Software Requirements Specification (SRS)

Serial Peripheral Interface (SPI)

Authors: Burners Team (Ahmed Salah, Basem Moufreh, Hassan El Gabass, Hazem El Morshedi, Mohamed Safwat).

Customer: NTI

Instructor: Mahmoud Ali, Ahmed Abd El Reheem.

Table of Contents

1 Introduction 2

2 System Overview 3

3 Functional Requirements 4

4 Non-Functional Requirements 5

5 State machine 6

6 Sequence diagram 8

National Telecommunication Institute

1 Introduction

The Serial Peripheral Interface (SPI) protocol is a widely used synchronous serial communication interface that enables the exchange of data between multiple devices in a master-slave configuration.

The primary objective of this Software Requirements Specification (SRS) document is to outline the functional and non-functional requirements for the development of a software system that supports the SPI protocol. By defining these requirements, the document aims to provide a comprehensive understanding of the system's behaviour, features, and constraints.

a. Purpose: The purpose of this Software Requirements Specification (SRS) document outlines the functional and non-functional requirements necessary for the successful implementation of SPI communication between master and slave devices.

b. The scope: This SRS document encompasses the software system's requirements related to SPI protocol support. It includes the configuration,

operation, error handling, and communication aspects of SPI for both master and slave devices

c. Definitions, Acronyms, and Abbreviations:

SPI: Serial Peripheral Interface

2 System Overview

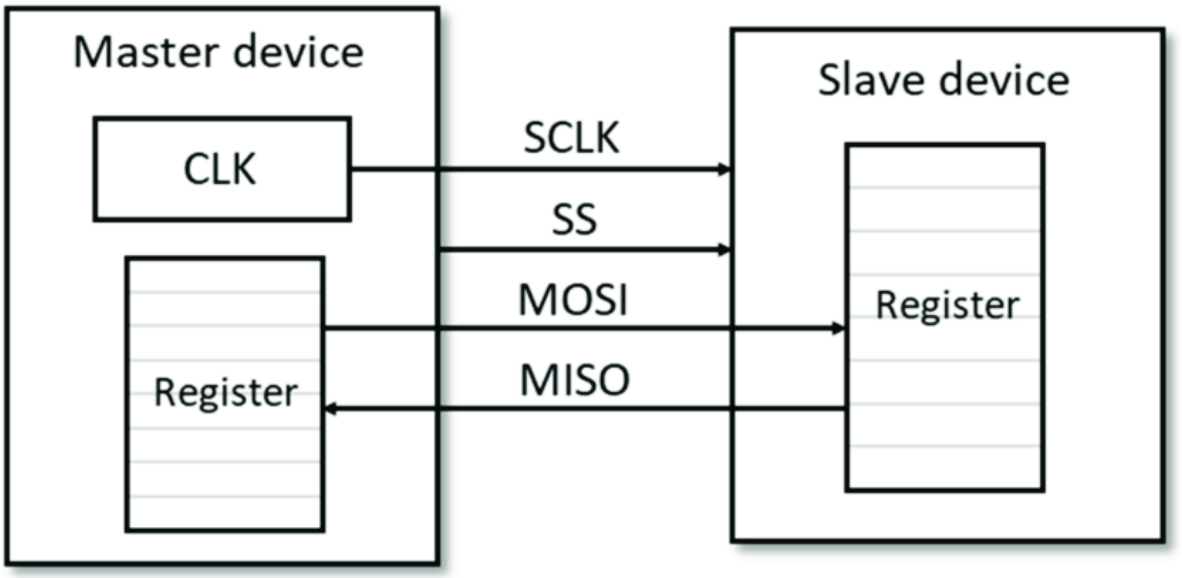
The software system supporting SPI protocol operates within the context of a larger digital system. It interacts with the SPI hardware interface provided by the microcontroller or other relevant devices. The SPI protocol enables communication between a master device that initiates data transfer and one or more slave devices that respond to the master's requests.

a. Product Features

The software system supporting SPI protocol should include the following features:

- SPI configuration: Ability to configure the SPI interface parameters, such as clock frequency, data order, and data mode.
- SPI master operations: Support for transmitting data from the master to slave devices and receiving data from the slaves.
- SPI slave operations: Support for receiving data from the master and transmitting data in response.
- Error handling: Ability to detect and handle errors, such as data corruption or communication timeouts.

b. Context Diagram



3 Functional Requirements

[SRS_SPI_100]: SPI Initialization

The software system should provide the capability to configure the SPI interface based on the following parameters:

- Clock frequency: The system should allow setting the clock frequency for SPI communication.
- Data order: The system should support both MSB (Most Significant Bit) and LSB (Least Significant Bit) data order.

[SRS_SPI_101]: SPI Master Operations

The software system should support SPI master operations, including the following functionalities:

- Transmit data: The system should enable the master device to transmit data to the slave devices connected via SPI.
- Receive data: The system should allow the master device to receive data from the slave devices during SPI communication.

[SRS_SPI_102]: SPI Slave Operations

The software system should support SPI slave operations, providing the following functionalities:

- Receive data: The system should enable the slave device to receive data from the master device during SPI communication.
- Transmit data: The system should allow the slave device to transmit data in response to the master's requests.

[SRS_SPI_103]: Data Transmission

The module shall provide a function to send data through the SPI.
The function shall accept a buffer of data and its length as parameters.
The function shall transmit the data synchronously based on the configuration.
The module shall handle any errors that occur during transmission, such as buffer overflow or transmission failure.

[SRS_SPI_104]: Error handling

The software system should incorporate error handling mechanisms for SPI communication, including the following capabilities:

- Error detection: The system should be capable of detecting errors, such as data corruption or communication timeouts.
- Error reporting: The system should provide appropriate error reporting mechanisms to notify the user or application about encountered errors.

4 Non-Functional Requirements

[SRS_SPI_120]: Performance

- The software system should provide efficient and reliable SPI communication with minimal latency.
- The system should support SPI communication at various clock frequencies and handle data transfers within the specified timing constraints.

[SRS_SPI_121]: Reliability

- The software system should ensure the accurate and reliable exchange of data between the master and slave devices over the SPI interface.
- The system should handle data integrity, error detection, and recovery to maintain reliable communication.

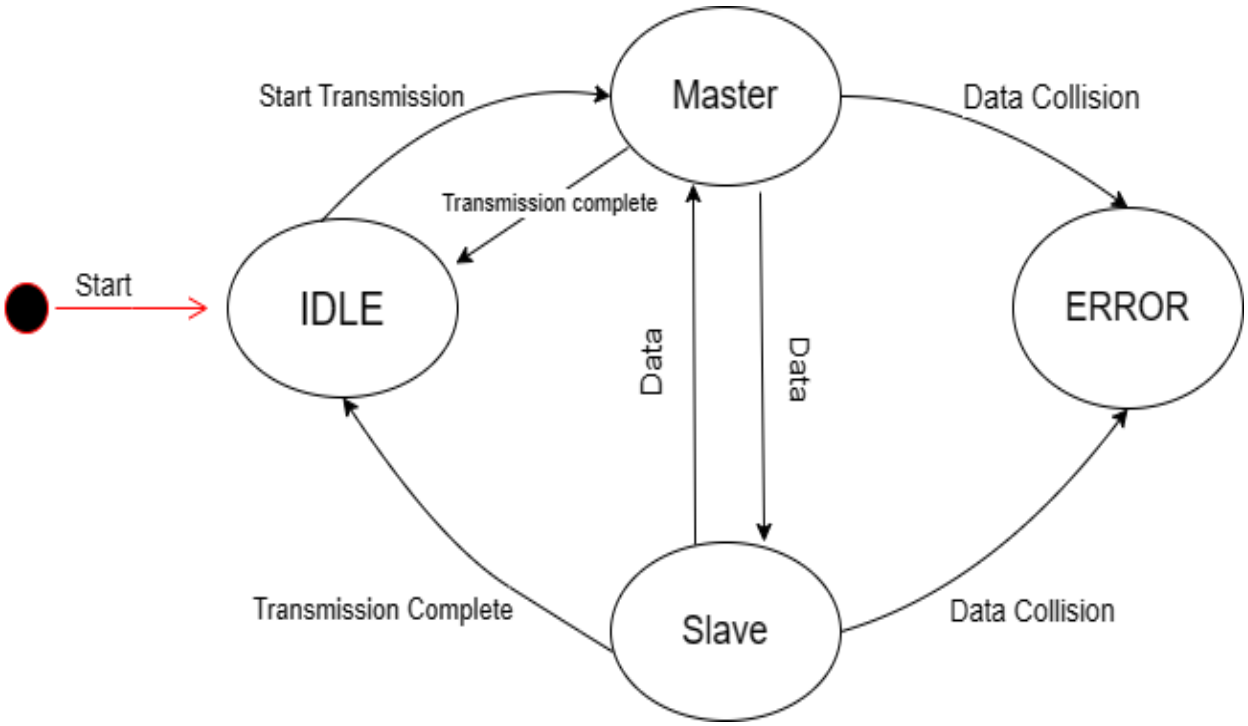
[SRS_ SPI _120]: Portability

The UART module shall be designed to be portable across different microcontroller platforms and architectures. The module shall be easily adaptable to different operating systems and development environments.

[SRS_ SPI _120]: Security

- The software system should incorporate security measures, such as data encryption or authentication, if required by the application or system.

5 State machine


Master
Start Transmission
Data Collision
Transmission complete
Start
IDLE
ERROR
Data
Data
Transmission Complete
Slave
Data Collision

6 Sequence diagram