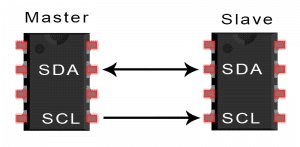
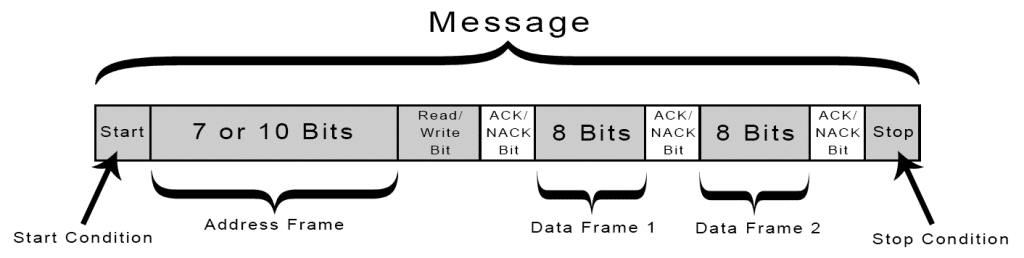
[1.] **BASICS OF THE I2C COMMUNICATION PROTOCOL**

****

**[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—***

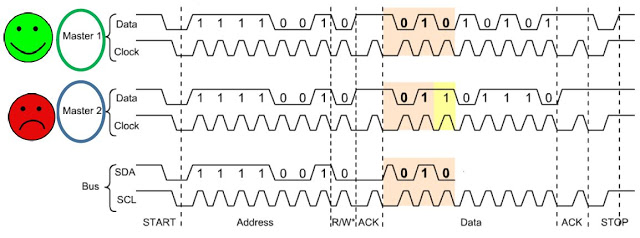
If a master writes something to the slave, for successful writing, the slave responds with a successful acknowledgement. If the slave writes something to master, then master responds with successful acknowledgement byte to the slave. This applies to every frame (data, register, address) in the I2C bus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

**The important difference between I2C and SPI ( I2C vs SPI ) communication protocol.**

|  |  |
| --- | --- |
| **I2C** | **SPI** |
| I2C can be multi-master and multi-slave, which means there can be more than one master and slave attached to the I2C bus. | SPI can be multi-save but does not a multi-master serial protocol, which means there can be only one master attached to the SPI bus. |
| I2C is a half-duplex communication protocol. | SPI is a full-duplex commination protocol. |
| I2C has the feature of clock stretching, which means if the slave cannot able to send fast data as fast enough then it suppresses the clock to stop the communication. | Clock stretching is not the feature of SPI. |
| I2C is used only two wire for the communication, one wire is used for the data and the second wire is used for the clock. | SPI needs three or four-wire for communication ((depends on requirement), MOSI, MISO, SCL, and Chip-select pin. |
| I2C is slower than SPI. | In comparison to I2C, SPI is faster. |
| I2C draws more power than SPI. | Draws less power as compared to I2C. |
| I2C is less susceptible to noise than SPI. | SPI is more susceptible to noise than I2C. |
| I2C is cheaper to implement than the SPI communication protocol. | Costly as compared to I2C. |
| I2C work on wire and logic and it has a pull-up resistor. | There is no requirement of a pull-up resistor in the case of the SPI. |
| In I2C communication we get the acknowledgment bit after each byte. | Acknowledgment bit is not supported by the SPI communication protocol. |
| I2C ensures that the data sent is received by the slave device. | SPI does not verify that data is received correctly or not. |
| I2C support multi-master communication. | SPI does not support multi-master communication. |
| I2C is a multi-master communication protocol that’s why it has the feature of arbitration. | SPI is not a multi-master communication protocol, so it does not consist of the properties of arbitration. |
| I2C is the address base bus protocol, you have to send the address of the slave for the communication. | In the case of the SPI, you have to select the slave using the slave select pin for the communication. |
| I2C has some extra overhead due to start and stop bits. | SPI does not have a start and stop bits. |
| I2C supports multiple devices on the same bus without any additional select lines (work on the basis of device address). | SPI requires additional signal (slave select lines) lines to manage multiple devices on the same bus. |
| I2C is better for long-distance. | SPI is better for a short distance. |
| I2C is developed by NXP. | SPI is developed by Motoro |