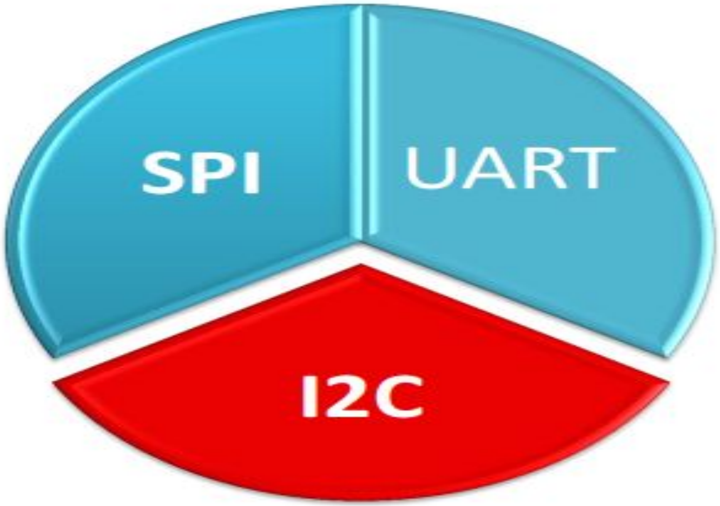
Embedded System Interfacing

Lecture 15
IIC

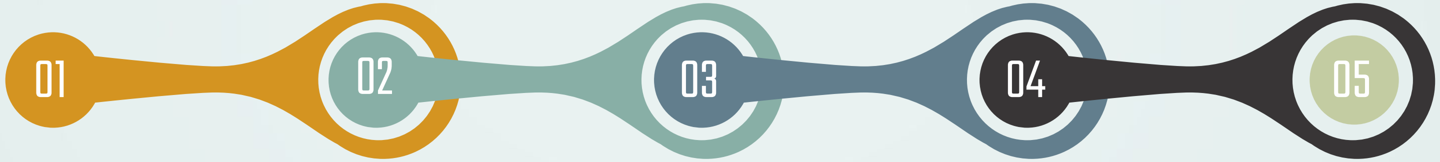
01

IIC

IIC



IIC Features



Half Duplex

Synchronous

Wired

Multi Master
Multi Slave

Serial

What is IIC ?

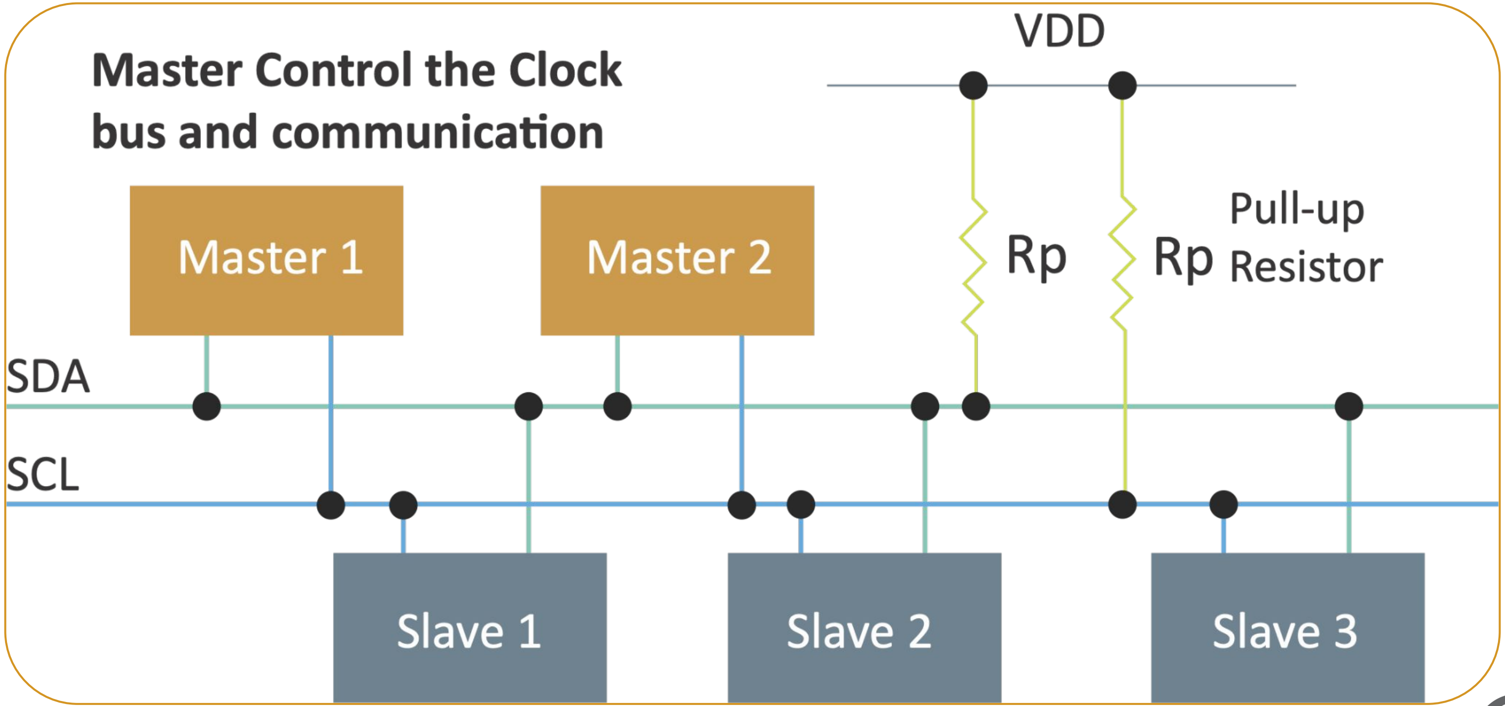
Inter-Integrated Circuit I^{2}C (Pronounced I Two C or I Squared C), is a serial communication protocol at which the devices are hooked up to the I2C bus with just ***two wires***.

It is sometimes referred to as ***Two Wire Interface***, or the ***TWI***

Devices could be the CPU, IO Peripherals like ADC, or any other device which supports the I2C protocol.

All the devices connected to the bus are classified as either being ***Master*** or ***Slave***.

IIC Bus Interface



IIC Physical Layer

As we know I2C is a pure master and slave communication protocol and can be a multi-master or multi-slave. But we generally see a single master with multi slave in I2C communication.

I2C Bus consists of only two wires and is named serial data line (**SDA**) and a serial clock line (**SCL**). At the physical layer, both SCL and SDA lines are of the open-drain design, thus pull-up resistors are needed. High-speed systems (and some others) may use a current source instead of a resistor to pull up only SCL or both SCL and SDA, to accommodate higher bus capacitance and enable faster rise times.

The SDA wire is used to transfer the data and the SCL wire is used to synchronize the master and slave with the clock signal. See the below image in which all the slave devices and master are connected to the same SCL and SDA lines in the I2C network.

IIC Physical Layer

Because all slave and master are connected with the same data and clock bus, here important point needs to remember that these buses are connected using the **WIRE-AND configuration** which is done by putting both wires in an open drain design. The wire-AND configuration allows in I2C to connect multiple nodes to the bus without any short circuits from signal contention.

The open-drain allows the master and slave to drive the line low and release to a high impedance state. So In that situation, when the master and slave release the bus, need a pull resistor to pull the line high. The value of the pull-up resistor is very important because the incorrect value of the pull-up resistor can be lead to signal loss.

A low value of a pull-up resistor is called a strong pull-up resistor (more current flows) and a high resistor value is called a weak pull-up resistor (less current flows).

IIC Physical Layer

For Summary

- ❑ **SDA** and **SCL** are *open-drain* (also called *open-collector*) lines.
- ❑ Normal IO Line (Called Push Pull Line) can be derived to GND by writing digital 0 (Pull) or can be derived to VCC by writing 1 (Push)
- ❑ Open Drain Line can be derived only to GND by writing digital 0. Writing digital 1 on open drain line makes it floating.
- ❑ Open Drain lines are used to implement open drain bus. Allowing many devices to be connected on one line, see the following example.

IIC Physical Layer

Push Pull Line

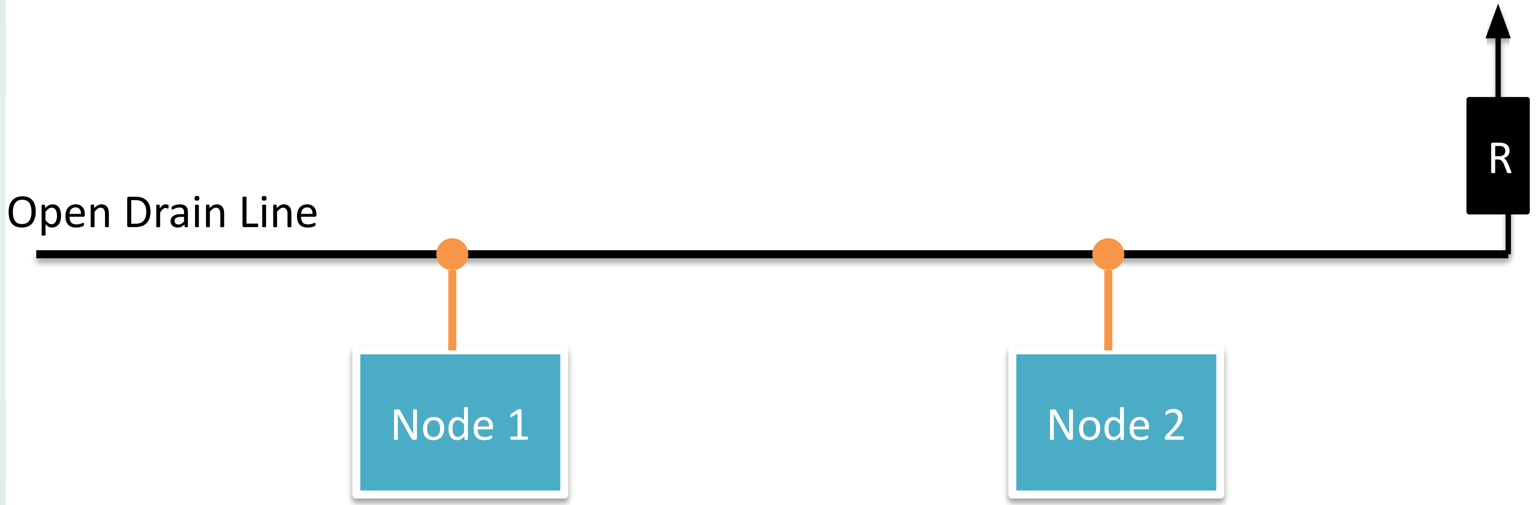
Node 1
Node 2

Looking to that bus and considering that Node 1, 2 and 3 can write on it. If the bus is Push Pull bus, then it can be derived to GND or to VCC, what would happen if Node 1 derived the bus to GND and Node 2 derived the bus to VCC ?!

The same line would have a connection to GND and a connection to VCC at the same time which makes a short circuit on the power source ! The network would be damaged

Node 1	Node 2	Result on Line
0	0	GND
1	1	VCC
0	1	Short Circuit
1	0	

IIC Physical Layer



IIC Physical Layer

If the bus is open drain line, then any node that writes 0 the bus would be derived to GND, if all nodes writes 1 then the bus would be floating, that's why we use a pull up resistor to connect the line to VCC which will be effective only if all nodes writes 1.

It looks like AND gate, any nodes writes 0, the bus derived to GND, if all nodes write 1, the bus derived to Vcc by the pull up resistor.

Node 1	Node 2	Result on Line
0	0	GND
1	1	VCC
0	1	GND
1	0	

IIC Physical Layer

Dominant Bit

A bit that appears on the bus if any node writes it. In our system the 0 is the dominant

Recessive Bit

A bit that appears on the bus if all nodes write it. In our system the 1 is the recessive

Master and Slave

Master

- Controls the SCL and generate the clock.
- Starts and stops data transfer.
- Controls addressing of other devices.
- Determines data transfer direction.

Slave

- Device address by master.
- Timing is controlled by the clock line.

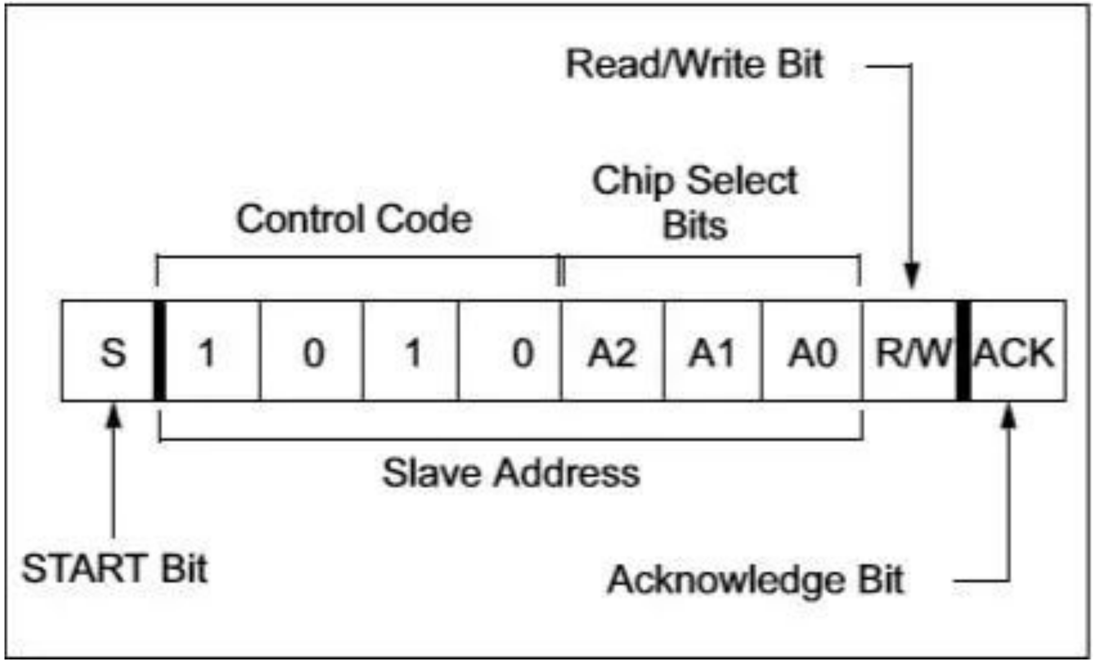
Working Principles

I2C is a simple chip-to-chip communication protocol. In I2C, communication is always started by the master. When the master wants to communicate with the slave then it asserts a **start bit** followed by the **slave address** with read/write bit.

After the asserting of the start bit, all slave comes in the attentive mode. If the transmitted address match with any of the slaves on the I2C Bus then an ACKNOWLEDGEMENT (**ACK**) bit is sent by the slave to the master.

After getting the ACK bit, the master starts the communication. If there is no slave whose address matches the transmitted address then the master received a NOT-ACKNOWLEDGEMENT (**NACK**) bit, in that situation either master asserts the stop bit to stop the communication or asserts a repeated start bit on the line for new communication.

Working Principles

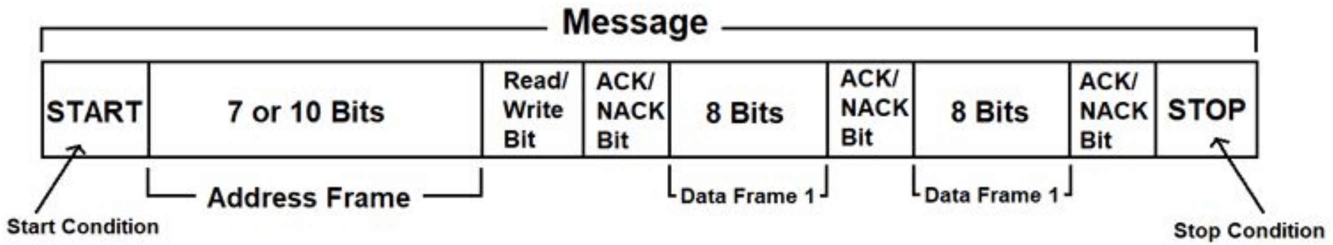


Data Frame Overview

Overview

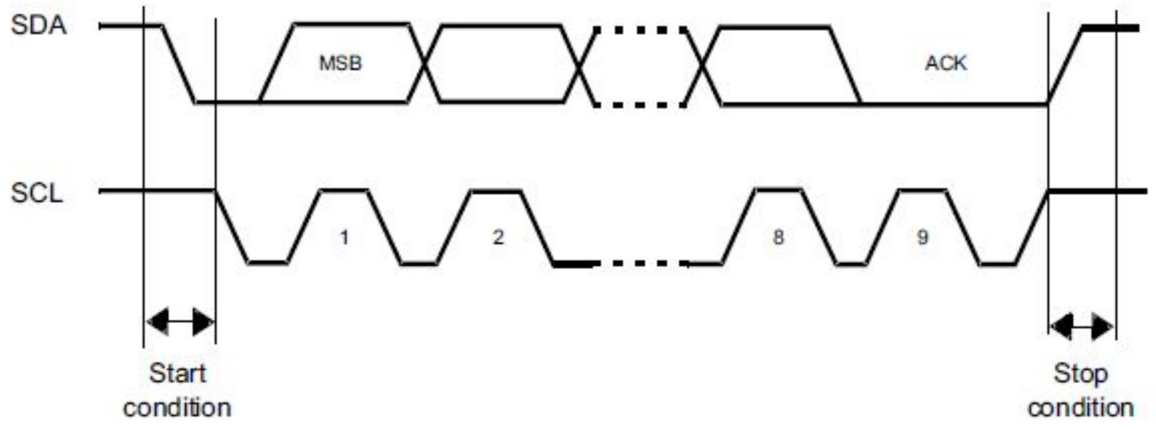
I2C is an eight-bit communication protocol and an ACK or NACK bit associated with each byte. In I2C data is transferred in messages. An I2C transaction may consist of multiple messages. Each message begins with a **start** bit, and the transaction ends with a **stop** bit. Master may send another **start** condition to retain control of the bus for another message (a “combined format” transaction).

Messages are broken up into frames of data. Each message has an address frame (slave address), and one or more data frames that contain the data being transmitted. The message also consists of read/write bits, ACK/NACK bits between each data frame. See the below Image,



Start and Stop Conditions

- **Start condition :** A high to low transition on **SDA** when **SCL** is high
- **Stop condition :** A low to high transition on **SDA** when **SCL** is high

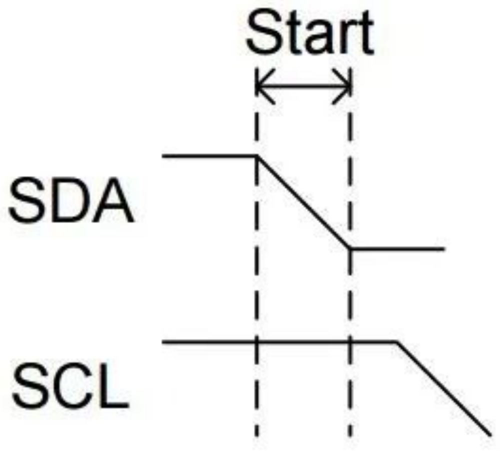


Start Condition

- The default state of the SDA and SCL line is high (Due to the pull-up resistors). A master asserts the start condition on the line to start the communication. **“**A high to low transition of the SDA line while the SCL line is high called the Start Condition**“.**

  In simple words, you can understand that whenever a master decides to start a communication, it switches the SDA line from high voltage level to a low voltage level before the SCL line switches from high to low. You can see the below image.

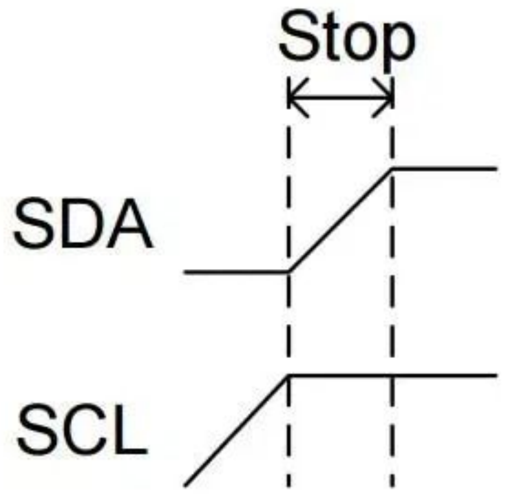
  You should remember that the Start Condition is always asserted by the master and the I2C bus is considered busy after the assertion of the START bit.



Stop Condition

The STOP condition is asserted by the master to stop the communication. “A Low to high transition of the SDA line while the SCL line is high called the STOP condition“. Whenever a master decides to stop communication, it switches the SDA line from low voltage level to high voltage level before the SCL line switches from high to low. See the below image.

The STOP condition is always asserted by the master. The I2C bus is considered free after the assertion of the STOP bit.



Repeated Start Condition

Repeated Start Condition is similar to the Start Condition but both have different meanings. The Repeated Start is asserted by the master before the stop condition (When the bus is not in an idle state).

The I2C Bus considered busy between each start and stop condition. If the master tries to initiate a new transfer and does not want to lose control over the I2C Bus before starting the new transfer, then it issues a new Start Condition. This asserted start condition is called a Repeated Start Condition.

The repeated start is beneficial for the master when it wants to start a new communication without asserting the stop condition.

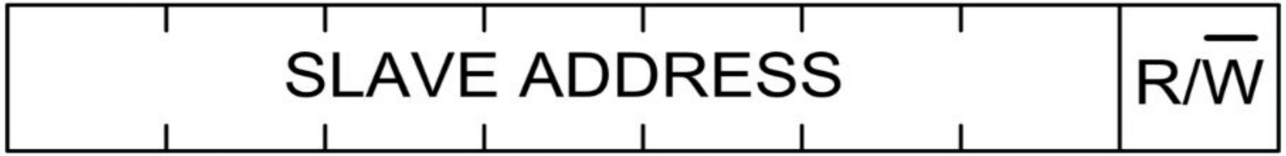
Note: Repeated start is beneficial when more than one master connected with the I2c Bus.

IIC Addressing

- Each node has a unique 7 (or 10) bit address
- Peripherals often have fixed and programmable address portions

✔ The first byte (immediately after the START condition) contains the I2C slave address.

✔ The last bit of the I2C address is a data direction bit).



Addresses starting with 0000 or 1111 have special functions:-

0000000 Is a General Call Address
0000001 Is a Null (CBUS) Address
1111XXX Address Extension
1111111 Address Extension – Next Bytes are the Actual Address

Read Write bit

If you will see the above-mentioned message, you will find that the address frame includes a single R/W bit at the end. This bit specifies the direction of data transfer. If the master wants to transfer the data to the slave device, the R/W bit will be ‘0’. Also, if the master wants to receive data from the slave device, the R/W bit will be ‘1’. We will see it in the below read/write operations.

Acknowledgment

If you will see the above-mentioned message, you will find that each frame in a message is followed by an ACK/NACK bit. Basically, it is a protocol overhead, ACK/NACK stands for Acknowledged/Not-Acknowledged bit. The sender will get an ACK bit if an address frame or data frame was successfully received by the receiver in I2C Bus.

Let’s see some scenarios, where NACK bit is generated

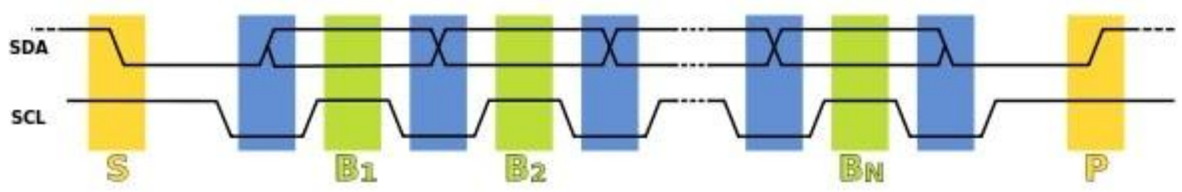
- When the receiver is unable to receive or transmit the data, in that situation it generates a NACK bit to stop the communication.
- During the communication, if the receiver gets any data or commands which are not understood by the receiver then it generates a NACK bit.
- During the transfer, if the receiver performs any real-time operation and not able to communicate with the master then assert a NACK bit.
- When Master is a receiver and reads the data from the slave, then after the reading of whole data it asserts a NACK bit on data lines to stop the communication.
- If there is no device present in the I2c bus of the same address which is transmitted by the master, then the master will not get acknowledged by any slave and treat this situation as NACK.

Acknowledgment Summary

- **From Slave to Master Transmitter:**
  - After address received correctly
  - After data byte received correctly
- **From Slave to Master Receiver:**
  - Never (Master Receiver generates ACK)
- **From Master Transmitter to Slave:**
  - Never (Slave generates ACK)
- **From Master Receiver to Slave:**
  - After data byte received correctly

Byte Format in I2C Protocol (Data Frame)

In I2C, every data which is transmitted over the SDA line must be eight-bit longs. It is very important to remember that in I2C the data bit is always transmitted from the MSB and we can send or receive any number of bytes in I2C between the start and stop condition.



The sender always gets NACK/ACK bit just after the data frame to verify the frame has been received successfully. You can also say that each data frame is immediately followed by an ACK/NACK bit.

Byte Format in I2C Protocol (Data Frame)

In I2C, one bit is always transmitted on every clock. A byte that is transmitted in I2C could be an address of the device, the address of the register, or data that is written to or read from the slave device.

In I2C, the SDA line is always stable during the high clock phase except for the start condition, stop condition, and repeated start condition. The SDA line only changes its state during the low clock phase.

Note: SDA can only change their state only SCL is low except for the Start Condition, Repeated Start Condition, and Stop Condition.

Handshaking Process in I2C Protocol

In I2C for each byte, an acknowledgment needs to be sent by the receiver, this acknowledgment bit is proof that data is properly received by the receiver and it wants to continue the communication.

A master starts the communication to assert a start condition on the bus. After the start condition master is transmitted a 7-bit address with associated read or write bits (here I am discussing a 7-bit address).

Handshaking Process in I2C Protocol

After the transmission of the address byte, the master releases the data lines to put the data line (SDA) in a high impedance state, which allows the receiver to give the acknowledgment bit.

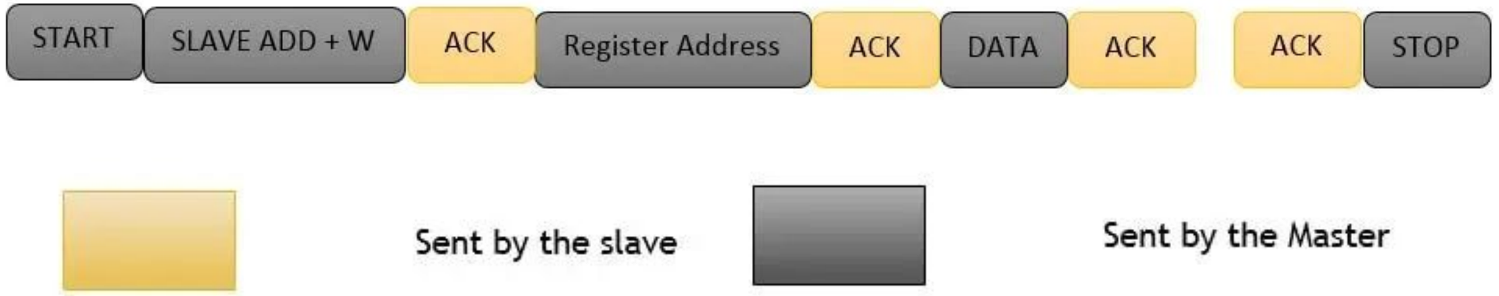
If this transmitted address is matched with any receiver then it pulls down the SDA lines low for the acknowledgment and after the acknowledgment, it releases the data lines. The master generates a clock pulse to read this acknowledgment bit and continue the read or write operation.

If this transmitted address is not matched with any receiver then nobody is pull down the data lines low, master understands it is a NACK and in that situation, the master asserts a stop bit or repeated start bit for further communication.

I2C write operation

In I2C before performing, the write operation master has to assert a start condition on the I2c bus with the slave's address and write control bit (for write operation R/W bit will be 0).

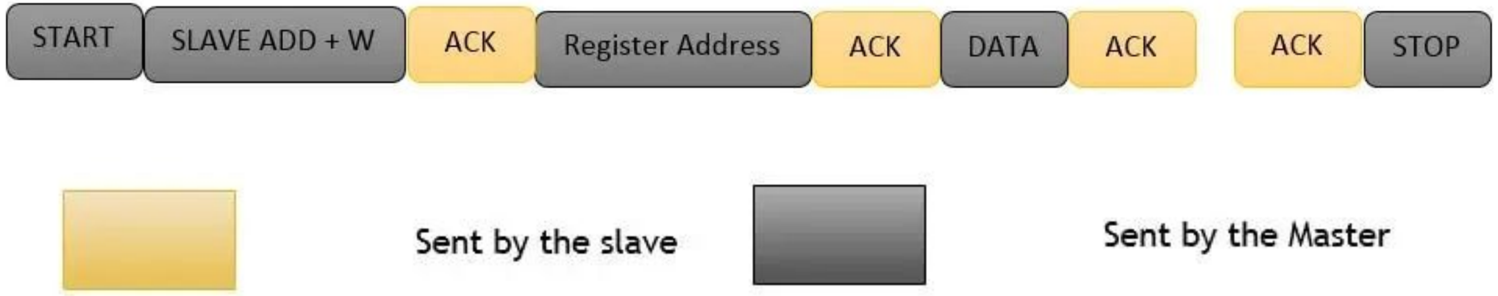
If the transmitted address match with any slave device (EEPROM) which connected to the I2C bus then master receives an acknowledge bit. After getting the ACK bit master send the address of the register, where it wishes to write, the slave will acknowledge again, letting the



I2C write operation

After getting this acknowledgment, the master will start sending the data to the slave. Master will get the acknowledgment of each transmitted byte.

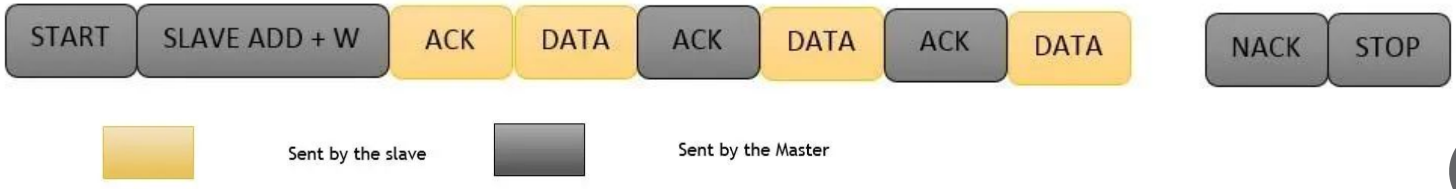
If the master does not get the acknowledgment from the slave then the master asserts a stop condition to stop the communication or either assert the repeated start to establish a new communication. There or another option to stop the communication when the master has sent all the data then the master is terminated the transmission with a STOP condition.



I2C Read operation

After getting this acknowledgment, the master will start sending the data to the slave. Master will get the acknowledgment of each I2C read operation same as the I2C write operation, In which the master asserts the start condition before the read operation. After the start condition master transmit the slave address with read control bit (for read operation R/W bit will be 1), if the transmitted address match with any device in the I2C Bus then it acknowledges to the master to pulling down the data bus(SDA). transmitted byte.

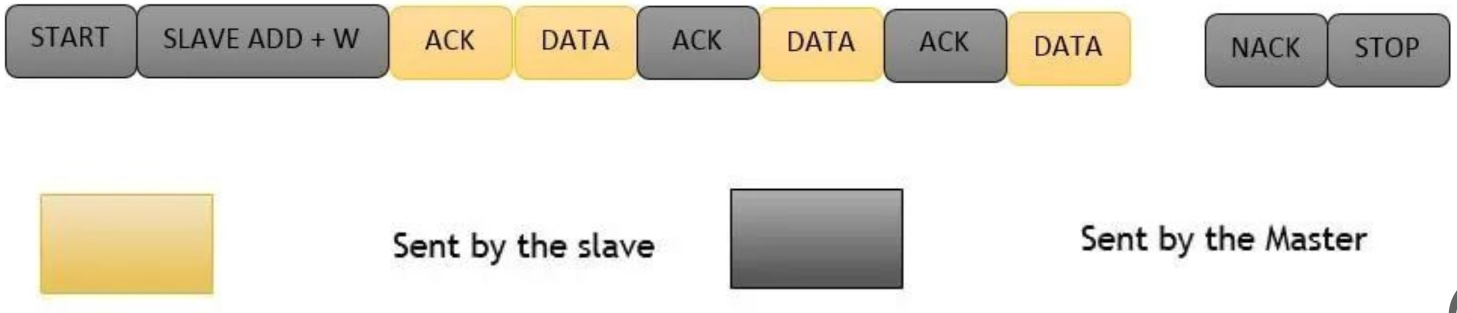
If the master does not get the acknowledgment from the slave then the master asserts a stop condition to stop the communication or either assert the repeated start to establish a new communication. There or another option to stop the communication when the master has sent all the data then the master is terminated the transmission with a STOP condition.



I2C Read operation

After getting ACK bit, the master releases the data bus but continues sending the clock pulse, in that situation master becomes the receiver and the slave becomes the slave transmitter.

In the read operation, the master gives the acknowledgment to the slave on receiving every byte to let the slave know that it is ready for more data. Once the master has received the number of bytes which it is expecting, it will send a NACK bit to release the bus and assert the stop bit to halt the communication.

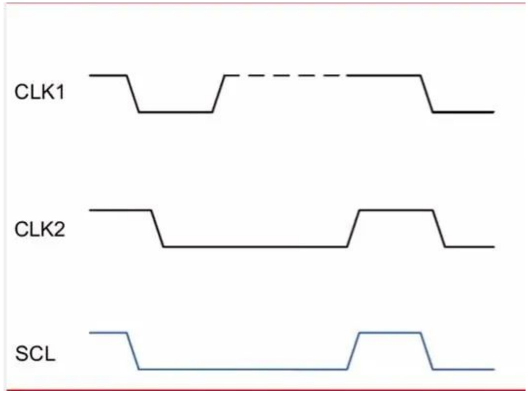


I2C Clock synchronization

I2C is synchronous communication, in which a clock is always generated by the master and this clock is shared by both master and slave. In the case of multi-master, all master generate their own SCL clock, hence the clock of all masters must be synchronized. In the I2C, this clock synchronization is done by wired and logic.

Let’s see an example for a better understanding, where two masters try to communicate with a slave. In that situation, both masters generate their own clock signal, master M1 generates clk1 and master M2 generates clk2, and the clock observed on the bus is SCL.

The SCL clock would be the Anding (clk1 & clk2) of clk1 and clk2 and most interesting thing is that the highest logic 1 of the SCL line defines by the master clock which has the lowest logic 1.

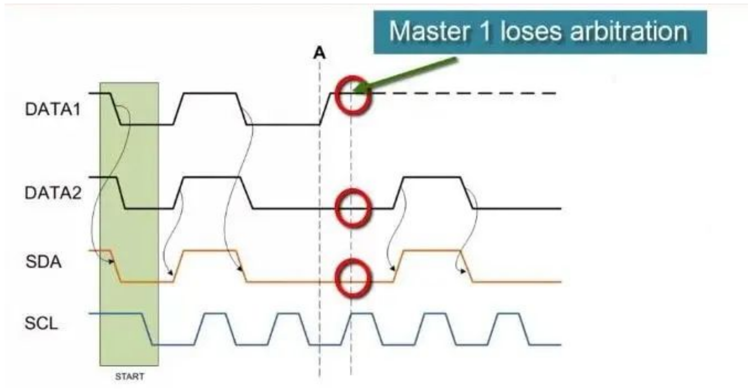


Arbitration in I2C Bus

Arbitration is required in the case of a multi-master, where more than one master is tried to communicate with a slave simultaneously. In I2C arbitration is achieved by the SDA line.

For Example, Suppose two masters in the I2C bus try to communicate with a slave simultaneously and assert a start condition on the bus. The SCL clock of the I2C bus would be already synchronized by the wired and logic.

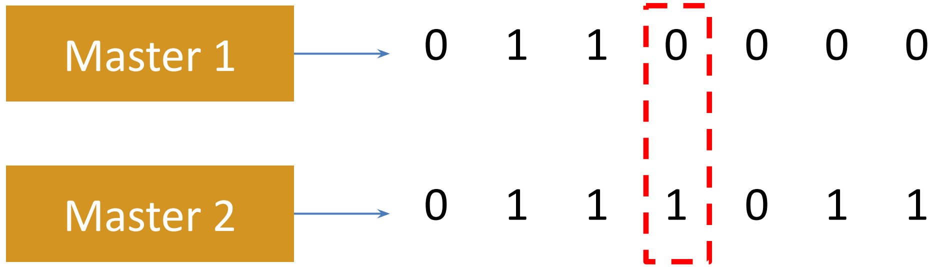
In this case, everything will be good till the state of the SDA line will the same as what is the masters driving on the bus. If any master sees that the state of the SDA line differs, what is it driving, then they will exit from the communication and lose their arbitration.



Note: Master who is losing their arbitration will wait till the bus become free.

Arbitration in I2C Bus

The arbitration is done first on the slave address, think of the following scenario:



Master 1 is sending data to slave address (0110 000) and Master 2 is sending data to slave address (0111 011), remember that the MSB is sent first.

At the first clock, Master 1 sends 0 and Master 2 Sends 0, then when they listen on the high level of the same clock they will found a 0 bit, the same for the second and third clock. But at the forth clock (Bit Number 3) Master 1 sends 0 but Master 2 Sends 1, because 0 is dominant it would appear on the bus. At the high level of this clock cycle Master 1 find 0 as it expects, while Master 2 finds 0 although it sent 1 !
At this stage Master 1 wins the arbitration and Master 2 become a slave.

Arbitration in I2C Bus

Question, which master would win the arbitration assuming the following scenario:

Master 1 Sending data to Salve 0111   111
Master 2 sending data to Salve 1000   000

Answer, Master 1 wins the arbitration, because the MSB is sent first, then Master 1 wins the arbitration at first clock !

Summary:
It doesn’t matter how many 0’s in the slave address, it matters only when the first 0 comes !

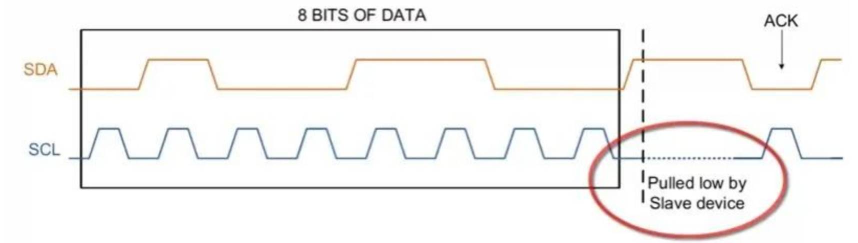
The rule says:
The Highest priority slave is the slave has the lowest address value

Clock stretching in I2C

Communication in the I2C bus can be paused by the clock stretching to holding the SCL line low and it cannot continue until the SCL line released high again.In I2C, the slave able to receive the data at a fast rate but sometimes the slave takes more time in the processing of received data. In that situation, slave pulls the SCL line low to pause the communication and after the processing of the received bytes, it again released the SCL line high to resume the communication.

The clock stretching is how slave drive the SCL line but it is the fact, most of the slave does not drive the SCL line

Note: In the I2c communication protocol, most of the I2C slave devices do not use the clock stretching feature, but every master should support the clock stretching.

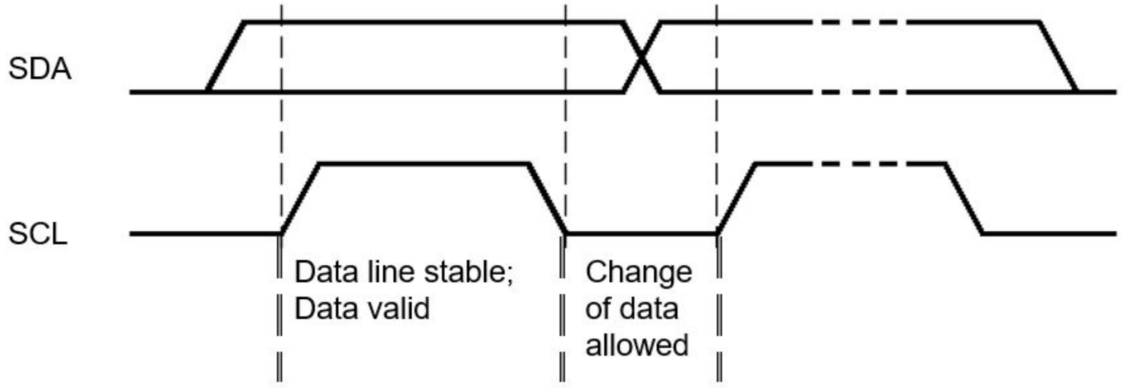


Notes for Data Transfer

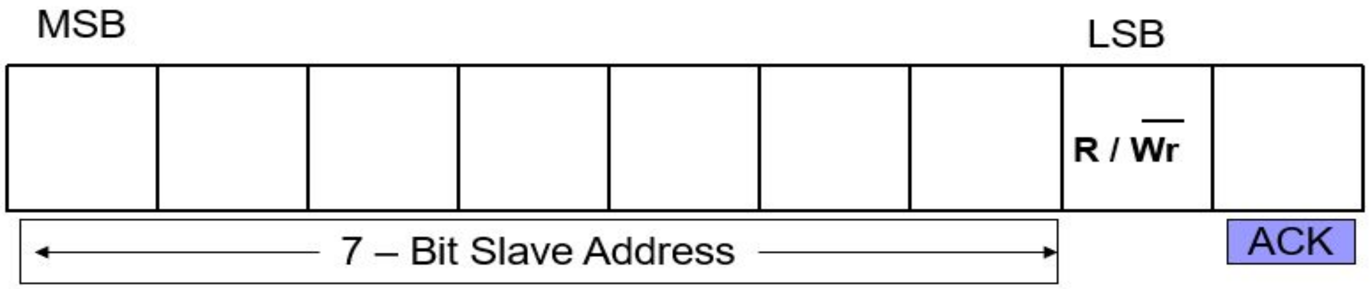
- ✔ Every Byte put on SDA must be 8 bit long
- ✔ Each Byte followed by Acknowledge bit by the receiver
- ✔ Transfer- MSB to LSB

- ✔ **When SCL is low- Data can be transfer**

  "to avoid the start and stop conditions"



Notes for Data Transfer



R/Wr

0 – Slave written to by Master

1 – Slave read by Master

ACK – Generated by the slave whose address has been output.

IIC Advantages

There is a lot of advantage of I2C protocol which makes the user helpless to use the I2C protocol in many applications.

- It is an asynchronous communication protocol, so no need for precise oscillators for the master and slave.
- I2C requires only two-wire, one wire for the data (SDA), and the other wire for the clock (SCL).
- It provides the flexibility to the user to select the transmission rate as per the requirements.
- In the I2C bus, each device on the bus is independently addressable.
- It follows the master and slave relationships.
- It has the capability to handle multiple masters and multiple slaves on the I2C Bus.
- I2C has some important features like arbitration, clock synchronization, and clock stretching.
- I2C provides ACK/NACK (acknowledgment/ Not-acknowledgement) features that provide help in error handling.

IIC disadvantages

- Open Collector driver at master needs pull up resistance 10k on each line
- High Power Requirement
- Low Speed relative to SPI

The End

IMT SCHOOL

IMT SCHOOL

www.imtschool.com

www.facebook.com/imaketechnologyschool/