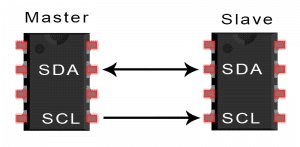
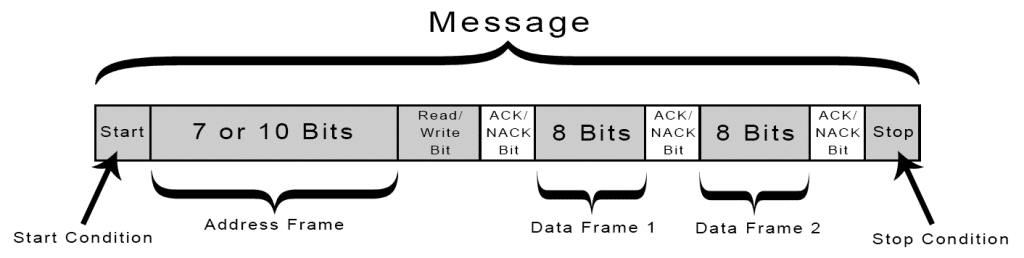
[1.] **BASICS OF THE I2C COMMUNICATION PROTOCOL**

****

**[2.] I2C Frame Format :**

****

**[3.] Start , Stop  and Data validation :**

***Data Stability—*The state change of bytes in SDA line only takes place when SCL line goes LOW to avoid the false START**

**[4.] Speed Modes of I2C :**

**Standard Mode = 100 Kbps**

**Fast Mode = 400 Kbps**

**High-Speed Mode = 3.4 Mbps**

**Ultra-Fast Mode = 5 Mbps**

**[5.] ACK**

**If a particular slave device is addressed, the slave acknowledges to master by holding SDA low for one clock cycle.**

***Acknowledgement and Not-Acknowledgment byte—***

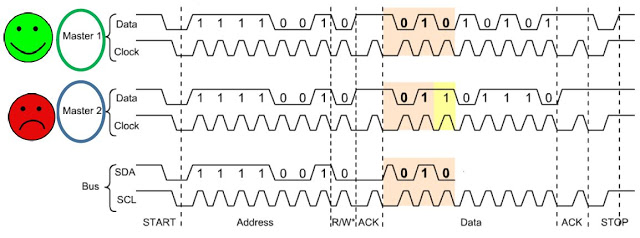
If a master writes something to the slave, for successful writing, the slave responds with a successful acknowledgement. If the slave writes something to master, then master responds with successful acknowledgement byte to the slave. This applies to every frame (data, register, address) in the I2C bus communication.

Before the receiver sends an acknowledgement byte, the transmitter releases the SDA line free; now if the receiver pulls the SDA line low during the low phase of the clock and if SDA remains stable low during the high phase of the clock, the transmitter gets a successful acknowledgement.

If the receiver does not pull the SDA line low during the low phase of the clock, it remains high or changing during the acknowledgement clock period or high clock period, the transmitter assumes it as a Not-acknowledgement byte. And there are various reasons for the NACK byte:

1. The receiver is busy doing some process so it is not able to send the ACK byte.
2. The receiver is not able to understand the data from the transmitter and doesn’t send any ACK byte.
3. No receiver is present at that moment or receiver is damaged during the process.
4. Master-receiver notifies the slave-transmitter for the end of the data transfer.

[6.]   ***Arbitration Process—***



**[7. ]** ***Clock Stretching—***

Clock stretching pauses the communication for some time and this is performed by slave only. We’ve studied that communication is mostly handled by master only but there is a case where slave isn’t able to handle the data or hasn’t processed the previous data yet, in that case, after maste**r releases the SCL line HIGH, the Slave pulls it LOW until it is ready to receive the next data**. After releasing the SCL line by a slave, the master controls the clock again.