DATA MONITORING AND ACQUISITION SYSTEM FOR THE I2C PROTOCOL

Mark Eduard GROSS, Dorin Marius PETREUȘ
Technical University of Cluj-Napoca, Cluj Napoca
grossmark13@gmail.com, petreus.dorin@utcluj.didatec.ro

Abstract: This paper represents a resume of the study about a data acquisition system for monitoring and debugging an I2C (Inter Integrated Circuit) communication. The system has the role of a sniffer: it does not interfere with the data communicated between the master and slave. It can be connected to only one I2C bus at the time and it sends the data using the USB (Universal Serial Bus) to the computer. The acquired data will be displayed using an interface created in LabVIEW, while also being saved in a separate file. This system consists of a microcontroller with integrated SPI (Serial Peripheral Interface) peripheral and a SPI to USB transceiver. The USB connection will also power the device. The system was only designed and tested for I2C communications up to 400kbit/s (Fast-mode) with a 7-bit address.

Keywords: *Data Acquisition, Communication Protocol, I2C, LabVIEW*

I. INTRODUCTION

Communication protocols are widely used in electronic circuits as a mean of sending data between devices. In some cases, it might be needed to monitor such a communication for different purposes. Debugging is one of them and it requires dedicated systems, which are often expensive. One of these might be a DSO (Digital Storage Oscilloscope) such as the RIGOL DS1074Z Plus [1]. Another option is a logic analyzer, such as the ones from the TLA Series made by Tektronix [2]. The oscilloscope is limited by the number of input channels, usually four. A logic analyzer solves this problem by having up to hundreds of input channels. When the only purpose is to debug a communication protocol, these two devices become expensive as they have a high complexity. [3]

An existing solution to the problem of monitoring and debugging an I2C communication is presented in the paper "Portable I2C monitor and debugger" [3]. In comparison to that system, the one elaborated in this article must be connected to a PC to show the data and has no ability to be a master or a slave in the I2C communication. On the other side, its advantage is the ability to save the data directly on the computer in a file and have a friendlier user interface.

II. ACQUIRING THE DATA

A. Software description

Figure 1 presents the block diagram of the system. It consists of a microcontroller connected to the I2C bus that has the role of a sniffer. The acquired data is sent to the PC using the SPI protocol, while being converted to USB by a transceiver. Once the data has reached the PC, it will be analyzed and processed by a program in LabVIEW.

The I2C [4] was invented in 1982 by Philips Semiconductor as a medium speed, synchronous communication protocol. It is used by many embedded systems that consist of EEPROMs, digital sensors or data converters [5]. The bus consists of two signals: Serial Data (SDA) and Serial Clock (SCL). For acquiring the data, two external interrupts of the microcontroller were used, one for each I2C signal. The acquired data bits are being translated to characters of *'1'* and *'0'*. When it comes to a start or stop command, these will be represented by characters *'S'* for start, *'R'* for repeated start and *'P'* for stop. An example of an I2C sequence might be *„S101011100R101011111P"*, which means: Start – Write to address *0x57* – Acknowledged – Repeated Start – Read from address *0x57* – Not Acknowledged – Stop.

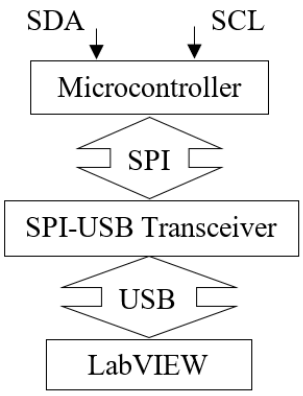

Figure 1. System block diagram.

Both external interrupts are set to trigger on any edge of the corresponding signal. A rising edge on the SDA interrupt during the high level of the SCL signal will represent a stop command, while a falling edge will translate to a start command. The other interrupt will acquire the data on the rising edge of the SCL signal and will save it via SPI on the falling edge. Figure 2 presents the block diagram of the software running on the microcontroller.

Manuscript received June 25, 2021; revised December 17, 2021

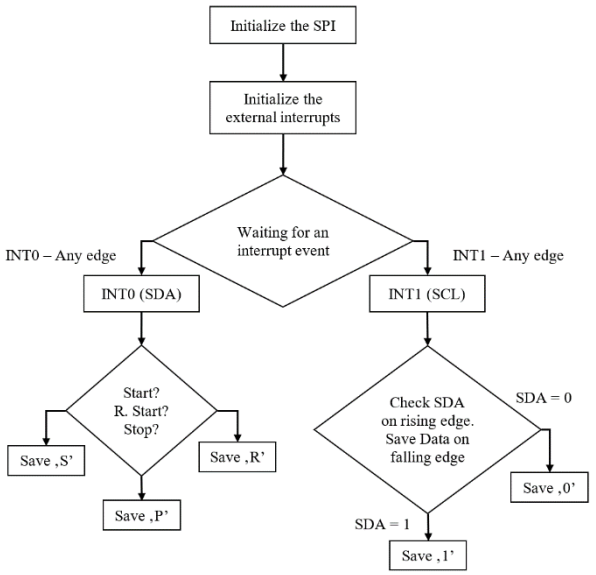

Figure 2. Block diagram for the I2C data acquisition.

Being part of the acquisition block, the microcontroller plays the role of the slave while communicating using the SPI protocol. As soon as the LabVIEW application, as SPI master, will start generating the clock signal, the buffered data will be sent to the PC trough the transceiver.

Regarding the slave, the programming algorithm is described using Figure 3. The first step is represented by the initialization of the ports (SCK – input, MOSI – input, MISO – output, SS – input) and the device as being a SPI slave. To send the data, the slave will load it into the buffer and will wait for the buffer to empty itself. After the data has been sent, the microcontroller will erase the data from its local buffer not to send it again.

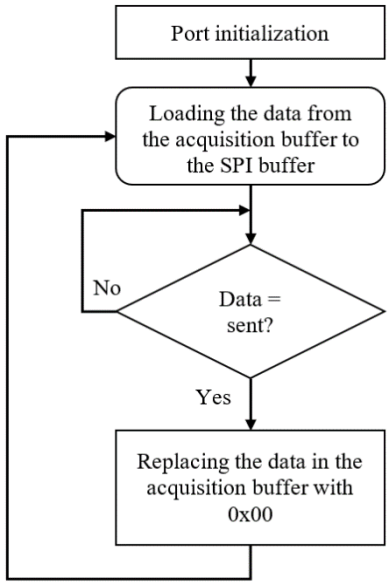

Figure 3. SPI communication (Slave)

The LabVIEW application uses a downloaded driver for the transceiver. Figure 4 describes the functioning of this application. The first step is the initialization of the communication channel and its characteristics. The SPI mode that is being used is mode *0*. The next step is the reading of the data being stored in the microcontroller. If the data is *0x00*, it will be ignored, otherwise it will be added to a character array. The character array will be processed as soon as the reading of the data will be stopped.

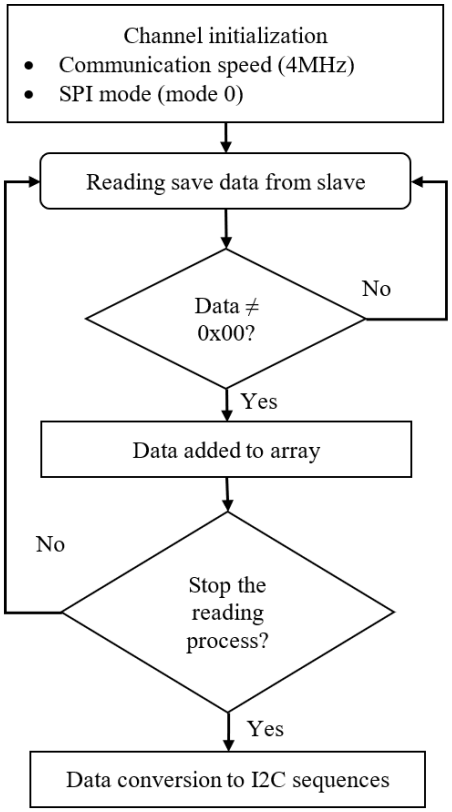

Figure 4. SPI communication (Master)

B. Hardware description

1) The processing block

This block consists of a microcontroller and its peripheral subcircuits, presented in Figure 5. The microcontroller is represented by an ATMEGA324P [6]. It is connected to a reset subcircuit, to its clock source, to the programming connector, to the data source and to a debugging LED.

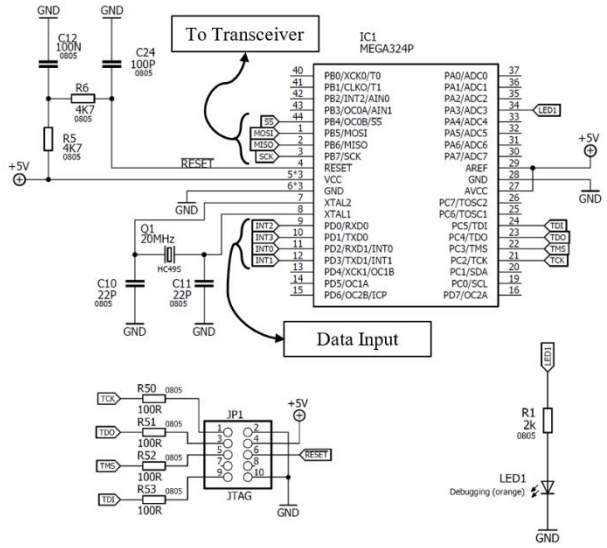

Figure 5. The processing block

2) The communication block

This part of the system consists of a circuit designed and built by Adafruit [7]. This system presented in Figure 6 contains a SPI to USB transceiver (FT232H) with its subcircuits and a USB Type-C connector. The USB connection is also used to power the entire system. The authors did not have any input designing the LabVIEW driver and its functions. It was designed by another person, and, with their approval, taken and used for the project.

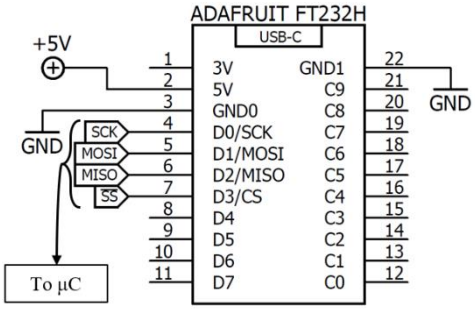

Figure 6. The communication block

3) The protection subcircuit

The input signal is in theory a digital one with two positive levels of voltage, 0V and 3.3V or 5V. To protect the circuit from voltages outside this domain and to recreate smooth logic transitions, a subcircuit such as the one shown in Figure 7 was designed for each input. It consists of two Schottky diodes that clamp the voltage value between 0V-V_F and 5V+V_F. The forward voltage V_F on a Schottky diode is between 150mV and 450mV, therefore being lower than the voltage drops on any internal junctions. Another component is the Trigger Schmitt buffer which will eliminate the noise and recreate a smooth digital signal.

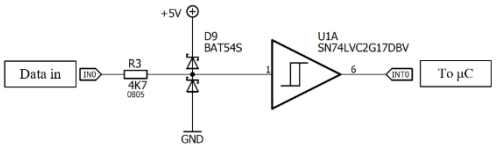

Figure 7. The protection subcircuit

III. PROCESSING THE DATA

Once the data is acquired it will be sent to the PC to be processed. The LabVIEW program will receive arrays of characters and translate them to I2C sequences in the format presented in Figure 8. These sequences will be displayed in
an interface and will be saved in a separate text file.

It can be observed that after each start command the instruction read or write will be followed by the corresponding address. Acknowledged and Not Acknowledged bits are being displayed as ‘A’ and ‘N’ characters.

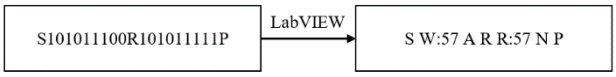

Figure 8. Data processing in LabVIEW

The algorithm used in LabVIEW is based on the diagram in Figure 9. The processing will run until the character array is empty.

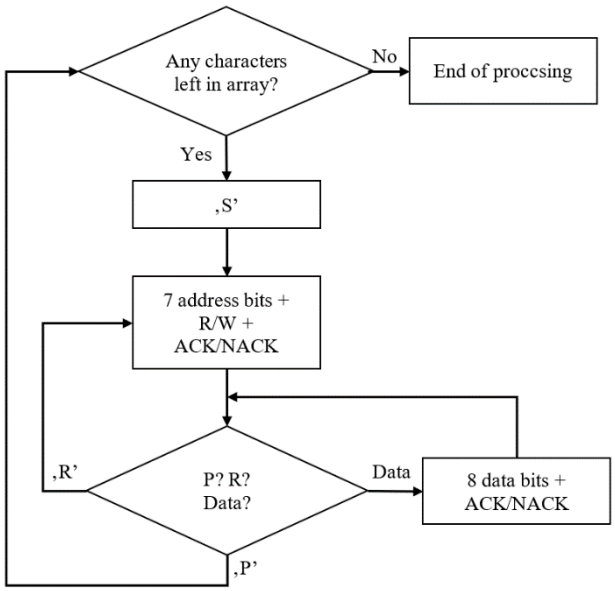

Figure 9. Data processing diagram

The processing will start when the program will find a ‚S’ or ‚R’ character. The next 8 bits are analyzed as followed: 7 address bits and one „Write/Read” bit. The next bit is an acknowledged or not acknowledged bit. From this point forward, a „Repeated Start” bit, a „Stop” bit or 8 data bits can appear. The first case will follow the initial steps. The second case will verify if there are any

characters left in the array. The third case will display two hexadecimal numbers and an acknowledged or not acknowledge bit. What follows is one of the 3 cases listed above. The processed sequences will be displayed in the LabVIEW interface, while also being written in a text file for future use.

IV. EXPERIMENTAL RESULTS

Using a RIGOL DS1074Z Plus oscilloscope, a communication between a microcontroller and a light sensor was monitored. The light sensor is transmitting the value for *green* to the microcontroller, as requested. At the same time, the system presented in this paper was doing the same. The results obtained by the oscilloscope can be observed in Figure 10 trough Figure 14, while the system obtained the data displayed in Figure 15.

Regarding the results, they are identical, which demonstrates the good functioning of the data monitoring and acquisition system.

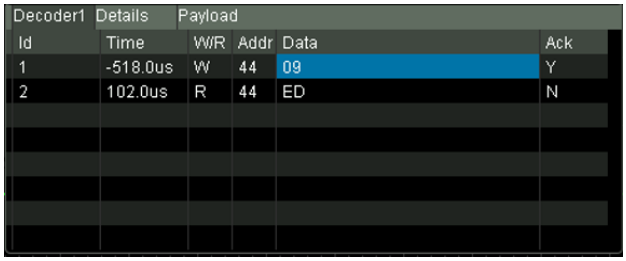

Figure 10. Data obtained by the oscilloscope (ED)

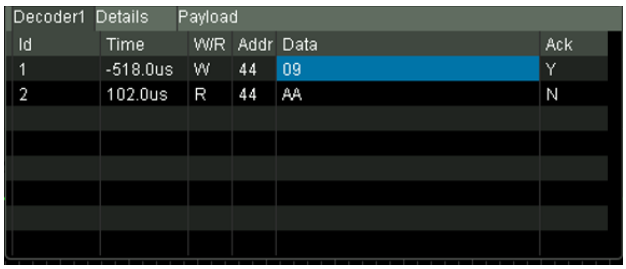

Figure 11. Data obtained by the oscilloscope (AA)

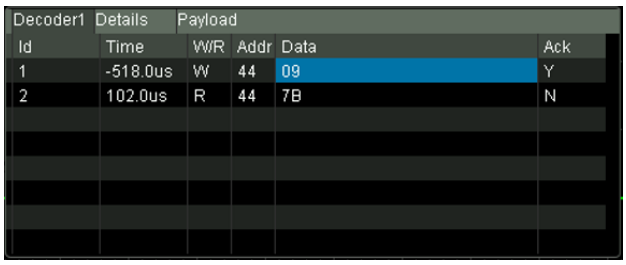

Figure 12. Data obtained by the oscilloscope (7B)

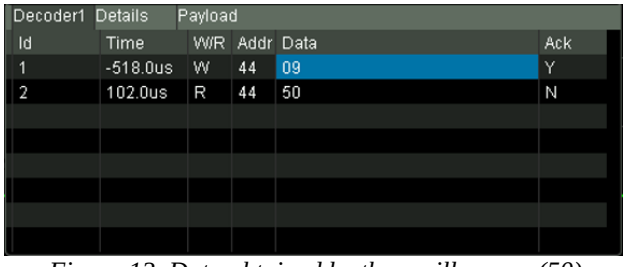

Figure 13. Data obtained by the oscilloscope (50)

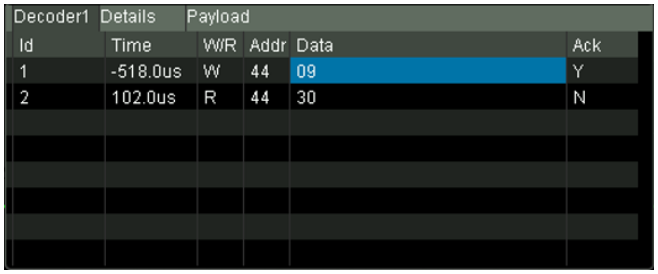

Figure 14. Data obtained by the oscilloscope (30)

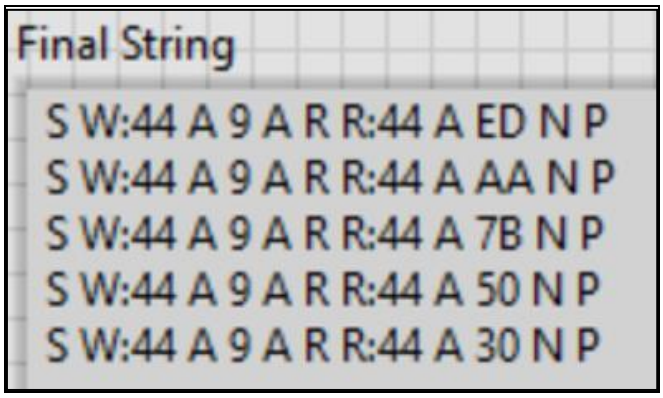

Figure 15. Data obtained by the monitoring system

V. CONCLUSIONS

This system design should offer a cost-effective solution to communication protocol monitoring and debugging. Used together with a laptop it becomes portable and flexible. Further development might include the addition of new communication protocols such as SPI, RS232, or another I2C bus, as the circuit has four signal inputs in the current stage. Another improvement might be the possibility to debug a 10-bit address I2C communication, as the current circuit is designed only for a 7-bit address.

REFERENCES

[1] All About Circuits – Rigol DS1074Z Plus Manual, https://www.allaboutcircuits.com/test-measurement/oscilloscopes/mso-ds1000z-plus-series-ds1074z-plus/manual/

[2] Tektronix – Logic Analyzer Fundamentals, https://assets.testequity.com/te1/Documents/pdf/logic-analyzer-fundamentals.pdf

[3] Grecu C., Iordache C. – "Portable I2C monitor and debugger", *2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pp. 131-134

[4] Texas Instruments – Understanding the I2C Bus, https://www.ti.com/lit/an/slva704/slva704.pdf?ts=1615100127956&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTCA9548A

[5] J. W. Bruce, Member, IEEE, M. A. Gray, Student Member, IEEE, and R. F. Follett, Member, IEEE – "Personal Digital Assistant (PDA) Based I2C Bus Analysis", pp. 1482-1487

[6] Microchip – Atmega324P Datasheet

[7] Adafruit – Adafruit FT232H Breakout, General Purpose USB to GPIO, SPI, I2C