

## Testing AVR universal bootloader on Atmega128\_

<http://www.scienceprog.com/testing-avr-universal-bootloader-on-atmega128/>

After project [source code](#) is developed there is always need to flash it to microcontroller. There are few ways to program AVR microcontrollers. Usually we used to flash AVR's with [ISP](#) adapter (or other programmer) that is not always handy especially when designing device for the end user who doesn't have ISP adapter or doesn't know much about flashing MCU. So it is better to flash a bootloader program AVR MCU once with programming adapter and later load [firmware](#) interactively when starting AVR.

Bootloader not only allows to update firmware without programmer but also enables to load different programs for different purposes depending on situation flexibly. But enough about this.

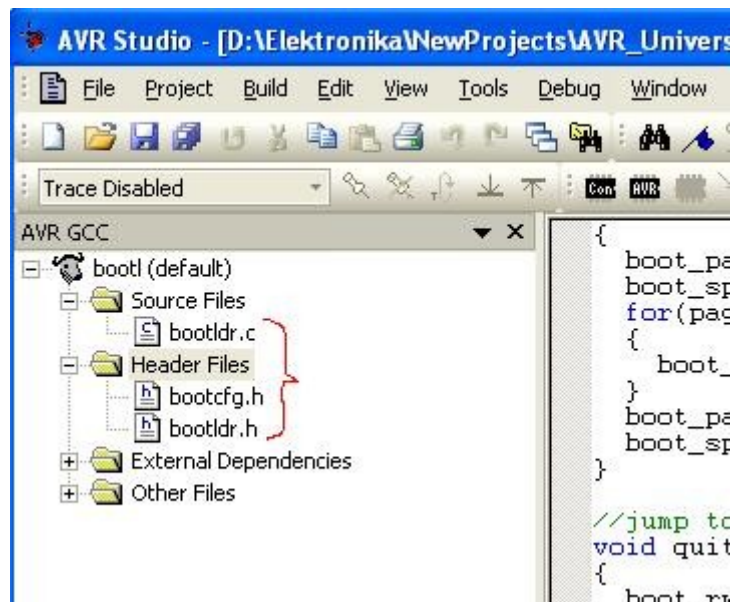
So my purpose today is to test AVR universal bootloader which is being developed by [Shaoziyang](#). His purpose was to develop universal bootloader that works on different types of AVR microcontrollers with minimal code modifications. Bootloaders you can find on the [Internet](#) are mostly available for special types of microcontrollers and nobody wants to do a lot of modifications to adapt to different MCU when needed.

This AVR universal bootloader can support most types of AVR microcontrollers (Mega series), which have self-programmable capability, boot section and UART. If the device have many serial ports, you can use any one of them. Bootloader supports RS232, RS485 and RS422. It can also supports USI, SPI, I2C interface if you made some modify. This bootloader occupies less than 1k words flash maximum, and it may occupy less than 300 words minimum. The space occupied by the bootloader is relative with device type, configuration parameters, functions you select, and the optimize grade that the compiler use.

Main features of bootloader are:

- Support many types of AVR microcontrollers;
- Support AVR device with multi-uart;
- Support RS232/RS485/RS422 mode;
- Support customize communication baud rate and system clock frequency;
- Automatically calculate the baudrate error:  
Write with AVR GCC, 100% C code;
- High optimized code, occupy small space;
- Cut out the function conveniently, and can meet different requirements;
- Support Watchdog;
- User may use the LED to show the upgrade status;
- Support to use super terminal as [download software](#) on PC;
- Support verification while write to FLASH;
- Can define the size of user program section;

I have Atmega128 board from [Piconomic Design](#) on my desk and I am going to try bootloader on it. First of all we need to download bootloader files from authors [site](#). You only need bootloader source files ([download](#) latest 3.1 version) and win32 [application](#) to communicate with target ([download](#) latest 3.1 version). Once files are downloaded, extract them to any location and leave them.



Now create new project in AVRStudio and copy following bootloader files (**bootldr.c**; **bootldr.h**; **bootcfg.h**) to project directory and add them to project tree. Now its time to modify project parameters that are in **bootcfg.h** file so they would fit Atmega128 settings. There are quite few parameters you may change, but most of them are OK. Lets see what I did.

- Changed MCU clock frequency
  - `<#define F_CPU 7372800UL>;`
- Changed Bootstart section address according to datasheet
  - `<#define BootStart 0xFC00>;`
- as Atmega128 has two USART ports, selected USART0
  - `<#define COMPORTNo 0>;`
- Turned off watchdog
  - `<#define WDGEN 0>` of course you can leave it, but you have to preserve it in your main application so MCU wont keep resetting;
- As my board has LED so defined where it is connected so I could see bootloading process

`//enable LED indication`

`#define LEDEn 1`

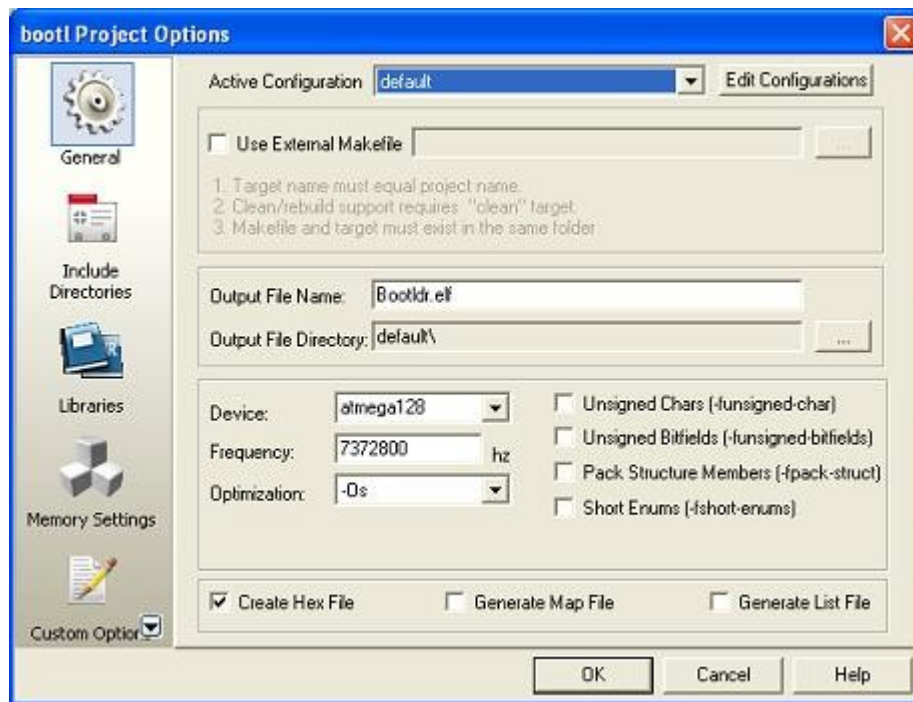
`//LED control port`

`#define LEDPORT B`

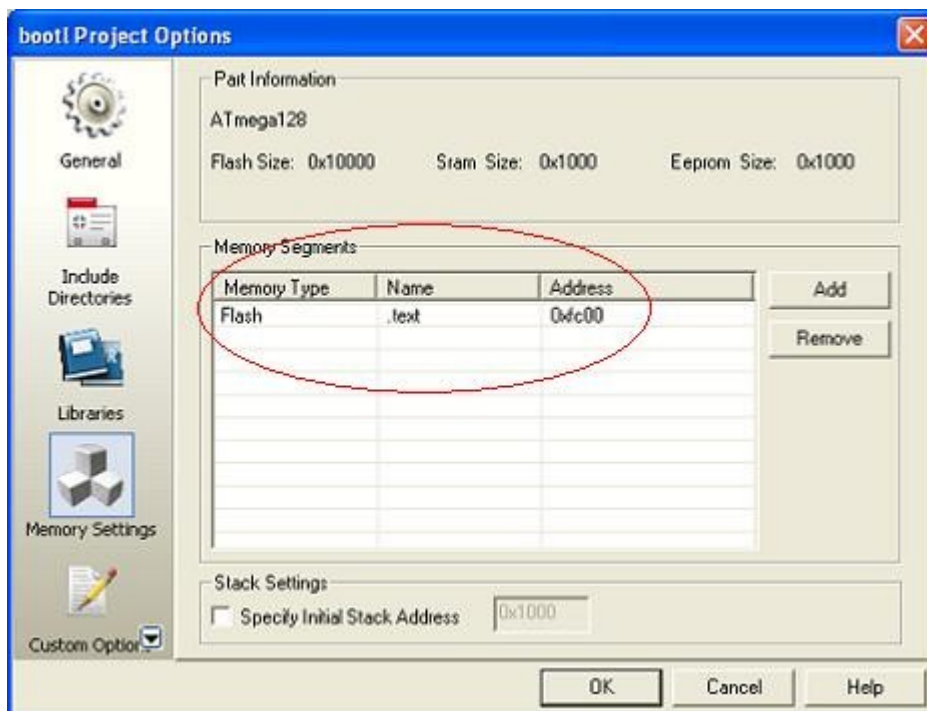
`#define LEDPORTNo PB6`

- Saved file – all modifications are done;

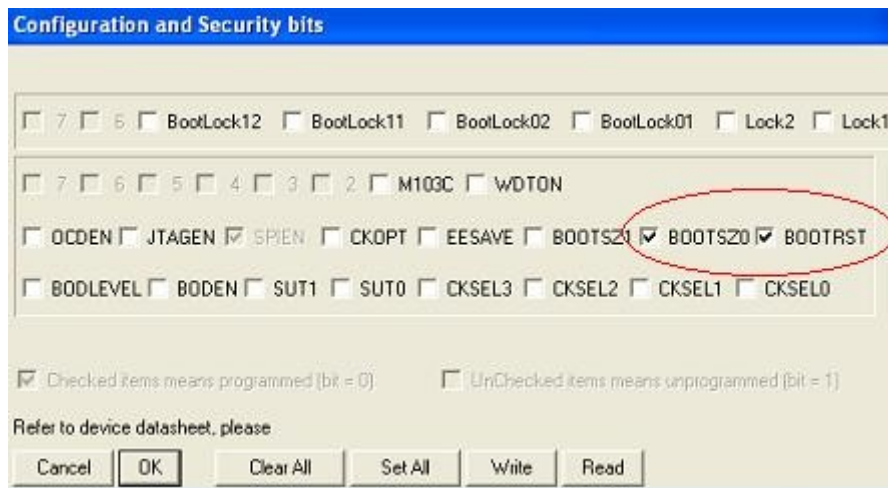
And there is last thing – we have to configure makefile to show what type of MCU is used, what frequency and where flash section starts. For this press “Edit Current Configuration Options” button in AVRStudio. First of all update *General* configs:



Then go to *Memory Settings* and create memory segment where bootloader starts. In my case it is 0xFC00:

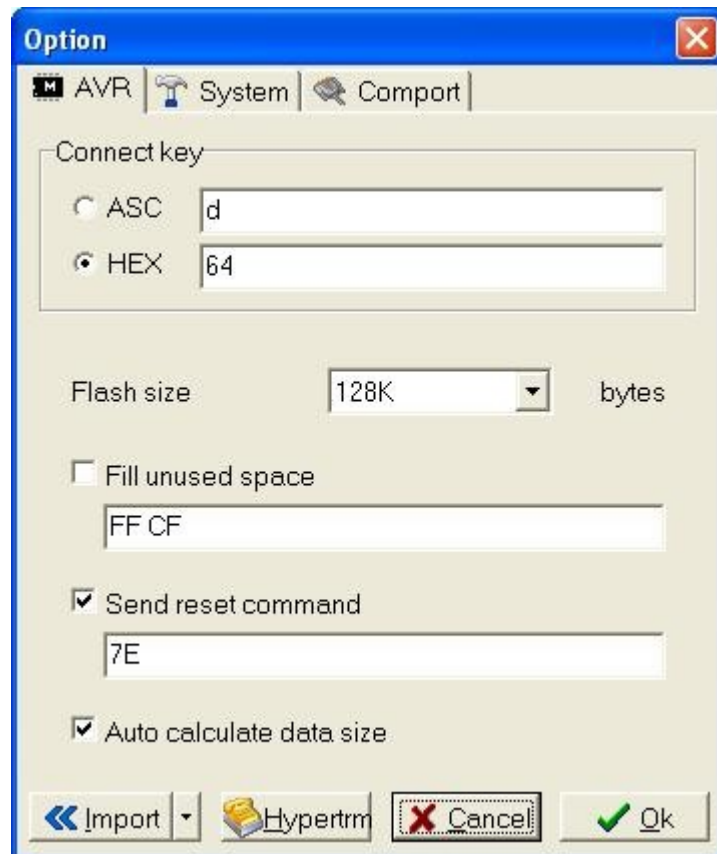


Then I built the project which took 1366 bytes (1.0% Full).  
 Flashed bootloader to Atmega128 via ISP and Ponyprog and then configured fuses for correct Boot section size and enabled Boot Reset Vector:

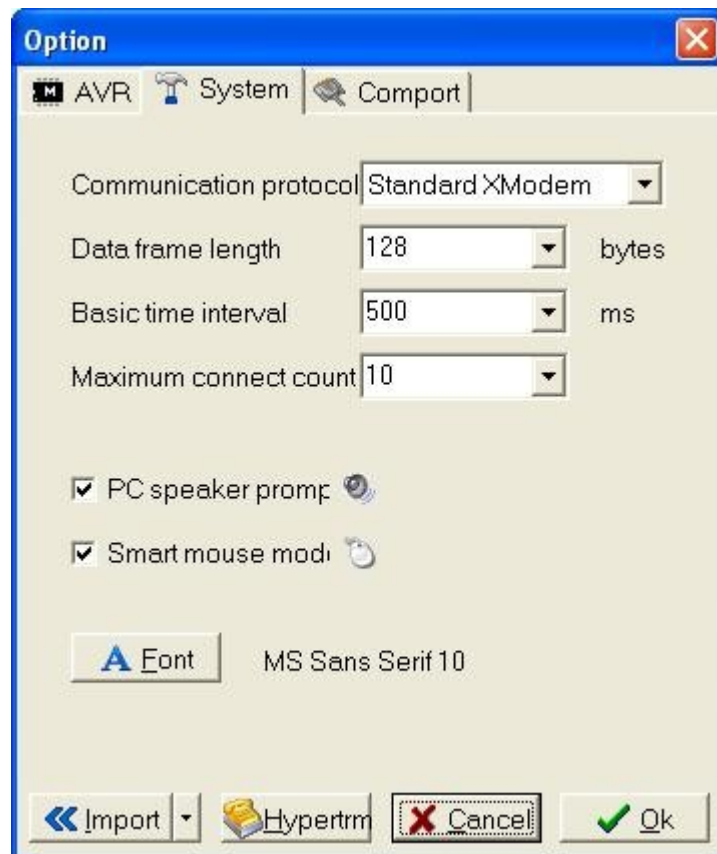


Checking BOOTRST means that bootloader after it finishes loading program resets to 0x0000 address where loaded program starts.

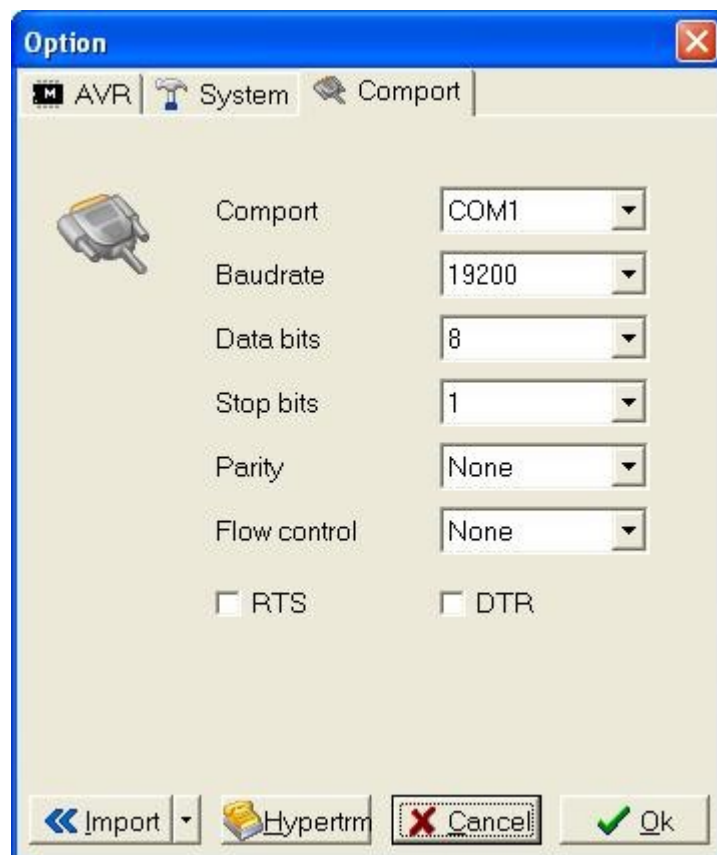
Now its time to connect target board to PC COM port and upload test program. For this open **avrubd.exe** application. And configure options so it could communicate correctly. For this go to Options->AVR and select:



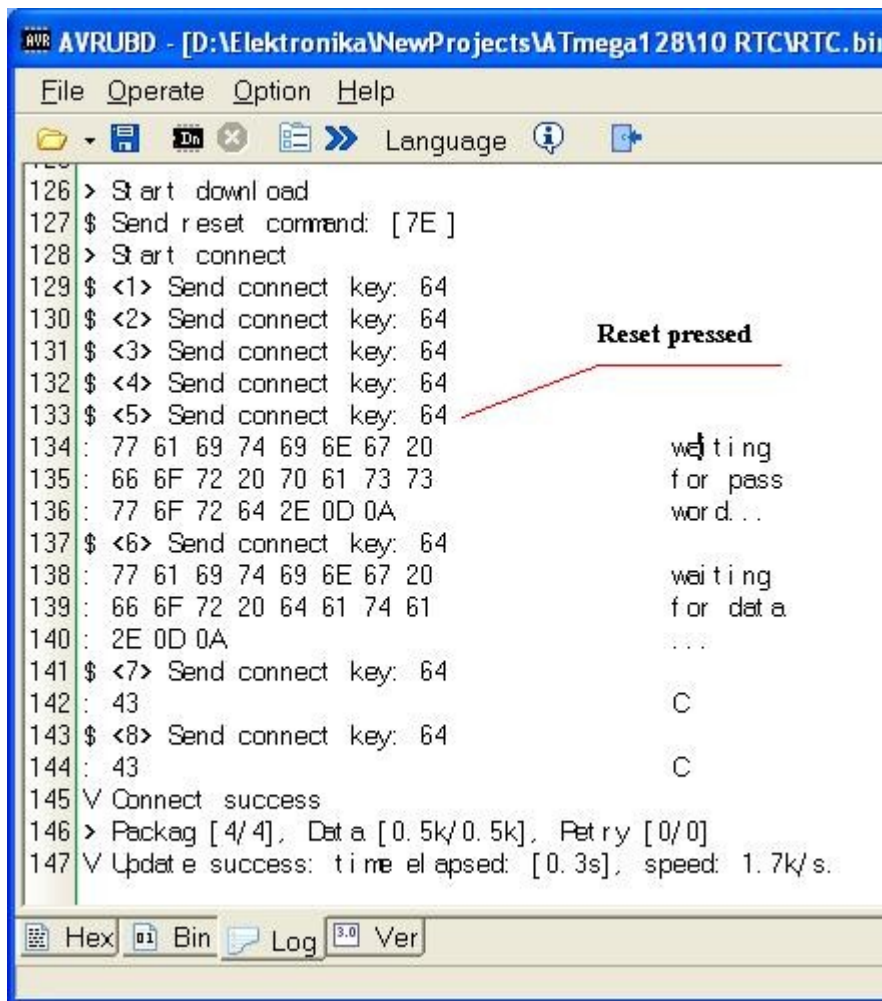
Options->System:



Options->Comport:



After options are set, open hex or bin file to be downloaded and press black **Dn** button. It will start sending connect key this means you have to reset target board in order to enter bootloader. If you wait too long – connection will fail. This is how successful [firmware update](#) messages looks like:



```
AVRUBD - [D:\Elektronika\NewProjects\ATmega128\10 RTC\RTC.bi
File Operate Option Help
Language
126 > Start download
127 $ Send reset command: [7E]
128 > Start connect
129 $ <1> Send connect key: 64
130 $ <2> Send connect key: 64
131 $ <3> Send connect key: 64
132 $ <4> Send connect key: 64
133 $ <5> Send connect key: 64
134 : 77 61 69 74 69 6E 67 20      waiting
135 : 66 6F 72 20 70 61 73 73      for pass
136 : 77 6F 72 64 2E 0D 0A         word...
137 $ <6> Send connect key: 64
138 : 77 61 69 74 69 6E 67 20      waiting
139 : 66 6F 72 20 64 61 74 61      for data
140 : 2E 0D 0A                     ...
141 $ <7> Send connect key: 64
142 : 43                            C
143 $ <8> Send connect key: 64
144 : 43                            C
145 V Connect success
146 > Packag [4/4], Data [0.5k/0.5k], Petry [0/0]
147 V Update success: time elapsed: [0.3s], speed: 1.7k/s.
Hex Bin Log Ver
```

In [documentation](#) text author says that bootloader has been tested with following AVR microcontrollers: ATmega8; ATmega64; ATmega128; ATmega168; ATmega169(Bufferfly); ATmega16; ATmega32; ATmega162. So it is really good [open source](#) bootloader project everyone can use easily.

---