INTERFACING

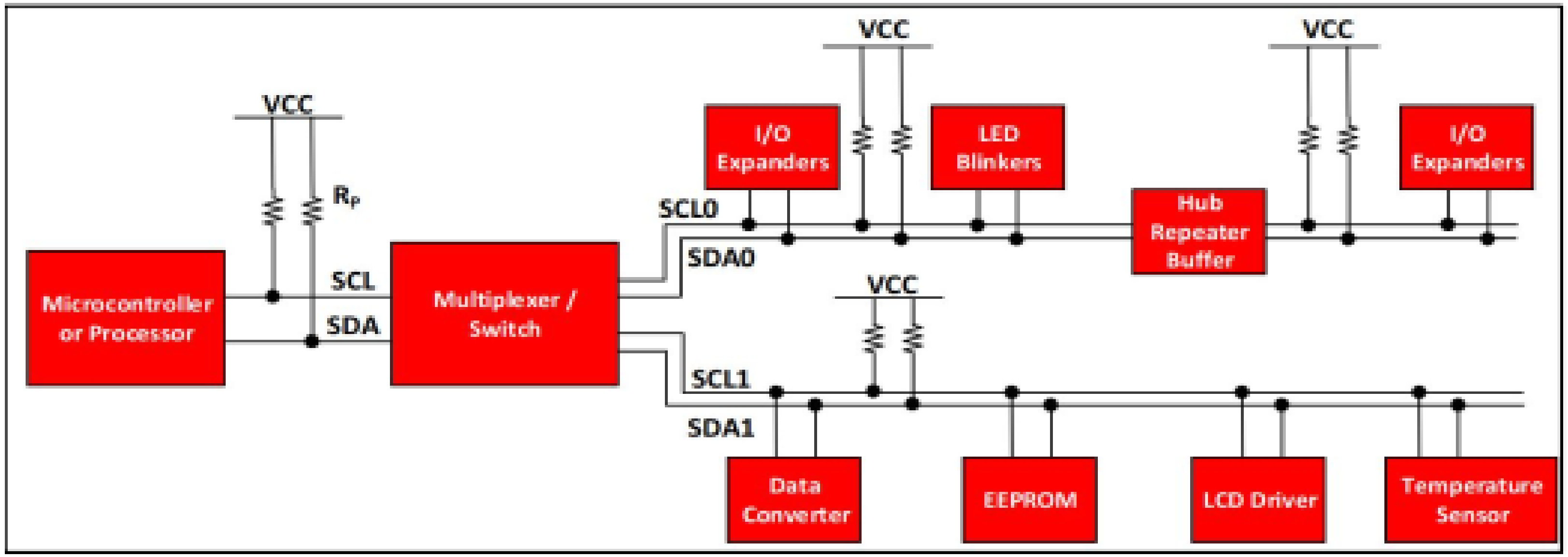
I2C

Our Agenda:

- What is I2c.
- I2C characteristic.
- Open-collector in I2C
- Hardware Connection of I2C.
- I2C Ideal frame.
- Clock Generation in I2C.
- Arbitration bus in I2C.
- Clock Stretching in I2C.
- I2C Advantages.
- I2C Limitations.
- I2C Register in ATmega32.
- Steps to program ATmega32 I2C

Internal Integrated Circuit:

- The Two-wire Serial Interface (TWI) is ideally suited for typical microcontroller applications. TWI is the same IIC with limited features.
- I2C uses only two bidirectional open-collector or open-drain lines: serial data line (SDA) and a serial clock line (SCL),

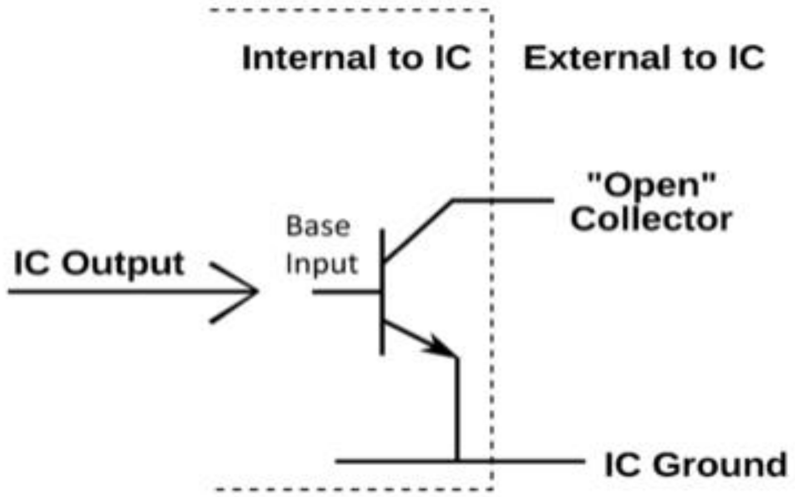


I2C characteristic:

- Wired.
- Serial.
- Multi Master Multi Slave.
- Synchronous.
- Half Duplex.
- Used for short-distance communications.
- Open-collector or Open-drain bus
- Used for connecting lower ICs to MCUs

Open-collector in I2C:

- An open collector is a common type of output found on many integrated circuits (IC), which behaves like a switch that is either connected to the ground or disconnected.
- Instead of outputting a signal of a specific voltage or current, the output signal is applied to the base of an internal NPN transistor whose collector is externalized (open) on a pin of the IC.



Open-collector in I2C:

- The emitter of the NPN transistor is connected internally to the ground pin. If the output device is a MOSFET, The output is called an open drain and functions similarly.

- NOT Gate also is used to make the output more logical.

Truth table without NOT Gate

B	C
0	float
1	0

Truth table with NOT Gate

B	C
0	0
1	float

I2C:

➢the float is not a signal, so the two buses of I2C must be Pulled-Up to become one, so if there Master2 writes “0” and Master1 writes “1”, the “0” signal Only will be appeared because of the pull-up resistor.

we have two concepts:

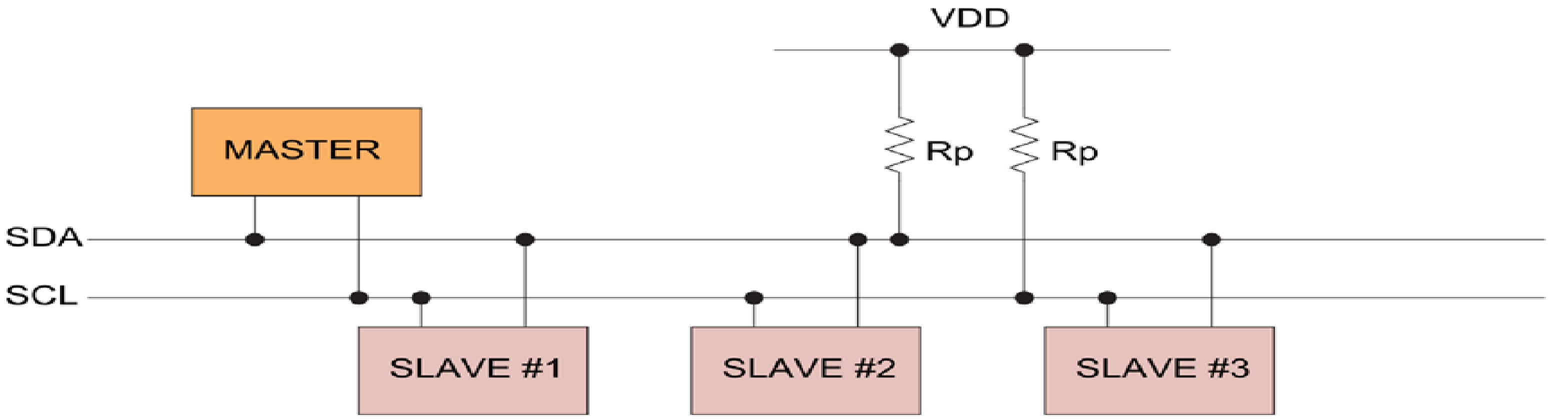
➢**Dominant Bit**:
it is the bit ones it writes to the bus, it clears and blurs the effect of other bit, ZERO is the Dominant in IIC.

B	C
0	0
1	1

➢**Recessive Bit**:
it is the bit that is cleared and blurred if a Dominant bit is written, ONE is the recessive bit in IIC.

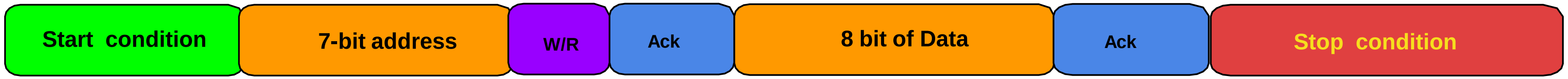
Hardware Connection of I2C:

- The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA).
- Because it’s an MMMS protocol, any master can start the communication at any time.



I2C Ideal frame:

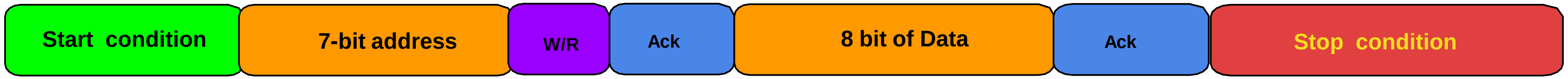
- Start Condition – 1 bit
- Slave Address – 7 bit
- Read/Write – 1 bit
- Acknowledge – 1 bit
- Data – 8 bit
- Acknowledge – 1 bit
- Stop Condition – 1 bit



Start condition | 7-bit address | W/R | Ack | 8 bit of Data | Ack | Stop condition

I2C Ideal frame:

- START and STOP : can be generated by keeping the SCL line high and changing the level of SDA. To generate START condition the SDA is changed from high to low while keeping the SCL high. To generate STOP condition SDA goes from low to high while keeping the SCL high, as shown in the figure below.



Start condition

7-bit address

W/R

Ack

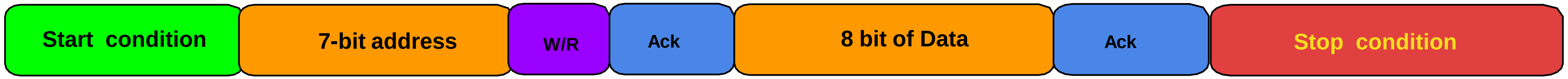
8 bit of Data

Ack

Stop condition

I2C Ideal frame:

- Read/Write Bit :A high Read/Write bit indicates that the master is sending the data to the slave, whereas a low Read/Write bit indicates that the master is receiving data from the slave.



Start condition

7-bit address

W/R

Ack

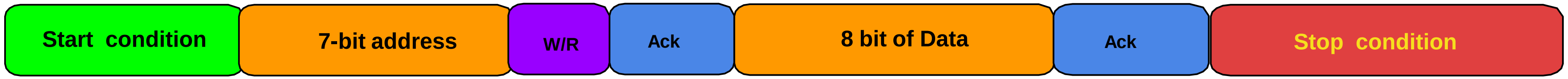
8 bit of Data

Ack

Stop condition

I2C Ideal frame:

- Addressing: The address frame is the first frame after the start bit. The address of the slave with which the master wants to communicate is sent by the master to every slave connected with it. The slave then compares its own address with this address and sends ACK.



Start condition

7-bit address

W/R

Ack

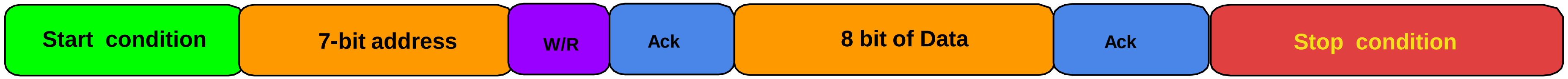
8 bit of Data

Ack

Stop condition

I2C Ideal frame:

- Read/Write Bit :A high Read/Write bit indicates that the master is sending the data to the slave, whereas a low Read/Write bit indicates that the master is receiving data from the slave.
- ACK/NACK Bit: After every data frame, follows an ACK/NACK bit. If the data frame is received successfully then ACK bit is sent to the sender by the receiver



Start condition

7-bit address

W/R

Ack

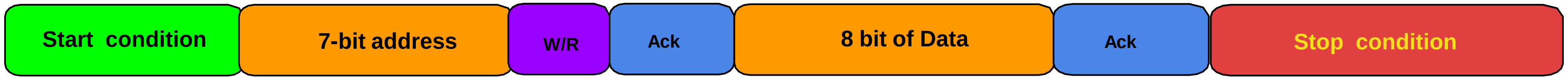
8 bit of Data

Ack

Stop condition

I2C Ideal frame:

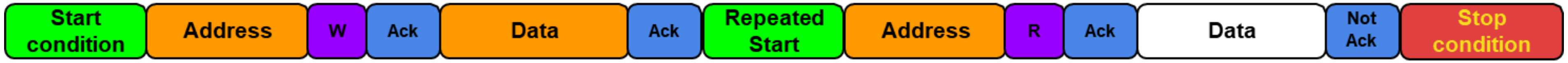
- I2C Packet Format : In the I2C communication protocol, the data is transmitted in the form of packets. These packets are 9 bits long, out of which the first 8 bits are put in SDA line and the 9th bit is reserved for ACK/NACK i.e. Acknowledge or Not Acknowledge by the receiver.



Start condition | 7-bit address | W/R | Ack | 8 bit of Data | Ack | Stop condition

Fram of I2C:

- Repeated Start Condition : Between each start and stop condition pair, the bus is considered as busy and no master can take control of the bus. If the master tries to initiate a new transfer and does not want to release the bus before starting the new transfer, it issues a new START condition. It is called a REPEATED START condition.



Start condition	Address	W	Ack	Data	Ack	Repeated Start	Address	R	Ack	Data	Not Ack	Stop condition

Clock Generation in I2C:

- The SCL clock is always generated by the I2C master. The specification requires minimum periods for the low and high phases of the clock signal. Hence, the actual clock rate may be lower than the nominal clock rate. in I2C buses with large rise times due to high capacitances.
- Note that maximum clock of the I2C is 400K

Arbitration bus in I2C:

- Several I2C multi-masters can be connected to the same I2C bus and operate concurrently. By constantly monitoring SDA and SCL for start and stop conditions, they can determine whether the bus is currently idle or not. If the bus is busy, masters delay pending I2C transfers until a stop condition indicates that the bus is free again.

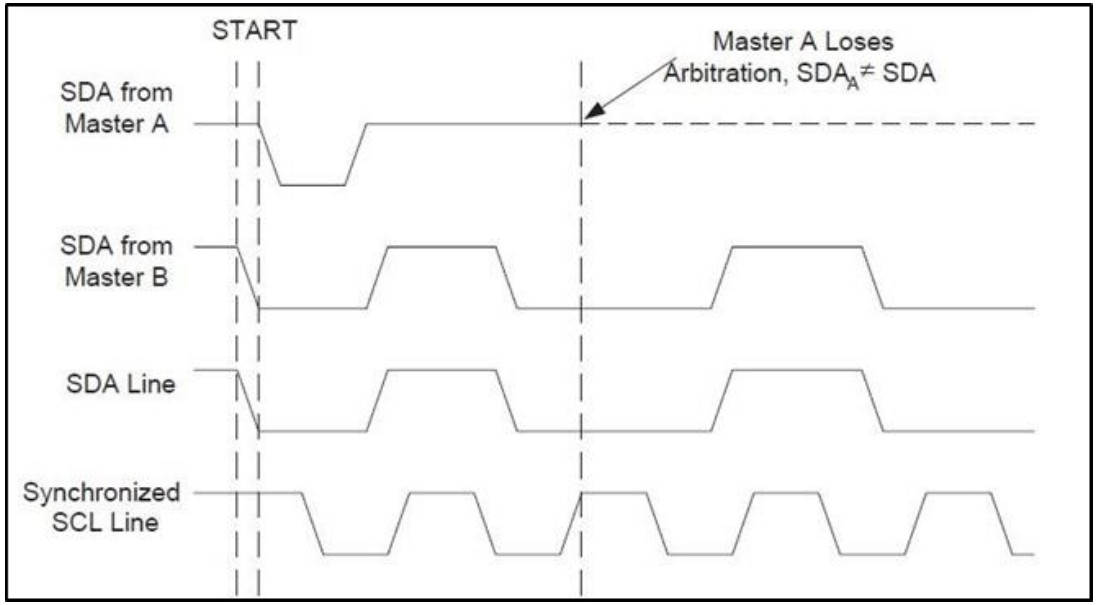
Arbitration bus in I2C:

- However, it may happen that two masters start a transfer at the same time. During the transfer, the masters constantly monitor SDA and SCL. If one of them detects that SDA is low when it should actually be high, it assumes that another master is active and immediately stops its transfer. This process is called arbitration.

Arbitration bus in I2C:

Arbitration is not allowed between:

- A REPEATED START condition and a data bit
- A STOP condition and a data bit
- A REPEATED START and a STOP condition



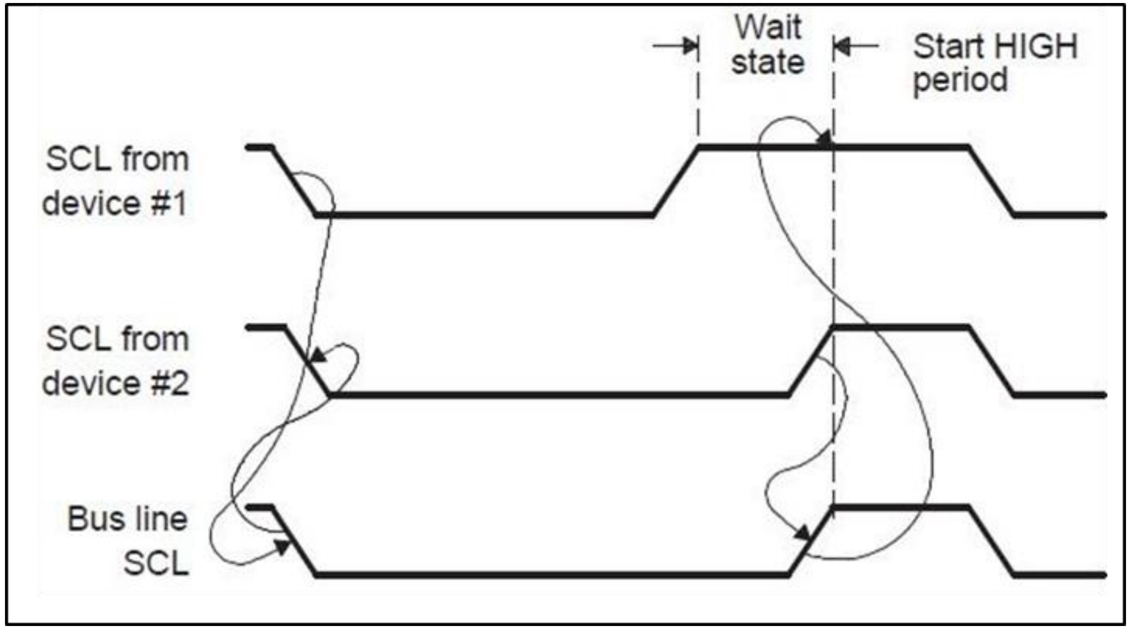
Clock Stretching in I2C:

- I2C devices can slow down communication by stretching SCL: During an SCL low phase, any I2C device on the bus may additionally hold down SCL to prevent it from rising again, enabling it to slow down the SCL clock rate or to stop I2C communication for a while. This is also referred to as clock synchronization.

- Note: The I2C specification does not specify any timeout conditions for clock stretching, any device can hold down SCL as long as it likes.

Clock Stretching in I2C:

- It is used to synchronize between the master and slower slaves or vice versa.
- The addressed slave will hold the clock pin low to indicate that it is not ready to receive the next bit.
- The master will wait for the
  clock pin to go high.



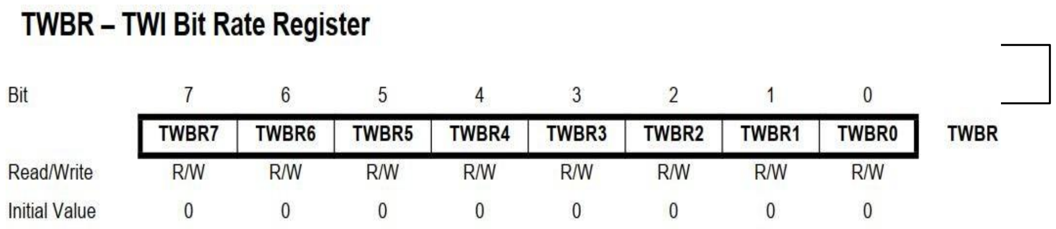
I2C Advantages:

- Can be configured in multi-master mode.
- Complexity is reduced because it uses only 2 bi-directional lines (unlike SPI Communication).
- Cost-efficient.
- It uses ACK/NACK feature due to which it has improved error handling capabilities.

I2C Limitations:

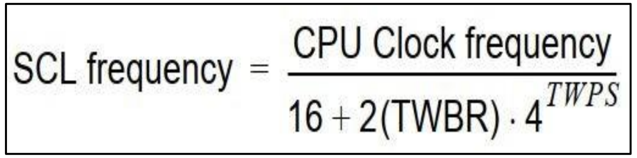
- Slower speed.
- Half-duplex communication is used in the I2C communication protocol.

I2C Register in ATmega32:

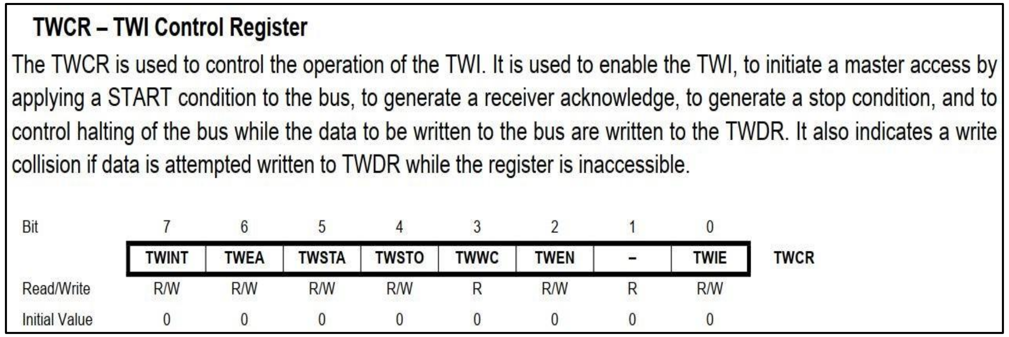


TWBR – TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	



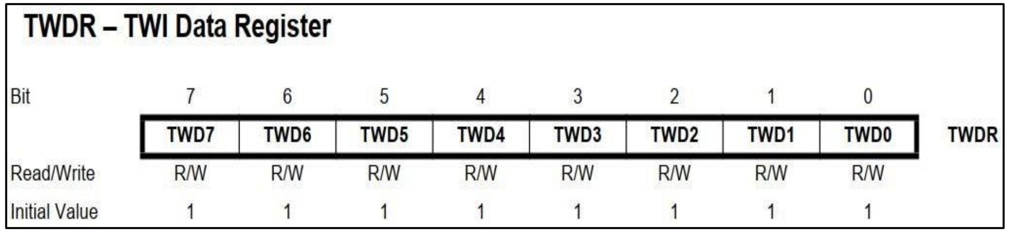
$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$



TWCR – TWI Control Register

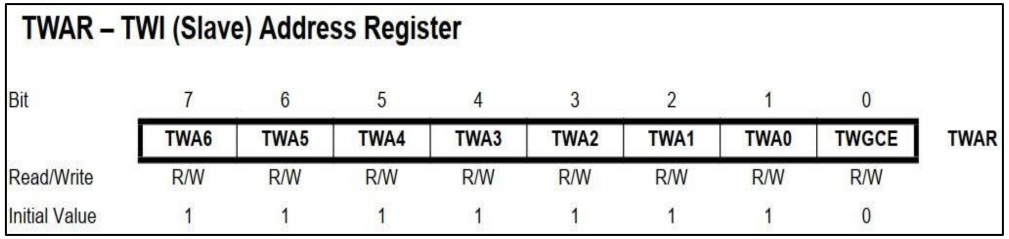
The TWCR is used to control the operation of the TWI. It is used to enable the TWI, to initiate a master access by applying a START condition to the bus, to generate a receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are written to the TWDR. It also indicates a write collision if data is attempted written to TWDR while the register is inaccessible.

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	



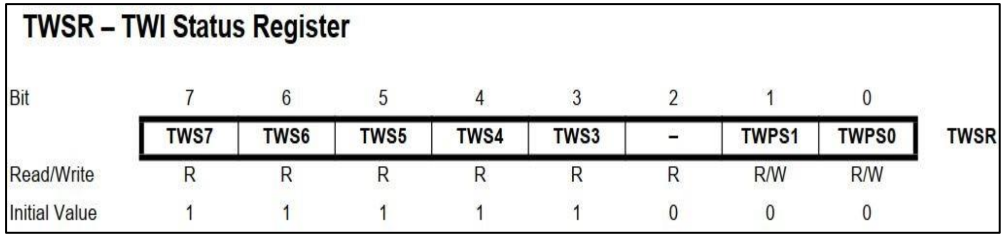
TWDR – TWI Data Register

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	



TWAR – TWI (Slave) Address Register

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	



TWSR – TWI Status Register

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

Steps to program ATmega32 I2C

- **Initialization**
  - Select Prescaler, **TWSR** register
  - Clear status code, **TWSR** register
  - Set bit rate, **TWBR** register
  - Set Address, **TWAR** register

Steps to program ATmega32 I2C

- **Master write**
  - – Send start condition
    - Clear TWINT flag, set TWSTA, and enable TWI, TWCR register
    - Wait for TWINT flag to be 1, TWCR register
    - Check status code for start is sent, TWSR register
  - – Send slave address and write bit
    - Write address + 0 into data register, TWDR register
    - Clear TWINT flag and enable TWI, TWCR register
    - Wait for TWINT flag to be 1, TWCR register
    - Check status code for start is sent, TWSR register

Steps to program ATmega32 I2C

- **Master write**
  - Send data
    - Write data into data register, TWDR register
    - Clear TWINT flag, and enable TWI, TWCR register
    - Wait for TWINT flag to be 1, TWCR register
    - Check status code for start is sent, TWSR register
  - Send stop condition
    - Clear TWINT flag, set TWSTO, and enable TWI, TWCR register
    - Wait for TWSTO to be cleared, TWCR register

Steps to program ATmega32 I2C

- **Master read**
  - Send start condition
    - Clear TWINT flag, set TWSTA, and enable TWI, TWCR register
    - Wait for TWINT flag to be 1, TWCR register
    - Check status code for start is sent, TWSR register
  - Send slave address and read bit
    - Write address +1into data register, TWDR register
    - Clear TWINT flag and enable TWI, TWCR register
    - Wait for TWINT flag to be 1, TWCR register
    - Check status code for start is sent, TWSR register

Steps to program ATmega32 I2C

- **Master read**
  - Read data
    - Select if reading with ACK or NOT, TWCR register
    - Clear TWINT flag and enable TWI, TWCR register
    - Wait for TWINT flag to be 1, TWCR register
    - Check status code for start is sent, TWSR register
    - Read data from TWDR register
  - Send stop condition
    - Clear TWINT flag, set TWSTO, and enable TWI, TWCR register
    - Wait for TWSTO to be cleared, TWCR register

I2CDriver:

Time To

Code

THANK YOU!