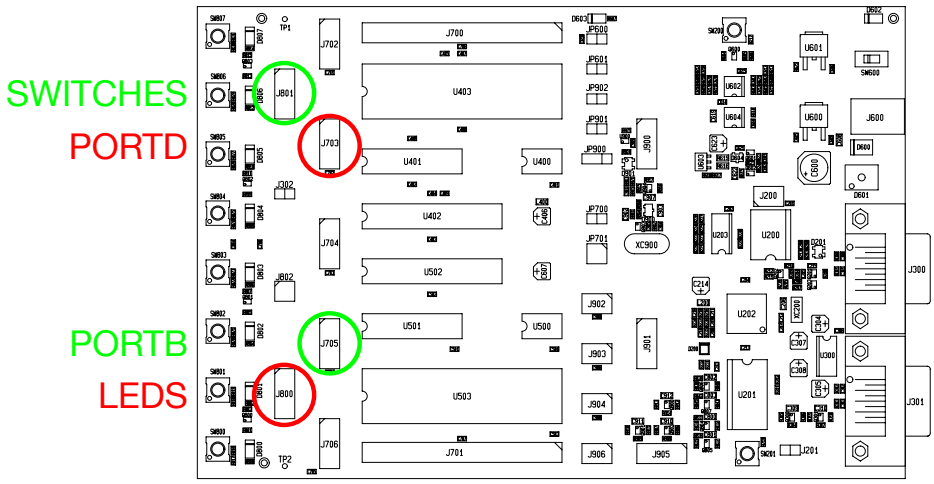
Eingebettete Vernetzte Systeme
Atmel AVR Experiment

Exercise 1 Let there be light

Connect the STK500 to your PC using the serial cable and the USB/Serial converter. The serial cable must be connected to port labeled with *RS232 CTRL*. Connect the pins with the label *LEDS* with the pins labeled *PORTD* and the pins of the switches to the *PORTB* pins. This is again shown in the schematic below. Dowload the skeleton code package *light.zip* from the course homepage and unzip it. In order to use the skeleton code you need the packages *avr-libc, avrdude, bintutils-avr* and *gcc-avr*. Further you need the rights to access the serial device e.g. */dev/ttyUSB0*. Edit the *CMakeLists.txt* and edit the *AVR_PROGRAMMER_DEVICE* variable to the real dev-file - if necessary. Connect the power-source to the board and turn it on using the power switch.



- Setup the build folder in the provided skeleton code and call *CMake*.
- To compile the program call *Cmake .. led*. After sucessfull compilation three files are generated *led-atmega168.elf, led-atmega168.hex* and *led-atmega168.map*.
- To deploy and run the generated code on the microcontroler you need to call *make upload_light*. All leds on the *STK500* should blink now.

Some suggestions about command:
sudo apt-get install avr-libc
sudo apt-get install avrdude
sudo apt-get install bintutils-avr
sudo apt-get install gcc-avr
sudo apt-get install cmake
(After you have downloaded the code files to the file folder Downloads)
cd Downloads/avrled/src

```
sudo mkdir build
cd build
cmake ..
cd ..
sudo chmod a+rw /dev/ttyUSB0
make upload_led
```

Exercise 2 Basic IO and the LEDs

Have a look at the official documentation of the AVR ATMega168 (http://www.atmel.com/images/doc2545.pdf). As you can see in the documentation the microcontroller offers 23 programmable I/O lines. The pins at the microcontroller are grouped into 8bit goups with the names B,C and D. Most pins can be used for reading and writing, which means that they either sense the current logical level at the pin or set it. To indicate how the pins are used there exist special data direction registers $DDR\{B,C,D\}$. If a bit is set the respective pin can be used for writing and if not the pin is indicated to be used for reading. To write a value the special defines $PORT\{B,C,D\}$ are available and to read the defines $PIN\{B,C,D\}$ exist.

- Open the *lettherebelight.c* in the source folder and try to understand what the code does. The function *_delay_ms* allows us to wait for specified amount of milliseconds. Can you toggle arbitrary leds?

- Change the example code to try to toggle the state of a single led and then toggle the next led to get an moving led light efffect. Use the following operations to set and unset single bits. You need to handle the case if you toggle the last led - you can either start again from the first led or you can go into the opposite direction. Try to implement both effects.

  Set bit N **in** A**:** $A = A \,|\, (1 << N)$

  Unset bit N **in** A**:** $A = A \,\&\, \sim (1 << N)$

- Change the data direction of the zeroth pin on B to use it as an input. There are defines to simplify the usage of the pins and also the code readability. E.g. the N th pin on port is also accessible by PBN. To check the current state of pin we can use for example $PINB \,\&\, (1 << PB0)$. Change your program so that the moving led behaviour is inverted while the button is pressed (There should be only on led that is off and it is moving).