LAB 4, PWM y ADC

1. LAB 04, Part A

1.1. Basic output: PWM

1.1.1. Summary

This lab introduces another basic output operation on MicroController Units (MCUs): PWM. PWM stands for Pulse Width Modulation and allows a pseudo analog output, i.e. the output of an analog voltage between 0V (GND) and your operating voltage (VCC) rather than just 0 or 1 (false or true / GND or VCC). This allows a huge variety of task, e.g. setting the speed of a motor or the brightness of an LED instead of switching it just on or off.
Your task is to properly configure one PWM unit of the microcontroller and to connect an LED to it. Then you shall slowly fade the brightness of this LED between 0 (off) and its maximum value (fully on) and back to zero. Repeat this pattern infinitely.

1.1.2. Objectives

By the end of the lesson, you shall have written a program which controls one LED. This LED shall fade between its minimum and maximum brightness.

1.1.3. Duration and evaluation

One lab of 80 minutes. You have to present the working software and your source code to the supervisors until the end of the lesson. Prepare to explain your code and the choices you made (PWM mode, PWM frequency, prescaler mode,) to the supervisors upon request.

1.1.4. Further Information

The microcontroller used in this lab is an ATmega328P on a developer board called ATmega328P Xplained mini. LEDs, resistors etc. are available as in the previous labs.

Also you can use MSP430F5529 from Texas Instruments.

The PWM is implemented as a socalled peripheral module . Peripheral modules add functionality outside of the CPU core and its digital arithmetics. Strictly speaking, even the GPIO functionality can be seen as a peripheral module each Port A, Port B etc. is one instance of several similar digital I/O peripheral modules. Peripherals are controlled by

means of special registers.
The direction (DDRB), output (PORTB) and input (PINB) registers are examples for that. Similarly, the PWM module is controlled by a set of registers. They are documented in the datasheet (see recommended reading).
The PWM module can operate in several different modes among them normal (no PWM, classic GPIO function) or fast PWM. There are more. In each mode, there are several behaviours, such as phase correct or frequency correct PWM. In addition, the PWM can operate at a certain base frequency or vary its frequency and it has additional input prescalers etc.
You have to read about the PWM module in the datasheet and understand what all of this is about. Recapitulating the according lecture will be very helpful in that.
Decide which mode and which behaviour you want to use, or what would be the effect of the different modes. Then find out which registers you have to set to which values in order to set these parameters.

1.1.5. 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.

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

1.2. Task1: Fade a LED between zero and its maximum brightness

1.2.1. 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.
- You will need a plan on how to connect the LED and resistor to the microcontroller. The same way as the last time should work.
- Decide which PWM mode and which behaviour you want to use. Why would you chose this one?
- Find out which registers you have to set to which values in order to set these parameters.
- How would you implement the fading effect? Is there an easy way to make it fade slowly between its two maximum points?

1.2.2. Class work

You are supposed to connect the LED to the microcontroller. Then write a program fulfilling above requirements. The result and the source codes must be shown to a supervisor.

1.2.3. CLI instruction cheat sheet

```
avr-gcc -Wall -Wextra -Wpedantic --std=gnu99 -mmcu=atmega328p -Os -o PWM.elf PWM.c

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

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

2. LAB 04, Part B

2.1. Basic input: ADC

2.1.1. Summary

This lab introduces another basic input option for MicroController Units (MCUs): ADCs. ADC stands for Analog to Digital Converter and allows to digitalise an analog input value, i.e. to obtain a number representing the voltage present on a certain pin. Once could say it "measures the voltage" on this pin.
NTCs are temperature variable resistors that can be used for measuring the ambient temperature. You are supposed to measure the voltage of such an NTC and to visualize the measurement result by varying the brightness of an LED (via PWM) and controlling a bar graph.

2.1.2. Objectives

By the end of the lesson, you shall have written a program which uses the ADC unit to obtain an 8 bit representation of the voltage applied to its input. The external circuit shall incorporate the NTC in a way that almost its entire temperature range from 40C to +100C can be measured. A LED attached to an analog output (8 bit PWM) shall represent this value (duty cycle = measured input voltage). The working hardware and the source code have to be shown and explained to the supervisors until the end of the lesson.

2.1.3. Duration and evaluation

One lab of 80 minutes. You have to present the working circuit, the running software and your source code to the supervisors until the end of the lesson. Prepare to explain your code and the choices you made (ADC mode, ADC frequency/prescaler mode, voltage reference, schematic for connecting the NTC) to the supervisors upon request.

2.1.4. 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.

Importan information: you have to use a different microcontroller from the part A.

The ADC is implemented as another peripheral module. The ADC unit in this microcontroller has 8 or 10 bit resolution and ranges from GND to VREF. VREF is the topmost

reference voltage and can be selected within certain limits. It can e.g. be applied externally on the VREF pin, or one of various internal voltages of the microcontroller can be selected as VREF. Hence, the bits of the digitalized ADC value are spread across this reference voltage and the resolution per bit varies with VREF and various other factors.
NTCs are Negative Temperature Coefficient resistors. This means that their resistance drops when they get warmer, and increases when they get colder. Typically, their nominal resistance is the one they have at 25C. The lab component kit includes a 10kOhm NTC type Vishay NTCLE413E2103F520L.
By integrating an NTC into a voltage divider, their properties can be used to measure the ambient temperature of this NTC together with an MCU this forms a digital thermometer! Your task is to build such a voltage divider and to properly configure the ADC of the microcontroller to measure the ambient temperature. Furthermore, you have to configure one PWM unit of the microcontroller and to connect an LED to it. The PWM value which determines the brightness of this LED shall be proportional to the measured temperature (the warmer it is, the brighter the LED shines). Design your circuit and code in a way that almost the full temperature range from -40C to +100C is covered. Equipment to sufficiently cool and heat the NTC for tests will be available.

2.1.5. Recommended reading

Atmega328 [3]**:** 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 [4]**:** provides the full developer board hardware documentation.

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

2.2. Task1: Fade a LED depending on the measured temperature

2.2.1. 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.
- You will need a plan on how to connect the LED and resistor to the microcontroller. The same way as the last time should work.
- You will need a plan how to connect the NTC to the ADC input of the microcontroller. What circuit would you use? Why? Think about the values of involved components (e.g. resistance, voltage).
- Learn about the ADC unit in this microcontroller. It has various modes and settings. Which ones would suit this application best? Find out which registers you have to set to which values in order to configure the ADC as desired.
- Doublecheck what you envisioned: Can your circuit really operate almost through the entire range from 40C to +100C? How will the input voltage on the ADC change throughout these temperatures? At which temperatures will the LED hit its darkest and brightest limits? If it doesnt yet: Are there ways to improve your design to really span this entire range?

2.2.2. Class work

You are supposed to connect the LED, NTC and necessary extra components to the microcontroller. Then write a program fulfilling above requirements. The result and the source codes must be shown to a supervisor before the end of the lesson.

2.3. Task2: Display the result on a bar graph

2.3.1. Required preparation

- All of task 1
- Be able to connect 6 additional LEDs to the microcontroller and control them (no pitfall, it is as easy as it sounds).
- Think of useful thresholds / steps to partition the ADCs value range: When should 1, 2, 3 all 6 LED light up? Is it a good idea to have an LED only turn on/off on the absolute maximum/minimum value? Why / why not?

2.3.2. Class work

Extend the code from task 1 to also control this bar graph. On very low temperatures, only one LED shall light up. On Very high temperatures, all LEDs shall light up. Partition the temperature range between these two points in a roughly linear manner. The result and the source codes must be shown to a supervisor before the end of the lesson.

2.3.3. CLI instruction cheat sheet

```
avr-gcc -Wall -Wextra -Wpedantic --std=gnu99 -mmcu=atmega328p -Os -o
ADC.elf ADC.c

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

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