The I2C protocol

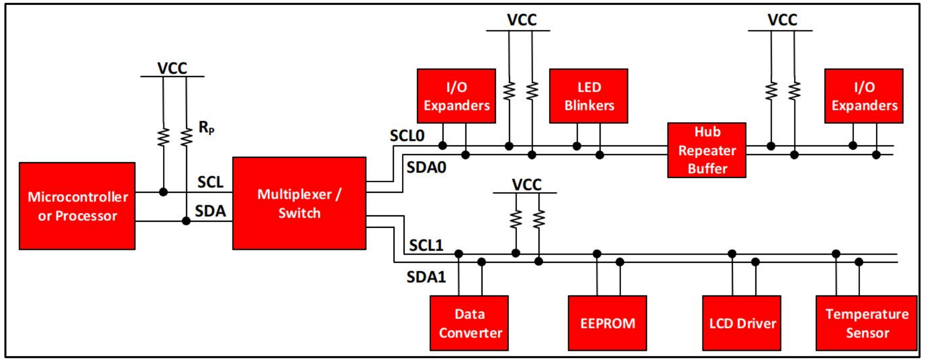
By: Yehia M. Abu Eita

Outlines

- What is I2C?
- Open-Drain bus
- I2C frame formats
- Bus arbitration mechanism
- Clock stretching mechanism
- EEPROM and its interfacing
- ATmega32 I2C registers
- Steps to program ATmega32 I2C

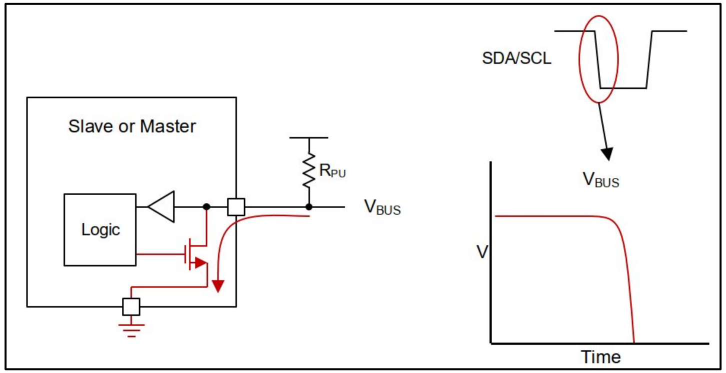
What is I2C?

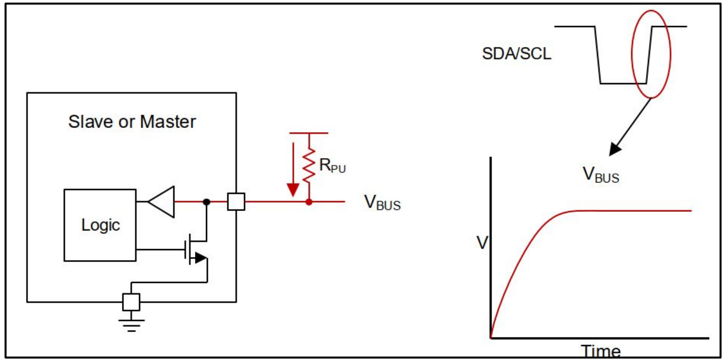
- It is acronym of **I**nter-**I**ntegrated **C**ircuit.
- It is also called **T**wo-**W**ire **I**nterface.
- **Characteristics:**
  - Synchronous
  - Half-Duplex
  - Multi-Master
  - Used to connect lower speed ICs to MCUs
  - Used for short distance communications
  - Open-Collector or Open-Drain bus



Open-Drain bus

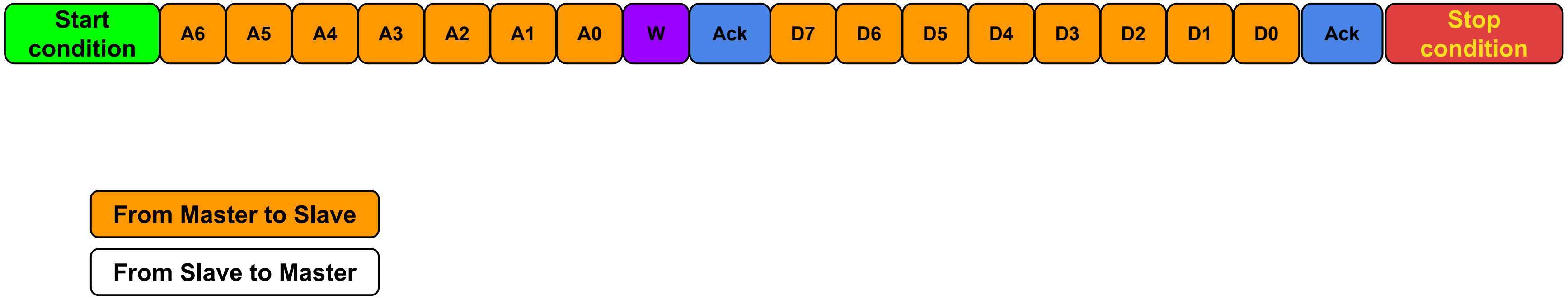
- Open-drain refers to a **type of output** which can **either pull the bus down**, or "**release**" the bus and let it be **pulled up by a pull-up resistor**.
- The **bus will never** run into a communication issue where one device may try to transmit a high, and another transmits a low, **causing a short circuit**.





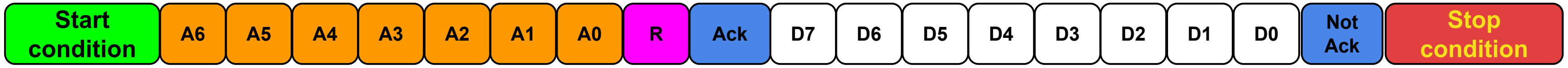
I2C frame formats

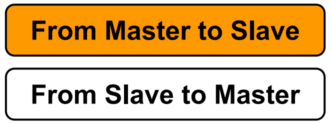
- **Single transmission**



I2C frame formats

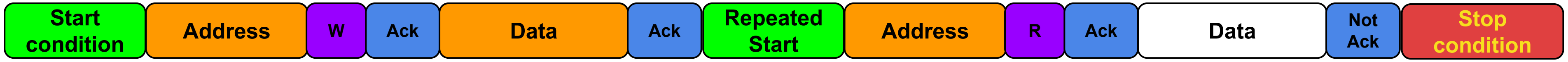
- Single read

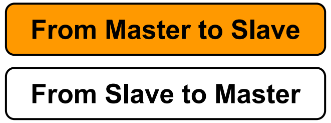




I2C frame formats

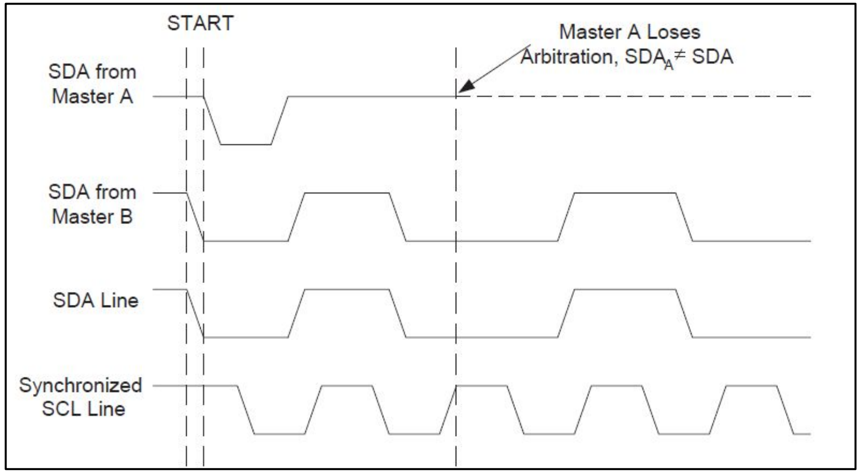
- Write then read





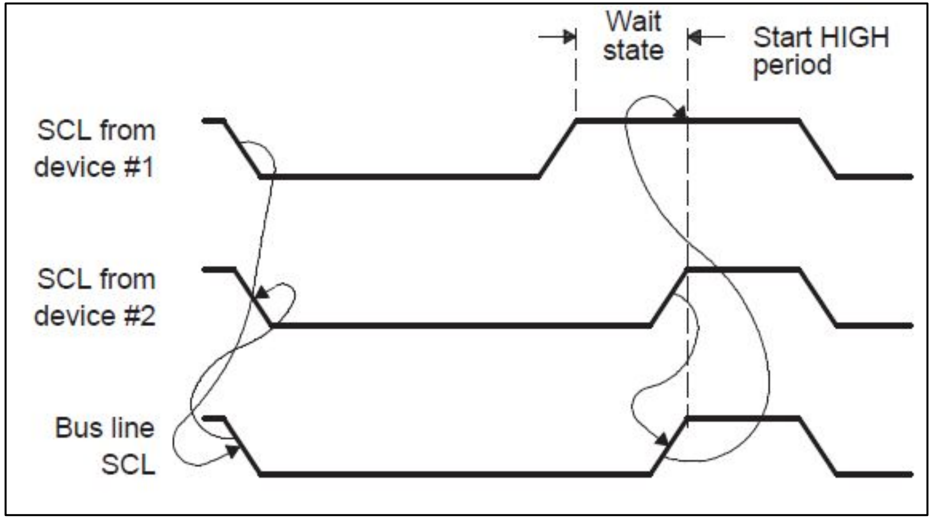
Bus arbitration mechanism

- **Use cases**:
  - **Identical communication** with the **same slave**.
  - Accessing the **same slave** with **different data** or **direction bit (R/W)**.
  - Accessing **different slaves**.
- **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

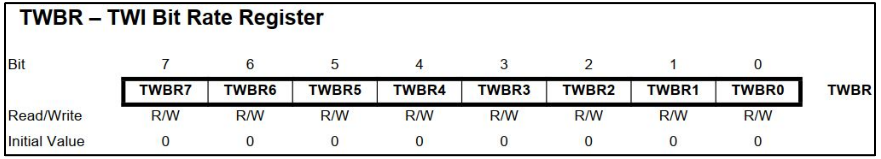


Clock stretching mechanism

- 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**.



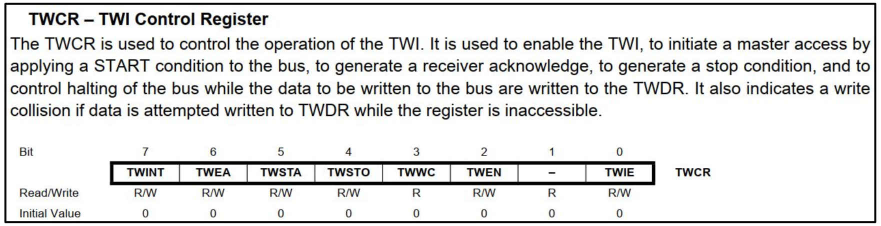
ATmega32 I2C registers



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	

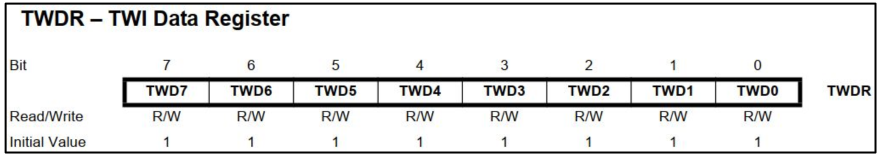
$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$



TWCR – TWI Control Register

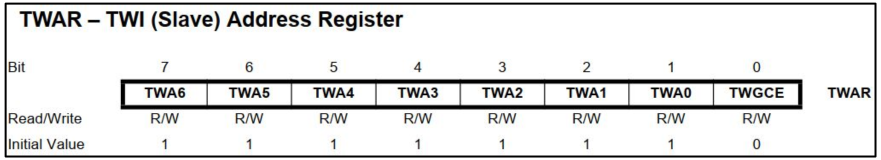
The TWCR is used to control the operation of the TWI. It is used to enable the TWI, to initiate a master access by applying a START condition to the bus, to generate a receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are written 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	



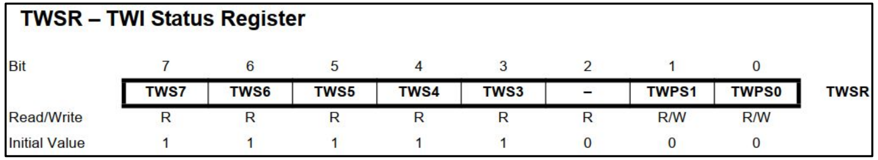
TWDR – TWI Data Register

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
- **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 + 1** 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 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

Summary

- Now you understand the I2C protocol, its connections and its frames
- Arbitration handles the multi-master communication
- An external EEPROM may be interfaced using I2C protocol