Application Requirements for Atmega32 Controller Project

Overview:

1. The application is intended to send characters from a computer to the Atmega32 controller using UART protocol. The received characters are concatenated to create a string that can be displayed on an LCD connected to another Atmega controller using either SPI or I2C protocol based on a pre-defined macro value. The application should also have two push buttons that control the string concatenation and clearing of the last character sent.

Functional Requirements:

2. The application should be able to:

2.1 Receive characters from the computer through UART protocol and store them in a buffer or any data structure.

2.2 Concatenate the received characters in the buffer to form a string.

2.3 Clear the last character sent from the string when the first push button is pressed.

2.4 Send the string to another Atmega controller using either SPI or I2C protocol based on a predefined macro value when the second push button is pressed.

2.5 Display the string on an LCD connected to the other Atmega controller.

2.6 Handle any errors that may occur during communication.

Technical Requirements:

3. The application should:

3.1 Be developed using Atmel Studio or any Preferred IDE.

3.2 Be written in the C programming language.

3.3 Use UART protocol for communication between the computer and the Atmega32 controller.

3.4 Use either SPI or I2C protocol for communication between the Atmega32 controller and the other Atmega controller based on a pre-defined macro value.

3.5 Use a buffer to store the received characters.

3.6 Use push buttons to control the string concatenation and clearing.

3.7 Use an LCD module that is compatible with the other Atmega controller.

3.8 Be tested thoroughly to ensure that it meets the requirements and works correctly.

4. Protocol Requirements:

4.1 UART Protocol:

- Baud Rate: The baud rate for the UART communication between the computer and the Atmega32 controller should be set to a value that both devices can support. A common baud rate for UART communication is 9600 bps.
- Data Format: The data format for UART communication should be set to 8 data bits, no parity, and 1 stop bit (8N1).
- Hardware Chip: The UART signals will use a TTL-to-USB converter chip, such as the FT232RL, to communicate with the computer's USB port.

4.2 SPI Protocol:

- Clock Rate: The clock rate for the SPI communication should be set to a value that both devices can support. A common clock rate for SPI communication is 1 MHz.
- Data Format: The data format for SPI communication should be set to 8 data bits and no parity.

4.3 I2C Protocol:

- Clock Rate: The clock rate for the I2C communication should be set to a value that both devices can support. A common clock rate for I2C communication is 100 kHz.
- Data Format: The data format for I2C communication should be set to 8 data bits and no parity.

Macro Option:

5. The application should provide a predefined macro value that allows the user to choose between using