Authors: Burners Team (Ahmed Salah, Basem Moufreh, Hassan El Gabass, Hazem El Morshedi, Mohamed Safwat).
Customer: NTI
Instructor: Mahmoud Ali, Ahmed Abd El Reheem.

Software Requirement Specification (SRS) for I2C Communication Protocol

1. Introduction

The **I2C (Inter-Integrated Circuit)** protocol is a widely used communication standard for connecting multiple integrated circuits within a system. It allows efficient data exchange between devices using only two bus lines: the serial data line (SDA) and the serial clock line (SCL). This document outlines the requirements for implementing I2C communication in our project.

2. Purpose

The purpose of this SRS is to define the functional and non-functional requirements for the I2C communication protocol. It serves as a guide for developers, testers, and stakeholders involved in the project.

3. Scope

The I2C protocol will be used to facilitate communication between various components within our system. This includes sensors, actuators, memory devices, and other peripherals.

4. Requirements

4.1. Functional Requirements

[SRS_I2C_200] - I2C Communication Protocol

The system shall support the standard I2C communication protocol, as specified in the I2C specification [insert version/reference].

[SRS_I2C_201] - Bus Speed Configuration

The system shall provide the capability to configure the I2C bus speed, allowing adjustment to meet specific communication requirements.

[SRS_I2C_202] - Multi-Master and Slave Support

The system shall support both multi-master and slave device operations on the I2C bus, enabling various devices to act as masters and communicate with one another.

[SRS_I2C_203] - Data Transmission

The I2C interface shall facilitate the reliable transmission of data to and from connected I2C devices.

[SRS_I2C_204] - Error Handling

The system shall incorporate robust error handling mechanisms to detect, report, and manage I2C communication failures effectively.

[SRS_I2C_205] - Bus Speed

The I2C interface shall operate within a range of bus speeds from [Insert Minimum Speed] to [Insert Maximum Speed], ensuring compatibility with various devices.

[SRS_I2C_206] - Device Limit

The system shall support a maximum of [Insert Maximum Number] I2C devices connected to the bus simultaneously.

[SRS_I2C_207] - Latency

The I2C interface shall respond to I2C commands with a maximum latency of [Insert Maximum Latency] to ensure timely communication.

[SRS_I2C_208] - Hardware Compatibility

The I2C interface shall be compatible with hardware components supporting the I2C standard, including the physical layer, voltage levels, and connectors.

[SRS_I2C_209] - API and Documentation

The system shall provide well-documented Application Programming Interfaces (APIs) and comprehensive documentation to allow developers to interact with the I2C interface effectively.

4.2 Non-Functional Requirements

[SRS_I2C_2010] - Reliability

The I2C interface shall be available for communication with a minimum uptime of 99.9% to ensure consistent and reliable operation.

[SRS_I2C_211] - Compatibility

The I2C interface shall be compatible with various operating systems and platforms, ensuring broad usability.

[SRS_I2C_212] - Scalability

The system shall support the addition of new I2C devices without the need for extensive modifications or disruptions to existing devices.

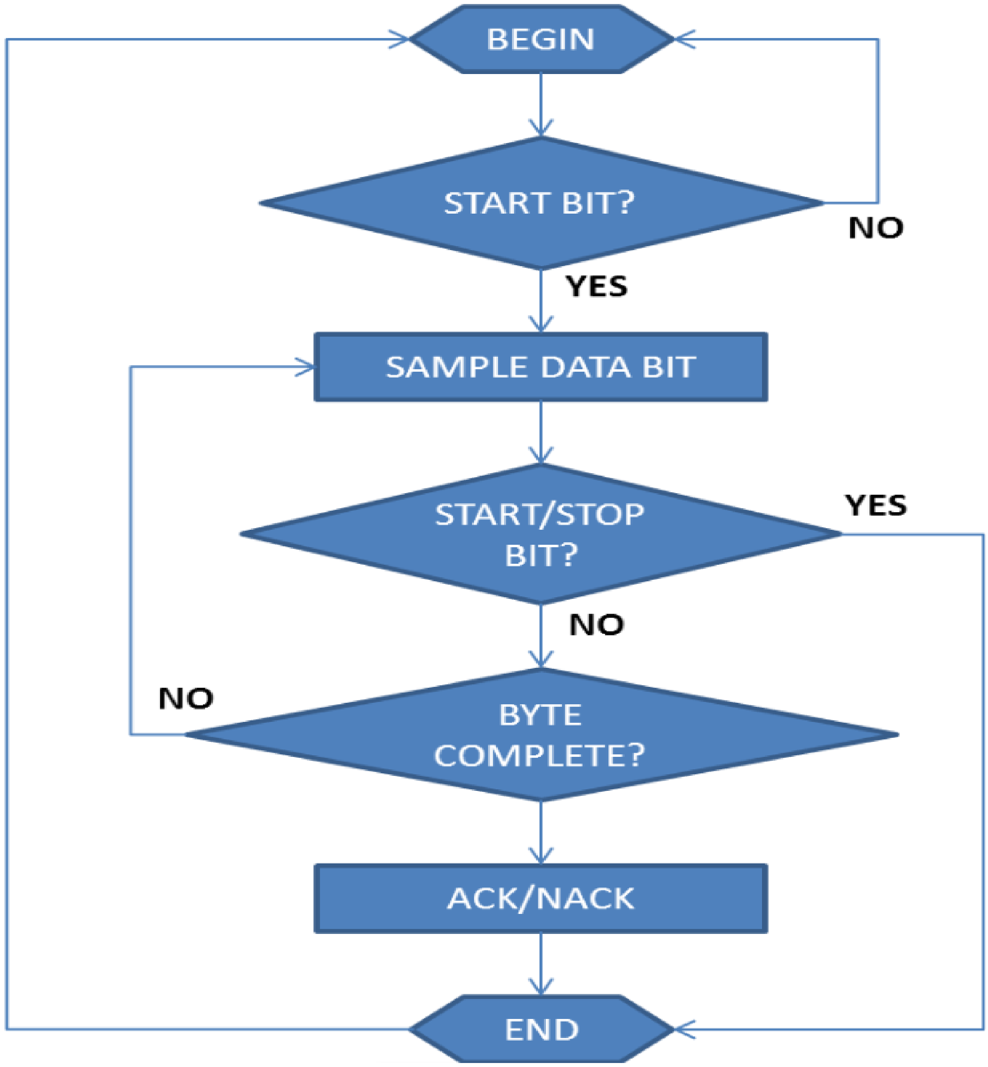
[SRS_I2C_213] - Usability

The I2C interface shall feature a user-friendly and intuitive design, promoting ease of use and navigation for all users.

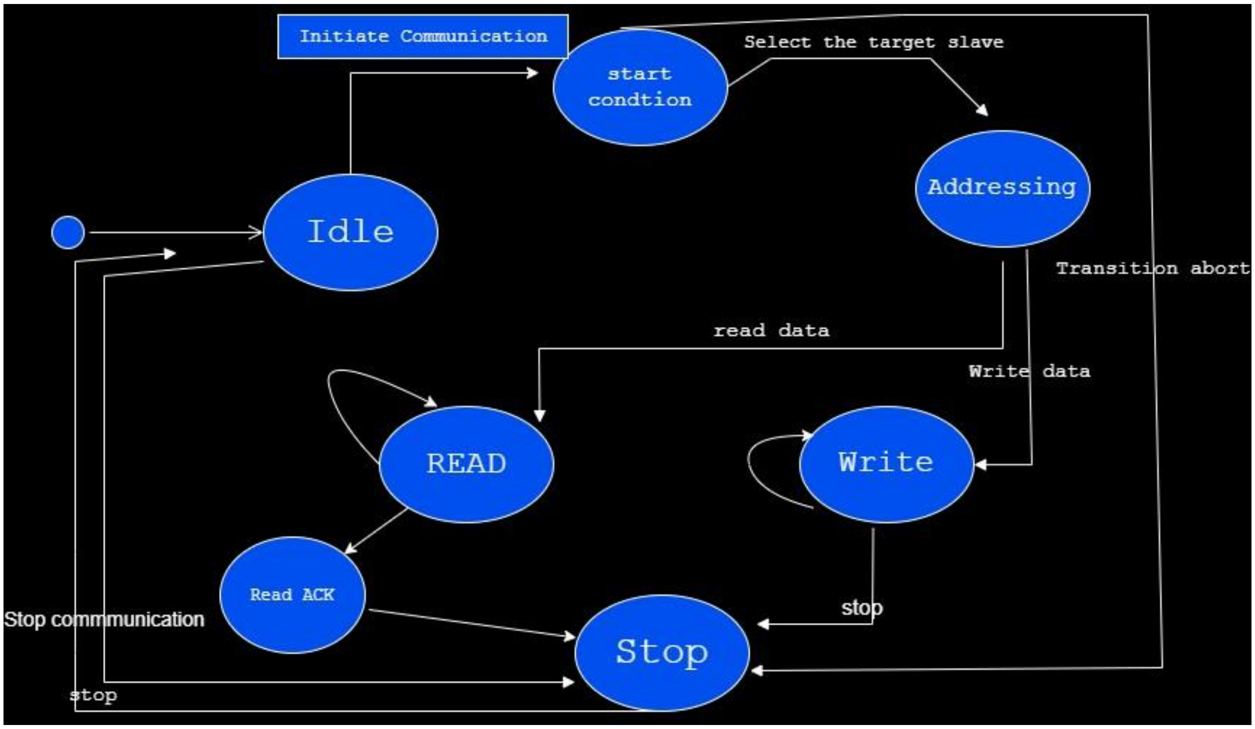
5. Constraints

- The I2C protocol operates at specific voltage levels (e.g., 3.3V, 5V).
- The maximum bus capacitance and pull-up resistor values must be within specified limits.

6. flowchart for the I2C



7. State Machine



The state machine includes the following states:

- **Idle State**: The initial state where the system is waiting for a new I2C transaction to begin or to abort an ongoing transaction.
- **Start Condition State**: Entered upon triggering a Start condition, indicating the initiation of a new I2C transaction.
- **Addressing State**: The state for selecting the target slave device by sending its address and specifying the read or write operation.
- **Write State**: Used for sending data to the selected slave device during a write operation.

- **Read State**: Engaged for reading data from the selected slave device during a read operation.
- **Read-NAK State**: A substate of Read State used to indicate the end of a read operation with a Not Acknowledged (NACK) signal.
- **Stop State**: Entered upon triggering a Stop condition, marking the conclusion of an I2C transaction.

4. Transition Descriptions

The state transitions are defined as follows:

- **Start Transition (Idle to Start Condition)**: Initiates an I2C transaction by sending a Start condition, marking the beginning of a communication.
- **Stop Transition (Idle to Stop)**: Aborts the ongoing I2C transaction by generating a Stop condition, returning to the Idle state.
- **Addressing Transition (Start Condition to Addressing)**: After Start condition, selects a target slave device by sending its address and specifying the operation type.
- **Stop Transition (Start Condition to Stop)**: Aborts the communication, generating a Stop condition and returning to the Idle state without addressing any specific slave.
- **Write Transition (Addressing to Write)**: Continues communication by sending data to the selected slave device during a write operation.
- **Read Transition (Addressing to Read)**: Initiates a read operation by generating a repeated Start condition, transitioning to the Read state.
- **Write Transition (Write to Write)**: Sequentially sends data bytes to the selected slave device during a write operation.

- **Stop Transition (Write to Stop)**: Concludes the write operation by generating a Stop condition, returning to the Idle state.
- **Read Transition (Read to Read)**: Continues reading data bytes from the slave device and sends ACK signals for more data, maintaining the Read state.
- **Read-NAK Transition (Read to Read-NAK)**: Receives a data byte from the slave and sends a NACK signal, indicating the end of the read operation and proceeding to the Stop state.
- **Stop Transition (Read-NAK to Stop)**: Concludes the read operation with a Stop condition, returning to the Idle state.