Comparison And Selection of Commonly Used Communication Protocols in Measurement and Control Instruments

Chenyu Zhuang*

Department of Xiamen University Malaysia, Xiamen University, Kuala Lumpur, 13600, Malaysia

* Corresponding Author Email: EEE2109306@xmu.edu.my

Abstract. In the contemporary landscape of economic globalization, the salience of the field of measurement and control has progressively intensified. Simultaneously, the utilization of serial bus technology within instrumentation has witnessed a considerable proliferation. Consequently, the judicious selection of an appropriate serial bus protocol has assumed paramount importance. It is incumbent upon practitioners to consider diverse factors, including the communication principles, performance attributes, and interfacing requirements, when making their choice, to ensure the efficacious and stable operation of their systems. This investigation employs a methodology that amalgamates a comprehensive literature review with experimental comparative analyses to ascertain the optimal suitability of Universal Asynchronous Receiver/Transmitter (UART), Inter-Integrated Circuit (I2C), and Serial Peripheral Interface (SPI) communication protocols across distinct application scenarios. Subsequent to a thorough comparative scrutiny, this study ultimately elucidates the respective merits and demerits of the UART, I2C, and SPI communication protocols, in tandem with their most judicious application contexts. This research endeavor serves to furnish engineers with valuable insights for the judicious selection of serial bus protocols.

Keywords: Communication protocol; Performance; Application scenarios.

1. Introduction

In the ever-evolving landscape of modern communication systems, communication protocols have assumed a progressively integral role within the realm of communication technology. The primary focus of this investigation centers on three prevalent communication protocols: UART, I2C, and SPI, which find extensive application within the domain of measurement and control instruments. These protocols serve as pivotal conduits for data transmission and hold widespread relevance in critical domains, including embedded systems. To cater to the diverse demands of various application environments, it becomes imperative to contemplate a spectrum of factors, encompassing not just performance and stability, but also energy efficiency, system intricacy, and related considerations [1].

In the field of measurement and control instruments, UART, I2C and SPI are widely used by engineers as the main communication protocols. They allow reliable data transmission between devices and promote interoperability between hardware components. They are used for serial communication, peripheral connection, sensor control and other scenarios. They are the core communication mode of embedded systems. As a common communication protocol, they are widely used in electronic devices and embedded systems, promoting interoperability between hardware components and being the core communication mode of embedded systems. However, the performance of different communication protocols in different application scenarios has its own advantages and disadvantages, so it is necessary to deeply compare the performance characteristics of different communication protocols to ensure the optimal pairing of communication protocols and application scenarios.

This study expects to find out the differences between UART, I2C and SPI communication protocols in terms of performance, data transmission rate and system complexity, summarize the applicability of the three communication protocols in different application scenarios, and provide system comparison and selection guidelines for measurement and control instrument designers.

2. Theoretical basic analysis

2.1. UART Communication Protocol

2.1.1 The main features of UART communication protocol

The Universal Asynchronous Receiver/Transmitter (UART) is an asynchronous communication protocol that achieves synchronization between transmitter and receiver devices without relying on clock signals. In this protocol, the transmitter derives its timing information from the bit stream rather than an external clock signal. Conversely, the receiver employs an internal clock signal to sample the incoming data and ensures synchronization by maintaining a consistent baud rate for both devices. This design obviates the need for simultaneous transmission of clock and data signals by the UART transmitter, ensuring accurate alignment of data flow [2].

2.1.2 The working principle of UART communication protocol

UART usually uses two data lines, one for sending and the other for receiving, to connect to the peer device through the serial port for data transmission. The sending end converts the data to be transmitted into serial data, sets the corresponding start bit, data bit, check bit and stop bit according to the communication protocol, and transmits data at an adapted baud rate [3]. The receiving end receives data and processes the data according to the communication protocol.

2.2. SPI Communication Protocol

2.2.1 The main features of SPI communication protocol

Serial Peripheral Interface (SPI) is a synchronous communication protocol. The sending end and the receiving device transmit data synchronously through a shared clock signal, which is synchronized. Data transmission is usually full duplex, allowing data to be sent and received simultaneously. Its main device needs to set a clock signal, and the clock rate can be adjusted as needed, with high flexibility.

SPI has a master-slave architecture, and one master device communicates with one or more slave devices. The master device is usually a microcontroller or processor and is responsible for controlling the clock and chip select signals. The slave device is a peripheral device and is responsible for responding to the instructions of the master device and transmitting data.

2.2.2 The working principle of UART communication protocol

The master device selects the corresponding slave device and selects the communicating slave device by pulling down the corresponding chip select (CS) signal. The master device generates the CLK (clock signal) and transmits one bit of data in each clock cycle. Depending on the selected transfer mode, the master and slave devices acquire data on the clock edge. At the same time, the master device sends data from the master-out slave-in (MOSI) line to the slave device and receives data from the slave device, and then passes the master-in-slave-out (MISO) [4].

After the communication operation is completed, the master device will release the chip select signal and switch to other slave devices for communication or end communication. The SPI protocol is usually used for short-distance high-speed data transmission communications and is widely used in embedded systems.

2.3. I2C Communication Protocol

2.3.1 The main features of I2C communication protocol

I2C is used for half-duplex communication and transmits data through clock lines and data lines. The clock signal for synchronous data transmission uses the clock line, supply chain logistics (SCL), while the actual data transmission uses the serial data line (SDA) [5]. In communication, data is divided into frames, and each frame includes a start bit, eight data bits, a check bit and a stop bit. Like SPI communication protocol, the clock speed of the I2C bus can be adjusted according to the application scenario.

2.3.2 The working principle of I2C communication protocol

The initiation of communication in the I2C protocol involves the master controller, which commences the process by emitting a start condition signal onto the bus. Within this signal, the master controller includes the address of the target device and the respective read and write bits, signifying the intended direction of communication. While this operation transpires, the SCL line is dedicated to transmitting clock signals, and the SDA line sequentially transmits data, bit by bit [6]. Notably, the initiation and termination of the communication cycle are denoted by specific conditions: the negative edge of SDA within the context of a high SCL level signifies the commencement, while the rising edge of SDA within the same high SCL level indicates cessation [7]. During other phases of signal transmission, the SDA signal remains unaltered when the SCL level is high. Specifically, the Start segment, which marks the commencement of data transmission, occurs when the SDA line shifts from high to low while the SCL level remains high. Conversely, the Stop segment, symbolizing the conclusion of communication, transpires when the SDA line transitions from low to high under the continued presence of a high SCL level. The graphical representation of these SDA and SCL conditions, for both initiation and termination, is depicted in Figure 1.

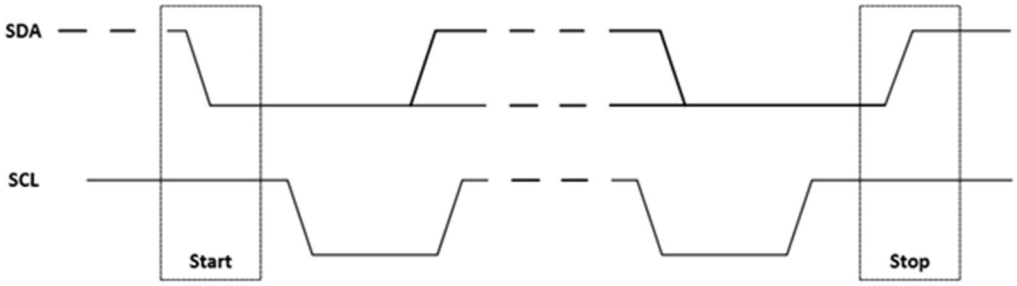

Fig. 1 SDA and SCL conditions to start and stop transmission signals [7]

The selected device responds to the master controller and, with the aid of a clock signal, transmits data over the data line. The master controller can choose to read or write more data, sending a stop condition to end the communication if needed. If there are other devices that require communication, the master controller can resend the start condition and select another device.

3. Verilog simulation

3.1. Experimental principle

3.1.1 UART communication protocol experimental principle

The loop module is used to generate enable signals and data, and then send data cyclically according to the busy status of the serial port sending module. Its inputs include the sending enable signal and data to be sent, as well as the sending status of the serial port sending module. The module's output includes the serial transmit port [8].

In the UART communication protocol simulation, the sending module is used to receive the data to be sent and the sending enable signal, control the clock according to the baud rate, send data bit by bit and generate start bits and stop bits to realize UART sending. The output of this module includes the serial port sending port and sending status flag [9].

The receiving module receives data from the UART receiving port and detects the start bit, count data bit and stop bit, and finally outputs the received data and reception completion flag. The inputs to this module include the UART receive port, while the outputs include the receive completion flag, receive data, and other control signals related to the receive process. The top-level module is used to integrate different UART modules and realize UART data loopback. Inputs to this module include the external system clock, system reset signal, and UART receive port.

3.1.2 SPI communication protocol experimental principle

In the SPI simulation experiment, the writing module is used to control the operation of writing data to the SD card. It receives the clock signal, reset signal, SPI serial input data signal of the SD card, and the user's write command and data. The module implements the SPI communication protocol internally, including sending write commands, sending data, sending CRC verification, and processing the response of the SD card. The module also generates a write data request signal and a write data busy signal to notify the external system when the next block of data can be sent.

The reading module is used to control the operation of reading data from the SD card. It receives the clock signal, reset signal, SPI serial input data signal of the SD card, and the user's read command. The SPI communication protocol is implemented inside the module, including sending read commands, receiving data, receiving CRC check, and processing the response of the SD card [10].

The top-level module is used to coordinate and control the read and write operations of the SD card. It receives the clock signal, reset signal, SPI serial input data signal of the SD card, and the user's read and write commands and data. This module selects the signal connected to the SD card interface according to the user's operation, so that the port signal is connected to the initialization module signal before the SD card initialization is completed.

3.1.3 I2C communication protocol communication protocol experimental principle

For the I2C communication protocol simulation experiment, the read-write module communicates with the EEPROM through the I2C communication protocol to simulate the process of writing and reading data. Its main function is to communicate with EEPROM and write or read data according to the I2C communication protocol. By simulating this module, the read and write performance of EEPROM can be tested, including read and write speed and correctness. You can simulate writing and reading different data and observe the simulation results to determine the performance characteristics of the EEPROM.

The top-level module is used to connect the EEPROM read-write module and the I2C communication protocol driver module. Pass the read and write operation request to the EEPROM read and write module and process the status and response information from the I2C communication protocol driver module [11]. By simulating this module, the functions of the entire EEPROM reading and writing system are tested, including starting and stopping reading and writing operations and handling error conditions.

The driver module is used to implement communication between the I2C communication protocol and EEPROM, including state machine, clock frequency division, data sending and receiving, and response processing. Through simulation, communication scenarios of different I2C communication protocols can be simulated and the performance of I2C communication, including clock frequency, data transmission speed and response processing, can be tested to evaluate the performance and correctness of the I2C driver module.

3.2. Simulation design and results

In order to more intuitively compare the performance differences of UART, SPI and I2C communication protocol, I used ModelSim to conduct Verilog simulation of the three communication protocols. Figure 2 shows the simulation results of the three communication protocols.

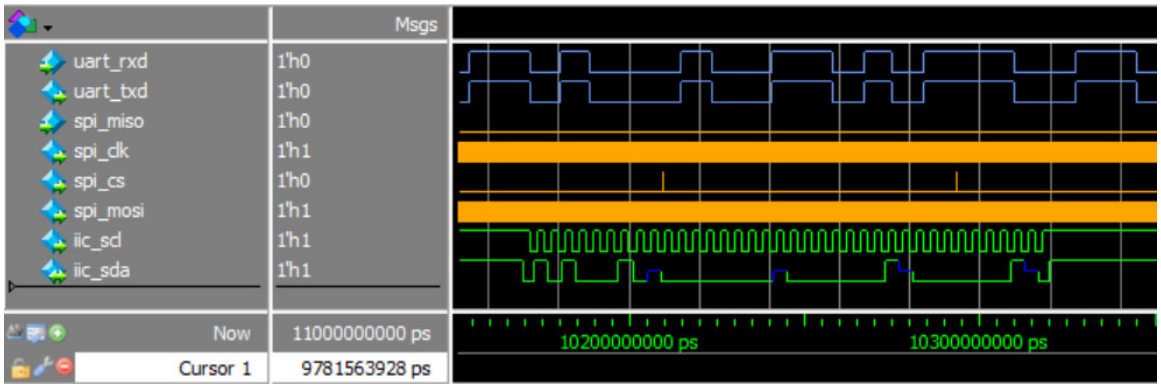

Fig. 2 Verilog simulation waveform (Photo/Picture credit: Original)

The blue waveforms are the sending data (uart_txd) line and receiving data (uart_rxd) line of the UART communication protocol respectively. The orange waveforms are the master input and slave output (spi_miso) lines, clock (spi_clk) lines, chip select (spi_cs) lines and master output and slave input (spi_mosi) lines of SPI communication protocol respectively. The green waveforms are the clock (iic_scl) lines and data (iic_sda) lines of I2C communication protocol respectively.

In Verilog simulation, for UART communication protocol, the input and output device sides are set to the same baud rate, so the sending and receiving data have the same timing. It can be seen from the simulated waveform that the UART communication protocol's sending data and receiving data are completely consistent.

It can be seen from the simulation of the SPI communication protocol that its serial data can be sent and received at the same time, and the clock can be changed by changing the frequency. The main module of the SPI communication protocol and the peripheral clock phase and polarity that communicate with it are completely consistent.

For I2C communication protocol, When the bus is idle, both the clock and data lines are high. When starting transmission, SCL remains high. After SDA changes from high level to low level, a delay of several microseconds occurs, and then SCL changes to low level. When SCL is at high level, SDA will transition from low to high.

It can be seen from the simulation results that SPI communication protocol has the highest rate and UART communication protocol has the lowest rate.

3.3. Performance analysis

UART communication protocol has only two data lines, used for sending and receiving. And UART communication protocol does not require a clock, it just needs to be consistent with the device-side rate. The counterpart of SPI and I2C communication protocol mainly corresponds to the memory configuration interface or memory chip, such as SD card. Moreover, the clocks of SPI and I2C communication protocol need to correspond to the data lines to represent timing, and the data transfer rate needs to be aligned through the clock. Therefore, UART is the simplest communication protocol.

In terms of device connection, UART communication protocol can only be connected to one device, that is, a one-to-one connection. SPI communication protocol can connect to several devices, but the number is limited. In an ideal world, since the I2C communication protocol device address has eight bits, there are 255 possible combinations. Each device has a unique address by which the host controller can select the device to communicate with. However, due to limitations in current, voltage, drive, etc. and the difficulty of hardware design, only three to four devices can be connected in actual situations.

The SPI communication protocol transceiver end has synchronization of data transmission under the shared clock signal, so under the premise of the parameters set in this experiment, the SPI communication protocol transmission rate is the fastest. The UART communication protocol has the slowest speed, the data line is the longest among the three, and the fault tolerance rate is also the highest. I2C communication protocol is generally on the same board, so the data line length is limited.

In addition, since SDA will transition from low to high when SCL is at a high level, the stop signal is a level transition timing signal rather than a level signal. The performance comparison and applicable scenarios is shown in table 1.

Table 1. Performance comparison and applicable scenarios

	UART	SPI	I2C
Speed	Low	High	Medium
Pin	Two wires	Multi-line	Two wires
Peer	Peer to peer	Support multiple slave devices	Support multiple slave devices
Hardware complexity	Simple	Complex	Medium
Distance	Long	Short	Short
Mode	Simplex or duplex	Full duplex	Simplex or duplex
Application scenarios	1. Long distance communication. 2. simple sensor	1. High speed data transmission. 2. Memory interface. 3. Multi-sensor configuration	1. Multi-slave device communication. 2. Low power applications

The parameters such as the transmission distance set in the experiment are feasible under ideal circumstances. In practice, problems such as line length limitations, peer equipment configuration limitations, and transmission errors may be faced. It needs to be measured and verified on the board to solve it.

4. Conclusion

In summary, this study conducts a multi-faceted comparison and evaluation of UART, I2C and SPI communication protocols, providing important reference and guidance for designers of measurement and control instruments. Based on our experimental results and analysis, the following conclusions can be drawn.

First of all, the UART communication protocol has good stability and reliability. Therefore, the UART communication protocol is an ideal choice for applications that require stable long-distance data transmission, especially for simple sensor applications.

Secondly, the SPI communication protocol has excellent performance in applications with large-scale data transmission or fast response. Therefore, in the design of high-demand measurement and control instruments, the SPI communication protocol should be highly valued and widely used in high-speed data transmission, memory interfaces and multi-sensor configuration.

The I2C communication protocol excels in multi-slave communication and low-power applications, making it the first choice for connecting multiple peripherals or sensors. In a multi-sensor configuration, the capability of the I2C communication protocol can enable collaborative work between different sensors and provide more information to the system.

The above conclusions may help designers of measurement and control instruments choose communication protocols in different application scenarios and help optimize system performance and complexity. It is hoped that this research can provide valuable guidance for future measurement and control instrument design and promote the continuous development and progress of measurement and control instrument technology.

References

[1] Zou L, Wang Z, Hu J, et al. Communication-protocol-based analysis and synthesis of networked systems: Progress, prospects and challenges. International Journal of Systems Science, 2021, 52(14): 3013-3034.

[2] Shi Wenqi, Gan Yujing, & Huang Guangming. Data acquisition IP core design based on NanEye 2D miniature image sensor. China Medical Equipment, 2019, 4(9): 55-58.

[3] Ingaleshwara S S. Heterogeneous architecture for reversible watermarking system for medical images using Integer transform based Reverse Contrast Mapping[J]. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 2021, 12(6): 308-315.

[4] H. Kareem and D. Dunaev. The Working Principles of ESP32 and Analytical Comparison of using Low-Cost Microcontroller Modules in Embedded Systems Design. 2021 4th International Conference on Circuits, Systems and Simulation (ICCSS), Kuala Lumpur, Malaysia, 2021: 130-135.

[5] Dinesh Sheoran. Assistant Professor, Department of ECE, MSIT, C-4, Janakpuri, New Delhi, India. Universal asynchronous receiver transmitter. 2020: 36-38.

[6] Wang Zhuangpeng, Xiao Bing, Liu Luoshi, & Ou Ben. Battery management system communication method based on I2C bus. Applied Science and Technology. 2020, 47(2): 48-52.

[7] C. Liu, Q. Meng, T. Liao, X. Bao and C. Xu. A Flexible Hardware Architecture for Slave Device of I2C Bus. 2019 International Conference on Electronic Engineering and Informatics (EEI), Nanjing, China, 2019, pp. 309-313.

[8] Peña E, Legaspi M G. Uart: A hardware communication protocol understanding universal asynchronous receiver/transmitter. Visit Analog, 2020, 54(4).

[9] Sharma P, Kumar A, Kumar N. Analysis of UART Communication Protocol. 2022 International Conference on Edge Computing and Applications (ICECAA). IEEE, 2022: 323-328.

[10] Trivedi D, Khade A, Jain K, et al. Spi to i2c protocol conversion using verilog. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). IEEE, 2018: 1-4.

[11] Rekha S, Reshma B, Dilipkumar N P, et al. Logically Locked I2C Protocol for Improved Security. International Conference on Communication, Computing and Electronics Systems: Proceedings of ICCCES 2019. Springer Singapore, 2020: 707-716.