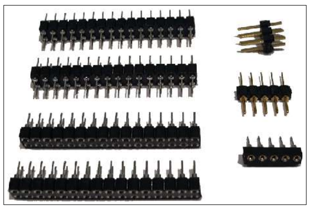
Crumb128 V2.0

Rapid Prototyping Module with ATmega128 µController.

Crumb128 is a low-cost, easy to use and small-formfactor module combining Atmels ATmega128 AVR microcontroller with a standard serial port with RS232 transceiver, USB2.0 (full speed) device interface, reset protection circuitry, reset jumper, status LED, standard 6-pin InSystemProgramming (ISP) header and all ATmega128 signals on two 32pin headers.

A preinstalled firmware (ATmegaBOOT) provides an SKT500 compatible bootloader, hence no ISP adapter is required for programming the ATmega128. A serial crossover cable connects the serial port to a PC running the programming software (e.g. AVR-Studio, uisp, avrdude, etc.). ATmegaBOOT can also be used for low level hardware tests of the ATmega128 microcontroller and peripheral circuits.

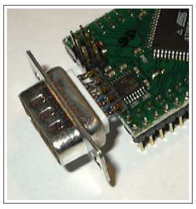


Connector Kit – Crumb128 is being shipped without pin headers mounted, since everybody has it's own favorates (pins male/female facing up/down or 90° angled, etc.). A set of high quality pin headers and receptacles are available as a Crumb128 Connector Kit (see picture).

Power Supply – Crumb128 requires a regulated +5V DC power supply. See pinout diagram for VCC pins (both VCC pins are internally connected). The AVCC pin is connected to VCC through coil L1 and decoupled by capacitor C5, hence it should not be connected externally.

System Reset – Reset circuitry C11, R7 and D3 provides a proper reset signal after power up and provide protection against irradiation. By closing jumper J1 a manual reset can be triggered.

System Clock – The CPU clock is generated by a 14.7456MHz crystal (X1) for error free common serial baudrates. An additional 32.768kHz asynchronous timer crystal can be used for asoftware realtime clock.



Serial Port – RS232 level signals are provided by the onboard MAX3221 (U2) RS232 transceiver, connected to UART0 of the ATmega128. U2 can be enabled/disabled by jumper J2, which should be set to 1-2 (disabled) or 2-3 (enabled) position before operation. The RS232 signals are available at CON4, which can be connected 1:1 to a male Sub-D-9 connector (see picture).

USB Port – The USB2.0 (full speed) device interface is based on the CP2101 chip by www.silabs.com. It converts TTL UART signals (ATmega128 RXD1/TXD1) to USB. A royality-free virtual COM driver is available for download at www.chip45.com. For more information about the USB interface see application note AN001 at www.chip45.com.

Status LED – LED1 is hardwired to pin 17 (OC2 OC1C PB7) of the ATmega128 and can be used by the application as a general purpose status indicator. The bootloader flashes the LED three times after power up.

In-System Programming (ISP) – CON3 is the standard 6-pin Atmel AVR ISP connector and can be used with most ISP adapters (e.g. stk200/300 compatible adapters like Crisp-LPT by chip45.com or Atmel's stk500) and software (e.g. uisp, avrdude, PonyProg, etc.). Keep in mind, that the ATmega128 shares the ISP signals PDI/PDO with UART0, hence serial communication might be blocked as long as the ISP adapter is connected.

Memories – The ATmega128 provides 128kbytes of onchip, non-volatile Flash memory for program code storage, 4kbytes of onchip application SRAM and additional 4kbytes of onchip non-volatile EEPROM memory.

Fuse Bits – The ATmega128 fuse bits are preset to the following values: high byte = 0xc8, low byte = 0xdf, ext byte = 0xff. Changes to the factory default are: ATmega103 compatibility mode disabled, OCD and JTAG disabled, CKOPT, preserve EEPROM during chip erase, 8k boot block, boot reset enabled, 4ms startup, high freq. oscillator.

Bootloader – The preinstalled bootloader ATmegaBOOT can be used to download application code into the ATmega128 flash memory via either of the UARTs, which means either via RS232 or USB.

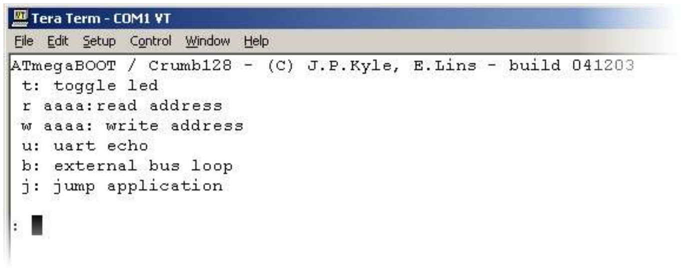
After power up, the status of PF7 and PF6 selects if the bootloader is being entered and which UART to be used for download (PF7 pulled low = UART0/RS232, PF6 pulled low = UART1/USB). If the flash memory is empty, i.e. no application has been programmed yet, the bootloader starts anyway. If neither PF7 or PF6 is set, UART0/RS232 is chosen by default. When the bootloader has started the onboard LED flashes three times.

When connecting the Crumb128 to a PC's serial or USB port, the bootloader acts as a standard stk500 ISP adapter, hence most ISP software tools (e.g. AVR-Studio, uisp, avrdude, PonyProg) detect the bootloader as stk500 and can be used in the usual way. Default COM parameters are 115200bps/8N1/no handshake.

chip45.com - Dr. Erik Lins
Development and Distribution
of Hardware and Software

35440 Linden/Germany
Email: info@chip45.com
http://www.chip45.com

The bootloader also provides low-level test features when pressing '!' three times, after the bootloader has started (LED flashed). The LED is switched on, a welcome message is being displayed and the following commands are available: 't' = toggle onboard LED, 'r aaaa' = read byte from memory, 'w aaaa' = write byte, 't' = toggle led, 'b' = external bus loop, 'u' = uart echo and 'j' = jump to application start (0x0000).



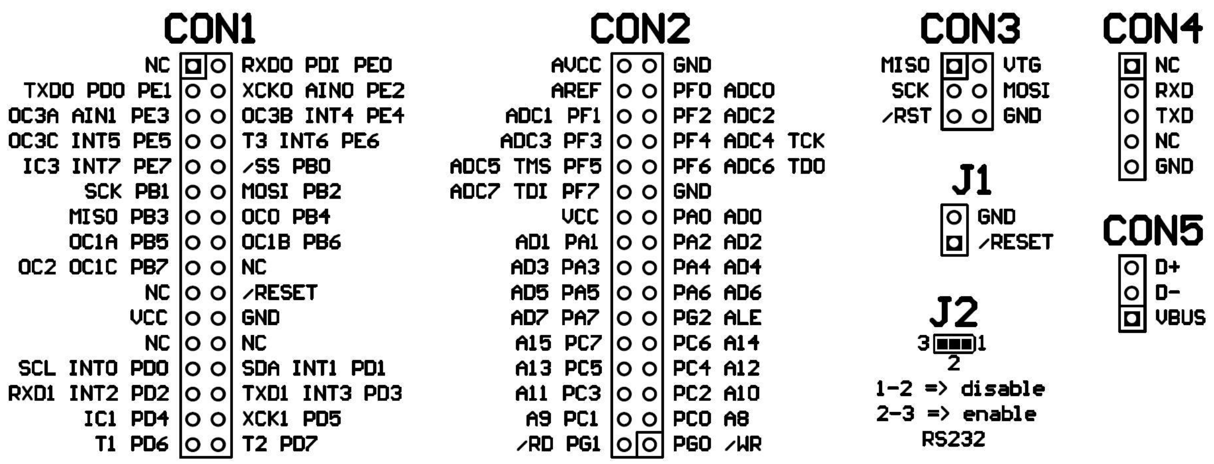
```
ATmegaBOOT / Crumb128 - (C) J.P.Kyle, E.Lins - build 041203
 t: toggle led
 r aaaa:read address
 w aaaa: write address
 u: uart echo
 b: external bus loop
 j: jump application

:
```

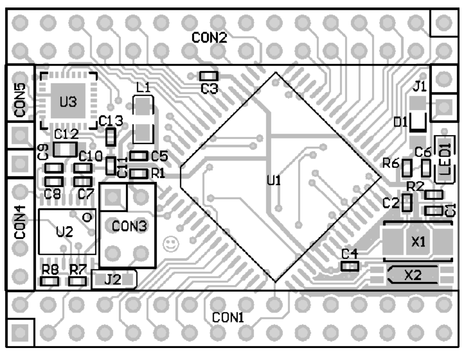
Note: The bootloader cannot be used to alter fuse bits, neither is it possible to overwrite or reinstall the bootloader by itself!!! An AVR ISP adapter has to be used for this purpose (e.g. Crisp-LPT by chip45.com).

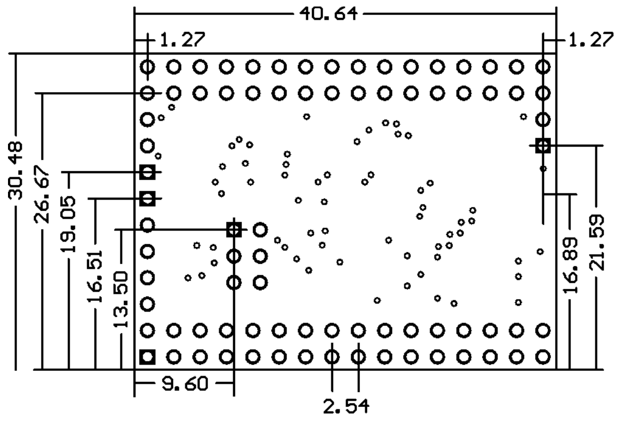
Note: The JTAG interface is disabled by default, since the bootloader uses PF6/PF7 for initial configuration.

Connectors - All connectors have standard 2.54mm spacing. CON1/2 provide all ATmega128 signals, CON3 is the ISP header, CON4 and CON5 provide the RS232 and USB interface signals.



Board Layout and Dimensions – Crumb128 is a ~40x30mm² double-sided FR4 PCB (1.6mm). Header holes have 0.9mm diameter, hence most standard round or rectangular pin headers can be mounted. See pictures for details.





Development Tools – Crumb128 is based on the ATmega128 AVR microcontroller, which can be programmed either in assembler (e.g. the original AVR Studio by Atmel: http://www.atmel.com/avr) or with several high level languages, including C/C++, Pascal or Basic. There exist several commercial C/C++ compiler suites (e.g. IAR Embedded Workbench or CodeVisionAVR) as well as the WinAVR GNU C/C++ compiler and tools suite (see http://winavr.sourceforge.net for details, the bootloader of Crumb128 was developed with WinAVR). A suitable and reasonably priced Basic compiler is BASCOM-AVR by http://www.mcselec.com. For a good Pascal environment please go to http://www.e-lab.de.

Further Information – Application notes and data sheets of the onboard components as well as the schematics can be downloaded at http://www.chip45.com. The official Atmel AVR homepage is http://www.atmel.com/avr. A valuable source of information dedicated to AVR microcontrollers is http://www.avrfreaks.net.