Comparative analysis of intra-board synchronous serial communication interfaces

Predrag Petronijević, *Vlatacom Institute of High Technologies, Milutina Milankovica 5, 11070 Belgrade, Serbia*
Vladimir Kuzmanović, *Faculty of Mathematics, University of Belgrade*

***Abstract* — Designing custom-made hardware for special purposes is a challenging process. During the development, it is essential to take into consideration the required performance of the device, component availability on the market as well as the final price of the developed and assembled product. Almost every modern hardware consists of various sensors, memories, AD/DA converters and a microcontroller to control and manage the interaction off all those devices. Based on the purpose of the device being developed, the engineer has to make a decision on the components that will be used in the final product. For this decision to be justifiable, the engineer needs to have a very high level of knowledge regarding the intricate world of interfaces required to establish the intercommunication of the components inside the device. Modern sensors, memories and AD/DA converters usually require some form of a high-speed serial interface, synchronous or asynchronous. In this paper we will analyze the three most commonly used serial synchronous communication interfaces: I²C, SPI and SPORT. Also, we will explain the hardware and software properties and limits of every mentioned synchronous serial interface. Finally, the benefits and drawbacks of the chosen communication interfaces will be considered and conclusions drawn.**

***Index Terms* — computer engineering, embedded systems, sensors, synchronous serial communication**

I. Introduction

One aspect of designing new hardware is defining its application and the other aspect is defining a set of features the final product has to meet. The desired set of features can be divided into a set of operational and environmental limits, e.g. thermal resistance or voltage, and a set of desired performance characteristics, e.g. bandwidth or noise levels. This set of features limits the number of possible components that can be used in the design of the hardware. Even when limited with operational and performance characteristics, the choice of available hardware components is enormous due to a large number of manufacturers. Making the correct choice of hardware in order to meet the desired characteristics requires extensive knowledge [1-2]. One key decision to make is the choice of the right communication interface that will be used for intercommunication of the chosen components.

Predrag Petronijević is with Vlatacom Institute of High Technologies, Milutina Milankovica 5, 11070 Belgrade, Serbia (e-mail: predrag.petronijevic@vlatacom.com).

Vladimir Kuzmanović is with the Faculty of Mathematics, University of Belgrade, Studentski trg 16, 11000 Belgrade, Serbia (e-mail: vladimir_kuzmanovic@matf.bg.ac.rs).

Modern devices usually consist of various sensors, memories, AD/DA converters and many other components controlled by a microcontroller. This control is achieved by establishing intercommunication between the microcontroller and every component inside the device. In modern devices, this communication is digital and standardized to conform to one or more of the standard communication interfaces in use today [3-4].

Interfaces used today can be divided into categories based on the way the data is transferred between the devices. Two criteria can be used for this division. The first criterion is defined by the number of channels used in the transmission of the data. If the data is transmitted bit by bit in a specific order over a single channel, such transmission is called serial. If the data is sent as multiple bits at the same time over multiple channels, such transmission is called parallel. The other criterion is defined by the way the data is sent. If the data is sent in the form of a byte or a single character with start and stop bits added to the data, such transmission is called asynchronous because it does not require synchronization. If the data is sent in the form of groups or frames, such transmission is called synchronous because it requires synchronization between sender and receiver. Synchronous transmission is more reliable and full-duplex, while asynchronous transmission is half-duplex [5].

Inter-Integrated Circuit (I^2C) was discussed by Patel et al. [6], Lynch et al. [7] and Blum [8]. Wootton in [9] described the use of Serial Peripheral Interface (SPI) as a means of communication between the CPU and various peripheral devices. Gay in [10] described the properties of SPI and its operation was described by Dogan in [11]. SPI and I^2C were compared in [12-13]. SPI, I^2C and UART were analyzed in [14-15].

In this paper we will address and compare the three most commonly used synchronous serial communication interfaces for intercommunication between various devices. Besides the well-known and widely used I^2C and SPI protocols, we will also introduce Analog Devices proprietary SPORT protocol and perform comparative analyses of the three serial protocols. Section 2 of the paper introduces all three interfaces with their hardware and software properties and requirements. The next section analyses the benefits and drawbacks of these serial interfaces. Finally, section 4 draws conclusions on serial interfaces described in the paper.

II. SYNCHRONOUS SERIAL COMMUNICATION

A serial communication protocol in which data is sent as a continuous stream at a constant rate is described as synchronous serial communication. For communication to be called synchronous it is required that the clocks are synchronized in both the transmitting and receiving devices. The term synchronized refers to the clocks running at the same rate, which enables the receiver to sample the signal at the same intervals used by the transmitter. Synchronization of clocks permits the omission of start and stop bits. As a consequence, more information can be passed over a circuit per unit of time than with asynchronous serial communication.

Serial communication can be established via a communication channel or a computer bus. It is mostly used for long-distance communication and computer networks where parallel communication is impractical. The development of technology has made serial computer buses more common at shorter distances, mostly as a basis for cheap and simple intra-board communication between two or more integrated circuits on the same printed circuit board connected by signal traces and not external cables.

The three most commonly used synchronous serial communication protocols for intra-board communication are inter-integrated circuit (I^2C), serial peripheral interface (SPI) and Analog Devices synchronous serial peripheral port (SPORT).

A. Inter-Integrated Circuit (I^2C)

Inter-Integrated Circuit (I^2C) is a synchronous, multi-master, multi-slave, packet-switched and single-ended serial communication bus invented in 1982 by Philips Semiconductors. Today it is widely used for interfacing with lower speed peripheral integrated circuits from a microcontroller in short distance intra-board communication.

I^2C emphasizes design simplicity and low manufacturing costs over speed. It is usually used for accessing low-speed AD/DA converters, controlling small displays, reading diagnostic sensors, etc. I^2C enables the microcontroller to control a network of devices with just two general-purpose input-output pins and software. Many other serial protocols offer similar functionality but require more pins and signals to interconnect multiple devices [16].

Hardware requirements for establishing I^2C communication are rather simple. Two bidirectional open collector or open drain lines with typical voltages of +5V or +3.3V are required for connecting the devices. These two lines are called Serial Data Line (SDA) and Serial Clock Line (SCL). I^2C bus speed can range from 10 kbit/s to 5 Mbit/s depending on the revision of the protocol. The bit rate is defined for transfer between master and slave without taking into consideration any protocol overhead. The overhead includes a slave address and usually a register within the slave device and finally per byte acknowledge (ACK/NACK) bits. This makes the actual bitrate lower than the bitrate used would imply. High-speed I^2C is widely used in embedded systems, while lower speed version is used in personal computers.

The reference design is a bus with clock (SCL) and data (SDA) lines with 7-bit addressing to which the devices are connected. Devices connected to the bus are referred to as nodes. The number of nodes is limited by the address space and by the total bus capacitance of 400pF. This restricts communication distances to a few meters. In practice, I^2C is restricted to intra-board communication due to its relatively high impedance and low noise immunity which requires a common ground potential.

There exist two roles for the node on the bus: master and slave. The device is referred to as the master if it generates the clock and initiates communication with the slaves. The device is referred to as the slave if it receives the clock and responds when addressed by the master. The protocol supports multiple masters and multiple slaves on the same bus. Also, the roles of the device can be changed during its operation.

The protocol defines four modes of operation for a given device on the bus: master transmits, master receives, slave transmits and slave receives. Usually, each device on the bus will use a single role with two predefined modes of operation. Besides 0 and 1 data bits, the I^2C defines special signals which represent message delimiters. These signals are called START and STOP signals which are distinct from data bits. The communication between devices is as follows:

- The master is in master transmit mode and initiates the transmission by sending the START signal followed by a 7-bit address of the slave it wants to communicate with which is followed by a single bit designating whether the master wants to write to or to read from the slave.
- If the slave with the given address exists on the bus it responds with the ACK bit for that address. Then, the master continues to transmit either in transmit or receive mode according to the bit set while the slave continues in complementary mode.

The address and the data over the I^2C bus are sent in MSB mode. The START signal is a high-to-low transition of the data line (SDA) with the clock (SCL) line high. The stop signal is a low-to-high transition of SDA with SCL high. All other transitions of SDA take place with SCL low. The device which is in transmitting mode writes the data byte by byte to the SDA line. The device in receive mode sends the ACK bit after every byte. I^2C transmission may consist of multiple messages. The master terminates a message with a STOP signal if it is the end of the transaction. If the master wants to retain control of the bus for another message it sends another START signal.

I^2C physical layer is shown in Figure 1.

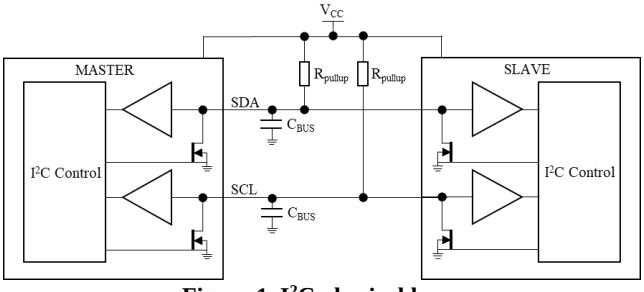

Figure 1. I^2C physical layer

B. Serial Peripheral Interface (SPI)

Serial Peripheral Interface (SPI) is a synchronous serial communication interface developed by Motorola in the mid-1980s [17]. It is used for short-distance communication in embedded systems. It is typically used for interfacing with memories, liquid crystal displays, sensors and AD/DA converters.

Devices communicating over SPI are organized as a master-slave architecture with a single master. The communication is achieved in full-duplex mode. The master device creates the frames for reading and writing. SPI supports multiple slave devices through selection with individual slave select lines. Sometimes, these lines are called chip select (CS) lines. The SPI bus specifies four logic signals:

- Serial clock (SCLK) – output from the master.
- Master Out Slave In (MOSI) – data output from the master.
- Master In Slave Out (MISO) – data output from the slave.
- Slave/Chip Select (SS/CS) – output from the master, active low.

For the communication to be established between devices, MOSI on a master device connects to MOSI on a slave device. Slave/Chip Select line is used instead of software addressing concept. Sometimes, MOSI on a slave device is labeled as Serial Data In (SDI) and MISO is labeled as Serial Data Out (SDO). This signal naming convention is used as an unambiguous way of labelling the pins of master and slave devices.

The SPI bus can operate with a single master device and one or more slave devices. Most slave devices have tri-state outputs so their MISO becomes high impedance when the device is not selected. This allows multiple slave devices to share common bus segments with each other.

For the communication to start, the master device has to configure the clock signal using a frequency supported by the slave. Then the master has to select the desired slave device with the logic level 0 on the appropriate SS/CS line. If the slave device requires a waiting period, the master device has to wait for at least that period of time before it starts issuing clock cycles on the SCLK line. During each cycle on the SCLK line, a full-duplex transmission occurs. The master sends a bit on the MOSI line and the slave reads it, while the slave sends a bit on the MISO line and the master reads it. This form of operation is maintained even when one-directional data transfer is intended.

Besides configuring the clock frequency, the master also needs to configure the clock polarity (CPOL) and clock phase (CPHA) with respect to the data. CPOL determines the clock polarity. CPOL value 0 defines a clock signal which idles at logic level 0 and each cycle consists of a pulse of 1. This translates to the leading edge being rising and the trailing edge is falling. CPOL value 1 defines the opposite. The clock idles at logic level 1 and each cycle consists of a pulse of 0. CPHA determines the timing of the data bits relative to the clock pulses. CPHA value 0 defines that the "out" side changes the data on the trailing edge of the preceding clock cycle, while the "in" side captures the data on the leading edge in the clock cycle. CPHA value 1 defines the opposite. The "out" side changes the data on the leading edge of the current clock cycle while the "in" side captures the data on the trailing edge of the clock cycle.

Finally, SPI supports word sizes that are not limited to 8-bit words but can range up to 32-bit words. Also, message size is arbitrary, as is its contents and purpose. The signal lines are shared between multiple devices, except for the slave select line which is unique per slave.

Its versatility, high speed and easy implementation coupled with board real estate savings compared to parallel buses have made it popular in many applications today. SPI interface is widely used in embedded systems for interfacing various sensors, control devices, memories and liquid crystal displays.

SPI physical layer is shown in Figure 2.

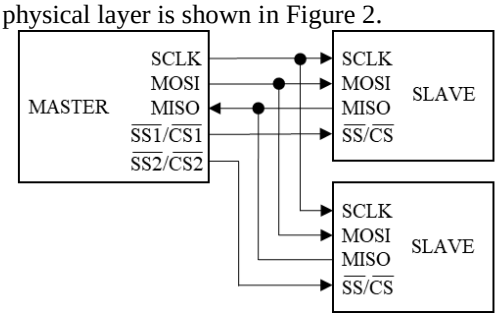

Figure 2. SPI physical layer

C. Synchronous Serial Peripheral Port (SPORT)

Synchronous Serial Peripheral Port (SPORT) is Analog Devices proprietary synchronous serial communication interface that supports a variety of serial data communication protocols. Key features of SPORT are continuously running clock and serial data words from 3 to 32 bits in length either most- or least-significant bit first. The protocol also supports two synchronous transmit and two synchronous receive data signals which double the total supported data stream. Finally, frames are synchronized with configurable synchronization signals [18].

For the SPORT interface to be established between two devices, the standard defines the following eight signals:

- Transmit Data Primary (DT0)
- Transmit Data Secondary (DT1)
- Transmit Clock (TSCLK)
- Transmit Frame Sync (TFS)
- Receive Data Primary (DR0)
- Receive Data Secondary (DR1)
- Receive Clock (RSCLK)
- Receive Frame Sync (RFS)

The values for clocks are independent and can be calculated by dividing the SCLK of the microcontroller with the correct value. The SPORT clocks are calculated with the following formula:

$$SPORTCLK = \frac{SCLK}{(2 \cdot (SPORTCLKDIV + 1))}$$

The smallest value the divisor SPORTCLKDIV can have is zero and the greatest value is 65535. TSCLK and RSCLK are

independent and thus can have different values of SPORTCLKDIV. Depending on the value of SCLK and SPORTCLKDIV, the clock values for SPORT can be as high as 60 MHz or as low as 1 kHz. By default, the primary transmit and receive channels are enabled while the secondary transmit and receive channels are disabled.

Frame sync signal can be divided into early frame sync and late frame sync. Early frame sync is active for one clock pulse and then deactivates. Once the signal has been deactivated, valid data will be available. Late frame sync signal frames valid data and is active for the length of time that valid data is available. The signal is deactivated once the word to transmit or receive is fully sent.

SPORT protocol is proprietary and is supported by a majority of Analog Device microcontrollers and various types of integrated circuits for numerous applications. Such applications range from AD/DA converters, sensors, memories, health applications, smart industries, etc. Also, with a range of clock and frame synchronization options, the SPORT interface allows a variety of serial communication protocols and provides a glueless hardware interface to many industry-standard data converters and CODECs [19-20].

SPORT physical layer is shown in Figure 3.

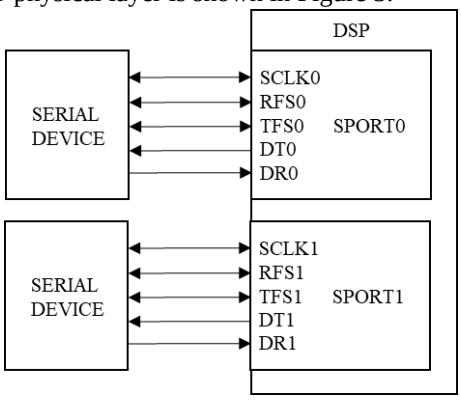

Figure 3. SPORT physical layer

III. COMPARATIVE ANALYSIS

I^2C, SPI and SPORT all are synchronous bidirectional serial interfaces with considerable differences. The first obvious difference is the number of signals needed to establish communication between devices. The signals and number of lines required for establishing communication with each interface are displayed in table 1.

Table 1. Signals required for establishing communication

I^2C	SPI	SPORT
SDA Serial Data	MOSI Master Out Slave In	DT Serial Data Transmit
SCL Serial Clock	MISO Master In Slave Out	DR Serial Data Receive
	SCLK Serial clock	TFS Transmit Frame Sync
	SS Slave select	RFS Receive Frame Sync
		TCLK Transmit Clock
		RCLK Receive Clock

Considering the number of signals it is obvious that SPI and SPORT are full-duplex, while I^2C is half-duplex. Also, one other property to note is that I^2C is a multi-master multi-slave interface, while SPI and SPORT are single-master multi-slave interfaces.

Data transfer should also be considered when choosing the protocol to be used in the final product. The limits for data transfer are displayed in table 2.

Table 2. Data transfer limits

I^2C	SPI	SPORT
100 kbit/s – 5 Mbit/s Predefined values depending on version	Depending on the implementation Usually in range n x MHz to 10n x MHz n – number of devices connected to a single master	SCLK/2 Mbit/s SCLK – processor clock frequency

The advantages of I^2C over SPI and SPORT are the ease of linking multiple devices and the fact that cost and complexity do not scale up with the number of devices. The limitation of I^2C is numerous. The first is its slave addressing scheme and its relatively low number of possible addresses which may lead to address collisions. One other limitation is the number of supported speeds which need to conform to a certain standard. Since I^2C is a shared bus there exists a possibility that a single device could hang the entire bus. This happens if any device holds the SDA or SCL lines low, which prevents the master from sending START and STOP signals and reset the bus. Also, starvation is possible where a slower device starves the bandwidth needed by faster devices and thus increases latencies when other devices are addressed. Taking all this into consideration it is advisable to use I^2C for communication with on-board devices that are accessed only occasionally with no need for low latencies and high-speed bidirectional communication.

The advantages of SPI over I^2C and SPORT are complete protocol flexibility with variable size words and arbitrary choice of message size, contents and purpose. Also, hardware interfacing is easy. Slaves do not need a unique address since they are addressed with a per slave chip select line and slave devices do not need precision oscillators since they use the master's clock. Disadvantages compared to I^2C are the increased number of pins required for communication and the lack of slave ACK which enables the master to transmit data to nowhere without knowing it. Also, SPI protocol supports only one master, does not have a formal standard so validating conformance is impossible and does not support dynamically adding nodes. Taking all this into consideration, SPI is applicable in situations where the data transfer is organized in packets of arbitrary size and full-duplex. Also, it is applicable when there are a number of slaves communicating with the same SPI modes, because frequent changes of SPI mode severely impact the performance of communication.

Compared to the other two protocols, the main advantage of SPORT protocol is the support for multichannel transmits and receives of up to 128 channels. Also, a wide selection of data sizes is also a benefit as is the programmable polarity of both frame sync signals and data receive and transmit clocks. Finally, significantly higher data rates and double-buffered data registers that allow continuous data stream are a big advantage compared to both SPI and I^2C. The main

disadvantages of SPORT are the fact that it is proprietary and supported only by Analog Devices products and that the complexity of supporting software components can be higher than that of competing schemes.

IV. Conclusion

In this paper we presented the three most commonly used synchronous serial protocols. The introduction showed that the engineer needs to have a broad knowledge regarding communication protocols to be able to make the right choice on the protocol to be used with respect to the desired operational and performance limits as well as to justify the proposed design. I^2C, SPI and SPORT are presented in detail and their properties, requirements and applications are discussed. Finally, the benefits and drawbacks of all three mentioned protocols are compared and analyzed which led to the conclusion on the suitability of the protocols in various scenarios. In the future, we intend to further research asynchronous communication protocols and their properties as well as inter-board communication protocols. We will focus on Controller Area Network (CAN) and Universal Asynchronous Receive Transmit (UART).

Acknowledgment

The research is founded by the Vlatacom Institute of High Technologies under project *#161 V155MM*.

References

[1] J. Staunstrup, W. Wolf, "Hardware/Software Co-Design: Principles and Practices," Springer Science & Business Media, 1997.
[2] P. Horowitz, W. Hill, "The Art of Electronics, 3rd edition," Cambridge University Press, 2015.
[3] J. Cowley, "Communications and Networking: An Introduction," Springer, 2007.
[4] IBM Corporation, "Data Communications Primer," Form C20-1668-0.
[5] J. Patrick, "Serial Protocols Compared," Embedded Staff, May 31, 2002, available at: https://www.embedded.com/serial-protocols-compared/.
[6] S. Patel, P. Talati, S. Gandhi, "Design of I2C Protocol," International Journal of technical innovation in Modern Engineering & Science, Vol 5, no. 3, pp. 741-744, 2019.
[7] K. M. Lynch, N. Marchuk, M. L. Elwin, "I^2C communication," Embedded Computing and Mechatronics with the PIC32, Newnes, 2016.
[8] J. Blum, "The I^2C Bus," Exploring Arduino; Tools and Techniques for Engineering Wizardry, 2nd Edition, 2019.
[9] C. Wootton, "Serial Peripheral Interface (SPI)," Samsung Artik Reference, Apress, Berkeley, 2016.
[10] W. W. Gay, "SPI Bus," Mastering the Raspberry Pi, Apress, Berkeley, 2014.
[11] I. Dogan, "Serial Peripheral Interface Bus Operation," SD Card Projects Using the PIC Microcontroller, Newnes, 2010.
[12] F. Leens, "An introduction to I^2C and SPI protocols," IEEE Instrumentation & Measurement Magazine, vol. 12, no.1, pp. 8-13, 2009.
[13] D. V. Gadre, S. Gupta, "Serial Communication: SPI and I2C," Getting Started with Tiva ARM Cortex M4 Microcontroller, Springer, 2017.
[14] A. Subero, "USART, SPI and I2C: Serial Communication Protocols," Programming PIC Microcontrollers with XC8, Apress, Berkeley, 2017.
[15] S. Shanthipriya, S. Lakshmi, "Design and verification of low speed peripheral subsystem supporting protocols like SPI, I2C and UART," ARPN Journal of Engineering and Applied Sciences, vol. 12, pp. 7368-7391, 2017.
[16] A.K. Oudjida, M.L. Berrandjia, R. Tiar, A. Liacha, K. Tahraoui, "FPGA Implementation of I2C & SPI Protocols: a Comparative Study," DOI: 10.1109/ICECS.2009.5410881, 2009 16th IEEE International Conference on Electronics, Circuits and Systems - (ICECS 2009), Yasmine Hammamet, Tunisia, 13-16. December, 2009.
[17] Motorola, Freescale, NXP, "SPI Block Guide v3.06," 2003.
[18] B. Prabhalika, M. Kiran Kumar, "Fpga implementation of Design and verification Synchronous serial port(S-PORT)," ISSN: 2321-9939, International Journal of Engineering Development and Research IJEDR, India, 2013.
[19] A. Vasudev Prabhugaonkar, J. Rayala, "Interfacing AD7676 ADCs to ADSP-21365 SHARC® Processors," Engineer-to-Engineer Note EE-248, Analog Devices, Rev 1, October 7, 2004.
[20] "Implementing UART Using the ADuCM3027/ADuCM3029 Serial Ports," Application Note AN-1435, Analog Devices, Norwood, MA, 2017.