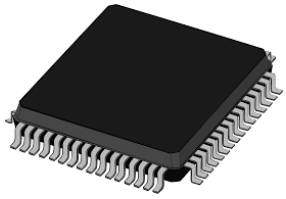
Computer Organization

Lab 2



EITF70

I/O Handling

Goals

- To understand what an I/O device is
- To be able to interface internal and external I/O devices
- To understand communication protocols
- To get an understanding of practical problems and how to solve them

Sidra Muneer, Steffen Malkowsky and Erik Larsson
2020
Christoffer Cederberg and Jonathan Sönnerup
2019

Contents

Introduction

Purpose

The purpose of the laboratory exercise is to learn and understand what an input/output (I/O) device is, get some experience on how to write programs that interact with an I/O device, understand the basics on how communication protocols work, get an understanding of practical problems and how to solve them, and get some knowledge on reading documentation.

Organization

The material is organized in two main parts; I/O ports and Universal Synchronous/Asynchronous Receiver/-Transmitter (USART). For I/O ports, we first detail I/O ports and then we describe the home assignments and the laboratory exercises on I/O ports. For USART, we first detail USART and then we describe the home assignments and the laboratory exercises on USART.

Hardware

The hardware used in the course includes several I/O devices. The external devices are shown in Figure 1 and 2. In Figure 1 the I/O devices are encapsulated in a red rectangle and labelled with a number In Figure 2 the corresponding blocks are coloured black.

1 - LED's Eight LEDs that are connected to Port B. By setting a pin high (3.3 V), the corresponding LED emits photons with the energy 2.8^{-19} J ;).

2 - Potentiometers Two potentiometers are connected to Port A on the microcontroller (PA0 and PA1). A potentiometer can be seen as a voltage divider where the resistance of the two resistors is changed when the button is turned.

3 - Buttons There are six buttons on the circuit broad. They are connected to Port A on the microcontroller. If a button is pressed, the corresponding pin will be set high (high = 1 in software or 3.3 V in the real world).

4 - Spare button This is a spare button, nothing fancy.

5 - Ultrasonic sensor This piece of hardware can be used to measure the distance to an object. This is done by measuring the time of flight of a burst of sound waves (from the sensor to an obstacle and back). This time is proportional to the distance to the object the sound reflects on.

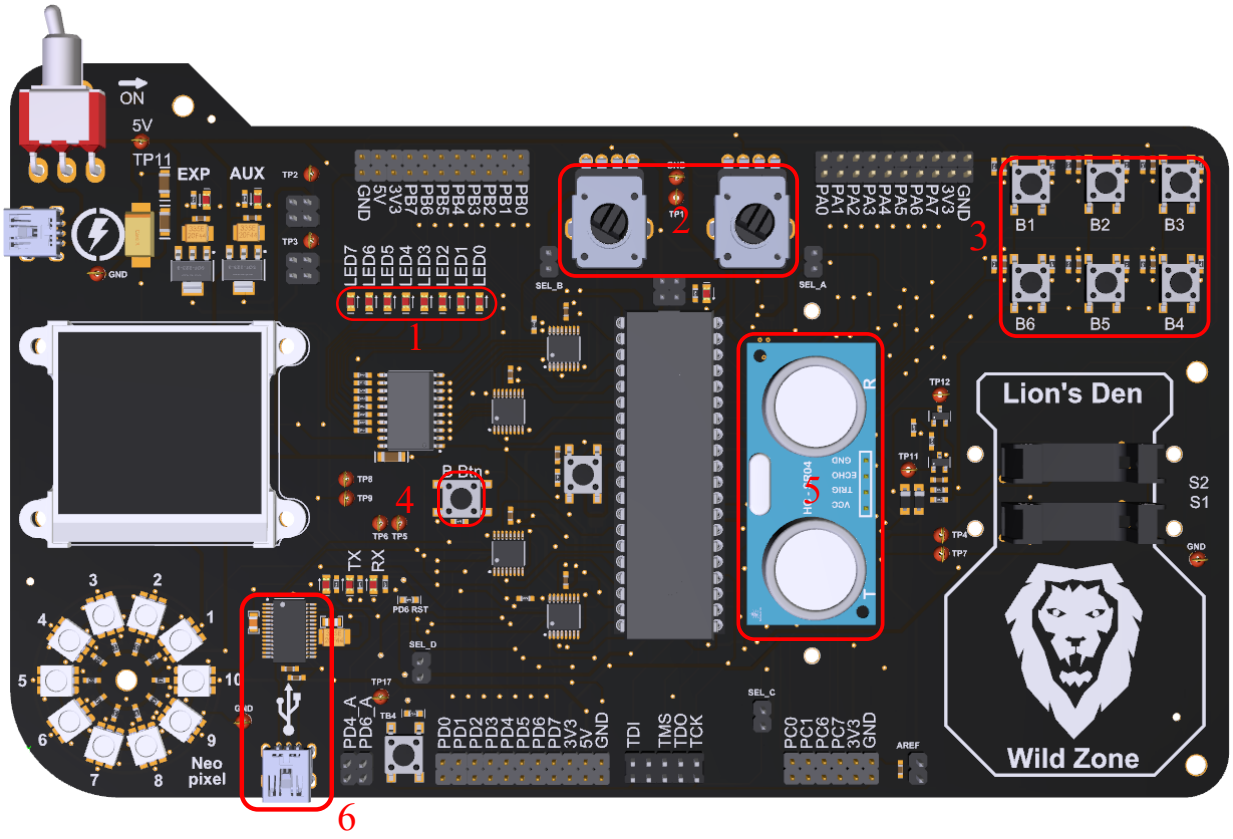


Figure 1: A picture of the circuit board with the used I/O devices marked.

6 - USB-UART interface This integrated circuit translates UART to USB (and vice versa) and enables the microcontroller to communicate with a PC. In order to send data back and forth to a PC, a USB cable needs to be connected between the USB mini connector on the circuit board and the PC.

Some of the I/O units need to be interfaced in a way that requires more finesse than what a simple, general-purpose I/O pin can achieve. For instance, the outputs of the potentiometers are analog signals. An I/O pin, when configured as an input, can only tell if a signal is above or below a certain threshold. For this reason, it lacks the ability to measure the actual amplitude of an analog signal. In case of the AVR microcontroller (and microcontrollers in general), each I/O pin can be connected (inside the microcontroller) to a couple of different sources. The name of the alternative sources can be seen in Figure 2. It is the text between the parentheses. When it comes to Port A, all of the I/O pins can be connected to the built-in analog-to-digital converter. See the name of the alternative source in Figure 2, `ADC0`, `ADC1` and `ADC2` and so on.

During the lab, two other internal I/O devices will be used, namely one of the two serial communication units and a timer. All of the used I/O devices, coloured black, can be seen in Figure 3.

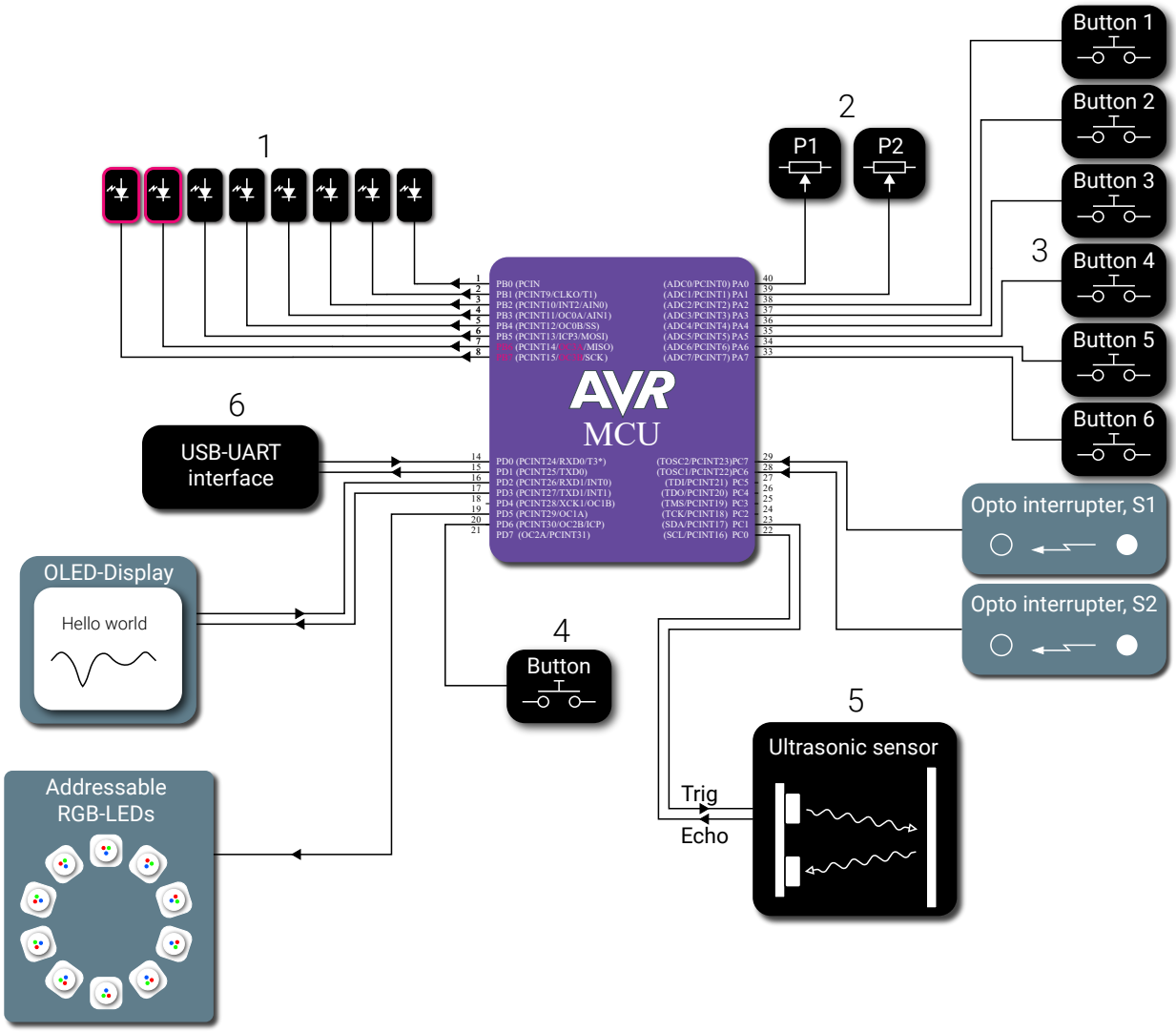


Figure 2: Block schematic of the circuit board with the used I/O blocks coloured black.

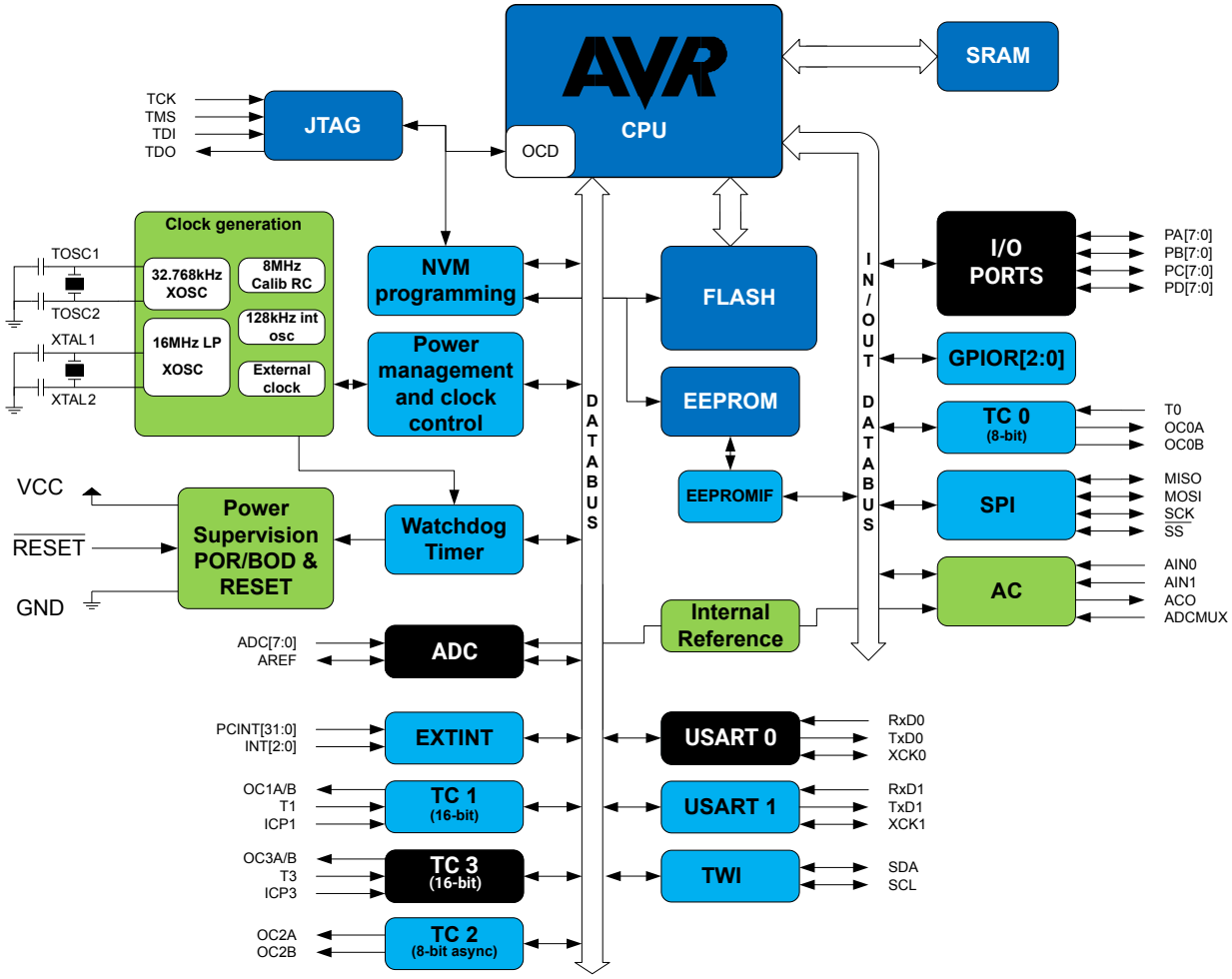


Figure 3: Block diagram for an ATmega1284P.

I/O Ports

In the previous laboratory exercise, two out of four 8-bit, bi-directional I/O ports were used. The name of the ports are Port A, B, C, and D. With this piece of hardware it is possible to change the state (high or low) of an individual I/O pin. This was done when turning an LED on and off. The I/O ports can also be configured to be an input. This function was utilized when checking the state of a button.

Each port has three different registers. For Port A the name of these registers are `DDRA`, `PINA` and `PORTA`. Each port contains 8 pins and thus the corresponding registers have the same length. In Table1 below, their functionalities are described.

Table 1: I/O port registers and their corresponding functionality.

Register	**Description**
DDRx	Specifies the data direction (input or output)
PINx	Read the state of the port, if it is an **input**
PORTx	Read and write to the port, if it is an **output**

To set the first pin on Port B, `PB0`, as an output and the pin high, follow the steps below.

- Set the data direction register to `0x01` (hexadecimal) or `0b00000001` (binary).
- Set the pin `PB0` to high by writing `0x01` or `0b00000001` to the port register.

When writing to a I/O port register (or any other register, for that matter) it is important to use the logic operator, OR (|), instead of directly assigning a value. By using the "|" operator, a bitwise OR operation is performed with the content of the register and the value `0b00000001`. By doing so, it is ensured that only the specified bit is changed. This is good, since in the majority of cases there will be different types of devices connected to the same port.

As mentioned earlier, the microcontroller has four I/O ports. See Table 2 for their addresses.

Table 2: I/O registers and their corresponding addresses.

Address	**Register**
0x20	PINA
0x21	DDRA
0x22	PORTA
0x23	PINB
0x24	DDRB
0x25	PORTB
0x26	PINC
0x27	DDRC
0x28	PORTC
0x29	PIND
0x2A	DDRD
0x2B	PORTD

Laboratory Exercises

Oh, You Again...

As may be remembered from the first laboratory exercise a state machine, see Figure 4, was implemented. In this assignment the same state machine should be implemented, but this time usage of the provided functions in the course library are not allowed. Each port's corresponding register addresses should be used instead. The addresses for the port registers can be found in Table 2. There are no restrictions regarding which LEDs or buttons (B1 to B6) to choose.

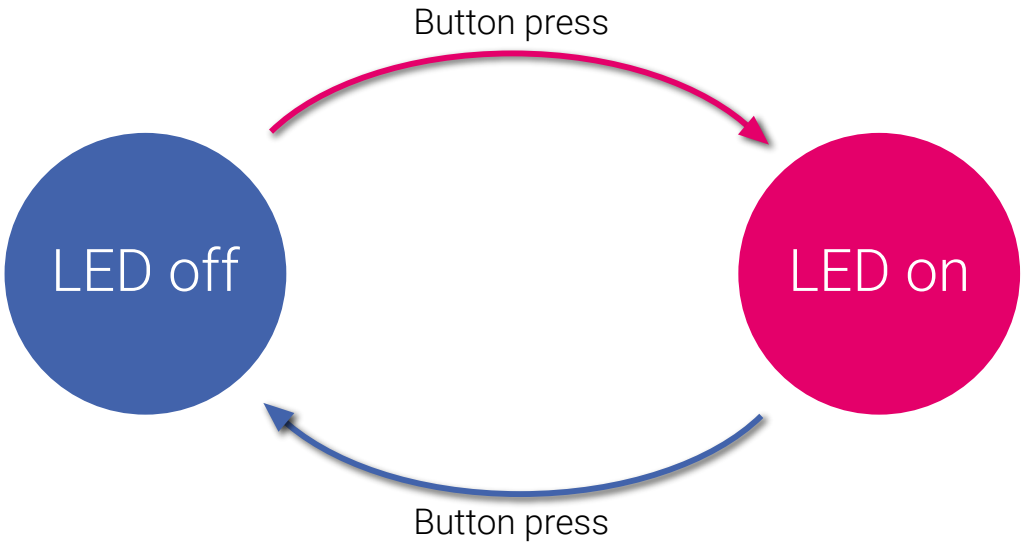


Figure 4: State diagram of the program.

Tasks:

- Create a new C project in Atmel Studio.

Home Assignment 1
The LEDs are connected to Port B. The address to the output register is as shown in Table 2. Using a pointer to the address, how do you turn LED number 3 (only) on?

Home Assignment 2
The buttons are connected to Port A. The address to the input register is as shown in Table 2. Using a pointer to the address, how do you read the value of (only) button 4?

Use the schematic in Figure 2 to see where the the button is connected.

- Implement the state machine with the provided register addresses.

Keep It Simple

To make life a bit more easy, Atmel has provided macros for all the memory-mapped I/O register addresses. Instead of writing the actual address and then type cast the number to a `volatile uint8_t` (or any other suitable data type) and then use it as a pointer, it is possible to just use the macro `PINA` instead.

Tasks:

- Create a new project in Atmel Studio.
- Copy and paste the code from the previous assignment and replace all the pointers with their corresponding macros, and verify that it works. Please refer to Table 2 if needed.

State Machine, once Moore

Here, we will investigate some of the physical properties of buttons.

Tasks:

- Now we will spice things up a bit. Change the currently used button to the spare button connected to PD7 (Port D). It is labeled with B_Btn and can be found on the left hand side of the MCU, see Figure 1. This button is of the same kind as button B1-B6. And remember to configure PD7 as an input only. There are other components connected to port D.

The correct macro for the pin labeled PD7 in the schematic and the datasheet, is `PORTD7`. Note that the macros are just numbers. `PORTD7` is simply equal to 7. To set the bit that corresponds to pin PD7, left shift a one `PORTD7` positions. (Do not forget that you should only affect the state of the specified pin.)

Lab Question 1
Is the application working as expected? Press the button *several* times. Do you notice anything strange? (*There is most likely nothing wrong with the code.*)

Home Assignment 3
When pressing a mechanical button, do you get a perfectly clean signal? If no, how do we handle such situations?

The origin of the problem comes from the mechanical properties of a button. When it is pressed the contact plate inside the button is bouncing up and down before settling, and thus a clean transition from zero to one is not obtained. See Figure 5. It has been taken by an oscilloscope and shows the button voltage as a function of time. Each horizontal division is equal to 1 ms.

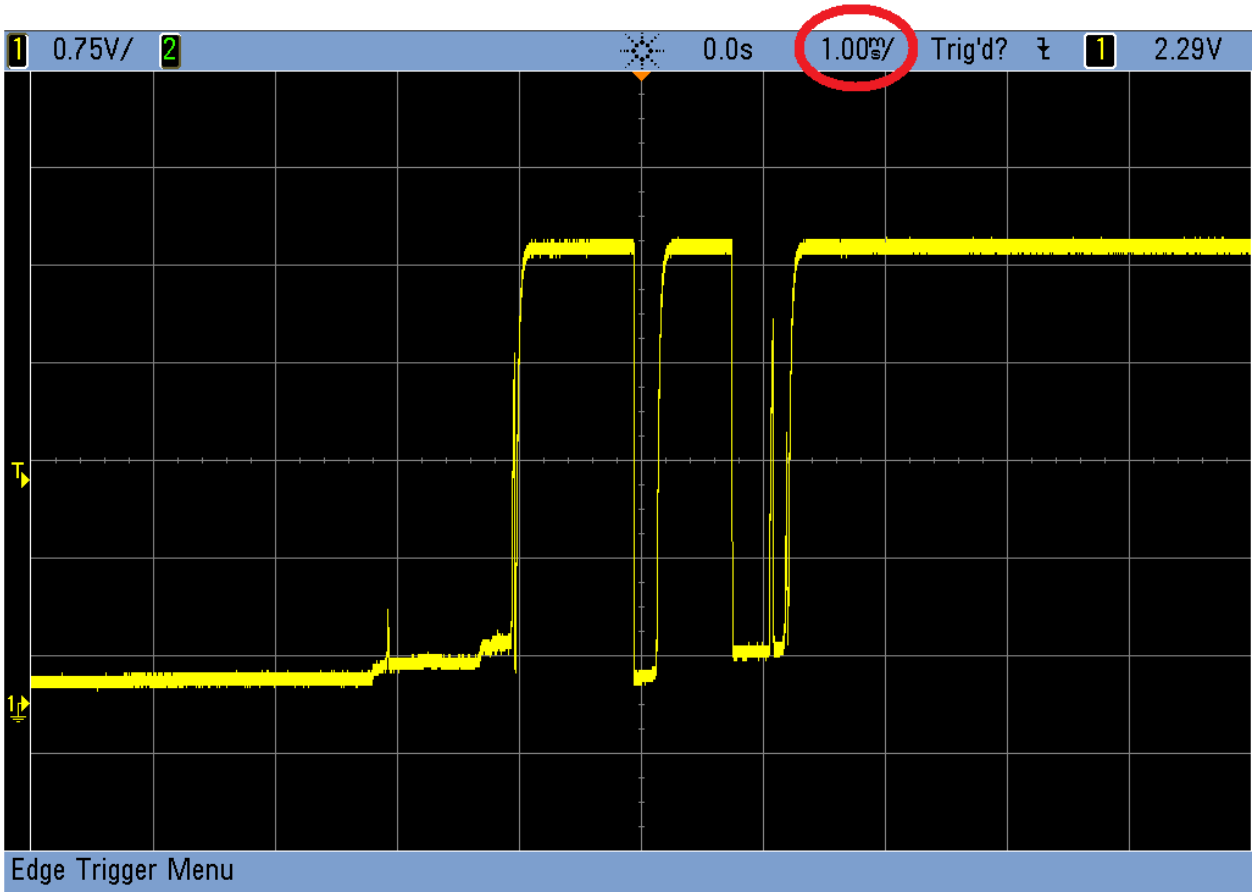


Figure 5: A picture of the button voltage taken by an oscilloscope.

Home Assignment 4
Create a function, `uint8_t button_read_reliably()`, that reads the button and excludes the bounces, shown in Figure 5.

- Test the code written you created in the previous home assignment.

You are now done with this part, show your work to a lab assistant! _____ ☑

USART - A Serial Communication Protocol

A microcontroller is often used to gather some sort of information which then is sent to a computer for analysis. Another typical application is that a computer sends commands to a microcontroller, which then performs an action that corresponds to the received command. This can be done with the USART peripheral device inside the microcontroller. The abbreviation USART stands for Universal Synchronous Asynchronous Receiver and Transmitter. With this device, the data is sent bit by bit. The transfer rate, that is, bits/s, is referred to as the baud rate. Each data package consists of one start bit, a number of data bits, a parity bit (which is optional), and one or two stop bits. See Figure 6.

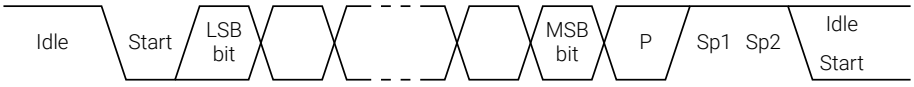


Figure 6: UART data package.

There are two versions of this type of communication, one is asynchronous and one is synchronous. The synchronous version requires that a clock signal is connected between the two devices. The more common is the asynchronous version, which is usually referred to as UART (Universal Asynchronous Receiver and Transmitter). This is the version that is going to be used during the laboratory exercises. Each UART device consists of a transmitter and a receiver. The receiver is often labeled Rx and the transmitter Tx. See Figure 7.

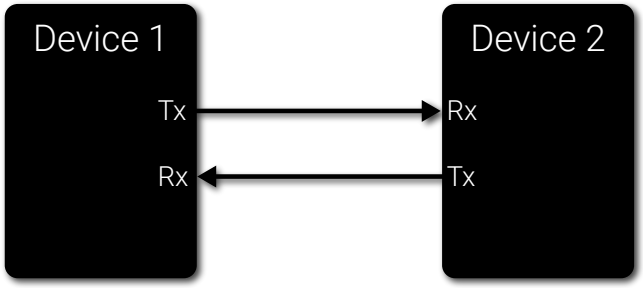


Figure 7: UART communication between two devices.

Before the communication starts, the devices needs to be configured, that is, the mode of operation, baud

rate, the number of data bits, stop bits, and parity[1] mode has to be selected. This is done by setting bits in the status and control registers of the USART device that corresponds to these options. The registers of the USART device for the used microcontroller can be seen in Table 3. For a detailed version, please refer to page 257-265 in the data sheet.

Table 3: Registers of the USART 0 unit.

Register	Description
UDR0	USART I/O Data Register 0
UCSR0A	USART Control and Status Register 0 A
UCSR0B	USART Control and Status Register 0 B
UCSR0C	USART Control and Status Register 0 C
UBRR0L	USART Baud Rate 0 Register Low byte
UBRR0H	USART Baud Rate 0 Register High byte

Data register

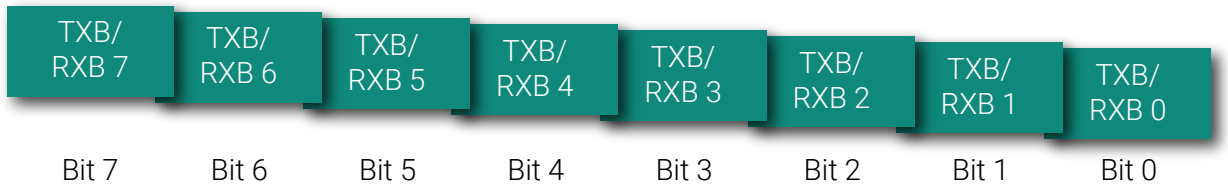


Figure 8: UART data Register.

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDRn[2]. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR1 Register location. Reading the UDRn Register location will return the contents of the Receive Data Buffer Register (RXB).

USART control and status

Control and status register A - UCSR0A:
Only two bits in this register is important for the laboratory exercises, UDRE0 and RXC0. They can be used while transmitting and receiving data.

Control and status register B - UCSR0B:
The only important bits in this register are the bits that enable the USART transmitter and receiver.

Control and status register C - UCSR0C:
The default value of UMSEL0[1:0], UPM0[1:0], USBS0 and UCSZ0[2:0] initializes the USART unit as following:

- USART mode is asynchronous,

[1]The parity is used as an error detection feature, but it will not be used during the lab exercise.
[2]When coding, the n in UDRn should be replaced with 0 since it is USART0 that will be used. This goes for all names containing an n.

- parity mode is disabled,
- one stop bit and
- the frame size is 8-bits.

During the laboratory exercises there is no need to change this configuration.

USART Baud Rate Register `UBRR0H` and `UBRR0L`

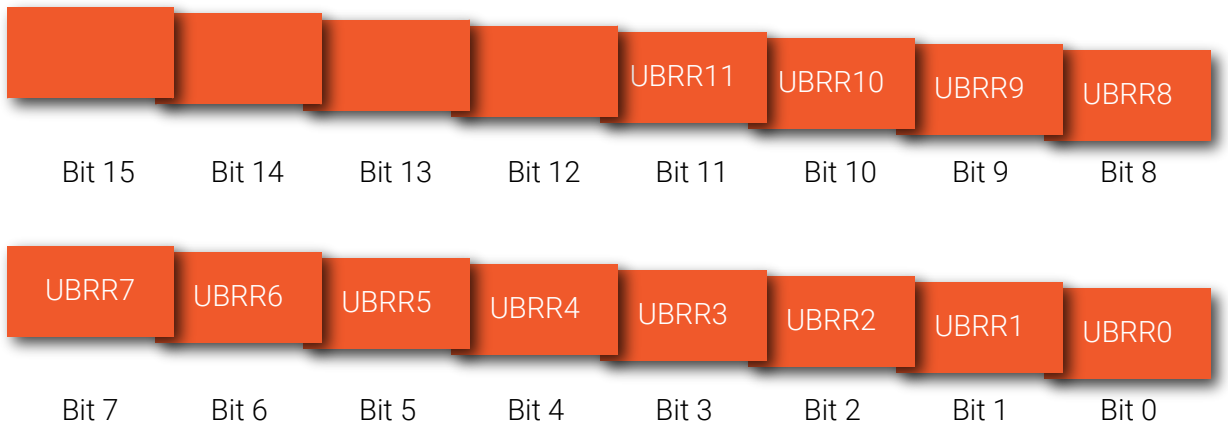


Figure 9: UART data Register.

Bits 11:0 – `UBRR011:0`: USART Baud Rate This is a 12-bit register which contains the USART baud rate. The `UBRR0H` contains the four most significant bits and the `UBRR0L` contains the eight least significant bits of the USART n baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing `UBRR0L` will trigger an immediate update of the baud rate prescaler.

Baud Rate [bps]	**fosc = 16.0000MHz and U2X = 0**	**Error rate**
2400	416	-0.1%
4800	207	0.2%
9600	103	0.2%
14.4k	68	0.6%
19.2k	51	0.2%
28.8k	34	-0.8%
38.4k	25	0.2%
57.6k	16	2.1%

To write a value to both `UBRR0L` and `UBRR0H` “simultanously” use the macro in the listing below.

```
UBRR0 = 103;
// alternatively
UBRR0H = (103 >> 8);
UBRR0L = 103;
```

Listing 1: Test.

Laboratory Exercises

Communicating with the World - Part I

In this assignment the computer will be used to send a command that toggles a LED.

In order to receive data from the computer, the USART hardware inside the microcontroller needs to be initialized. On the computer side, a serial terminal has to be used to send and receive data. For this purpose a software called YAT (Yet Another Terminal) should be utilized. Instructions on how to use this software will be given later. There are two USART units in the Atmega1284, USART 0 and USART 1. In this assignment the USART 0 will be used. In the data sheet there are some useful tables when selecting the baud rate (page 255-256). Just remember that the CPU frequency is 16 MHz.

Tasks:

- Create a new C project in Atmel Studio.

During this exercise the data sheet may need to be consulted, specifically the pages 258-264. Do not worry, you will not need to read everything.

Home Assignment 5
How do you enable the transmitter and receiver for USART0?

Home Assignment 6
How do you set the data length to 8 bits and 1 stop bit for USART0?

Home Assignment 7
How do you set the baud rate for USART0? You are free to choose any applicable value.

- Implement a "`void usart0_init()`" function using the answers to the above home assignments.

Home Assignment 8
How do you read the received data from the USART0? Remember that you need to wait until there is data to read.

- Implement a "`uint8_t usart0_receive()`" function.

Home Assignment 9
How do you transmit data via USART0?

- Implement a "`void usart0_transmit(uint8_t)`" function.

Maybe there are some examples in the data sheet (page 244 and 247) for receiving and transmitting data. But remember, a correct explanation of all code is required to pass the laboratory exercises.

- Watch the following video to configure YAT properly: `https://youtu.be/qrL23q0g4VU`. Make sure that all the boxes are marked/unmarked according to the video.
- In YAT, enter your baud rate, the number of data bits, and how many stop bits that are going to be used.
- At this point, it is time to check if the transmit and receive functions work as intended. This can be done by sending back (echo) the data that is received (from the microcontroller's point of view). Write an application that echoes back the received data, using your implemented functions.
- Run the application on the microcontroller and send a character from YAT. You should see the character printed twice in YAT (why?).

Lab Question 2
What happens if the baudrate setting in YAT is changed?

- Modify your application to turn an LED on and off based on the character sent from YAT. Choose any characters of your liking.

Lab Question 3
Try sending a non-printable character, such as "\x00" or similar. Is it still working?

You are now done with this part, show your work to a lab assistant! _____ ☑