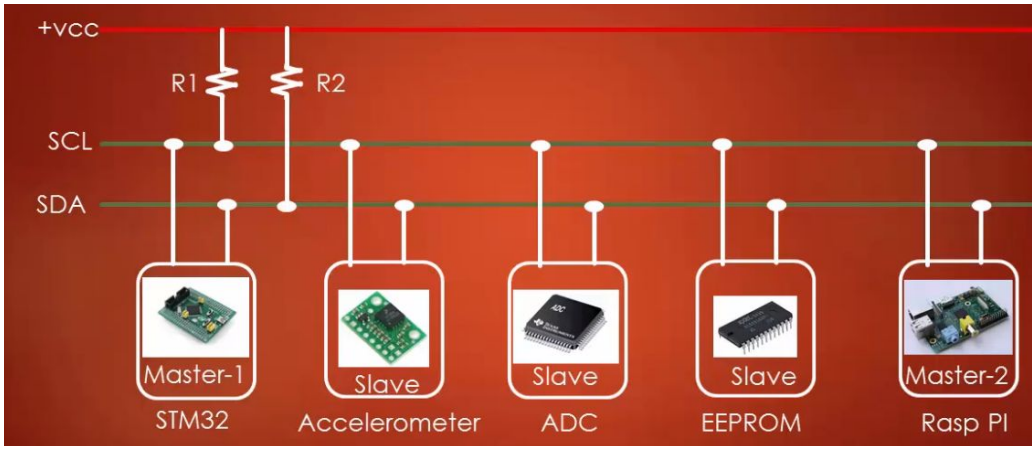
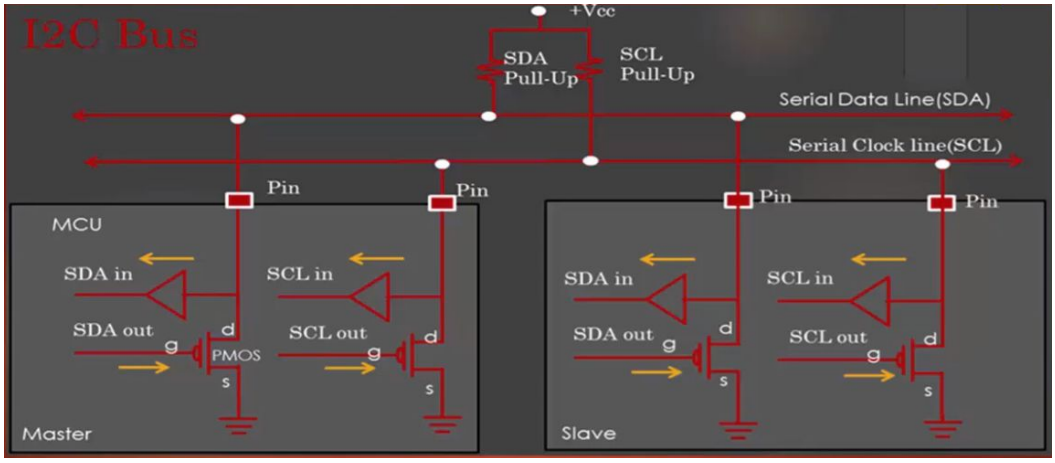
Inter-integrated circuits protocol

- Inter-integrated circuits , also known as I2C
- Two wire protocol
- Ideal for slow communications
- Each slave is specified by a seven bit or 10 bit address
- Multi master capable
- Supports hardware ack
- Half duplex
- Slave Controls clock if it is busy



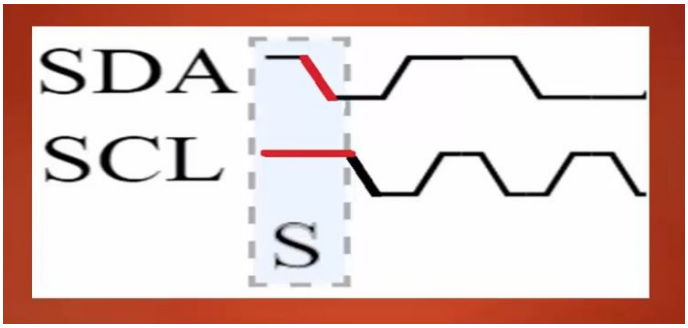
Hardware



Terms used in I2C:

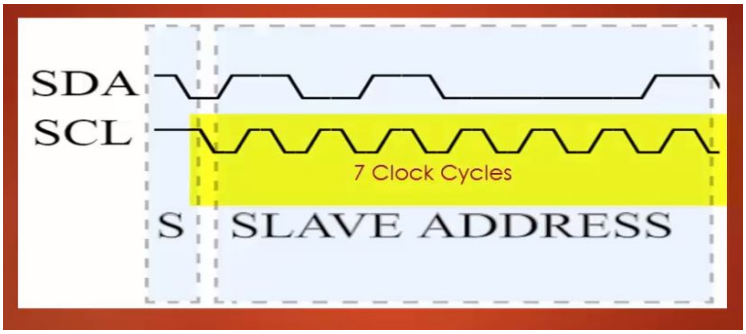
- Start condition
- Address phase
- Read / write bit
- Ack
- Nack
- Stop condition
- Repeated start
- Clock stretching

Start condition

- Always generated by master
- If the SDA line makes a transition from high to low when SCL is high.
- When SCL is high SDA is not allowed to make any cransition , as they are control signals
  - 

Address phase

- Where master sends 7 bit slave address in seven clock cycles
- Data transition happens only when the clock is low

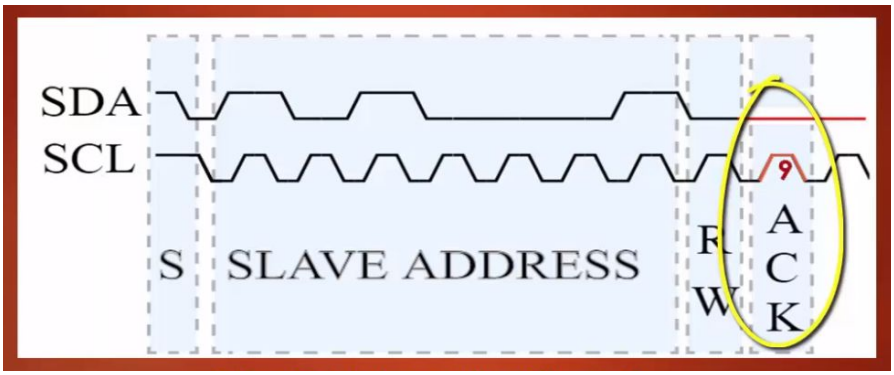


Read / write bit

- Once address bits are sent , master sends read or write bit in 8th clock cycle.
- In 8th clock cycle if the SDA is low , it is interpreted as write
- In 8th clock cycle if the SDA is high , it is interpreted as read

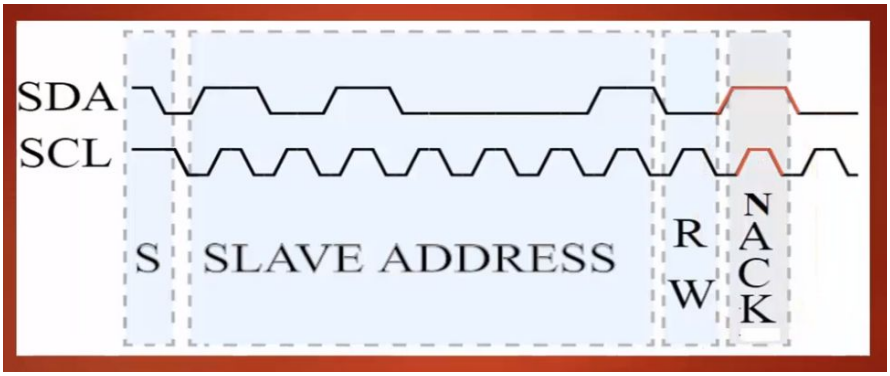
Ack

- In the ninth clock if the SDA line goes low it is called ACK from the salve
- If the address sent to slave matches its own address , the slave pulls the SDA to low at the ninth clock cycle



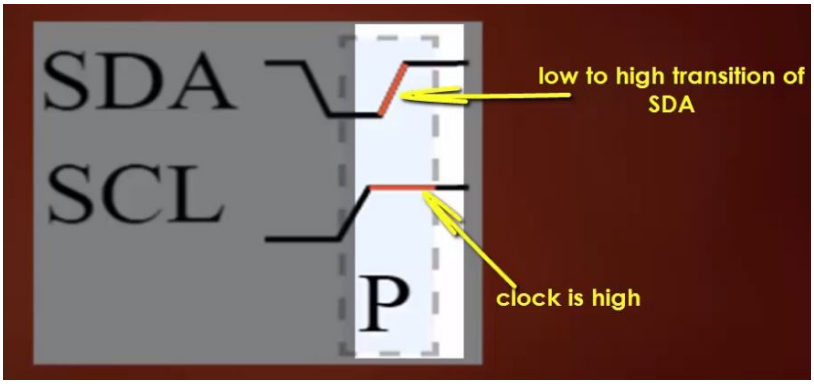
Nack

- In the ninth clock if the SDA line goes high it is called NACK from the slave
- If the address sent to slave does not matches its own address , the slave pulls the SDA to high at the ninth clock cycle
- On receiving NACK . master decides if it needs to send stop condition or send start condition



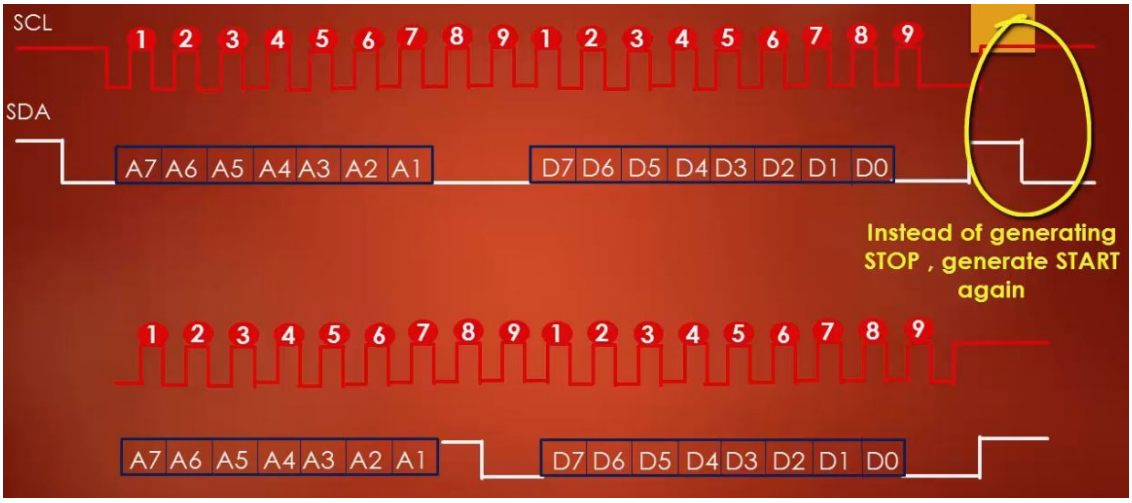
Stop condition

- Always generated by master
- If the SDA line makes a transition from low to high when SCL is high.
- When SCL is high SDA is not allowed to make any cransition , as they are control signals
- The bus is released and can be used by other bus masters



Repeated start

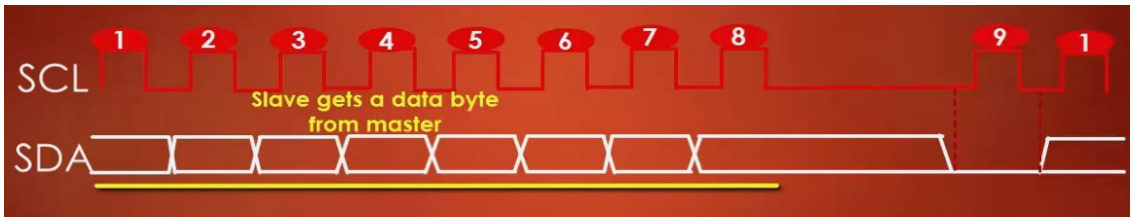
- Start without a stop
- Generating another start without generating intermediate stop

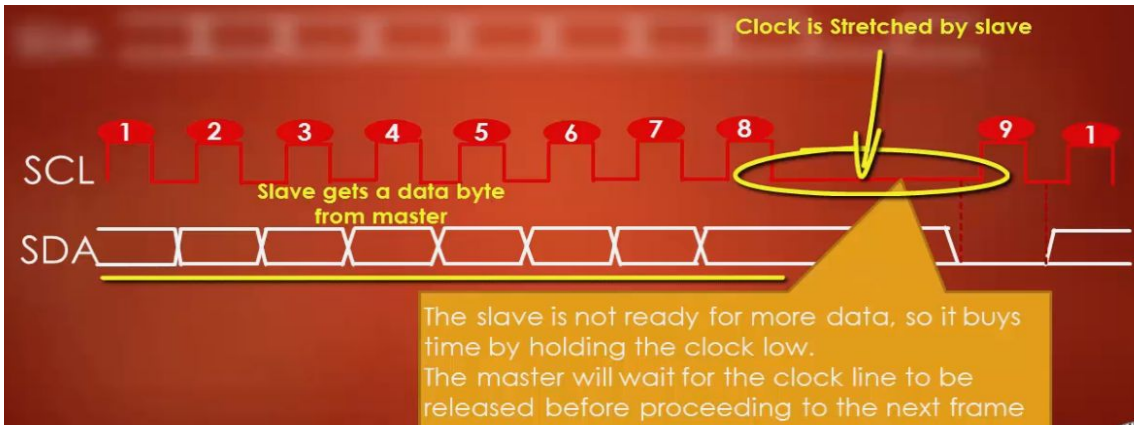


Clock stretching

- It means holding the clock to zero or ground level.
- The moment sclk is held low , the whole i2c interface pauses till clock is given upto its normal operation level
- This feature is used by master/ salve to slow down the communication , by holding the clock to low , which actually prevents the clock to rise high again and the i2c communication stops for a while

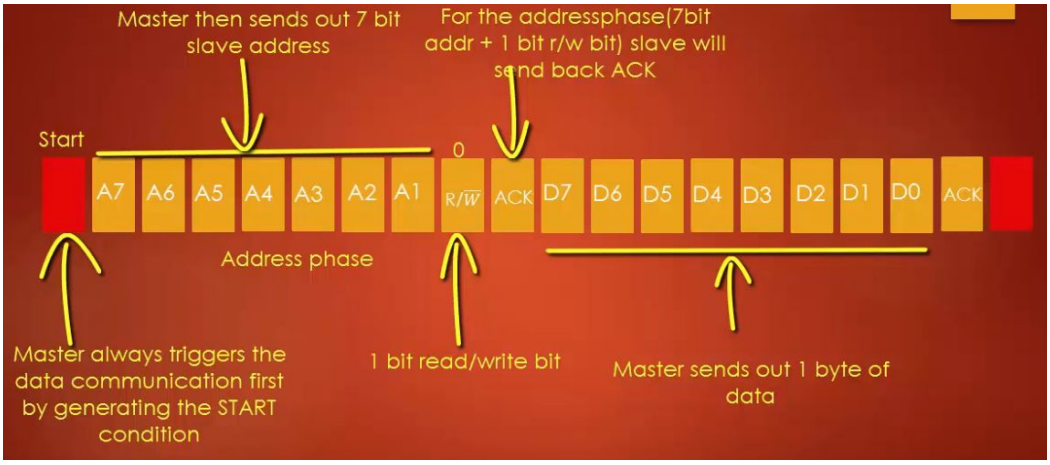
- If the slave device is not able to cooperate with clock speed generated by master and it needs to slow down a little , slave takes advantage of clock streatching
- When slave stretches the clock to low on ninth clock pulse , the master thinks its a NACK (if clock stretching is disabled),
- By stretching the clock slave tells the master “ i am busy please hold on i will give you acknowledgment
- If enabled Clock stretching and unstretching is controlled by hardware so need not do it in code (untill its a soft i2c driver)
-

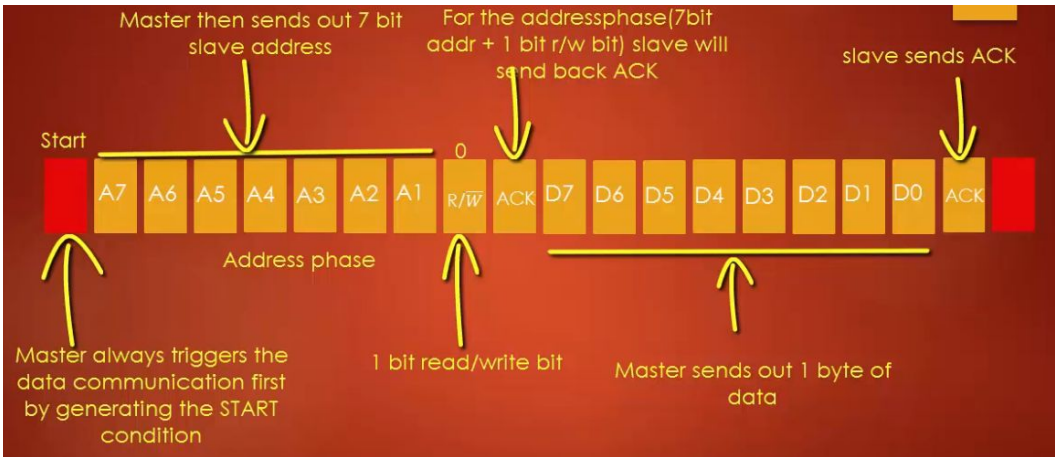


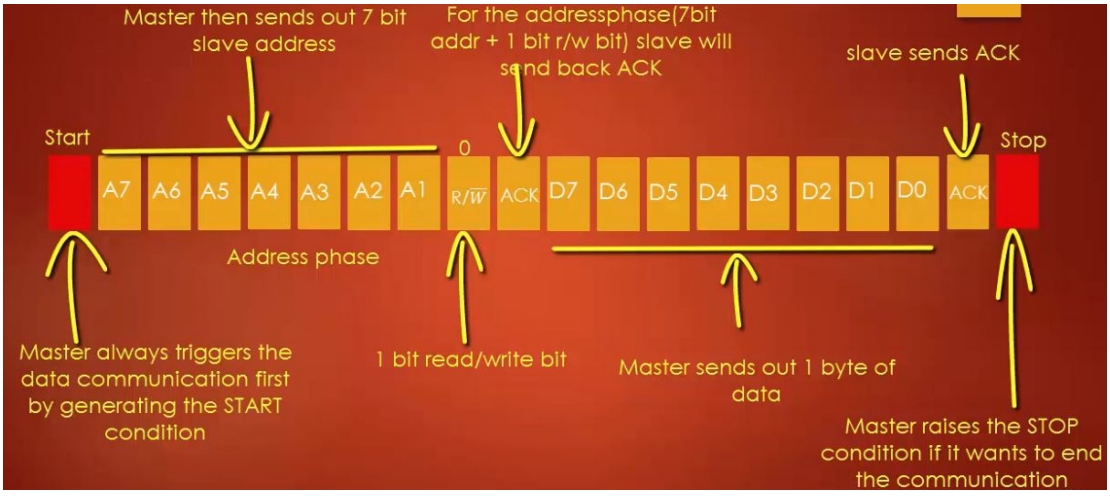


Protocol:

- ***<u>One byte write operation by master:</u>***
    - Master triggers the communication first by sending the start signal
        - Bus is under master's control.
    - Master send seven bit address of the slave
    - Master send one bit for read or write bit (also known as address phase)
        - If this bit is one , master is expecting data from slave (read)
        - If this bit is zero , master will send out data to slave (write).
    - When address phase (read/write bit) finishes , the slave will send an ACK if the address matches it's own address .
    - On receiving the ack , master will transmits data bytes to the slave
    - Slave sends ack for each data byte received by slave
    - If master wishes to terminate the communication , it sends stop signal where the bus will be released by both master and slave.the bus state goes to normal (pulled high).

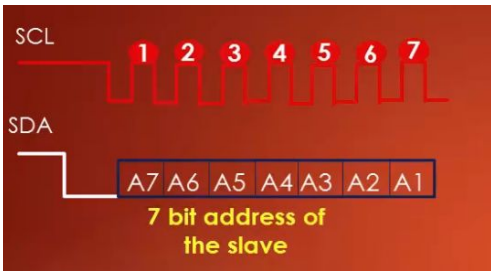

Master then sends out 7 bit slave address
For the addressphase(7bit addr + 1 bit r/w bit) slave will send back ACK
Start
A7 A6 A5 A4 A3 A2 A1
0
R/W
ACK
D7 D6 D5 D4 D3 D2 D1 D0
ACK
Address phase
Master always triggers the data communication first by generating the START condition
1 bit read/write bit
Master sends out 1 byte of data


Master then sends out 7 bit slave address
For the addressphase(7bit addr + 1 bit r/w bit) slave will send back ACK
slave sends ACK
Start
A7 A6 A5 A4 A3 A2 A1
0
R/W
ACK
D7 D6 D5 D4 D3 D2 D1 D0
ACK
Address phase
Master always triggers the data communication first by generating the START condition
1 bit read/write bit
Master sends out 1 byte of data


Master then sends out 7 bit slave address
For the addressphase(7bit addr + 1 bit r/w bit) slave will send back ACK
slave sends ACK
Start
Stop
A7 A6 A5 A4 A3 A2 A1
0
R/W
ACK
D7 D6 D5 D4 D3 D2 D1 D0
ACK
Address phase
Master always triggers the data communication first by generating the START condition
1 bit read/write bit
Master sends out 1 byte of data
Master raises the STOP condition if it wants to end the communication

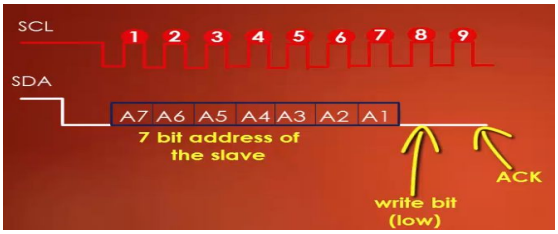
- ***Multi byte write operation by master:***
  - Master generates a start condition

  - Master sends out address of seven bits

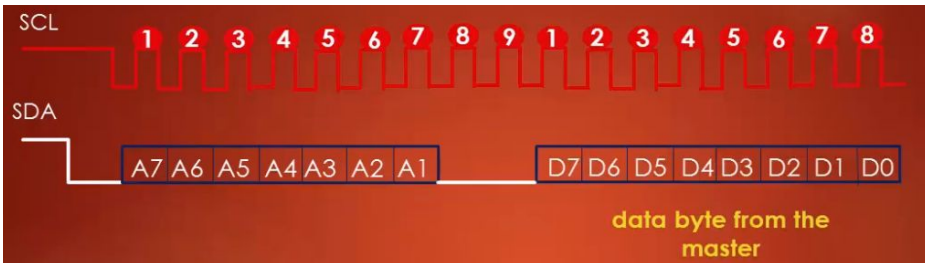


  - Master send the write bit which is low at the 8th clock cycle

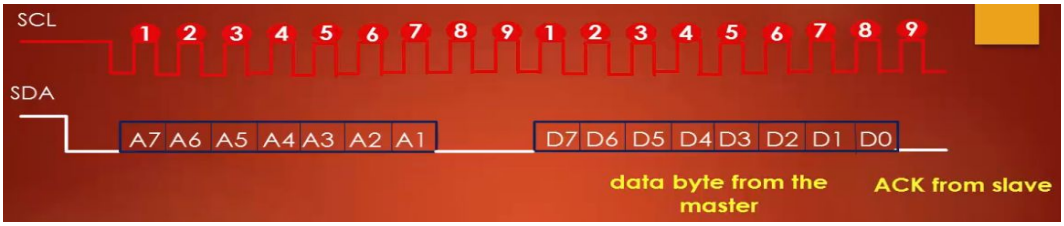
  - Slave sens the ack at the ninth clock



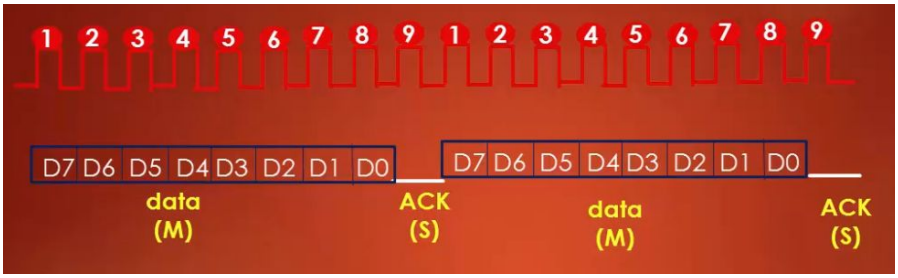
  - Master sends first data byte which is eight byte long



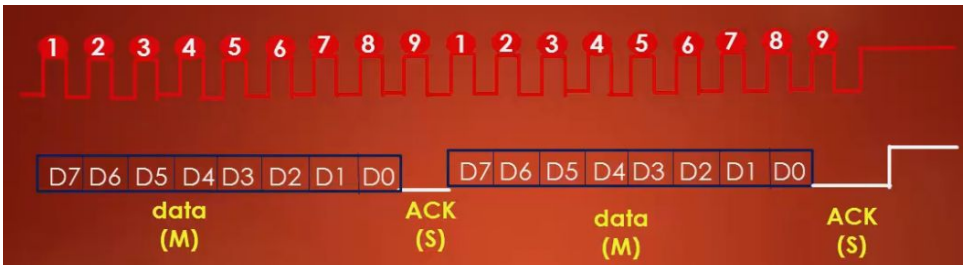
  - Slave send the ack for first data byte

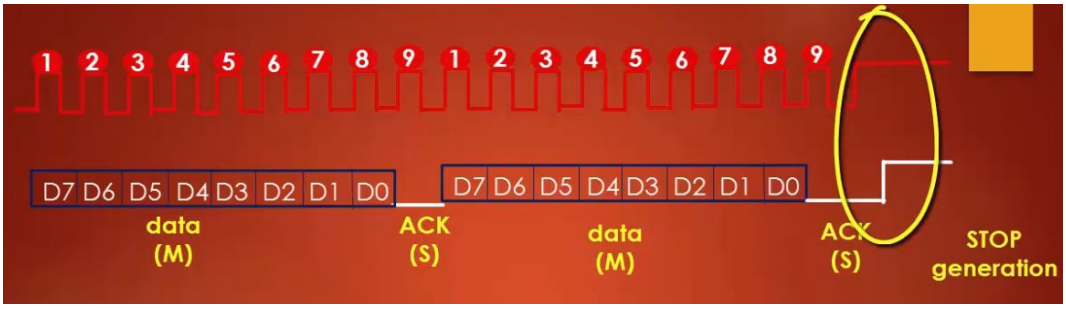


- Master again sends the next data byte
- Slave again sends ack to master



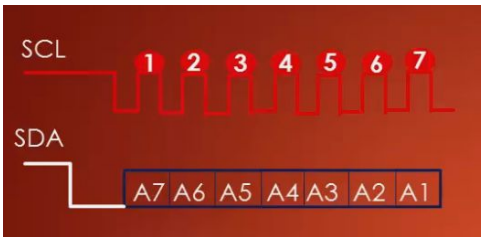
- This continues till last data byte to be sent by master
- If there is no data to be sent to slave , master terminates the connection by generating stop condition



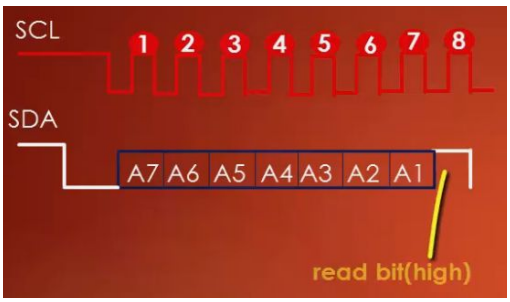


- ***Multi byte read operation by master:***
  - Master generates a start condition

  - Master sends out address of seven bits

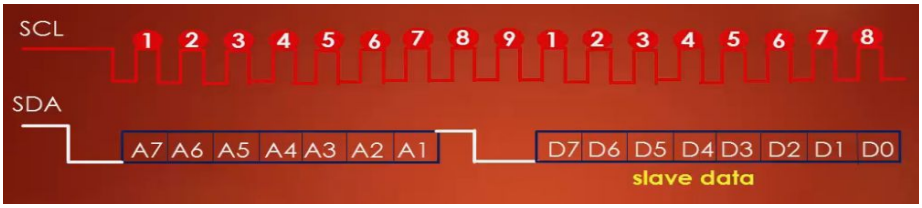


  - Master send the read bit which is low at the 8th clock cycle

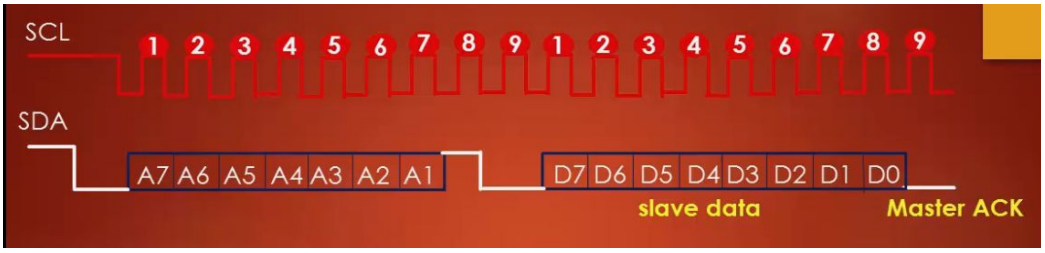


  - Slave sends the ack at the ninth clock

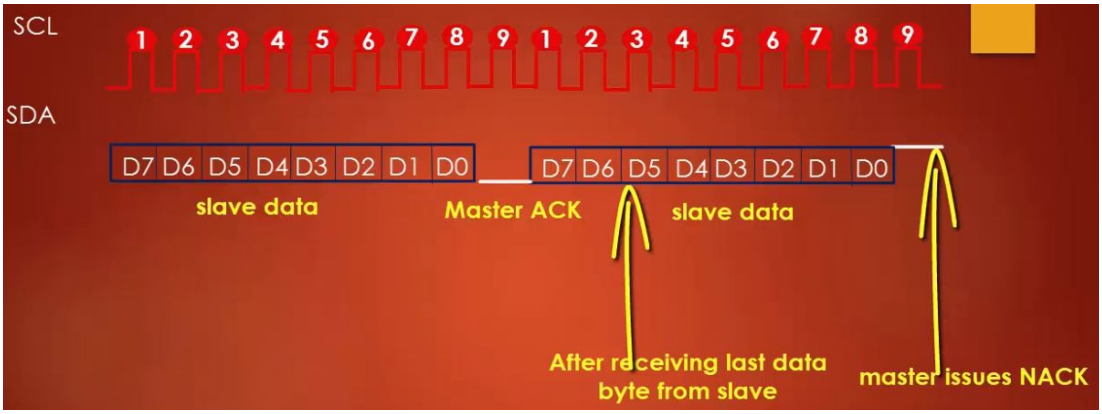
  - Slave will start sending out data byte to the master



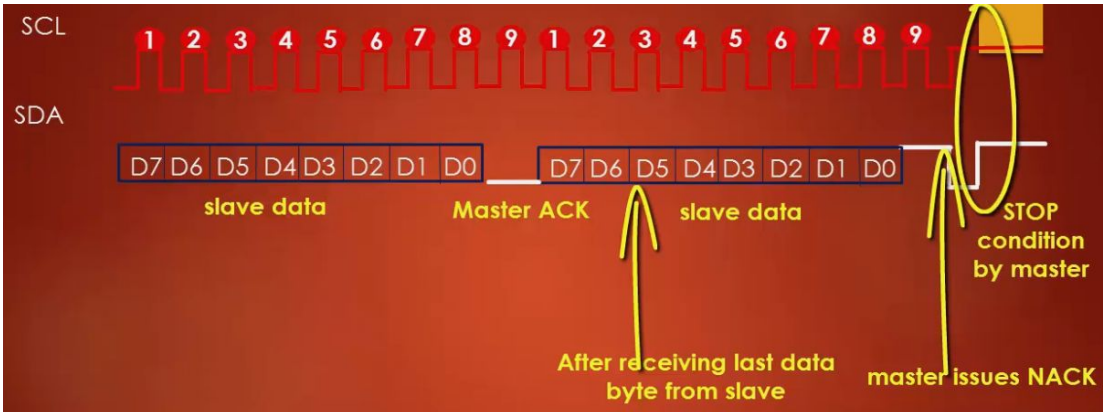
  - For each data byte received master will send the ack to slave



- When master decides , it no longer needs data bytes from slave ,after receiving last data byte , master sends NACK

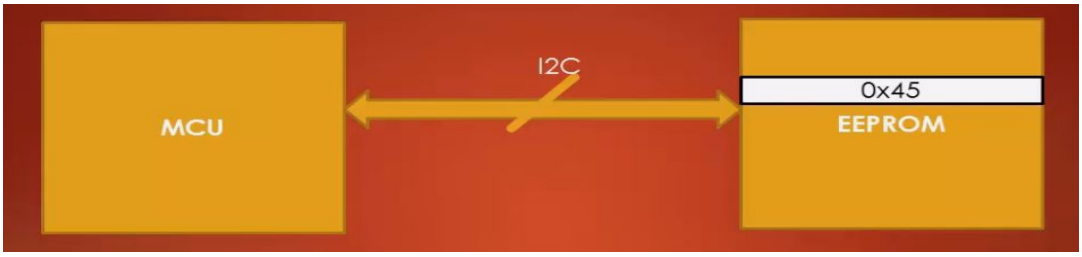


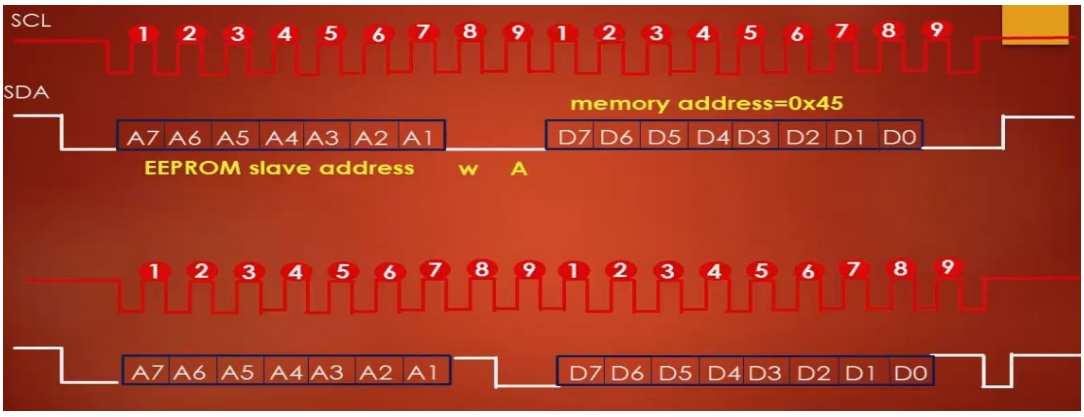
- Master generates stop condition to stop the communication

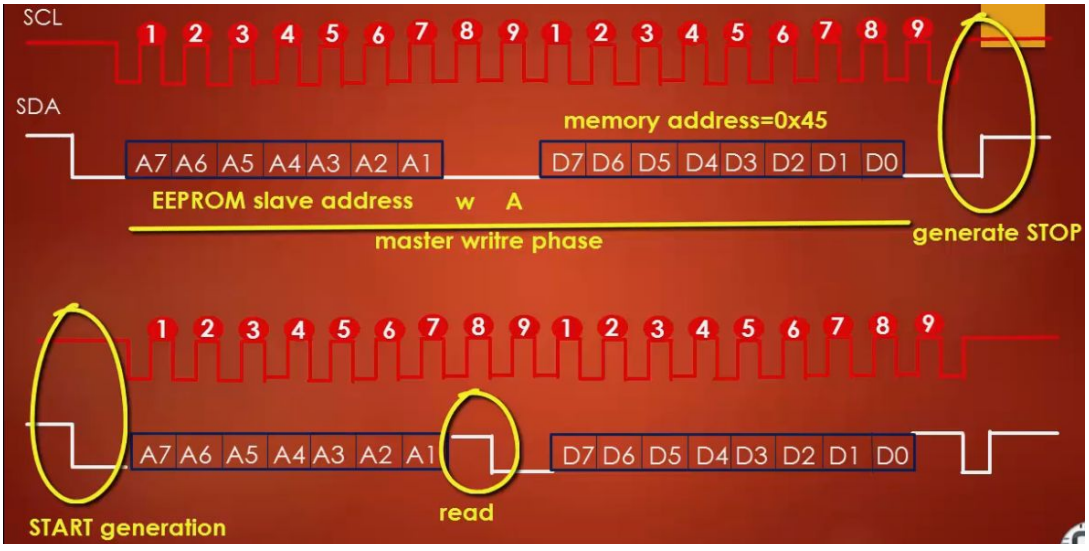


- ***<u>Multi byte read after write operation by master:</u>***
  - **Let's take an example for reading from a particular location in eeprom where** Master wants to write to the slave then it wants to read back
  - If master wants to read content of eeprom at address 0x45
    - Master writes to the slave the memory address from which it wants to fetch
    - Master generates start condition
    - Master sends device address
    - Master send write bit
    - Slave send ack
    - Master sends address to be read from (eg 0x45)
    - Slave send ack
    - As slave will not send back data even if master produces more clock (because salve has received write bit , slave will never send data till it receives read bit as it is in master write phase)
    - Now there are two ways to read the data

- **First way is a stop condition followed by start condition**
  - Generate a stop condition
  - Trigger a start condition
  - Send slave address followed by read / write bit equals to one
  - Slave will send ack
  - Slave will send data to master byte by byte
  - Master need to send ack after each byte
  - Master need to send NACK after last byte
  - Master need to generate stop condition







- **Second way is restart condition**
  - Instead of generating stop condition , generate an another start condition (repeated start).
  - Send slave address followed by read / write bit equals to one
  - Slave will send ack
  - Slave will send data to master byte by byte
  - Master need to send ack after each byte
  - Master need to send NACK after last byte
  - Master need to generate stop condition