Ročník 2017 electroscope Číslo II

Error rate of USART in NRWW Section

L. Novák [1,2], P. Šteffan[1]
[1] Brno University of Technology, Department of FEEC,
Technická 3058/10, Brno
[2] Central European Institute of Technology
Purkyňova 123, Brno
E-mail : xnovak0b@vutbr.cz, steffan@vutbr.cz

Anotace:
Článek popisuje Universální Asynchronní a Synchronní sériovou linku a základy jejího nastavení. Dále je ve článku popsáno rozdělení Flash paměti u mikrokontrolérů ATmega a částečná implementace Bootloaderu a uložení v konkrétní sekci. Později je ve článku popsána problematika přenosu dat po sériové komunikaci, při běhu programu v sekci „Not Read While Write Section". V poslední části článku je popsáno řešení k dané problematice přenosu dat.
Abstract:
This paper describes a Universal Synchronous and Asynchronous serial Receiver and Transmitter and its basic setting. Further it describes a divided Flash memory at ATmega chip. The function of the Bootloader and the storage in Flash memory is described too. Than it describes the transfer data problematic in Not Read While Write Section by the serial communication device. At the end of the paper is proposed the solution of received data problematic in the application section.

INTRODUCTION

Universal Synchronous and Asynchronous serial Receiver and Transmitter (in short USART) is serial communication device implement in ATmega microcontroller. This serial communication device is used in industrial application. The basic serial frame of USART (see in **Fig.1**) is defined with 1 start bit, 5 to 9 bits of data, no, even or odd parity and 1 or 2 stop bits. The most used data frame is with 1 start bit, 8 bits data, without parity and 1 stop bit. When a frame is transmitted, the next frame can be sent immediately. In case all data are sent, the communication line is set an idle state. The idle state is represented by logical high level.

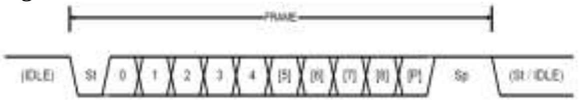

Fig. 1: Serial frame of USART.

Before any serial communication is started, the USART must be initialized. The initialization contains settings of baud rate, setting frame format and enable receiver or transmitter of the device or interrupt. The Baud Rate is depended on the connected crystal. The crystals can be used internal, external clock signal or full swing crystal oscillator. In the microcontroller ATmega is two internal crystals: 8 MHz crystal and 128 KHz low power crystal. Usage of the internal crystal is problematical, because of their inaccuracy. The better option is used the external clock. It can reach greater accurate, but it still depends on the connected clock. The better usages are used to full swing crystal. The full swing crystal does not reach considerable accuracy than the external clock, but the implementation is very simple with minimum components. In this work were used two full swing crystal. Size of crystals in the paper was used 14.7456 MHz and 16 MHz. After the crystals were selected, the value is recalculated for the specific baud rate of USART (see in **Eq. 1**).

$$UBRR = \left(\frac{fosc}{16*Baudrate} - 1\right) \qquad (1)$$

The calculated value is set in the register UBRR. By this equation, the calculated value for 38.4k Baud Rate with 16 MHz is 25. By back calculation to the Baud Rate, the value is 38.461k. As can be seen, the actual Baud Rate value is little different than the set value. This difference value is called Error Rate, and the formula for calculate percentage Error Rate values are calculated using the following equation:

$$Error\ rate[\%] = \left(\frac{Baudrate_{ClosedMatch}}{Baudrate} - 1\right) * 100\ \% \qquad (2)$$

Nominally percentage of Error Rate is tolerated around 10 %. The smaller this value is, the more likely it is the Transmit/Receive data will not be lost. Table of value uses speeds in the paper is in **Tab.1**. By setting register UBRR to the specific value is used Baud Rate for asynchronous operation.

Tab. 1: Table of values for settings register UBRR.

Baud Rate [bit per second]	***f_{osc}* = *14.7456 MHz***		***f_{osc}* = *16.0000 MHz***	
	UBRR [-]	**Error [%]**	**UBRR [-]**	**Error [%]**
38.4k	23	0.0 %	25	0.2 %
57.6k	15	0.0 %	16	2.1 %
115.2k	7	0.0 %	8	-3.5 %
230.4k	3	0.0 %	3	8.5 %
250k	3	-7.8 %	3	0.0 %
0.5M	1	-7.8 %	1	0.0 %
1M	0	-7.8 %	0	0.0 %

The USART serial communication can be used only for transfer data between two devices. In industry, USART is used in combination with the RS485 standard. By this usage, it can microcontrollers communicate more than two devices. By adding RS485 chip must be communication device data frame used with addressing packet.
The microcontroller ATmega has divided Flash memory into two sections. The first section is RWW (Read-While-Write). When the program is running in this section, it cannot be read itself. But the same section can read data from NRWW section. The second section is NRWW (Not-Read-While-Write) (see in Fig. 2). When the program is running in this section, we can read or write into the RWW section. In situation the program running in NRWW section and we want to read the same memory, we cannot. If we try it, CPU is halted during the entire operation. RWW is named Application part, and NRWW is named Bootloader part.

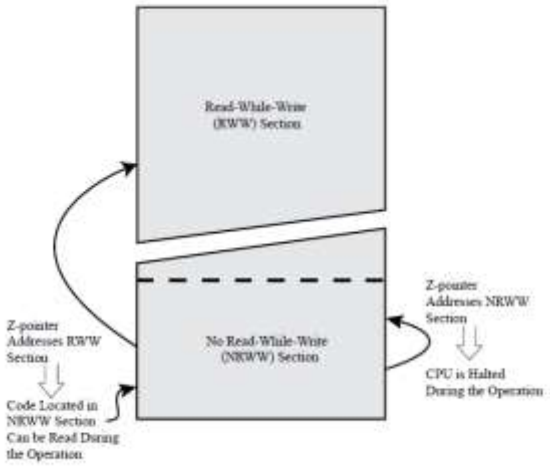
Read-While-Write (RWW) Section
No Read-While-Write (NRWW) Section
Z-pointer Addresses RWW Section
Code Located in NRWW Section Can be Read During the Operation
Z-pointer Addresses NRWW Section
CPU is Halted During the Operation

Fig. 2: Read-While-Write vs. No-Read-While-Write [1].

So basically we can rewrite Application part by Bootloader part without an external programmer. This property is helpful for update firmware when we find a bug, and we do not have physical access to the device. In the Bootloader part, we can use all peripheral like we were in the application part. One of the problems is the interrupt. When we were in the bootloader part, and we want to use interrupts in the bootloader section, we must set specifics register. When we set this register, the interrupt vector is moved to the start of bootloader section. By this operation, we can use all attributes of the interruption.
The most typical usage of interrupt is for Universal Asynchronous Receiver Transceiver. It is not necessaries uses interrupt for the USART, but meanwhile, data is not received by USART, we cannot use another function of MCU. If the interrupt is used, the microcontroller can perform another instruction meanwhile. By the particular register, we can set the speed of the USART.

PROBLEM OF NRWW SECTION

The initial testing board was assembled with 14.7456 MHz crystal. The microcontroller used in the initial board has two USART device. First USART device is used for transfer data between MCU and computer by FTDI chip. This chip is used as the converter from USB to USART. On the second is the connected converter from USART to RS485 bus. It is used to connect more testing board on the bus then one. The microcontroller is set Baud Rate by UBRR as mentioned before. At the computer, the Baud Rate is set in program Real Term. This program is used to communicate and debugging firmware inside of microcontroller.
Initial test communication on the testing board device was done in several steps:

- Test number 1: Transfer data between computer and microcontroller in the Application section.
- Test number 2: Transfer data between computer and microcontroller in the Bootloader section.
- Test number 3: Transfer data between master microcontroller and slave microcontroller. The Master microcontroller is set in the application section. The slave is set in the bootloader section.

Each step of the test was performed with 14.7456 MHz and 16 MHz crystal. The first and second test was performed with the size of 256 B data transfer. The last test was performed with reduced size packet of 9 Byte. Each packet is contained randomly generated data. The newly generated data is provided that the previous measurement has not affected on the next one.
On the testing board was connected logical analyzer for spying bus. The usage logic analyzer was used Saleae Logic 8 channel. In the program for Saleae can be the measured speed of the serial connection. Example of the measured value for the size of crystal 16 MHz is shown in the following table. In the same table is calculated the Error Rate of every usage Baud Rate.

Tab. 2: Předpokládaná čísla časopisu

Targeted Baud Rate [bps]	***Measured Baud Rate of receive line [bps]***	***Measured Baud Rate of transmit line [bps]***	***Error Rate of receive line [%]***	***Error Rate of transmit line [%]***	***Real Error Rate [%]***
38.4k	38400.10	38461.54	0.0003	0.1603	0.1606
57.6k	57692.42	58823.53	0.1604	2.1242	2.2846
115.2k	115385.50	111111.11	0.1610	-3.5494	-3.3884
230.4k	230771.01	250000	0.1610	8.5069	**8.6680**
250k	250000	250000	0	0	0
0.5M	500000	500000	0	0	0
1M	1000000	500000	0	0	0

In the table is bold Error Rate at Baud Rate of 230.4k. On this speed, the error rate was too high, so a problem has occurred with serial communication. Some data were lost from the packet. At other speed was not the problem in received data. Similar data were measured in test number 2. The problem with received data happened not only at 38.4k Baud Rate but at other speed expect of speed with Error Rate 0%. The Test number 3 was performed the same as previous, but the size of the packet was only 9 Byte. This is normally standard for flow-bus with

As an example of the Test number 2, the 256 Bytes of the packet from PC to MCU was sent, but microcontroller was received something about 240 Bytes. During each test, the different number of data was received. Sometimes the MCU is received a 250 Bytes, other times it is received fewer data. The biggest problem was in case the MCU received all 256 Bytes, but some data was not the same, as sent from the computer. There were found three solutions to this problem. The first solution includes dual control of data. The firmware features control of data by checksum. The second control of data could involve sending the data second time. In this case, it will be speed line slow down to the half. In some cases, slowing down the communication speed may not be desirable. The second method deals with change of the communication speed at a value where the error rate is zero. The last solution was the combination of the previous two solutions.

The first solution has created the table of 256 Bytes randomly scattered on the table. Each packet of data is sent from the computer, and it was calculated checksum by bits negate. After the 256 Bytes of data are sent, the computer sends one byte of the checksum. The same table and calculation was done on the MCU site and compare with received checksum byte. If the checksum was not the same as received, then the MCU sends not-acknowledge byte as a request for resending of data. The last solution was made with the combination of the previous solution for the double check of transfer data. Because none of the testing boards were bought, the problem was solved by the change to the communication speed where the Error Rate is zero percent for next work. But it is not the problem include part of the code for checksum in future.

CONCLUSIONS

In summary, in the work is described serial communication device USART implemented at Atmel microcontroller. Then is described the Flash memory inside of chip and differences between Application and Bootloader section. Further, the calculation of Error Rate is described and the table of filled the values. After the basic description of the microcontroller and serial communication device USART is mentioned the testing board and steps of testing. Three steps are selected and the resulting 16 MHz crystal included into the table. Calculation of Error Rate is recapitulated in the table and in the descriptions. During testing has been found the troubles in data transfer for Application section. Also there are described three methods of improving of this issue. For this issue was chosen the second solution, with the option of inserting checksum code by the time.

ACKNOWLEDGMENT

The article was supported by project no. FEKT-S-17-3934, utilization of novel findings in micro and nanotechnologies for complex electronic circuits and sensor applications.

REFERENCE

[1] “Datasheet: ATmega644P/V”, 8-bit AVR Microcontrollers, 2016. [Online]. Available: http://www.atmel.com/Images/Atmel-42744-ATmega644P_Datasheet.pdf. [Accessed: 20-Oct.-2016].

[2] L. Novák, “Bootloader for Sci-Trace”, Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016.

[3] “AVR106: C Functions for Reading and Writing to Flash Memory”, Atmel Corporation, 2017. [Online]. Available: http://www.atmel.com/Images/Atmel-2575-C-Functions-for-Reading-and-Writing-to-Flash-Memory_ApplicationNote_AVR106.pdf. [Accessed: 08-Nov.-2017].