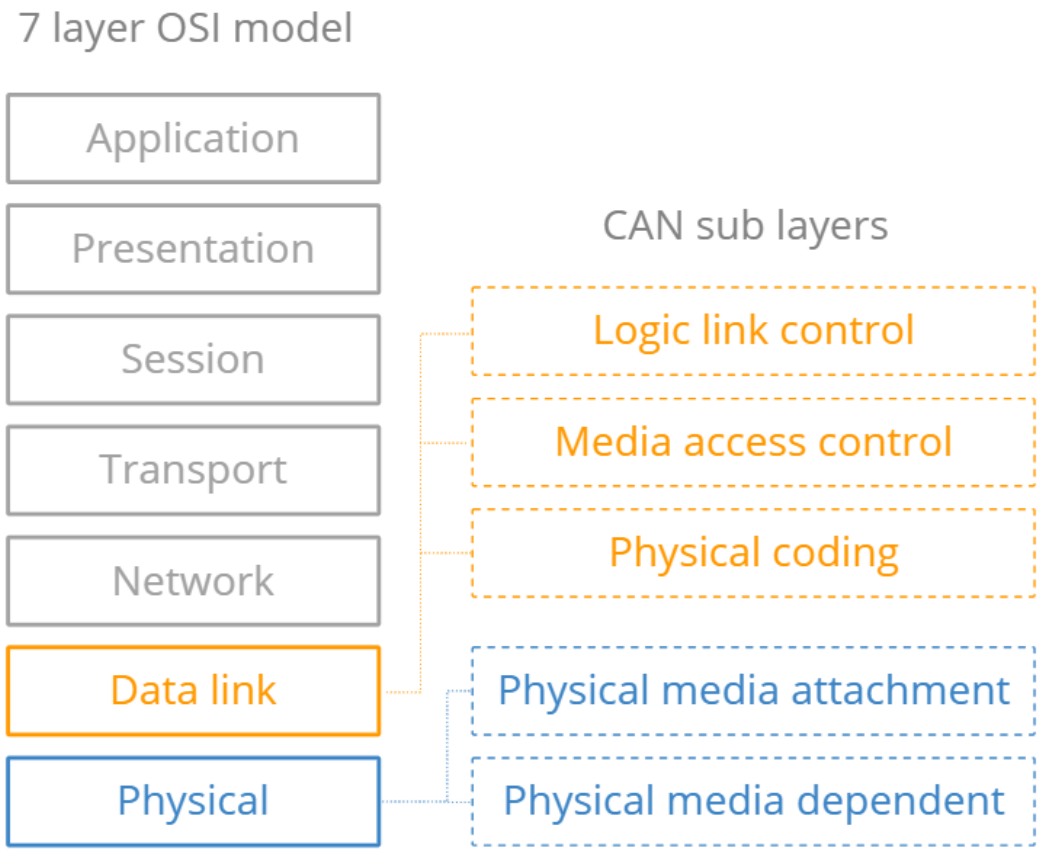
- Logic 1 is a recessive state. To transmit 1 on CAN bus, both CAN high and CAN low should be applied with 2.5V, then the actual differential voltage would be zero volt.
- Logic 0 is a dominant state. To transmit 0 on CAN bus, CAN high should be applied at 3.5V and CAN low should be applied at 1.5V, then the bus's actual differential voltage would be 2 volts.
- The ideal state of the bus is recessive.
- If the node reaches the dominant state, it cannot move back to the recessive state by any other node.



CONTROLLER AREA NETWORK

CAN ARCHITECTURE

- The OSI model partitions the communication system into 7 different layers but the CAN layered architecture consists of two layers, i.e., data-link layer and physical layer.



CONTROLLER AREA NETWORK

CAN ARCHITECTURE

- **Data-link layer:**
  - This layer is responsible for node to node data transfer, allows you to establish and terminate the connection.
  - It is also responsible for detecting and correcting the errors at the physical layer.
  - Data-link layer is subdivided into two sub-layers:
    - MAC (Media Access Control): It defines how devices in a network gain access to the medium by providing Encapsulation and Decapsulation of data, Error detection, and signalling.
    - LLC (Logical link control): It is responsible for frame acceptance filtering, overload notification, and recovery management.
- **Physical layer:**
  - The physical layer is responsible for the transmission of raw data.
  - It defines the specifications for the parameters such as voltage level, timing, data rates etc.,

CONTROLLER AREA NETWORK

CAN MESSAGE FRAME FORMAT

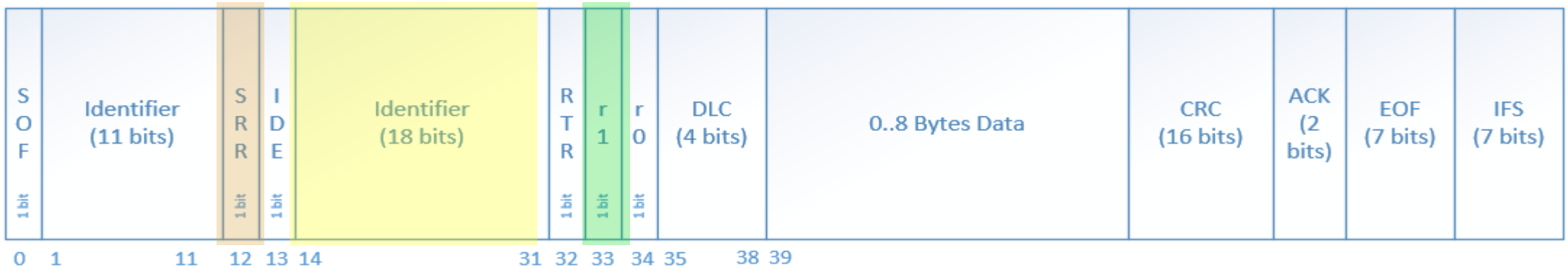
- Messages in CAN are sent in a format called frames. Framing of message is done by MAC sub layer of Data Link Layer.
- There are two type of frames standard or extended. These frames can be differentiated on the basis of identifier fields.
- A CAN frame with 11 bit identifier fields called Standard CAN and with 29 bit identifier field is called extended frame
- Standard Frame Format for Passenger Vehicles and Extended Frame Format for Heavy Vehicles
- Binary values in CAN protocol are termed as dominant (logic 0) and recessive (logic 1) bits.

CONTROLLER AREA NETWORK

CAN MESSAGE FRAME FORMAT

CAN Standard (Base) Frame Format

Applies to data and remote frames.



CAN Extended Frame Format

Applies to data and remote frames.

CONTROLLER AREA NETWORK

CAN MESSAGE FRAME FORMAT - STANDARD

- SOF bit: A dominant start of frame bit marks the start of a message. It is used to synchronize all the nodes on a bus after being idle. Transmitted by the sender.
- 11-bit Identifier: Identifier is NOT a destination node address. It serves dual purpose, one to determine which node has access to the bus and second to identify the type of message.
- RTR bit: The Remote Transit Request bit differentiates between data and remote frames. In data frames, this bit is dominant and in remote frames this bit is recessive.
- IDE bit: The Identifier Extension bit distinguishes between standard and extended frames. In standard frames this bit is dominant, in extended frames this bit is recessive.
- r0 bit: This bit is reserved for future CAN bus standards user. Always recessive.
- 4-bit DLC: Data Length Code (DLC) contains the number of bytes (0 to 8) that will be transmitted. DLC values from 9-15 are not allowed.

CONTROLLER AREA NETWORK

CAN MESSAGE FRAME FORMAT - STANDARD

- 0-8 bytes Data: This is the data payload. Up to 8 bytes can be sent in a single packet, as long as it is a data frame. For a remote frame, there must be no data bytes.
- 16-bit CRC: The Cyclic Redundancy Check (CRC) is used to detect errors in the packet. It consists of a 15-bit CRC value followed by a delimiter.
- 2-bit ACK: It comprises of the ACK slot and the ACK delimiter. When the data is received correctly the recessive bit in ACK slot is overwritten as dominant bit by the receiver. The 2nd bit is the ACK delimiter and is driven recessive by the transmitter.
- 7-bit EOF: The End Of Frame is marked with 7 recessive bits.
- IFS: Inter Frame Space specifies minimum number of bits separating consecutive messages. It provides the intermission between two frames and consists of three recessive bits.

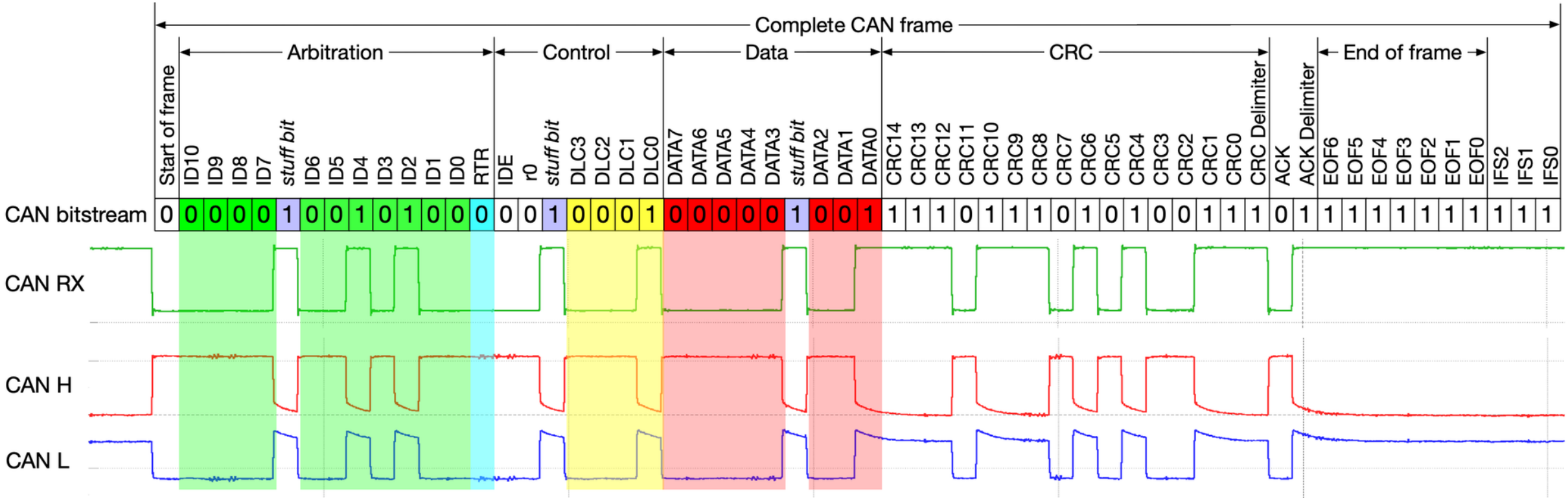
CONTROLLER AREA NETWORK

CAN MESSAGE FRAME FORMAT - EXTENDED

- The extended frame is the same as the above standard frame, except for the differences described below.
- SRR bit: The Substitute Remote Request bit is transmitted in extended frames at the position of the RTR bit in standard frames. It is always recessive.
- r1: An additional reserve bit for extended frames only. Must be recessive.
- 18-bit Identifier: Another 18-bits that can be used as part of the identifier, giving a total of 29-bits for the identifier in an extended frame. 11-bit identifiers have a higher priority than 29-bit identifiers.

CONTROLLER AREA NETWORK

CAN DATA TRAMISSION – TIMING DIGRAM



Stuff Bit: To avoid the excessive DC component In CAN, a complimentary bit will be added after the 5 successive same bits.

CONTROLLER AREA NETWORK

CAN MESSAGE FRAME - TYPES

- There are four different frames which can be used on the CAN bus.

1. Data frames- Most commonly used frame for data transmission from one node to another node. Used to transmit a data payload of up to 8 bytes.

2. Remote frames – seeks permission for data transmission from another node. It is similar to data frame without data field and RTR bit is recessive.

3. Error frames – If transmitting or receiving node detects an error, it will immediately abort transmission and send error frame consisting of an error flag made up of dominant bits and error flag delimiter made up of eight recessive bits.

4. Overload frame - If a CAN node receives messages faster than it can process them, then the CAN node can generate Overload Frames to delay the next data/remote frame. An overload frame contains an overload flag consisting of 6 dominant bits and then an overload delimiter of eight recessive bits.

CONTROLLER AREA NETWORK

HOW CAN PROTOCOL WORKS?

- Each message in CAN has a unique identification number.
- The identification number is specified according to the content of the message and stored in message identifier.
- This identification number is unique within the network so when the transmitting node places the data on the network for access to all nodes it checks unique ID number to allow the message to pass through the filter and rest are ignored
- All nodes receive the CAN frame and based on the ID nodes decides whether to accept it or not.
- With message based protocol other nodes can be added without re-programming since the units connected to the bus have no identifying information like node addressing.

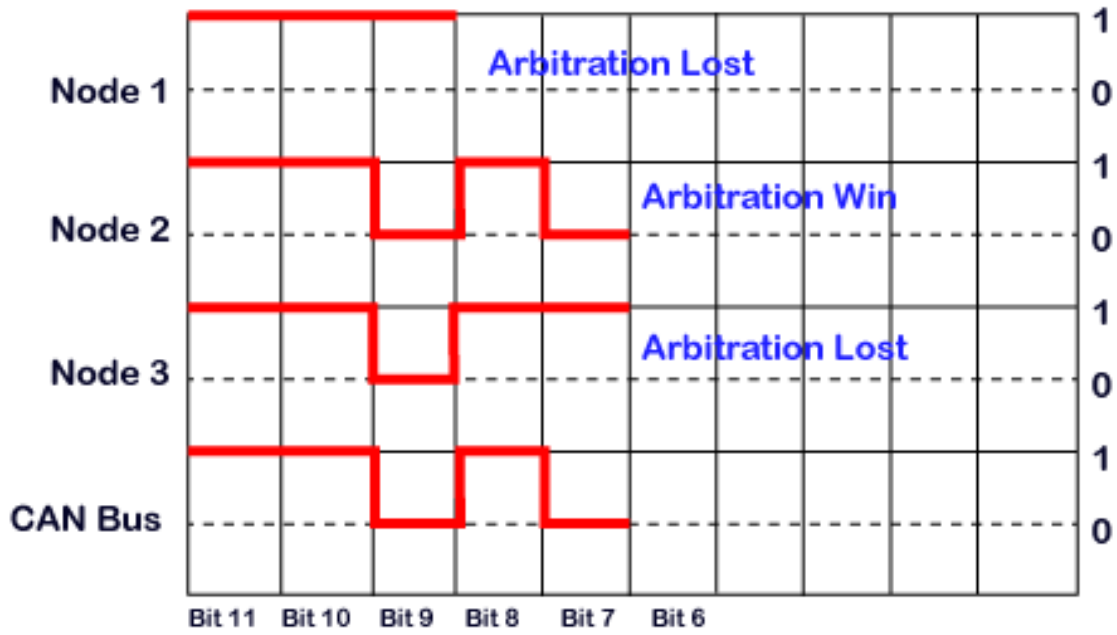
CONTROLLER AREA NETWORK

HOW CAN PROTOCOL WORKS? - ARBITRATION EXAMPLE

- As per CAN standard, two fields of data or remote frame decides the arbitration of CAN message over CAN line – Identifier(11-bit) and RTR (1-bit)

- If we consider three nodes, i.e., Node 1, Node 2, and Node 3, the message identifiers of these nodes are 0x7F3, 0x6B3, and 0x6D9, respectively.

CAN Node	Identifier (Hex)	Identifier (Binary)
1	0x7F3	11111110011
2	0x6B3	11010110011
3	0x6D9	11011011001



CONTROLLER AREA NETWORK

HOW CAN PROTOCOL WORKS? - ARBITRATION EXAMPLE

- The transmission of all the three nodes with the most significant bit is shown in diagram.
  - 11th bit: As all the three bits of nodes are recessive, so bus bit will remain recessive.
  - 10th bit: All the nodes have 10th bit as recessive, so the bus will remain recessive.
  - 9th bit: Node 1 has recessive bit while other nodes have a dominant bit, so the bus will also remain dominant. In this case, node 1 has lost the arbitration, so it stops sending bits.
  - 8th bit: Both node 2 and node 3 are sending recessive bit, so that the bus state will remain recessive.
  - 7th bit: The node 2 is sending dominant bit while node 3 has sent recessive bit, so that the bus state will remain dominant. In this case, the node 3 has lost the arbitration, so it stops sending the message while the node 2 has won the arbitration means that it will continue to hold the bus until the message is received.

CONTROLLER AREA NETWORK

ADVANTAGES

- Cost-Effective: Reduces the need for complex wiring, resulting in cost savings
- Reliable: Highly immune to EMI and excellent error detection and error handling
- Flexibility: can be easily connected / disconnected
- Broadcast capability

DISADVANTAGES

- Due to electrical loading, maximum number of connected devices is limited to 64.
- Cable length is limited to 40 meters in length which could limit some applications.
- Maximum speed is 1 Mbit/second, which is very less.
- Produce excessive electric noise due to different voltage levels

BLUETOOTH

BLUETOOTH

INTRODUCTION

- Bluetooth is short-range, low-power, low-cost, wireless data protocol designed for sending and receiving data between electronics devices over a secure 2.4GHz network.
- It was originally conceived during the late 90's as an in-house project at Nokia, however, it quickly spread to become a standard for wireless data.
- Bluetooth is a dynamic standard where devices can automatically find each other, establish connections, and discover what they can do for each other on an ad hoc basis.
- It is designed to be energy efficient, making it suitable for use in battery-powered devices like smartphones, smartwatches, wireless headphones, and IoT devices.
- Bluetooth technology is widely used in various applications, including wireless audio streaming, file transfer between devices, wireless printing, wireless keyboards and mice, automotive hands-free systems, IoT devices, and smart home automation.

BLUETOOTH

VERSIONS

Bluetooth 1.0	Bluetooth 2.0 + EDR	Bluetooth 3.0 + HS	Bluetooth 4.0
1998.10 – 2003. 11 "Base Rate" - 1Mbps data rate - V1.0 - Draft - V1.0A - published on 1999.7 - V1.0B Enhanced the Interoperability - V1.1 - IEEE 802.15.1 - V1.2 Enhanced the compatibility	2004. 11 – 2007. 7 "Enhanced Data Rate" - Higher ordered modulation for data payload - 2Mbps or 3Mbps physical data rate - V2.0 - V2.1	2009. 4 "HS Mode" - AMP Alternative MAC/PHY - Implement high data rate by using 802.11 protocols. - Facing the Challenge from Wi-Fi - V3.0	2010. 6 – 2014. 12 "Low Energy" - Facing the IoT application - Changed the protocol greatly, almost a new technology - V4.0 - V4.1 - V4.2

- Bluetooth will have distance coverage to about 1 to 100 meters based on power class supported on Bluetooth devices. Class-1 covers 1 meter, Class-2 covers 10 meters, class-1 covers 100 meters.

BLUETOOTH

VERSIONS

	Released In	Transfer Speed	Range	Distinct Features
Bluetooth 5	July 2016	50MB per second	200 meters	Improved range, speed and stability. Support for multiple simultaneous connections
Bluetooth 5.2	Dec 2019	50MB per second	200 meters	EATT, ISOC, and LE Audio
Bluetooth 5.3	July 2021	50MB per second	200 meters	Periodic Advertising Enhancements, Encryption Key Size Control Enhancements, Connection Sub Rating, and Channel Classification Enhancement

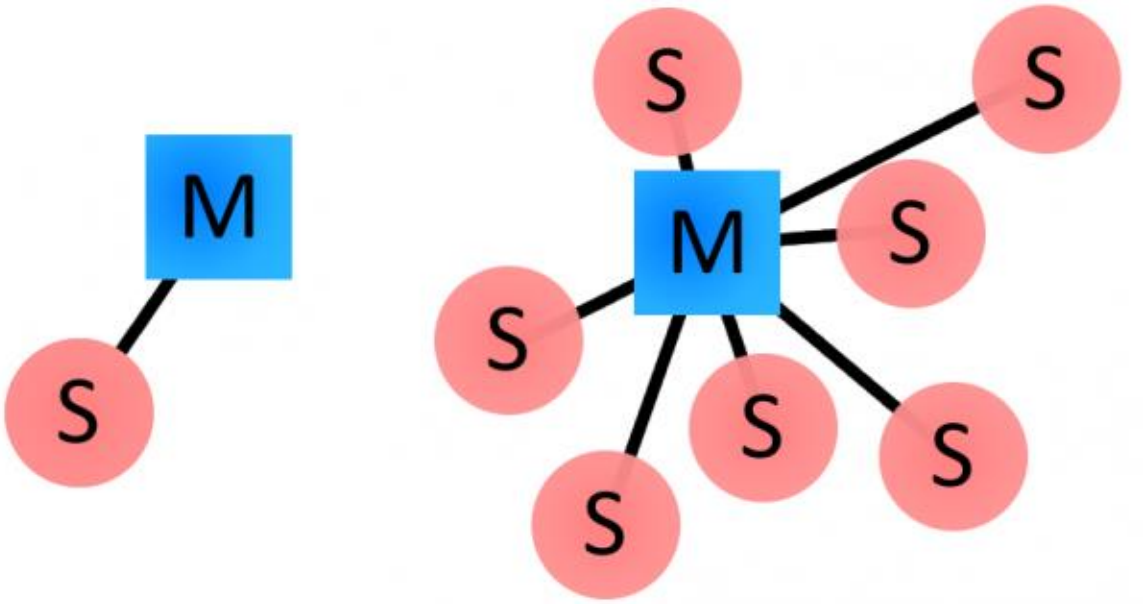
BLUETOOTH

BLUETOOTH NETWORK

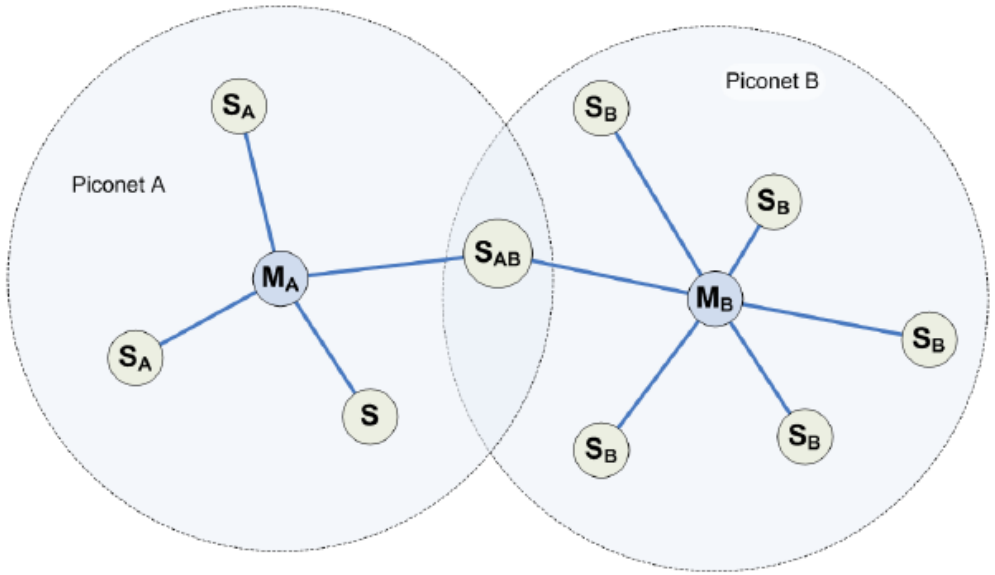
- Bluetooth networks (also referred as piconets/scatternet) use a master/slave model to communicate with other devices.
- In this model, a single master device can be connected to up to seven different slave devices.
- The master coordinates communication throughout the piconet, it can send/request data to/from any slaves.
- Any slave device in the piconet can only be connected to a single master. Slaves are only allowed to transmit and receive from their master and can't communicate with other slaves.
- Combinations of multiple piconets is known as scatternet. A device can participate in multiple piconets.

BLUETOOTH

BLUETOOTH NETWORK



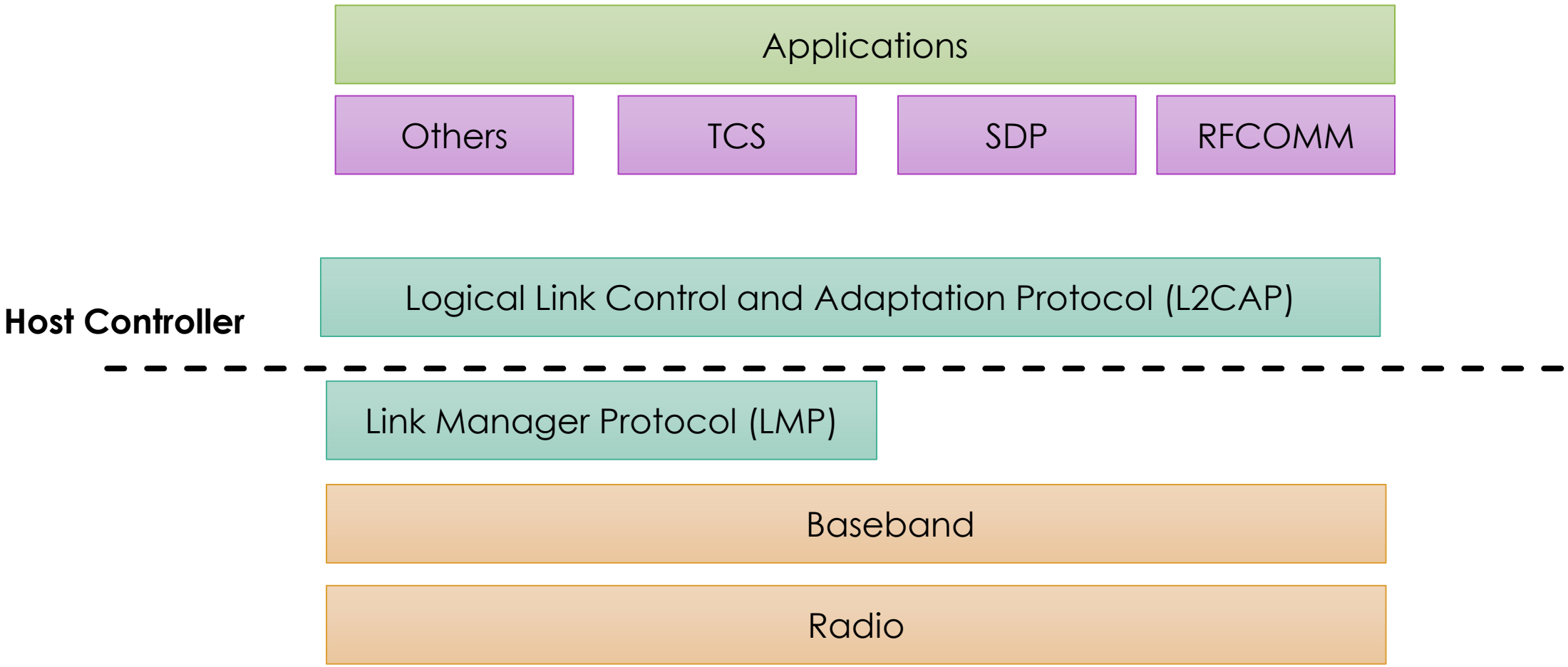
Examples of Bluetooth master/slave piconet topologies.



Scatternet

BLUETOOTH

BLUETOOTH ARCHITECTURE



BLUETOOTH

BLUETOOTH ARCHITECTURE

- **Radio:** Defines the requirements for a transceiver to operate in the 2.4 GHz ISM band such as frequency bands, frequency hopping specifications, and modulation techniques.

- **Baseband:** Describes the specification of the Bluetooth Link Controller (LC), and carries baseband protocol. It defines the addressing scheme, packet frame format, timing, and power control algorithms.

- Two types of link can be created in baseband are ACL and SCO.

Asynchronous Connection Less (ACL)

- Packet switched data
- Slave can have only one ACL link to master
- Used for correct delivery over fast delivery
- Maximum data rate of 721 kbps

Synchronous Connection Oriented (SCO)

- Real time data transmission
- Damaged packet cannot be retransmitted
- Data rate of 64 kbps

BLUETOOTH

BLUETOOTH ARCHITECTURE

- LMP: Used by the Link managers for link set-up and control. The other main functions of LMP are device authentication, message encryption, and negotiation of packet sizes.
- Host Controller Interface (HCI): HCI provides a command interface to the Baseband LC and Link Manager, to access hardware status and control registers.
- L2CAP: Logical Link Control and Adaptation Protocol (L2CAP) supports higher level protocol multiplexing, packet segmentation and reassembly, and also conveys the QoS.
- RFCOMM: RFCOMM protocol provides emulation of serial ports over the L2CAP protocol.
- SDP: The Service Discovery Protocol (SDP) provides a means for applications to discover, which services are provided by or available through a Bluetooth device
- TCS: Telephony control protocol for telephony service
- Applications: This includes the application profiles that allow the user to interact with the Bluetooth applications.

BLUETOOTH

BONDING AND PAIRING

- When two Bluetooth devices want to connect, they can be bonded together. Bonded devices automatically establish a connection whenever they're close enough.
- Bonds are created through a process called pairing. On paring devices share their addresses, names, and profiles, and they are stored in memory.
- They also share a common secret key, which allows devices to bond in future whenever required. Pairing requires an authentication to establish connection between devices.
- Sometimes pairing is a simple "Just Works" operation, where the click of a button is all it takes to pair (this is common for devices with no UI, like headsets).
- Older, legacy (v2.0 and earlier), pairing processes involve the entering of a common PIN code on each device. The PIN code can range in length and complexity from four numbers (e.g. "0000" or "1234") to a 16-character alphanumeric string.

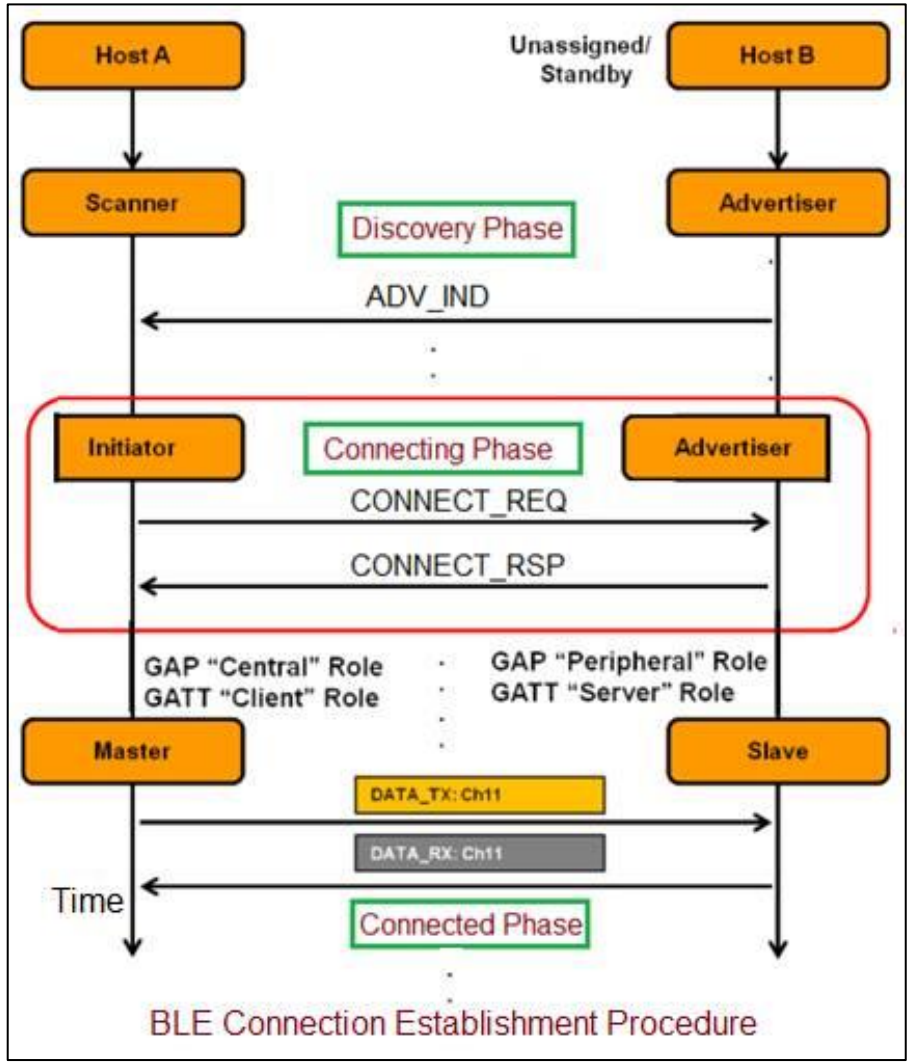
BLUETOOTH

CONNECTION PROCESS

- ✓ **Step-1 Inquiry (Discovery)** - If two Bluetooth devices know absolutely nothing about each other, one must run an inquiry to try to discover the other. On receiving the inquiry the device which listens to such a request will send back address, name and other information

- ✓ **Step-2 Pairing (Connecting)** – Process of forming a connection between two Bluetooth devices.

- ✓ **Step-3 Connection** – After a device has completed the pairing process, it enters the connection state. The mode of operation of device is decided in connection phase.



BLE Connection Establishment Procedure

BLUETOOTH

CONNECTION MODES

1. Active Mode – regular connected mode, where device send and receive data (up-to 7 devices)

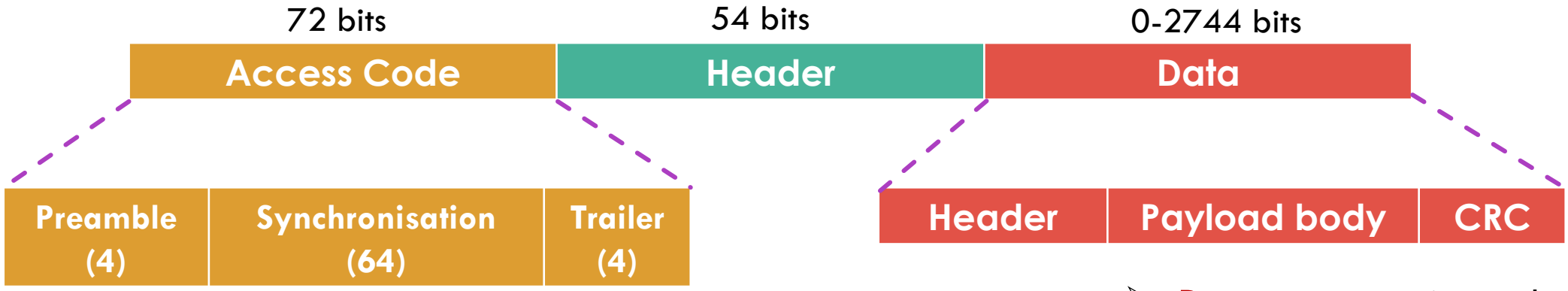
2. Sniff Mode – power saving mode, less active mode (frees slave for predetermined period of time, recurring fixed time slot)

3. Hold Mode – temporary mode, device will be sleeping for a particular period of time (frees slave for predetermined period of time, one time slot)

4. Park Mode – sleep mode with infinite time, slave will wake only if master tells (max. 255 devices)

BLUETOOTH

MESSAGE FORMAT



Access code (AC)

- Used for packet identification
- Once packet is received, receiver in piconet will compare the incoming signal with AC, if any mismatch is found packet will be ignored
- 72 bit AC is derived from master identity
- It is used for synchronization
- Three types
  - Channel AC – targets piconet ID
  - Device AC – individual devices
  - Inquiry AC – used during pairing

- Data - contains data or control information from upper layer
- Header – identifies logical channel
- Payload body – contains user data
- CRC- 16 bit checksum

BLUETOOTH

MESSAGE FORMAT

3 bits	4 bits	1 bits	1 bits	1 bits	8 bits
AM_ADDR	Type	Flow	ARQN	SEQN	HEC

Header

- Address – it can address up to 7 slave devices, if it is 0, then messages will broadcasted
- Type – defines type of incoming data
  - ACL – Data medium (DM) or Data High (DH)
  - SCO - Data Voice (DV) and High Quality Voice (HV)
- 12 types of packet for each SCO and HV
- 4 common control packets
  - Flow- 1 bit flow control
  - ARQN – 1 bit acknowledgement
  - SEQN – 1 bit sequential numbering scheme for packet ordering
  - HEC – Header error check
- Header = 3 X 18 bits = 54 bits

BLUETOOTH

ADVANTAGES:

- Adhoc based more secure wireless connectivity
- Low cost & low power consumption.
- No issue of interoperability among different Bluetooth vendor products
- Can be used for voice and data transfer.
- Less interference compare to other wireless technologies

DISADVANTAGES:

- Data rate is lower compare to WiFi
- Limited coverage distance

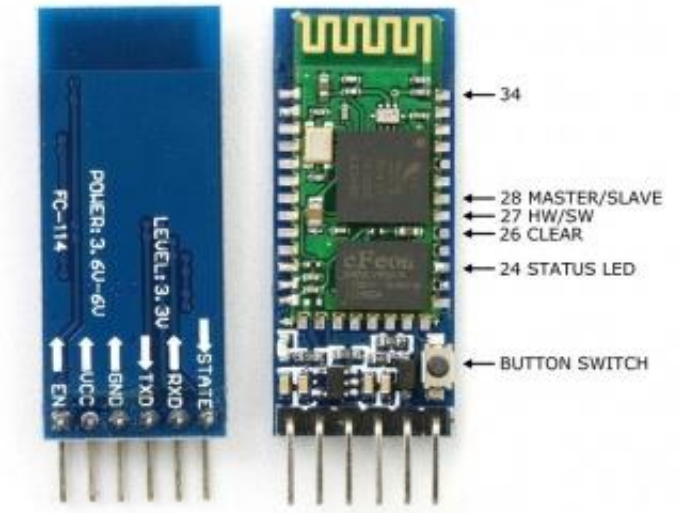
BLUETOOTH

HC-05 BLUETOOTH MODULE

- HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.
- This serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps modulation with complete 2.4GHz radio transceiver and baseband.
- The Bluetooth module HC-05 is a MASTER/SLAVE module. By default the factory setting is SLAVE.
- The Role of the module (Master or Slave) can be configured only by AT COMMANDS.
- Master module can initiate a connection to other devices. The user can use it simply for a serial port replacement to establish connection between MCU and GPS, PC to your embedded project, etc.

BLUETOOTH

HC-05 BLUETOOTH MODULE

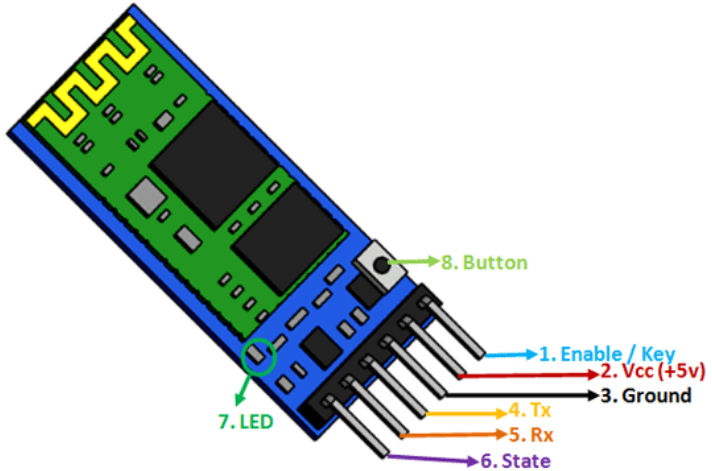


Hardware Features

- Up to +4 dBm RF transmit power.
- 3.3 to 5 V I/O.
- PIO(Programmable Input/Output) control.
- UART interface with programmable baud rate.

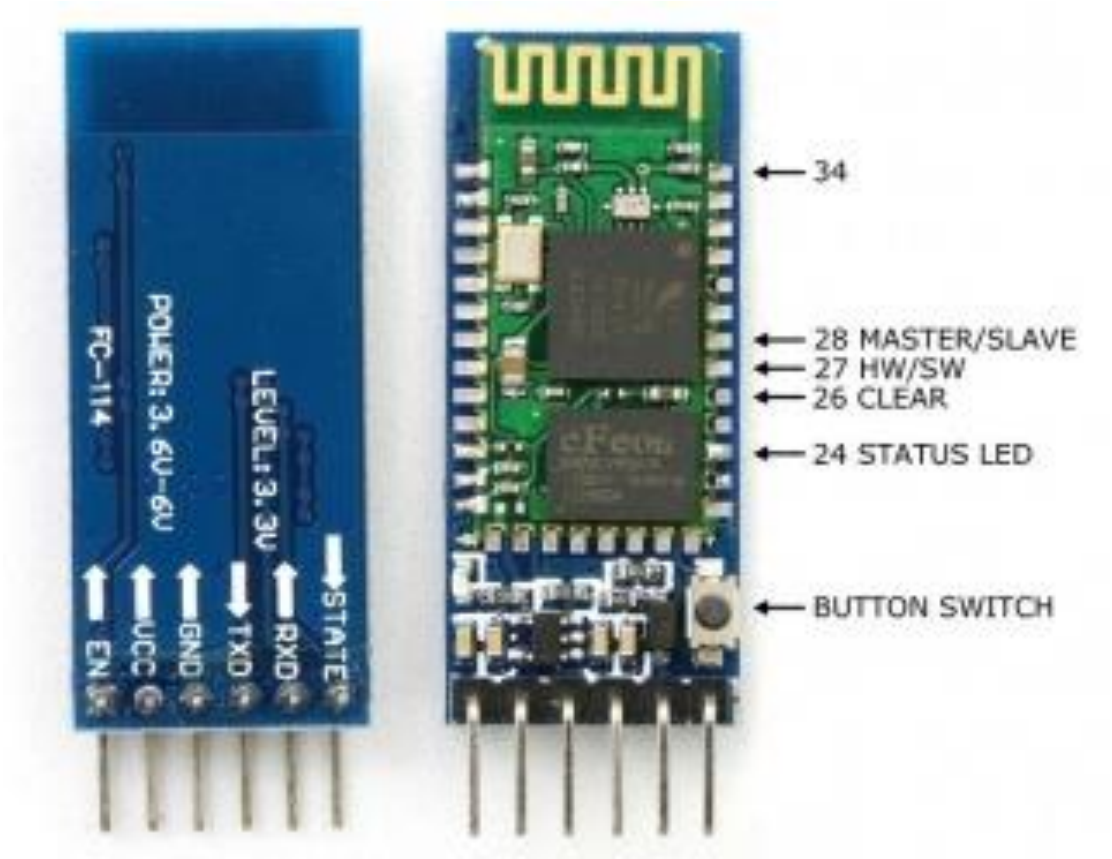
Software Features

- Slave default Baud rate: 9600, Data bits:8, Stop bit:1,Parity:No
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"1234" as default



BLUETOOTH

HC-05 BLUETOOTH MODULE



Pin Configuration

Pin Number	Pin Name	Description
1	Enable / Key	This pin is used to toggle between Data Mode (set low) and AT command mode (set high). By default it is in Data mode
2	Vcc	Powers the module. Connect to +5V Supply voltage
3	Ground	Ground pin of module, connect to system ground.
4	TX – Transmitter	Transmits Serial Data. Everything received via Bluetooth will be given out by this pin as serial data.
5	RX – Receiver	Receive Serial Data. Every serial data given to this pin will be broadcasted via Bluetooth
6	State	The state pin is connected to on board LED, it can be used as a feedback to check if Bluetooth is working properly.
7	LED	Indicates the status of Module • Blink once in 2 sec: Module has entered Command Mode • Repeated Blinking: Waiting for connection in Data Mode • Blink twice in 1 sec: Connection successful in Data Mode
8	Button	Used to control the Key/Enable pin to toggle between Data and command Mode

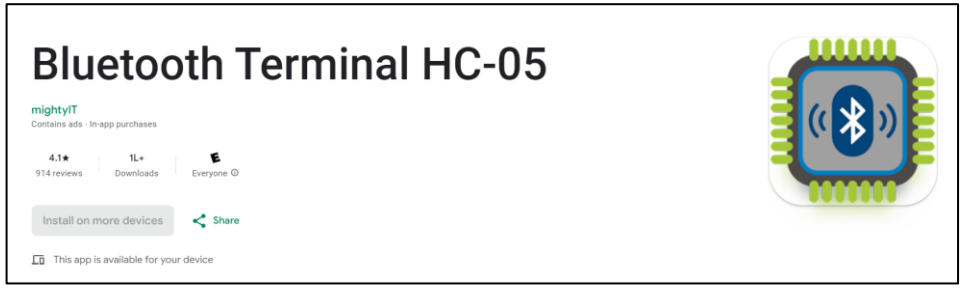
HC-05 DAT SHEET: https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf

BLUETOOTH

Example-1:Data transfer between NUCLEO HC-05 and Smartphone

Write a C++ code with mbed APIs to communicate with Smart phone via Bluetooth module by transferring ASCII values between them. Values received by NUCLEO board will be displayed in PC Teraterm and value received by smartphone displayed on the App screen. Implement this logic on Nucleo board and HC-05 Bluetooth module using Keil studio cloud online compiler.

Note: use "Bluetooth Terminal HC-05" app from Google play store on smartphone for communication

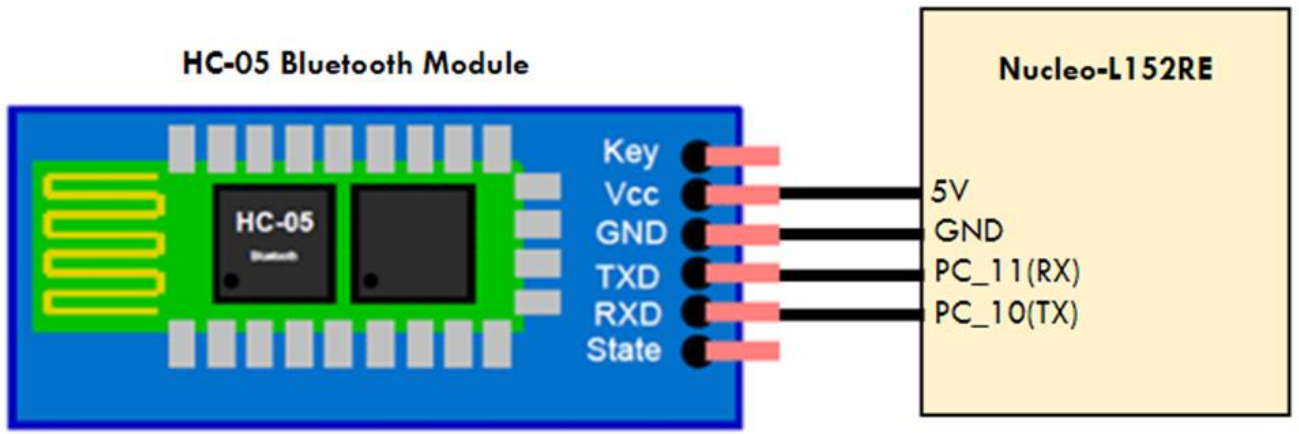


BLUETOOTH

Program

```
#include "mbed.h"
Serial pc(USBTX,USBRX);
Serial bt(PC_10,PC_11);
int main(void){
char ch;
bt.baud(9600);
pc.baud(9600);
pc.printf("Hello World!\n\r");
while(1){
if(bt.readable())
{
ch=bt.getc();
pc.printf("%c",ch);
}
if(pc.readable())
{
ch=pc.getc();
bt.printf("%c",ch);
}
}
}
```

Connection diagram



BLUETOOTH

Example-2: Bluetooth based Home automation

Write a mbed C++ program to design a Home automation system to control the home appliances through smart phone. This System consist of HC-05 Bluetooth module, USB serial monitor and buzzer interfaced with Nucleo. The system receives the input from smart phone through Bluetooth app : based on the value, it controls the LED and buzzer as per the below:

- If the received character is '1' , LED should be OFF and display the message 'LED ON' on serial monitor.
- If the received character is '2' , LED should be ON and display the message 'LED OFF' on serial monitor.
- If the received character is '3' , Buzzer should be ON and display the message 'Buzzer ON' on serial monitor.
- If the received character is '4' , Buzzer should be OFF and display the message 'Buzzer OFF' on serial monitor.

Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE

BLUETOOTH

Program

```
#include "mbed.h"
Serial pc(USBTX,USBRX);
Serial bt(PC_10,PC_11);
DigitalOut led(LED1);
DigitalOut buzz(PB_8);
int main(){
    char ch;
    bt.baud(9600);
    pc.baud(9600);
    pc.printf("Lab Task 2\n\r");
    while(1){
        if(bt.readable()){
            ch=bt.getc();
            if(ch == '1'){
                led = 0;
                pc.printf("Led off\n\r");
            }
            if(ch == '2'){
                led = 1;
                pc.printf("Led On\n\n\r");
            }
            if(ch == '3'){
                buzz = 0;
                pc.printf("Buzzer Off\n\r");
            }
            if(ch == '4'){
                buzz = 1;
                pc.printf("Buzzer On\n\r");
            }
        }
    }
}
```

BLUETOOTH

Exercise: Temperature sensor LM35 interfacing

Write a program to design a temperature monitoring system using LM 35 and HC-05 Bluetooth module, buzzer and STM32 Nucleo-64 board.

- Read the temperature continuously from LM35 temperature sensor and display it on smart phone
- When the temperature goes above 35 C, turn on the buzzer.

Design and verify this logic on Nucleo 152RE board using online Keil Studio platform.

ZIGBEE

ZIGBEE

INTRODUCTION

- Zigbee is a wireless communication protocol designed for low-power, low-data-rate, and short-range applications.
- Introduced in 2003 by the Zigbee Alliance, Zigbee is based on the IEEE 802.15.4 standard and is widely used in IoT and wireless sensor network (WSN) applications.
- Zigbee supports different network configurations for the master to master or master to slave communications.
- ZigBee also supports a larger number of nodes than Bluetooth, which makes it more suitable for large-scale deployments.
- ZigBee versions: (i) ZigBee-2004,2006,2007 (ii) ZigBee-2007,2015,2017 (iii) ZigBee 3.0- 2016

ZIGBEE

ZIGBEE CHARACTERISTICS

- Low Power Consumption
- Low Data Rate (20- 250 kbps)
- Short-Range (75-100 meters)
- Network Join Time (~ 30 msec)
- Support Small and Large Networks: (up to 65000 devices (Theory), 240 devices (Practically))
- Low Cost and Cheap Implementation (Open Source Protocol)
- Uses AES encryption for security
- 3 frequency bands with 27 channels (Only one will be selected for use in a network):
  - Channel 0: 868 MHz (Europe),
  - Channel 1-10: 915 MHz (the US and Australia)
  - Channel 11-26: 2.4 GHz (Across the World)

ZIGBEE

ZIGBEE DEVICES

- ZigBee network consist of three type of devices (i) coordinator (ii) router (iii) end device
- Zigbee Coordinator: It plays a crucial role in managing and coordinating the activities of the network. Major responsibilities of the coordinator in a Zigbee network includes:
  - ✓ Network Formation: Initiate the formation of the network by allowing devices to join and assigning network addresses to them.
  - ✓ Device Management: It ensures that only authorized devices are part of the network by controlling their association, disassociation, and rejoining processes.
  - ✓ Beacon Management: Periodically transmits beacon frames to synchronize and maintain communication within the network also, configures the beacon interval to ensures consistent beacon transmission.
  - ✓ Address Assignment: It manages the address allocation process to prevent address conflicts and ensure proper communication between devices.
  - ✓ Channel Selection: Selects and manages the operating channel by monitoring channel conditions, interference, and noise levels to optimize network performance and reliability.

ZIGBEE

ZIGBEE DEVICES

- **Zigbee Router:** It refers to a device that extends the coverage and connectivity of the network by forwarding data packets between devices. Major responsibilities includes:
  - ✓ **Routing Data Packets:** They examine the destination address of incoming packets and determine the most efficient path for forwarding the packets to their destinations.
  - ✓ **Network Formation and Maintenance:** They assist in establishing the network topology, coordinating device joining, and updating routing information as devices move within the network.
  - ✓ **Routing Table Maintenance:** It maintains and updates the routing table (information about the network topology and routes) based on changes in the network and ensures efficient data routing.
  - ✓ **Link Quality Estimation:** Routers monitor the quality of links between neighbouring devices by measuring factors such as signal strength, packet loss, and latency.
  - ✓ **Network Optimization:** Routers optimize network performance by dynamically adjusting routing paths, managing network congestion, and balancing traffic load across multiple routes.

ZIGBEE

ZIGBEE DEVICES

- **Zigbee End device:** In a Zigbee network, end devices serve as the endpoints that interact with sensors, actuators, and other peripheral devices. Major responsibilities includes:
  - ✓ **Data Sensing or Generation:** End devices collect data from sensors or generate by monitoring environmental conditions, detect events, or measure parameters as per their designated function.
  - ✓ **Data Transmission:** Transmit data to other devices within the ZigBee network or to a centralized coordinator using ZigBee communication protocols.
  - ✓ **Message Processing and Actuation:** They process incoming messages received from other devices and interpret commands, execute control actions, or trigger events based on the message content.
  - ✓ **Power Management:** End devices employ power-saving techniques such as sleep modes, and low-power wake-up to conserve energy when idle and extend battery life.
  - ✓ **Application-Specific Functions:** End devices perform application-specific functions tailored to their intended use case like home automation, industrial monitoring, healthcare, asset tracking, etc.,

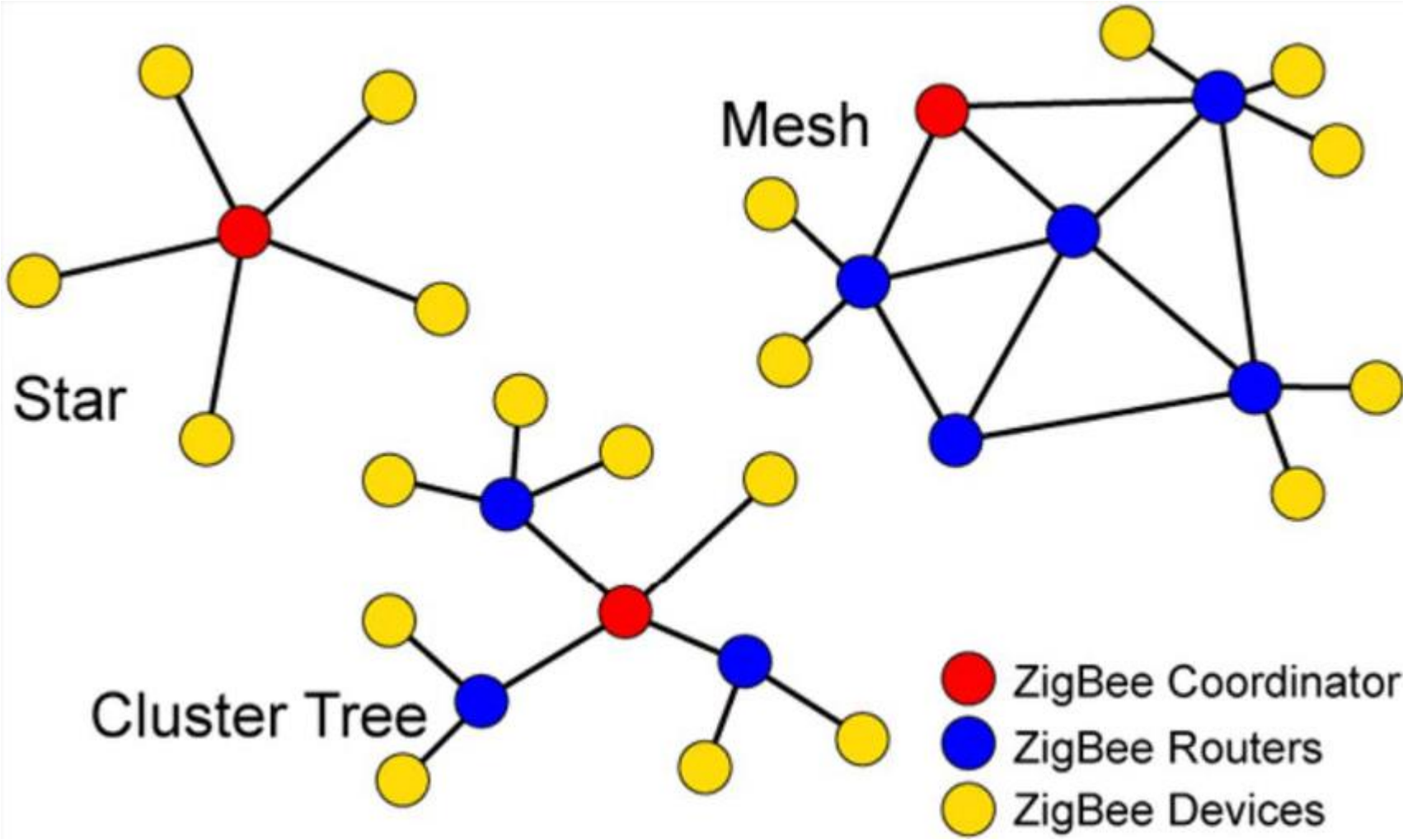
ZIGBEE

DIFFERENT NETWORK TOPOLOGY IN ZIGBEE

- Network topology refers to the way in which network has been designed. There are three type of network topologies supported by Zigbee, (i) Mesh (ii) Star (iii) Cluster Tree
- Mesh : In Mesh Topology, every node is connected with each other node except the end device because end devices can't communicate directly. Main advantage of Mesh network is that if one of the links becomes unusable, it does not affect the entire system.
- Star: In a star topology, each device has a dedicated point-to-point connection to a central controller (Coordinator). If one end device wants to send data to another, it sends the data to the coordinator, which further sends the data to the destination device.
- Tree: In Tree network, end devices are generally clustered around each router. It's not very different from a mesh configuration except the fact that there routers are not interconnected.

ZIGBEE

DIFFERENT NETWORK TOPOLOGY IN ZIGBEE



ZIGBEE

ZIGBEE CHANNAL ACCESS

- There are two methods how smart devices can access the ZigBee network.

1. Contention-free method (beacon-enabled network)

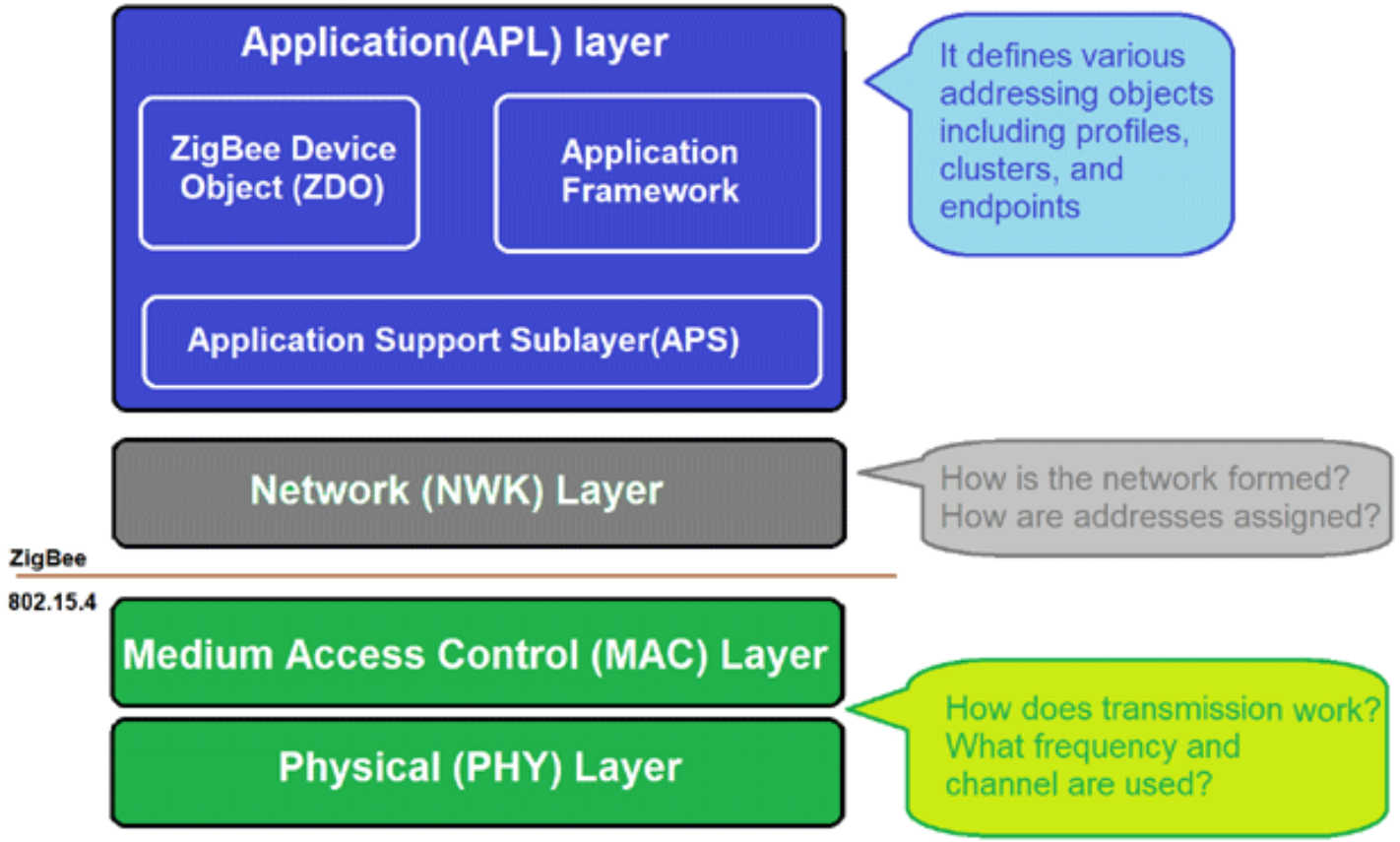
- Coordinator dedicates a specific time to each device, called the guaranteed time slot (GTS).
- The time slots for every device is guaranteed and therefore can not overlap.
- The coordinator transmits a synchronization messages (called beacon) periodically to each device in the network to synchronize the clock of every device.

2. Contention method (non beacon-enabled network)

- The network use CSMA/CA, when an end device wants to transmit data, first the device switches into the receiver mode and detects if there is any signal in the channel.
- If there is no signal, the device switches into the transmit mode and transfers the data.
- But if there is already a signal in the channel, the device backs off of for a random period of time and restart the process to transfer data until the message is sent

ZIGBEE

ARCHITECTURE



ZIGBEE

ARCHITECTURE

- Physical Layer: Perform modulation and demodulation operations upon transmitting and receiving signals respectively.
- MAC Layer: This layer is responsible for reliable transmission of data by accessing different networks with CSMA/CA. This also transmits the beacon frames for synchronizing communication.
- Network Layer: This layer takes care of all network-related operations such as network setup, end device connection, and disconnection to network, routing, device configurations, etc.
- Application Support Sub-Layer(APS): This layer enables the services necessary for ZDO and application objects to interface with the network layers for data managing services.
- Zigbee device objects (ZDO): Provides an interface between application objects and the APS in Zigbee devices. It is responsible for detecting, initiating, and binding other devices to the network.
- Application Framework: It provides two types of data services (i) Generic message is a developer-defined structure (ii) key-value pair is to get attributes within the application objects.

ZIGBEE

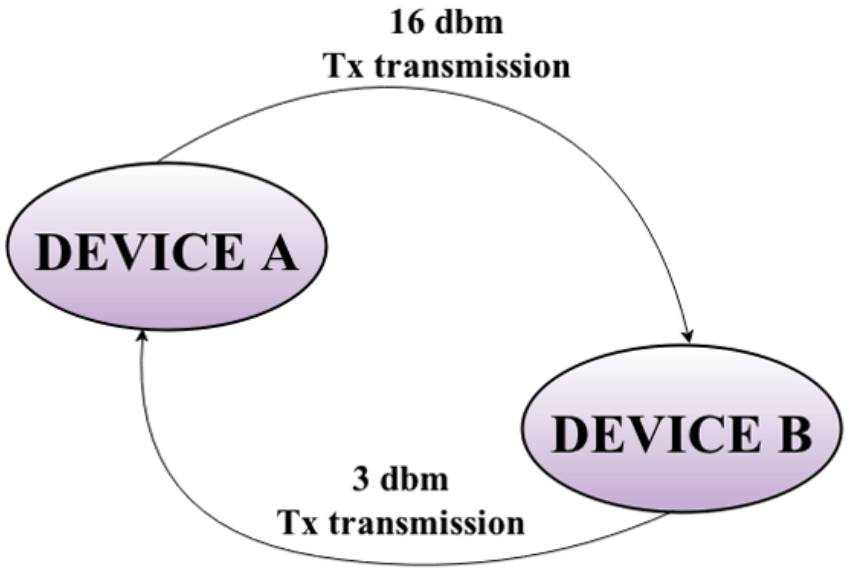
HOW ZIGBEE COMMUNICATION WORKS?

- Each device in a Zigbee network is assigned a unique 64-bit IEEE address and a shorter 16-bit network address. Devices use these addresses to communicate with each other.
- This 64-bit address is unique universally; it is firmed inside the ZigBee module by the manufacturer. A device receives a 16-bit address from coordinator which should be unique locally, when it joins a ZigBee network.
- Routers maintain routing tables to determine the best paths for data transmission based on destination addresses.
- Zigbee communication is packet-based, with data transmitted in frames consisting of headers, payload, and optional footer. The four basic frame types defined in 802.15.4: Data, ACK, MAC command, and beacon.
- Devices in a Zigbee network can use acknowledgments to confirm successful data transmission and request retransmissions in case of packet loss or errors.

ZIGBEE

DATA TRANSMISSION IN ZIGBEE - UNICAST

- Unicast transmissions in ZigBee route data from one source device to another destination device.
- The destination device could be an immediate neighbour of the source device, or it could have several hops in between the way.
- An example is shown below in the figure explaining mechanism for recognizing the reliability of the bi-directional link.



Neighbour of A : B

Outgoing Link : Reliable

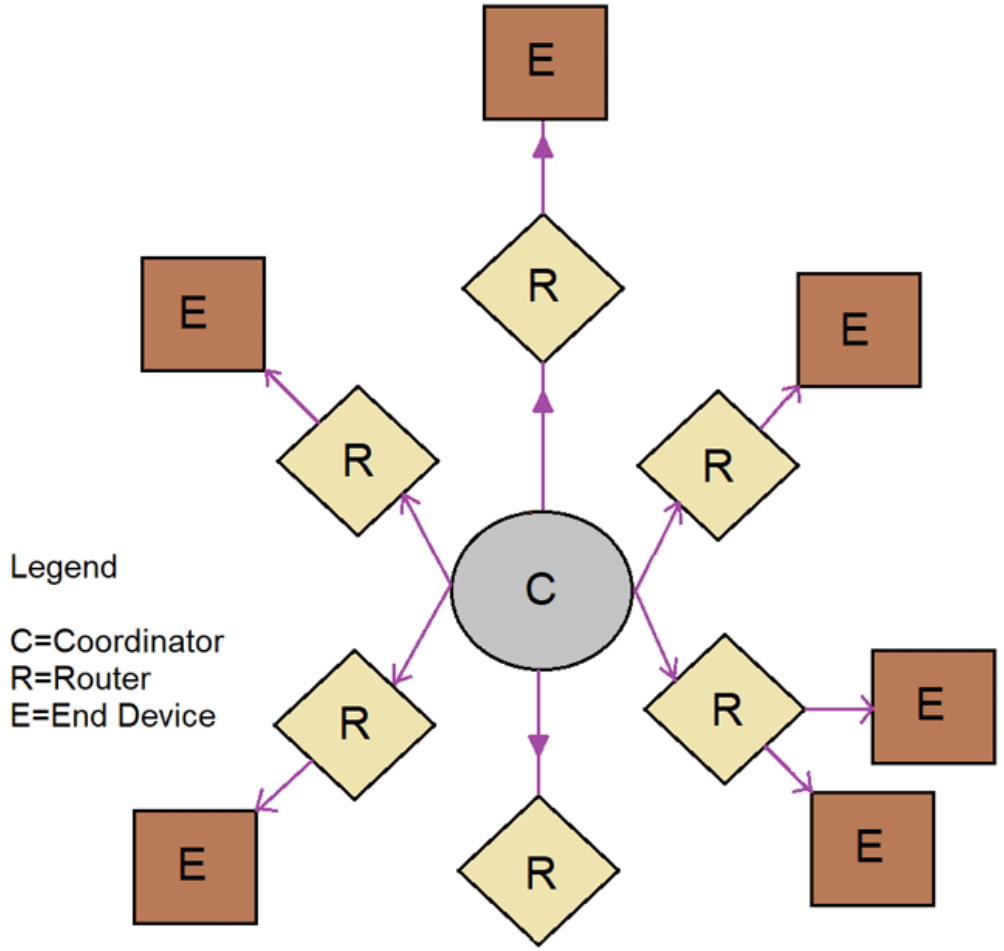
Neighbour of B : A

Outgoing Link : Poor
Incoming Link :

ZIGBEE

DATA TRANSMISSION IN ZIGBEE - BROADCAST

- Broadcast transmissions with the ZigBee protocol are propagated in the whole network such that all nodes receive the transmission.
- To accomplish this, the coordinator and all routers that receive a broadcast transmission will retransmit the packet three times.



ZIGBEE

ADVANTAGES:

- Setting up the network is very simple and easy.
- Low power consumption due to the low transmission rate.
- Faster response speed, generally only 15ms from sleep to working state.
- The network is scalable and it is easy to add/remote end device to the network.

DISADVANTAGES:

- Relatively short range compared to other wireless communications protocols
- Less suitable for applications that require high-speed data transfer.
- Not widely used (unlike IoT) which make it difficult to find devices that are compatible
- Security features are not robust like IoT making it more vulnerable to security threats.

NFC

NEAR FIELD COMMUNICATION

INTRODUCTION

- NFC is a short-range half-duplex wireless communication technology that enables devices to communicate with each other when they are within close proximity (few cm).
- NFC operates with low-data rate at 13.56 MHz and allows for data transfer and communication between devices usually by tapping or holding them near each other.
- It supports data rate of 106 Kbps , 212 kbps and 424 Kbps.
- NFC is part RFID (radio-frequency identification) and part Bluetooth.
  - Unlike RFID, NFC tags work in close proximity, giving users more precision.
  - NFC supports one and two-way communication, which gives the NFC an upper hand in use cases where transactions are dependent on data from two devices (e.g., card payments).
  - NFC also doesn't require manual device discovery and synchronization as Bluetooth.

NEAR FIELD COMMUNICATION

NFC FEATURES

Features	NFC Support
RF Carrier Frequency	13.56 MHz
Distance	less than 10 cm
Data Rate	106 or 212 or 424 Kbps
NFC Network Devices	Tags and Readers
NFC Tag Types	Type 1 to 5
Network configuration	Peer to peer
NFC Network Device Modes	Card Emulation, Reader/Writer, Point to Point
NFC devices communication modes	Active-Passive or Active-Active
Connection establishment time	Few seconds (approx. less than 0.1 sec)
Data Coding Schemes	NRZ-L, Manchester, Modified Miller
Data Modulation Schemes	Amplitude Shift Keying (ASK), Binary Phase-shift keying (BPSK)
Collision mechanism (i.e. MAC)	Anti-collision support

NEAR FIELD COMMUNICATION

APPLICATIONS

- NFC technology allows two devices housing NFC chip to communicate for various applications,
  - Data communication between smartphones
  - Access authentication for doors or offices
  - Banking payments using NFC compliant credit card
  - Mobile payments, such as Apple Pay and Google Pay
  - Ticket redemption at a concert or theatre
  - Transit card payments

NEAR FIELD COMMUNICATION

NFC DEVICES

- NFC devices can be classified into two types: Passive and Active NFC devices.

1. Passive NFC devices:

- These devices include tags and small transmitters that can only send information to other NFC devices without the need for a power source of their own.
- These devices don't really process any information sent from other sources, and can not connect to other passive components.
- Example: Bank credit/Debit cards, access cards, .

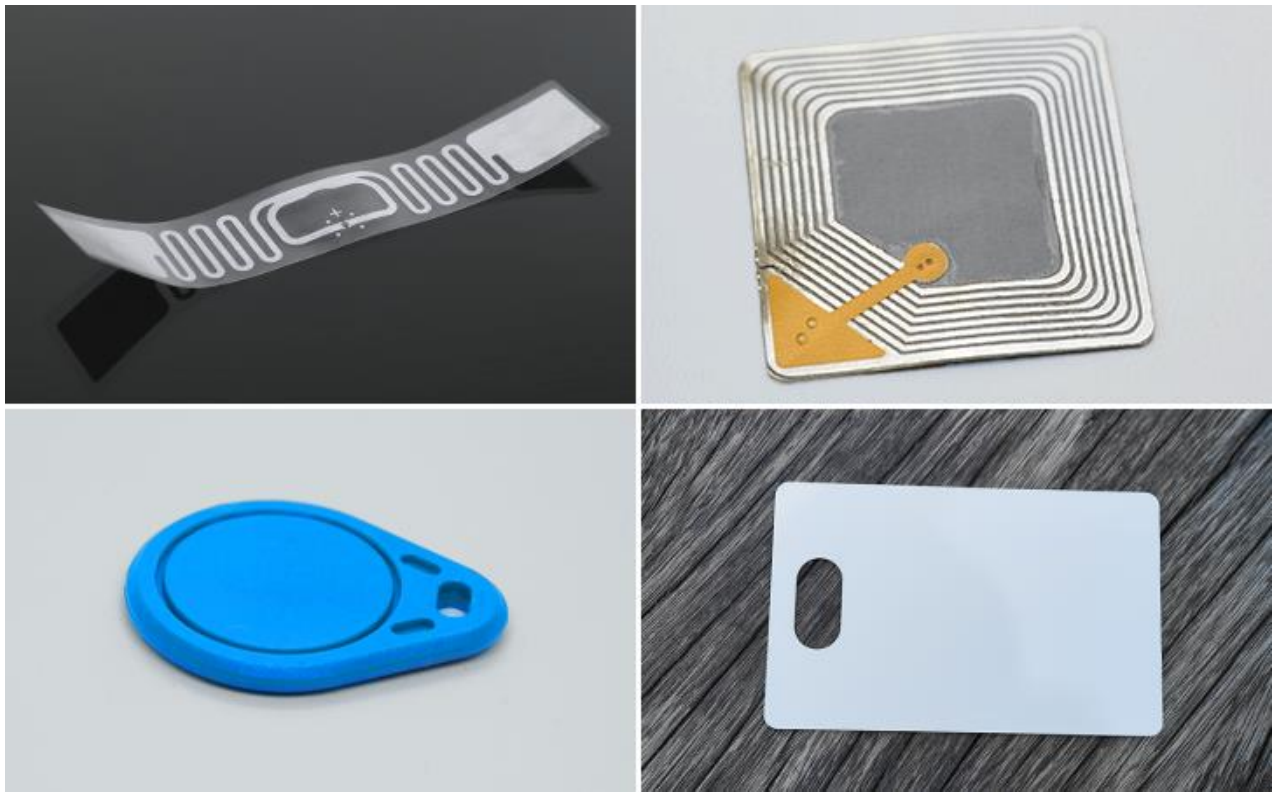
2. Active NFC devices:

- These devices can send and receive data.
- They can communicate with each other as well as with passive devices.
- Examples: Smartphones, NFC card readers in public transport and retails stores.

NEAR FIELD COMMUNICATION

NFC DEVICES

NFC Card Readers (Active Devices)



NFC Tags (Passive Devices)

NEAR FIELD COMMUNICATION

NFC – TAG TYPES

- NFC tag are categorized into five subtypes, labelled as Type 1 through 5.

NFC TAG TYPE	TYPE 1	TYPE 2	TYPE 3	TYPE 4	TYPE 5
DATA ACCESS	Read/write, can be configured as read-only.	Read/write, can be configured as read-only.	Pre-set as read-only or read/write.	Pre-set as read-only or read/write.	Pre-set as read-only or read/write.
MEMORY CAPACITY	96 bytes, expandable to 2 kB.	48 bytes, expandable to 2 kB.	1/4/9 kB.	32 kB.	32 kB.
COMMUNICATION SPEED	106 kbit/s.	106 kbit/s.	212-424 kbit/s.	106-424 kbit/s.	424 kbit/s.

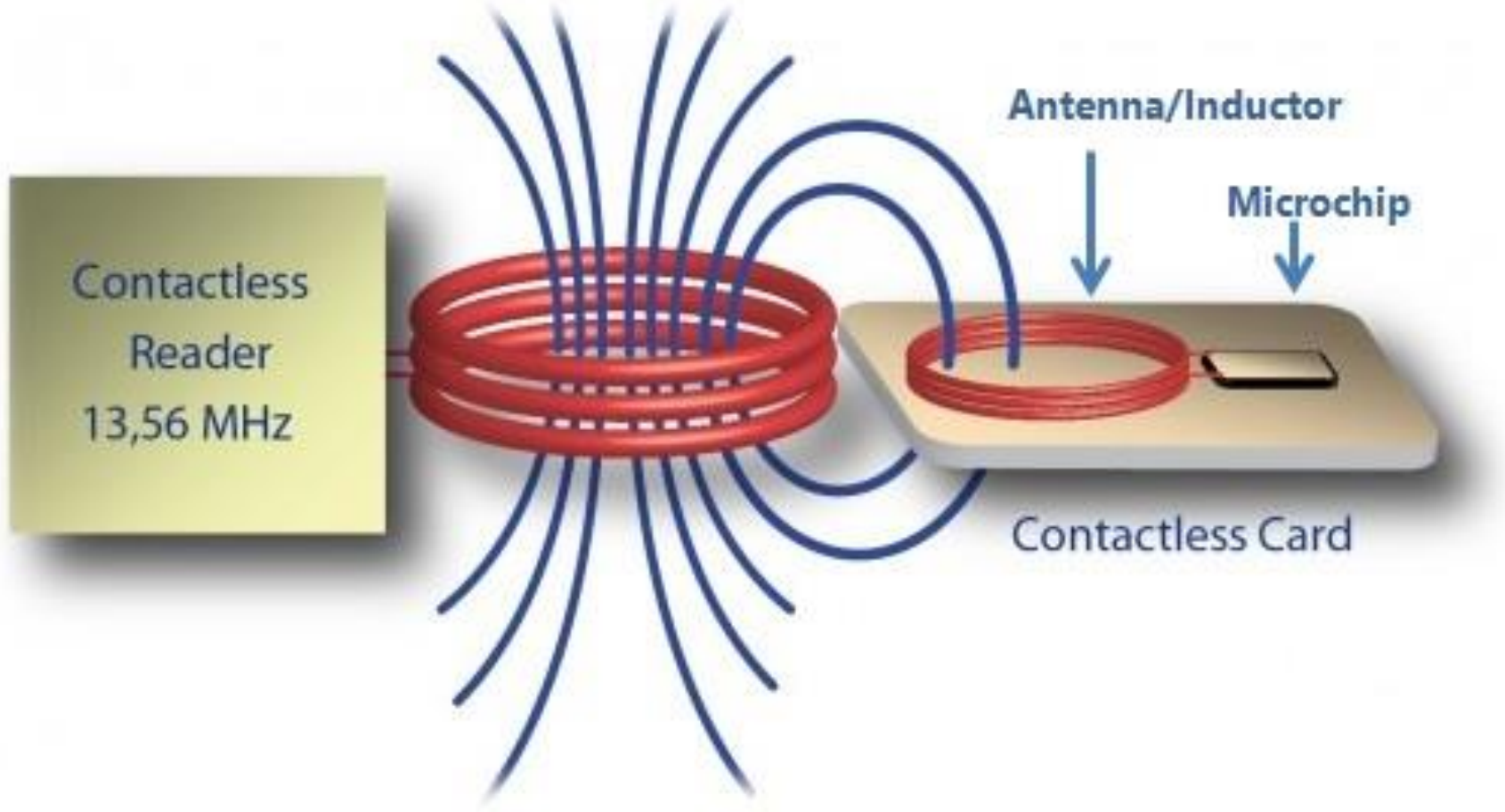
NEAR FIELD COMMUNICATION

HOW NFC DEVICE WORKS?

- NFC communication typically begins when one NFC-enabled device, known as the initiator or active device, generates a radio frequency (RF) field using electromagnetic induction.
- This RF field provides power to the passive device and establishes a communication link.
- When an NFC-enabled passive device, such as a tag, sticker, card, or another device, enters the RF field generated by the initiator then it wakes up.
- Once the passive device is awake, data exchange between the initiator and the passive device can occur.
- This exchange can involve reading data from the passive device, writing data to the passive device, or performing other actions based on the application.

NEAR FIELD COMMUNICATION

HOW NFC DEVICE WORKS?



NEAR FIELD COMMUNICATION

NFC – OPERATION MODES

Card Emulation

The NFC phone emulates a contactless card

- Payment
- Ticketing
- Access control

Reader/writer mode

The NFC phone reads tags

- Read posters
- Interactive advertising
- Launch mobile internet, SMS or make a call

Peer-to-Peer mode

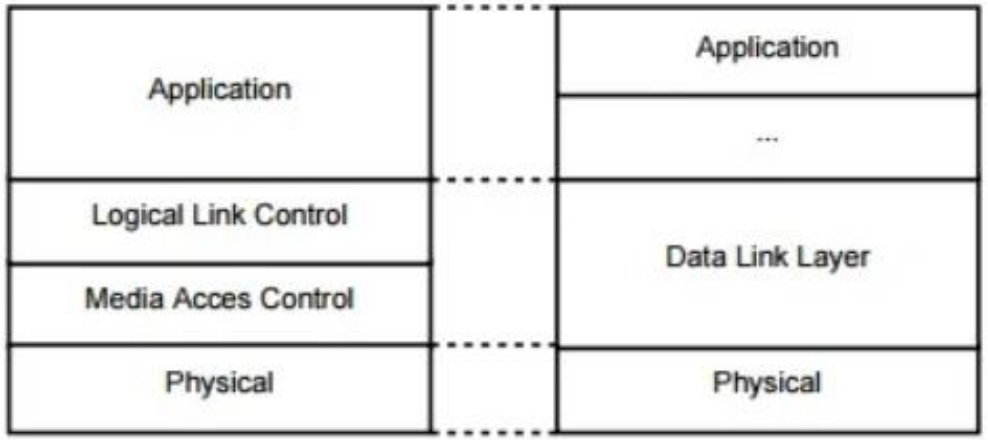
NFC devices can exchange data

- Setup Bluetooth
- Share business cards

NEAR FIELD COMMUNICATION

NFC – PROTOCOL STACK

1. Application layer takes care of format of the data to be exchanged between NFC devices or between NFC device and Tags. NFC uses NDEF (NFC Data Exchange Format).
2. Data link layer takes care of different modes of operation and anti-collision mechanism. LLCP establishes communication between two peer devices.
3. Physical layer takes care of modulation, coding and RF related parameters such frequency, power etc.



NEAR FIELD COMMUNICATION

NFC DATA EXCHANGE FORMAT (NDEF)

- NFC Data Exchange Format (NDEF) is a standardized data format that enables the exchange of messages between NFC devices.
- NDEF has been designed to offer a simple yet effective format so that it can be used by both active and passive NFC devices.
- Being a lightweight format, NDEF does not add much to the overheard of the messages allowing low data rates to be maintained, thereby saving on power
- NDEF is exchanged in messages that comprise of a sequence of records.
- NDEF message is composed of one or more NDEF records depends upon the application in use and the tag type used.

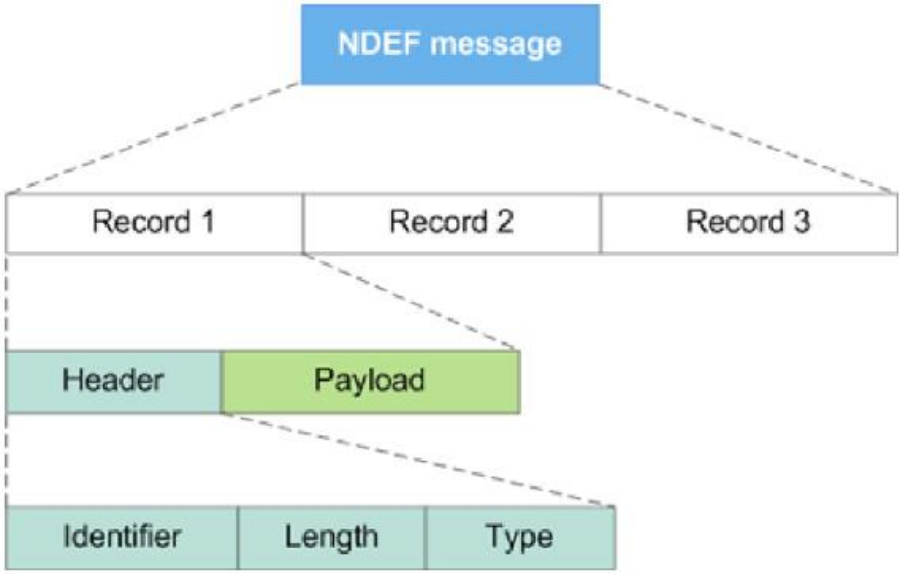
NEAR FIELD COMMUNICATION

NFC Data Exchange Format (NDEF)

- Each record consists of two parts:

1. Header: The header for the NDEF exchange includes indicator for a number of elements.

   - Payload length: This field is always included in the NDEF header; one octet for short records but for normal records it is four octets long.
   - Payload type: This field indicates the kind of data being carried in the payload of that record.
   - Payload identification: This optional field allows the applications to identify the payload.



2. Payload: The payload can be of one of a variety of different types: URL or NFC-specific data type.

NEAR FIELD COMMUNICATION

- ADVANTAGES:
  - Ease of Use
  - Short Setup Time
  - Lower Battery Consumption
  - Does not require complex search and pair procedure like Bluetooth
- DISADVANTAGES:
  - Limited Range
  - Low data rates
  - High cost

Wi-Fi

WIRELESS FIDELITY (Wi-Fi)

INTRODUCTION

- Wi-Fi stands for Wireless Fidelity, it is a wireless networking technology based on the IEEE 802.11 family of standards and is primarily a local area networking (LAN) designed to provide in-building broadband coverage.

- Wi-Fi allows electronics devices such as computers, smartphones, tablets, and other compatible devices to connect to the internet and communicate with one another without using physical cables.

- All Wi-Fi networks are contention-based systems, where the access point and the mobile stations use the same channel hence, all Wi-Fi networks are half duplex.

- Advanced routers offer remote management capabilities, allowing you to monitor and configure your Wi-Fi network from anywhere using a mobile app or web interface.

WIRELESS FIDELITY (Wi-Fi)

Wi-Fi VERSIONS

Version	Year	IEEE Standard	Frequency band	Maximum speed	Distance coverage
Wi-Fi 1	1999	IEEE 802.11a	5 GHz	54 Mbps	35-120m
Wi-Fi 2	1999	IEEE 802.11b	2.4 GHz	11 Mbps	35-140m
Wi-Fi 3	2003	IEEE 802.11g	2.4 GHz	54 Mbps	35-140m
Wi-Fi 4	2009	IEEE 802.11n	2.4 and 5 GHz	600 Mbps	70-250m
Wi-Fi 5	2013	IEEE 802.11ac	5 GHz	1.3 Gbps	35-140m
Wi-Fi 6	2019	IEEE 802.11ax	2.4 and 5 GHz	Up to 10 Gbps	70-250m

WIRELESS FIDELITY (Wi-Fi)

Wi-Fi - EMBEDDED APPLICATIONS

- There are many applications for embedded devices with a Wi-Fi interface:
  - ✓ Smart Home Devices: Wi-Fi enables connectivity for smart home devices such as smart thermostats, smart lighting systems, security cameras, and smart appliances.
  - ✓ Wearable Technology: Many wearable devices, including smartwatches, fitness trackers, and health monitors, incorporate Wi-Fi connectivity.
  - ✓ Wireless Audio Systems: Wi-Fi-enabled speakers, soundbars, and home theatre systems allow users to stream audio wirelessly.
  - ✓ Connected Cars: Wi-Fi is used in connected car systems to provide internet access for in-car entertainment, navigation, real-time traffic updates, and vehicle diagnostics.
  - ✓ Industrial IoT (IIoT): Wi-Fi is employed in industries for communication between machines and systems, monitoring and controlling equipment, collecting sensor data, asset tracking, predictive maintenance etc.,
  - ✓ Healthcare: Wi-Fi facilitates communication and data exchange in healthcare, supporting applications such as telemedicine consultations, medical imaging, and patient monitoring systems.

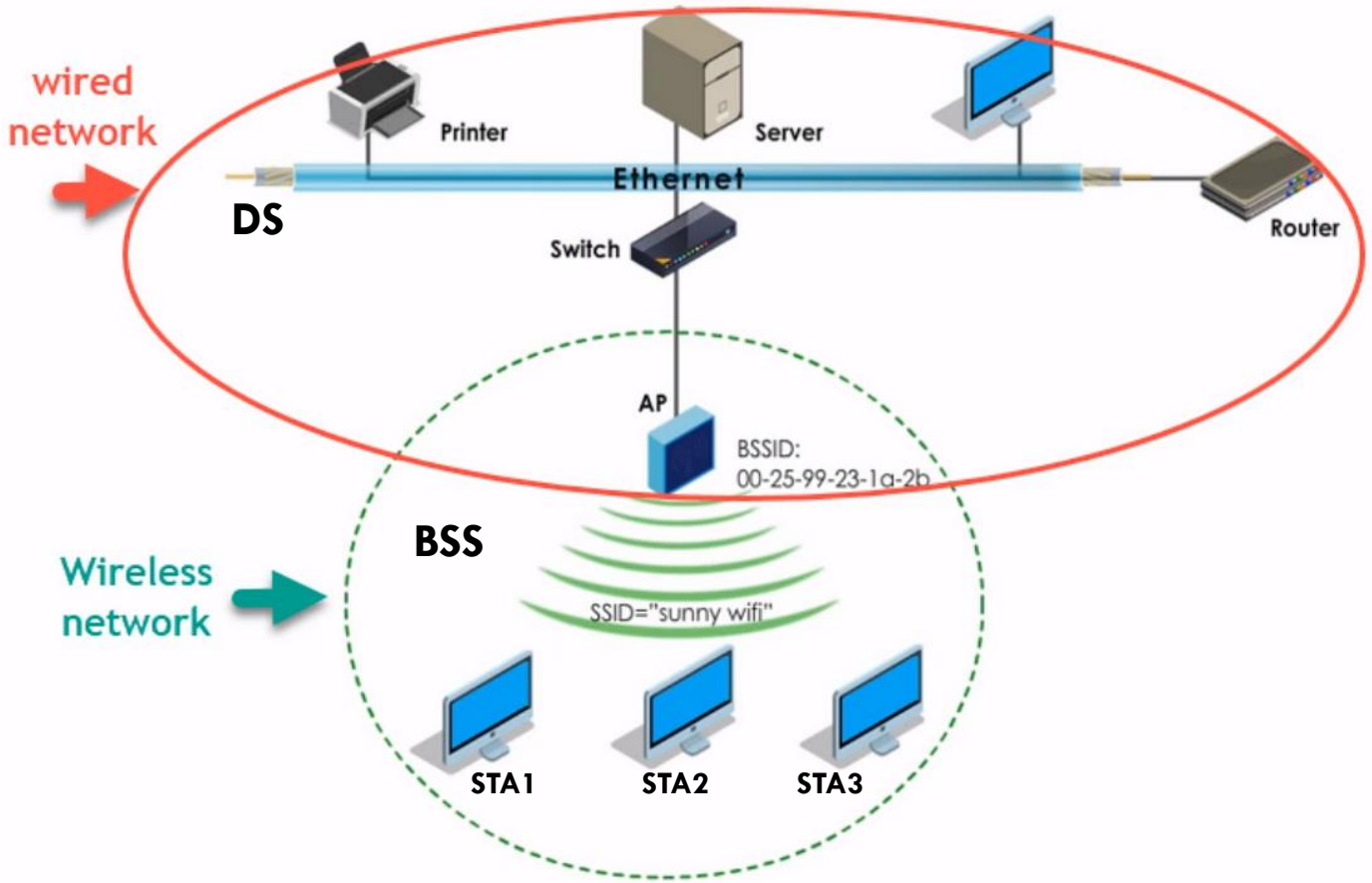
WIRELESS FIDELITY (Wi-Fi)

Wi-Fi ARCHITECTURE

- All wireless devices that join a Wi-Fi network, are called wireless stations (STAs), it might be a PC, a laptop, a phone, Wi-Fi module interfaced with μp/μc.
- When one or more wireless stations (STA) and the Access Point (AP) are wirelessly connected, they form a Basic Service Set (BSS).
- BSS is the basic building block of a Wi-Fi network. All wireless devices trying to join the BSS must associate with the AP.
- A major role for an AP is to extend access to wired networks via Ethernet and typically links onto the Internet for the clients of a wireless network.
- In a typical home network, there is one Wi-Fi router (access point + router) that connects BSS to the internet.

WIRELESS FIDELITY (Wi-Fi)

Wi-Fi ARCHITECTURE



WIRELESS FIDELITY (Wi-Fi)

Wi-Fi ARCHITECTURE

- An AP provides access to its associated STAs is called the distribution system (DS).
- The DS is an architectural component that allows communication among APs,.
- In IEEE 802.11, a set of BSS's can be interconnected by a Distribution system (DS) to form an Extended Service Set (ESS).
- The AP can be identified by Service set identifier (SSID).
- An ESS is a Wi-Fi network of arbitrary size and complexity.
- The network name, or SSID, must be the same for all APs participating in the same ESS.

WIRELESS FIDELITY (Wi-Fi)

Wi-Fi OPERRATING MODES

- Depending upon the mode of operation, BSS can be categorized into the following types:

1. Infrastructure BSS:

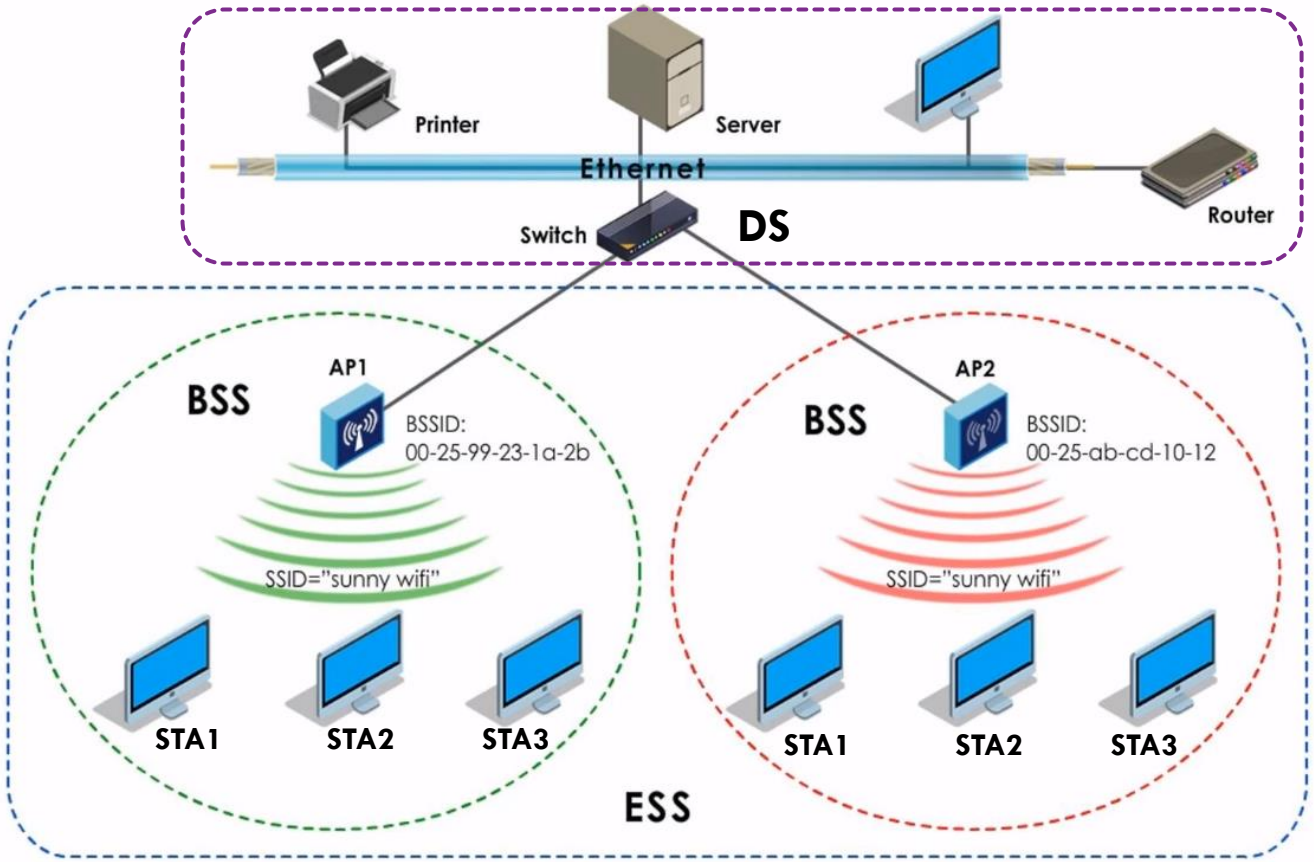
- Communication between Wi-Fi devices takes place through AP.
- The AP and its associated wireless clients define the coverage area and form the BSS.
- This is the most common mode used in homes, offices, and public Wi-Fi networks.

2. Independent BSS (IBSS):

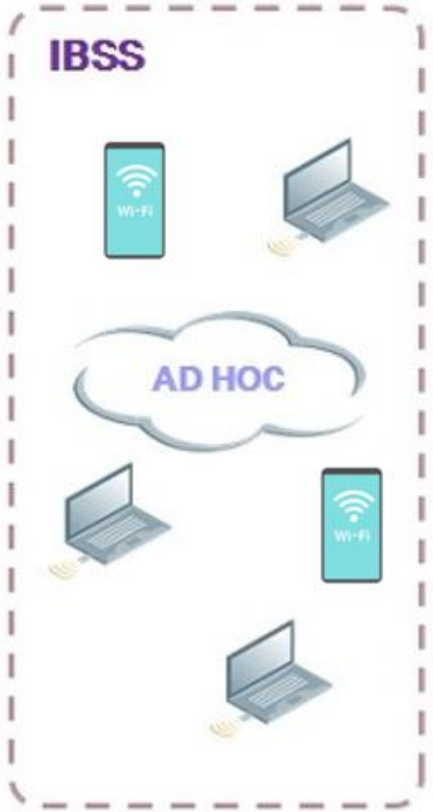
- Also known as peer-to-peer mode or ad-hoc mode.
- Enables Wi-Fi devices to communicate directly with each other without AP.
- Spontaneously create a network and does not support access to wired networks.
- This mode is useful for creating temporary networks to exchange data among them.

WIRELESS FIDELITY (Wi-Fi)

Wi-Fi OPERRATING MODES



Infrastructure BSS



Independent BSS

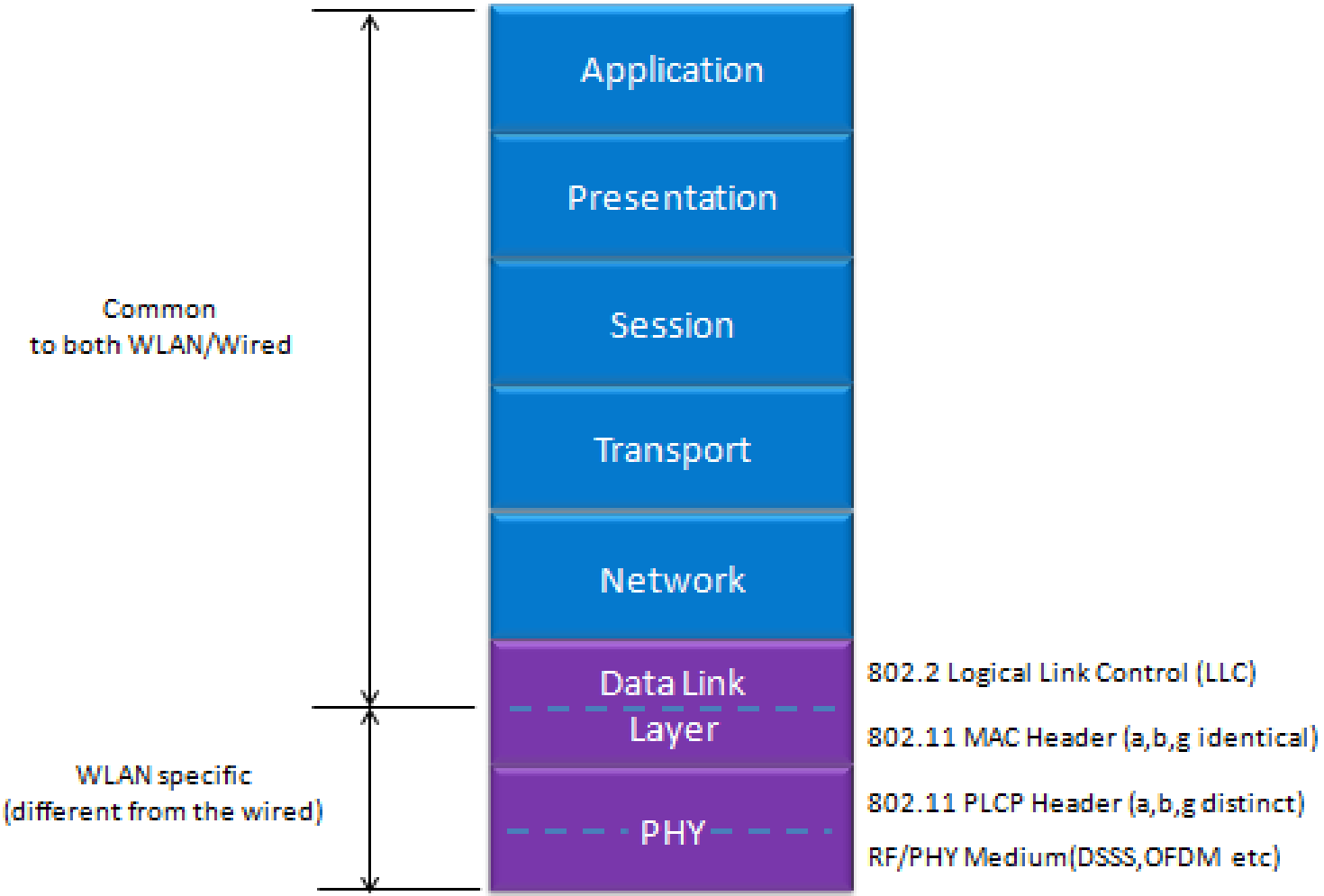
WIRELESS FIDELITY (Wi-Fi)

HOW Wi-Fi WORKS?

- When powered on, a Wi-Fi station will scan the available channels to discover active networks where beacons are being transmitted.
- It then selects a network, be it in ad hoc mode or infrastructure. In the latter case, it authenticates itself with the access point (AP) and then associates with it.
- While being part of a network, stations can keep discovering new networks and may disassociate from the current AP in order to associate with a new AP due to strong signal.
- Stations can roam this way between networks that share a common distribution system, in which case seamless transition is possible.
- A station can sleep to save power, and when it finishes infrastructured mode operation it can deauthenticate and disassociate from the AP.

WIRELESS FIDELITY (Wi-Fi)

Wi-Fi PROTOCOL STACK



WIRELESS FIDELITY (Wi-Fi)

Wi-Fi PROTOCOL STACK

- The Wi-Fi protocol stack is based on the OSI (Open Systems Interconnection) model, which consists of multiple layers, each responsible for specific tasks.

- Physical Layer (PHY):
  - Deals with transmission and reception of raw data bits over the physical medium, such as radio waves.
  - It defines characteristics such as modulation, coding, frequency bands, data rate and transmission power for various standards such as 802.11a/b/g/n/ac/ax.

- Data Link (MAC & LLC):
  - MAC layer is responsible for managing access to the wireless medium and coordinating the transmission of data frames between devices. It handles functions such as addressing, frame synchronization, error detection, and collision avoidance.
  - The LLC sublayer is responsible for establishing logical connections between devices and providing error control and flow control mechanisms. It ensures reliable communication between devices and provides a common interface for higher-layer protocols.

WIRELESS FIDELITY (Wi-Fi)

Wi-Fi PROTOCOL STACK

Internet Protocol (IP) Layer:

- Responsible for logical addressing, routing, and packet forwarding between different networks. It uses IP addressing to identify devices and routes packets between them, enabling communication across interconnected networks.

Transport Layer:

- This layer provides end-to-end communication services for applications and ensures data reliability and integrity. Protocols like TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) operate at this layer to establish connections, manage data transmission, and handle error recovery.

Session, Presentation, and Application Layers:

- These layers are responsible for managing the communication sessions, data formatting, encryption, and interaction with applications.

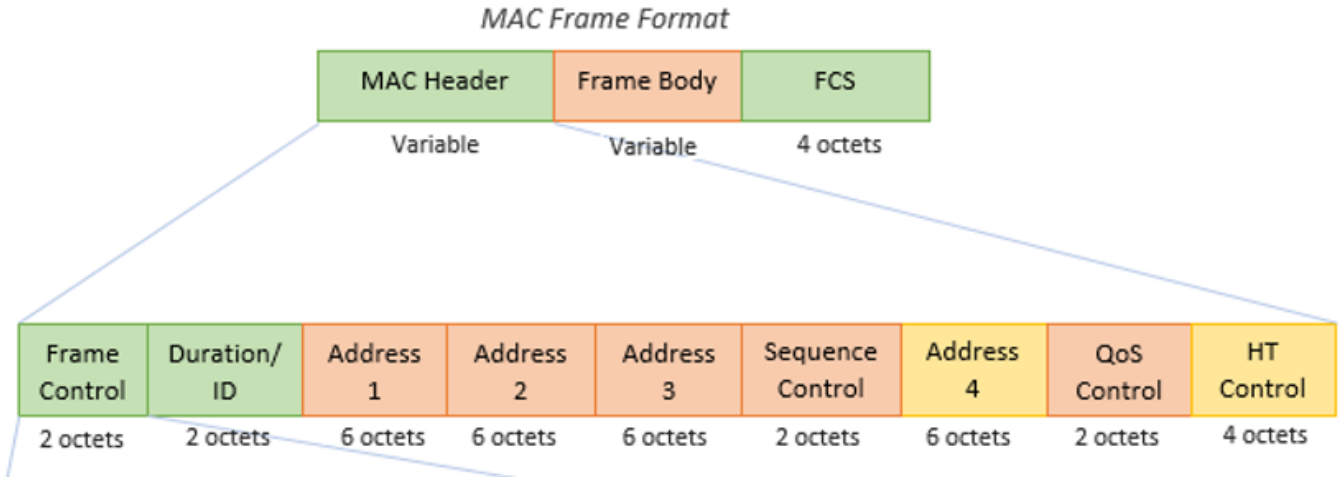
- In Wi-Fi networks, these layers typically leverage existing protocols and standards used in higher-layer applications, such as HTTP (Hypertext Transfer Protocol) for web browsing, FTP (File Transfer Protocol) for file transfer, and SSH (Secure Shell) for secure remote access.

WIRELESS FIDELITY (Wi-Fi)

Wi-Fi FRAME FORMAT

- The Wi-Fi frame format specifies how data is organized and transmitted over a Wi-Fi network.
- IEEE 802.11 supports 3 types of frames :

1. Management frames: Used for the initial communication between station and AP's. It is used for station association and disassociation with the AP's, timing and synchronization and authentication procedure.
2. Control frames: Used for accessing the channel and acknowledging the frames.
3. Data frames: Used for data transmission.



WIRELESS FIDELITY (Wi-Fi)

Wi-Fi FRAME FORMAT

- MAC Header:
  - FC : Defines the type of frame and some control information.
  - D : Defines the duration of e transmission.
  - Addresses : There are four address fields, each of 6 bytes long.
    - Address 1 is always the address of the next device.
    - Address 2 is always the address of the previous device.
    - Address 3 is the address of the final destination station (host) if it is not defined by address 1.
    - Address 4 is the address of the original source station which is not the same as address 2.
  - Sequence control : This field defines the sequence number of a frame to be used in flow control.
  - QoS Control: It is used to specify QoS parameters for the frame transmission, allowing for prioritization and management of traffic within a Wi-Fi network.
  - High Throughput (HT) Control: It is used to provide control information related to the transmission and reception of HT frames. to Wi-Fi standards that support HT mode, such as 802.11n.
- Frame body : This field can be 0 and 2312 bytes, contains information based on the type.
- FCS : The FCS field is 4 bytes long and contains a CRC-32 error detection sequence.

WIRELESS FIDELITY (Wi-Fi)

- ADVANTAGES:
  - Wireless Connectivity
  - Ease of Installation
  - Scalability
  - Cost-Effective
- DISADVANTAGES:
  - Limited Range
  - Interference and Congestion
  - Security Concerns
  - Wi-Fi access points require a constant power supply to operate

BLE vs ZIGBEE vs NFC vs Wi-Fi

Key parameters	Bluetooth	Zigbee	NFC	Wi-Fi
Specifications authority	Bluetooth Special Interest Group	Zigbee Alliance	NFC Forum	IEEE Association
Standard	802.15.1	802.15.4	802.2	802.11 (a, b, g, n)
Frequency band	2.4 GHz	2.4 GHz, 850 – 930 MHz	13.56 MHz	2.4 GHz and 5 GHz
Data rate	Upto 50 Mbps	20-250 Kbps	106 , 212, 424 Kbps	Upto 54 Mbps
Transmission range	Up to 100m	Up to 100m	Up to 10cm	Up to 100m
Power consumption	Very low	Low	Low	High
Network topology	point to point, Piconet	Mesh, star, tree	point to point	ESS, BSS, ad hoc
Network Size	8	64,000	2	255
Security	62 bit, 128 bit	128 bit AES	AES	Authentication service set ID (SSID)
Application	Wireless audio streaming and data transfer, smart wearables and fitness trackers	Home automation and control, industrial monitoring sensor network	Mobile payments, asset tracking & management, access control	WLAN, broadband Internet access, IoT

THANK YOU