HOCHSCHULE
RHEIN-WAAL
Rhine-Waal University
of Applied Sciences

LABORATORY

Microcontroller

8

EXPERIMENT:

Temperature Data Logger

Please fill in your name to every sourcecode file you edit in this lab. All files should contain all names of the persons who made changes. Use the @author tag for this in the headline of the templates.

1 Temperature Data Logger

A data logger is a devices, that records data. Here we want to create a temperature data logger.

This devices are used in the commercial environment, for example to make sure that a cold chain for foods was not broken.

A temperature data logger needs to do the following jobs:

- Measure the temperature
- Know the current time/date (Real Time Clock)
- Write the measured value to a memory, together with the time/date
- React on buttons to clear the memory and restart the data logging
- Output the logged values:
  - to a display
  - with an interface to a Personal Computer

There are all needed parts inside the MyAVR set to build a temperature data logger.

For the memory we use the EEPROM add-on module, for the real time clock the DS1307 add-on module.

To create the hardware please connect:

- **Key 1** (Button Read): PB0
- **Key 2** (Button Write): PB1
- LCD Display **add-on**
- EEPROM **add-on**
- RTC **add-on**

1.1 Temperature sensor

The temperature can be measured with an internal temperature sensor. The Atmel Mega88PA has this function as a part of the analog to digital converter.

The ADC has a resolution of 10 Bit only. This is very bad for the temperature sensor. To have a better resolution use oversampling.

Use 128 measurement cycles for the oversampling and try to get two more bits for the resolution. For this job, the ADC should not be set to autorun (autoupdate) mode!

The sensor inside the chip must be calibrated to use it. It is not necessary to measure very accurate here. Just calculate a simple offset to have a good result on room temperature (approx 22°C in the lab). A change of $1mV$ correlates to a change of 1°C.

The offset voltage is individual for every microcontroller chip!

1.2 Data storage

To store the data we have just one small EEPROM with 256 Byte (24C02). So it is not a good idea to store every temperature value together with the hole time/date. This would use too much memory.

Better is to store just the starting point (time/date), the number of recorded temperatures and after that the temperature values itself.

So we need a fixed memory of:

- Number of values, 1 Byte
- Year, 1 Byte
- Month, 1 Byte
- Day, 1 Byte
- Hour, 1 Byte
- Minute, 1 Byte
- **Sum: 6 Byte**

And for every temperature value we need two bytes.

1.3 User Interface

To operate, the device should have two buttons. This buttons are for read and write.

After the device is powered it should be in standby. The LCD shows:

- the current time
- the current temperature

If the button write is pressed in standby mode the device should start to record the temperature. The recording should not start, if there is already data in the memory! Please record one value every minute. While recording the LCD shows:

- "Recording data"

- how many temperature values are recorded
- the current temperature

The record mode stops if the memory is full or if both buttons are pressed.

If the button read is press in the standby mode, the device should show if there is no data in the memory or how many data-sets are stored and the associated time.

With every next button press it should show the next value together with the time when this temperature was recorded. Pressing both buttons in read mode is to delete the data.

1.4 Task

Task 1:
Create the temperature data logger with the MyAVR hardware.

Task 2 (Optional):
Append the program to output the recorded data via the UART (serial) interface. The output should be like this:
2018/05/29 12:01 25.0
2018/05/29 12:02 25.1
2018/05/29 12:03 25.0
Please use 9600 Baud for the serial speed.

To check if your output works there are two scripts in the template:

- MySmartUsbProg.sh
- MySmartUsbUart.sh

With this scripts the MySmartUSB on the MyAVR board can be set to UART mode and back to programming mode. If the MysmartUSB is set to UART mode, there are two green LED's on. In programming mode it is one red LED.

In the UART mode, use the program 'gtkterm' to connect to the microcontroller. Use 'make terminal' to start the program with the right options.