18. april 2016

1 Kveld 1: LEDs and stuff

In this session we will get to know AVR Studio and hopefully compile our first program.

- Task 1 will take you through starting AVR Studio, opening a project, compiling it, uploading it and debugging it.
- Task 2 will force you to look in the datasheet for information and if gods willing make your board respond to switch presses.
- Task 3 will throw you into the arcane world of assembly and register addresses. But we will hold your hand.

Task 1

1. Connect the ATmega324PB to the USB port on your computer.
2. Start AVR Studio. You will now see a start page with information about the ATmega324PB kit.
3. Unzip the session1.zip file provided.
4. Go back to Atmel Studio. Go to File − > Open − > Project/Solution
5. Open the file Session1/code/Task1/Task1.atsnl
6. Have a good gander at the code. Guess its purpose.
7. Look in the datasheet on the pages referenced in the comments. Read a bit about I/O Ports in general.
8. Going back to task1.c, press F7 to compile your project. No errors?
9. Connect as shown in the picture above.
10. Press the Play button (Start Without Debugging), or press CTRL + ALT + F5.
11. You will now have a blinking light on your Atmega kit.

Task 2

1. Open session1/code/task2/Task2.avrsln in AVR Studio.
2. Now you have to do some things to be able to read button presses.
3. Because of the way the buttons are connected, pressing a button will ground the pin the button is connected to, giving 0V and logical zero.
4. This is only interesting if the voltage on the pin is something other than 0V to begin with, otherwise there will be no change to detect.
5. Insert code to activate the internal pull-up (connecting the pin to 5V VCC through an internal resistor), ensuring a positive voltage on the pin.
6. Read in the datasheet, especially under Ports and Register Description to get a quick overview over capabilites.

7. Note that, since the button pulls the pin to earth, a pressed button will read as LOW, whereas a released button reads as HIGH.

Task 3

1. Open session1/code/task3/task3.avrsln in AVR Studio.
2. This time, the code is slightly more complex. Before looking at the code, think about how you would implement this
3. You will want to have an infinite main loop, that checks and logs the button's state. If it has changed, you will want to see if it was pressed or released. If it was pressed, you will want to toggle the LED
4. Notice that the light sometimes bugs. Why is that? (hint: the processor clocks really fast, and the button is a mechanical device.)
5. In order to get rid of this bouncing, you will have to implement anti-bouncing. How can you be sure that the button has settled in a final state? (hint: use a delay)

Task 4 (if you got time)

1. Open session1/code/task4/task4.avrsln in AVR Studio.
2. This time, we will use the extension board OLED1 Xplained Pro
3. Connect the OLED1 to the EXT1 on the ATmega324PB.
4. Read the user guide for the OLED1 extension card to see which pins the different LEDs and buttons are connected to.
5. Read the ATmega324PB datasheet to see which ports and pins on the microcontroller that is connected to the different pins coming from the OLED1 on the EXT1.
6. Try to map the different buttons to the different LEDs on the OLED1 extension card.
7. Make the LEDs blink on the buttonpresses like you did in Task 2.