ATMega 128 Microcontroller

This tutorial should give an overview about the Atmel ATMega128 Microcontroller and its features. This device should be used as the main controller and CPU of the CanSat.

Microcontrollers in general:
Microcontrollers can be thought as CPUs in computers. They can execute commands, evaluate expressions and therefore able to run some code to perform various tasks. They consist of CPU, memory and integrated peripherals. The main differences between computers and microcontrollers are speed and instruction set. Usually microcontrollers are much slower than a CPU in a state of the art computer (their clock frequency is in the order of 10MHz, while this parameter is in the GHz region at computers). Their instruction set is usually smaller (reduced) which means they can’t execute very difficult commands in one step, but this should be done by breaking down the difficult operations into elementary ones, which can be executed.

In order to operate a microcontroller it should be programmed with some code, which describes the operation in detail and the corresponding output responses for the input excitations. This code is stored in the memory (usually flash memory) of the microcontroller and is executed right after power on. If the memory is empty, the device will not execute anything, therefore this memory almost always contains the application and/or the boot loader.

Boot loader: This is a special code, running on the microcontroller and it initializes some of its interfaces (usually communication interfaces) in order to provide some basic functions which are necessary during development. In our case the boot loader sets up the UART interface of the microcontroller and provides access to the memory which can be loaded with the operational code after start-up.

Interrupts: These are special signals which are used to sign to the CPU that something happened at a peripheral and it needs attention. This task could be performed by polling as well, but that would be more time consuming. There are two types of interrupts according to the peripheral it comes from. If it is from an integrated peripheral it is called internal interrupt, otherwise external interrupt. In general there is no difference between executing these, and its done as follows: a device generates and interrupt request (usually just called interrupt), the CPU decides whether this interrupt should be served right at the moment or should wait. After accepting an interrupt the CPU halts its operation and jumps to the code which is assigned to the certain type of interrupt in the interrupt services register. After completing the operation it returns to normal mode and continues to execute the general code, where it halted. Interrupts can be used to implement event driven behavior.
ATMega128:
The provided microcontroller is an ATMega128 with clock frequency of 14.7456MHz, programmable flash memory of 128kbyte and with various types of integrated peripherals (see below).

It can be programmed in C, using the preferred AVR Studio development environment with WinAVR compiler and libraries. The code can be downloaded to it via RS-232/UART after start-up. The boot loader initializes the serial interface of the microcontroller and waits for connection with the AVR Studio’s programming tool (AVR Prog) for about a second. If there is no connection to the tool, it continues to execute the main code found in the memory.

While programming the compiled .hex file should be downloaded to the flash.

Peripherals: The following integrated peripherals are available in the ATMega128, or on the Crum128 board.

1. General I/O ports: These ports can be programmed to operate as digital inputs or outputs. There is a data direction register for each pin of these ports where the intended feature can be set. If the DDR contains logical one for a certain pin, it will be an output, if the content is zero it is an input. Further operations should regard the functionality of the port, therefore an input should only be read and the output should only be written (set).

2. UART: Serial port for general purpose and for programming the microcontroller. It has 2 separated UARTs, UART0 and UART1. Both of them by default are 5V TTL level ports. UART0 is connected to a MAX3221 level converter circuit which converts it to a true RS-232 level UART. UART1 is connected to a CP2101 circuit which converts this port to USB interface.

3. I2C: The microcontroller has an I2C controller which can be programmed to implement a bus. For further information on this bus refer to the datasheet of the device and the provided tutorial about protocols.

4. A/D Converter: Some ports can work as Analog to Digital Converters and the result of the comparison can be accessed internally. The sample rate and the reference signal can be set as well.

5. SPI: The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega128 and peripheral devices.

6. JTAG: This port of the microcontroller follows the JTAG standards and therefore it is intended to be used for testing debugging.