1 Lab 6: UART: Pointer, Strings and structures

1.1 Summary

This lab is about programming techniques: You will uses pointers to process strings, which greatly simplifies the sending of text messages. You will use structs to organise data in a tidy manner. And you will learn more about serial communication and the meaning of its parameters.

1.2 Objectives

By the end of the lesson, you shall have completed the provided files. They extend the serial communication code from the last lab with a new function, which takes and sends an entire string. And you shall modify these files to use a structure for handling all USART parameters. If done correctly, the provided test program prints several messages. Find out about the following points:

- Why are some messages received incomplete?
- What is the matter with the parity bit? When used on this hardware, it breaks the communication. Why that? Hint: Check the entire chain of data transmission from inside your microcontroller until the PC.

The working appearing text messages in your terminal program and the source code have to be shown and explained to the supervisors until the end of the lesson. Be prepared to spontaneously use other parameters if requested to do so - your code must be generic for all combinations of the parameters specified below.

1.3 Further Information

The microcontroller used in this lab is an ATmega328P on a developer board called ATmega328P Xplained mini. LEDs, resistors, NTCs etc. are available as in the previous labs, but actually not needed this time.
To complete this lab, it is important to understand the meaning of the parameters baud rate, parity, stop bits and data bits. How do they change the bits and signals that go over the physical line?
Hint: Think about where this line actually is (physically). Keep in mind that the mEDBG chip on these boards translates the (true) serial communication from the ATMega328P microcontroller into USB packages, and that this mEDBG chip is limited to 57600 baud/s

maximum. Slower baud rates work, higher dont. Consider the possibility that this chips firmware, the USB driver or your terminal program on the PC side might have further limitations, too.
Hint: If you ever compute big numbers in your source code, keep their bit sizes in mind. Even if the final result fits in a 16 bit variable, you might have intermediate values that exceed this size and cause overruns and computation errors.

1.4 Recommended reading

Atmega328 [1]**:** contains the datasheet and further information about the ATMega328p microcontroller. Datasheets with register descriptions are a thrilling lecture and should not be missed. Seriously knowing which registers there are (and what they can do) is essential for embedded systems programming and especially for using peripheral modules. Have a look at it!
Xmini [2]**:** provides the full developer board hardware documentation.
Man minicom: gives you the manual of the minicom terminal program (if it is installed type this command on your CLI). Despite minicom, there are several other terminal programs which you can use, e.g. gtkterm or CuteCom. They are all installed in the VM image provided to you for this course. Use whichever you want.
A search engine of your choice should point you towards further information about serial interfaces, their historic development and the meaning of their parameters. This might explain why settings like 5 data bits exist at all, and which meaning stop and parity bits once had. A little background knowledge might also help you to deal with the pitfalls of modern serial implementations (not every manufacturer supports all theoretically possible modes).

[1] `http://www.atmel.com/devices/atmega328p.aspx`
[2] `http://www.atmel.com/tools/mega328p-xmini.aspx`

2 Task1: Complete the provided source code and establish communication with a terminal program on your PC in different baud rates

2.1 Required preparation

- You need to bring your own device (e.g. laptop) with a working installation of the avrgcc crosscompiler and the avrlibc library. See Lab 2: Hello, World!" on Atmel AVR Microcontrollers for details.
- Understand how serial communication works. What are baud rates, parity, data bits, start and stop bits about? What are they good for?
- Check the provided source code. You are supposed to extend the file USART/USART_implement_me.c. You can reuse a lot of code from the last lab, but you will also need to bring some more. Prepare these lines - setting the register values for all the different parameters is actually easy, but you should know the values in advance. Note that these values are stored in a struct.
- If you are not sure anymore, find out about structs (and how to access their members).
- Switch-case-statements provide a beautiful way of evaluating the struct members in this case. Use them where it makes sense!
- Familiarize yourself with a terminal program of your choice (suggestions see above). How can you change the required parameters in it?

2.2 Class work

You are supposed to add the missing code lines in USART/USART_implement_me.c. First, add support for the transmission of strings. Then change your init function to accept and evaluate a struct. This struct shall contain the following parameters:

- Baud rate (any standard value between 1200 and 57600 shall work)
- Number of data bits (5 to 8, support for 9 bits is not necessary)
- Number of stop bits (1 or 2, the ATMega328P does not support 1.5)
- Parity (none, even or odd; carefully read all text above, esp. the Objectives)

The evaluation of this struct should partially use switch case statements. This allows to create very neat and maintainable code. If you detect parameters that your init function does not support, return an error code. Note that any combination of the specified parameters must work, not just the example ones which you have to complete in USART_2.c.
Note that you have to specify the members of this struct, too! It is located in USART/USART_implement_me.h, which makes it accessible to all .c files that include this header.
Configure your terminal program to see how the test strings from USART_2.c are received. Note that you will most likely have to reconfigure your program by hand and then press the reset button on the developer board. Adapting the terminal settings between the four test messages would be very tricky.
Feel free to modify the messages and parameters and try out different things. Focus on the question which combinations work and which ones dont. Why wont they? The result and the source codes must be shown to a supervisor before the end of the lesson. You might be asked to change the parameters in your program and the terminal and demonstrate that this combination works, too. (The supervisors will not request combinations that are known to not work).

2.3 CLI instruction cheat sheet

```
avr-gcc -Wall -Wextra -Wpedantic --std=gnu99 -mmcu=atmega328p -Os -o USART_2.elf USART_2.c USART/USART_implement_me.c

avr-objcopy -j .text -j .data -O ihex USART_2.elf USART_2.hex

avrdude -p atmega328p -F -c stk500v1 -P /dev/ttyACM0 -b57600 -D -V -U flash:w:USART_2.hex:i

minicom -s
man minicom
cutecom
gtkterm
```