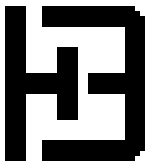


AVR

How to use

This document describes how to setup the STK500 board related to AVR programming.

Table of contents

1 SW development tools

1.1 AVR studio (4)

Used to program the device (since Atmel Studio 6 contains bugs, programming is done with AVR studio 4)
Location of install file: C:\Work\AVR\Tools\AVRStudio4_install

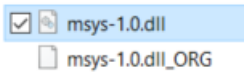
1.2 Installation of WinAVR

The winAVR (toolchain) contains the compiler and .h files.

1.2.1 Version 2004

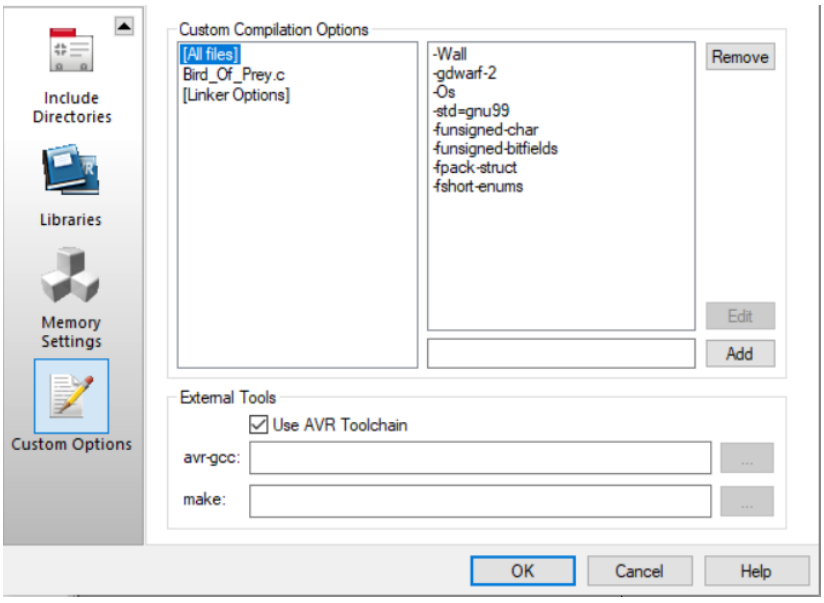
Step 1) Location of install file: C:\Work\AVR\Tools\winAVR2004, run the installer.

Step 2) Copy the file msys-1.0.dll from AVR\Tools\WinAVR2004 directory to the 'winAVR'\utils\bin directory

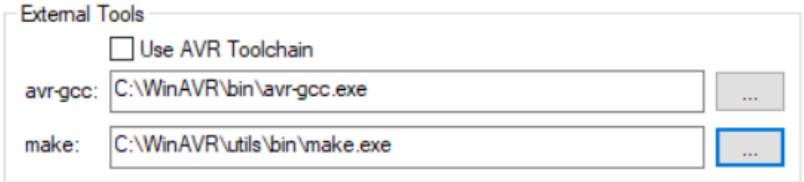


If this step is not done, then compiling a file will result in an error that the dep/... dependency file is not found.

Menu: Project > Configuration Options > Custom Options > External Tools: Use AVR Toolchain (or WinAVR)
The avr-gcc and make tools are by default not setup. See screenshot:



Uncheck the "Use AVR Toolchain" setting and browse to these locations:



Example code to test the toolchain:

```
#include <avr/io.h>
#include <avr/delay.h>

int main (void) {
	//Set PORTA to all outputs
	DDRA = 0xFF;

	while(1) {
		PORTA |=(1<<0);
		_delay_loop_2(10000);
		PORTA &= ~(1 << 0);
		_delay_loop_2(10000);
	}
}
```

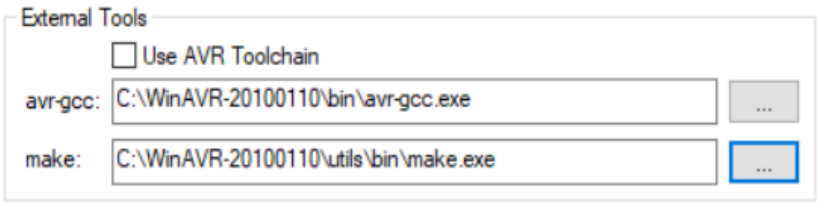
1.2.2 Version 2010

Step 1) Location of install file: C:\Work\AVR\Tools\winAVR2010, run the installer.
Manual: C:/WinAVR-20100110/WinAVR-user-manual.html

Step 2) Copy msys-1.0.dll from AVR\Tools\WinAVR2004 directory to the winAVR-20100110\utils\bin directory

Menu: Project > Configuration Options > Custom Options > External Tools: Use AVR Toolchain (or WinAVR)
The avr-gcc and make tools are by default not setup. See screenshot above of version 2004.

Uncheck the "Use AVR Toolchain" setting and browse to these locations:



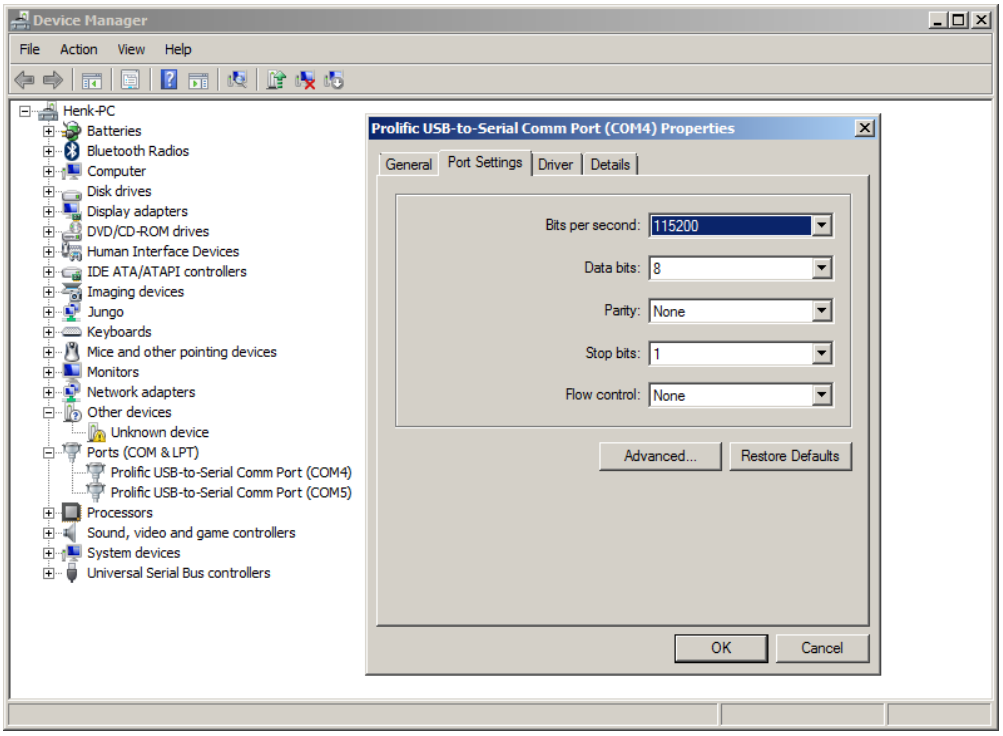
Example code to test this toolchain:

```
#include <avr/io.h>
#define F_CPU 1000000UL
#include <util/delay.h>

int main (void) {
	//Set PORTA to all outputs
	DDRA = 0xFF;

	while(1) {
		PORTA |=(1<<0);
		_delay_ms(500);
		PORTA &= ~(1 << 0);
		_delay_ms(100);
	}
}
```

Note: this toolchain contains functions like '_delay_ms()'.

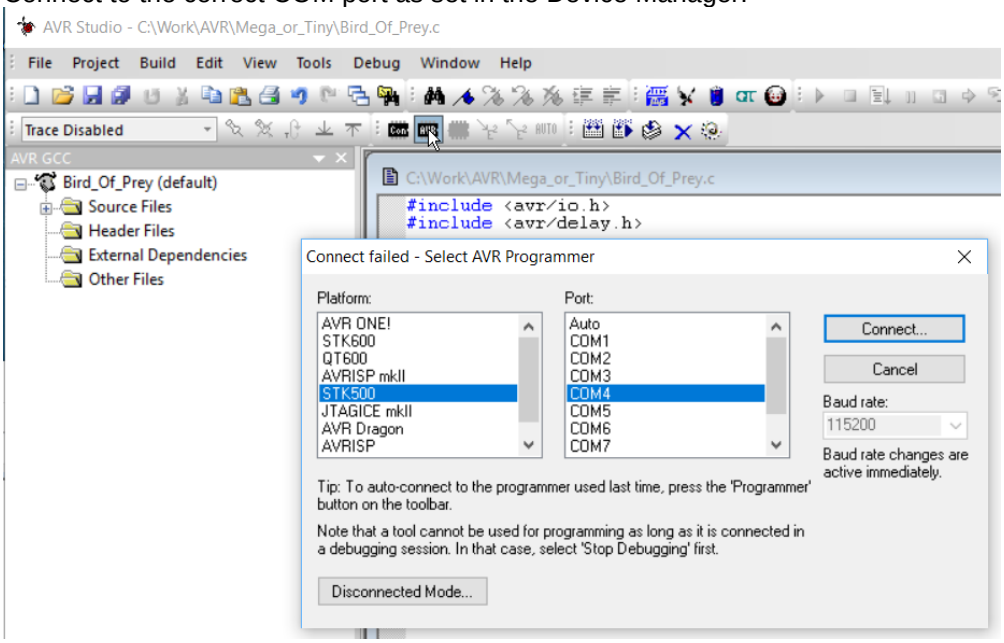
1.3 Serial interface

For serial communication a RS232 to USB converter cable is used. Install the drivers of this converter.
Open the “Device Manager” and per cable, set the port settings, in this case:
COM4, used for STK500 programming (RS232CTRL): 115k2,8,N,1
COM5: used for serial interface (RS232SPARE): 19k2,8,N,1

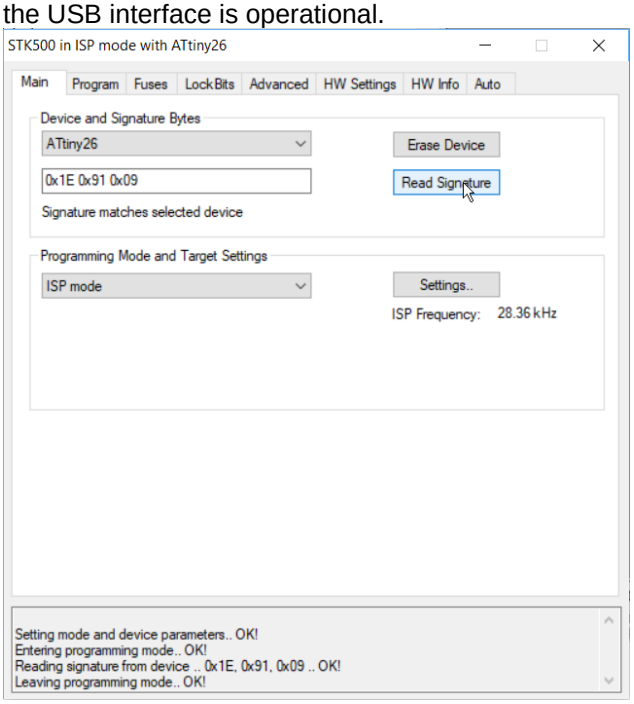


Note: On windows 10 the prolific chipset is not supported by the newest drivers. User driver version 3.3x.
File location: C:\Work\AVR\Tools\USB2Serial_correct_prolific_driver

Use the AVR Connect button (button before the button where the mouse pointer is shown).
Connect to the correct COM port as set in the Device Manager.

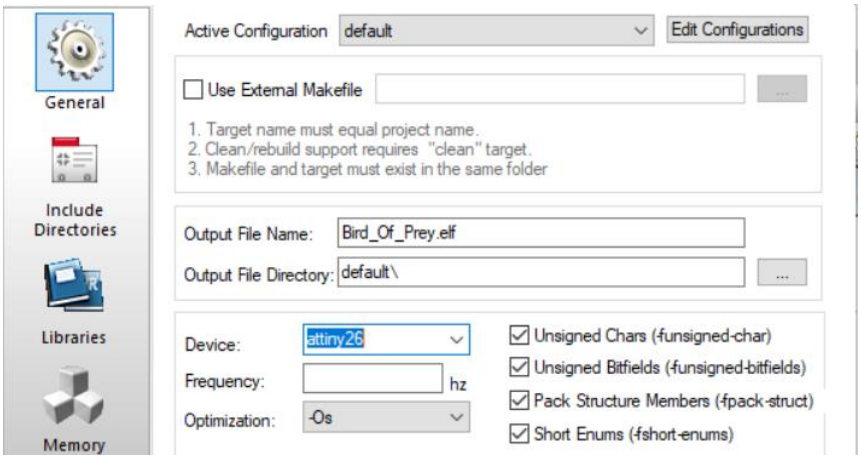


Set the device. If not clear, close AVR Studio and reopen it, select the project, a window is shown related to how to debug, choose 'simulator' and select the device. Now connect and read the signature. This should work as a test that the USB interface is operational.



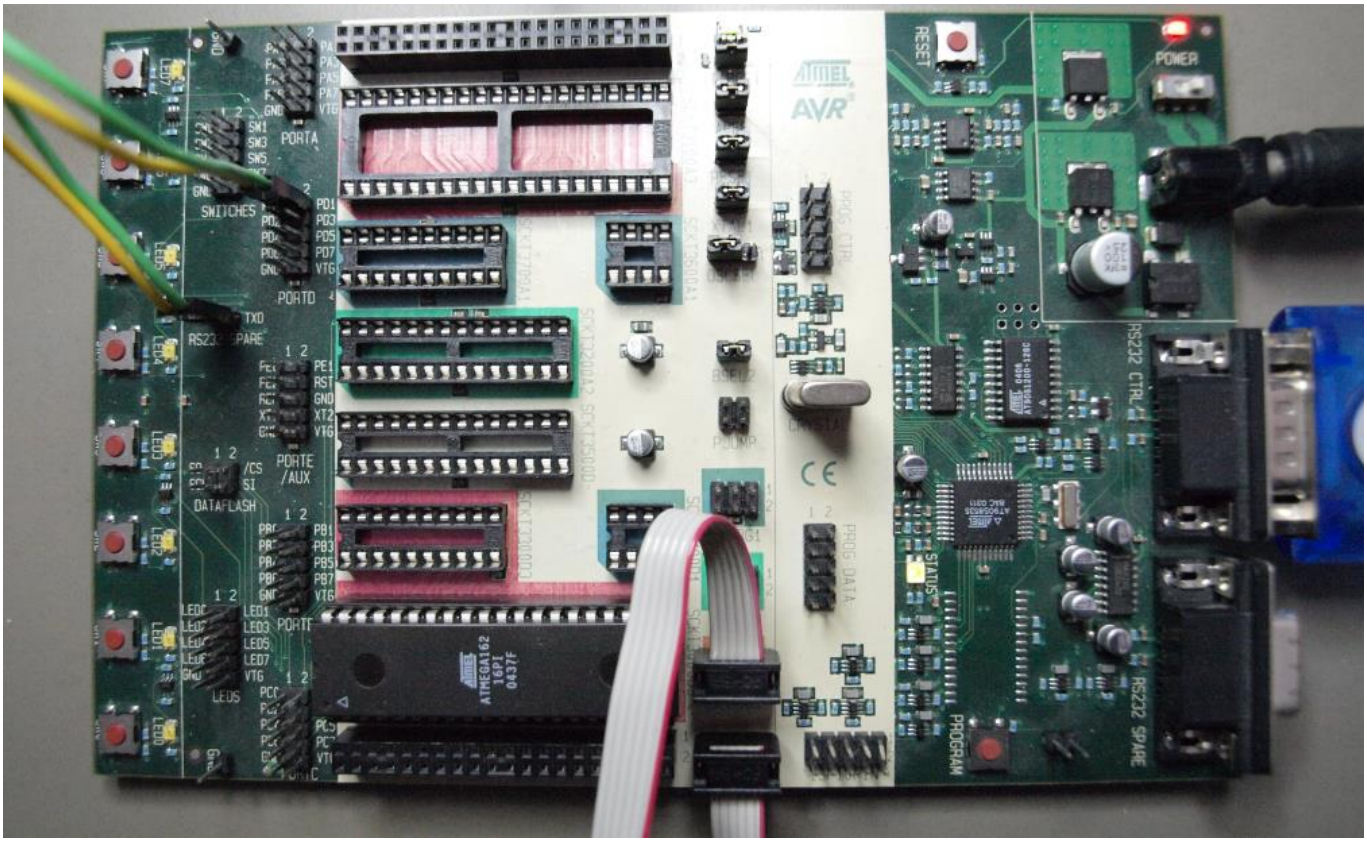
1.4 Setup Device

Menu: Project > Configuration Options > General
Do not forget to setup the correct target device.



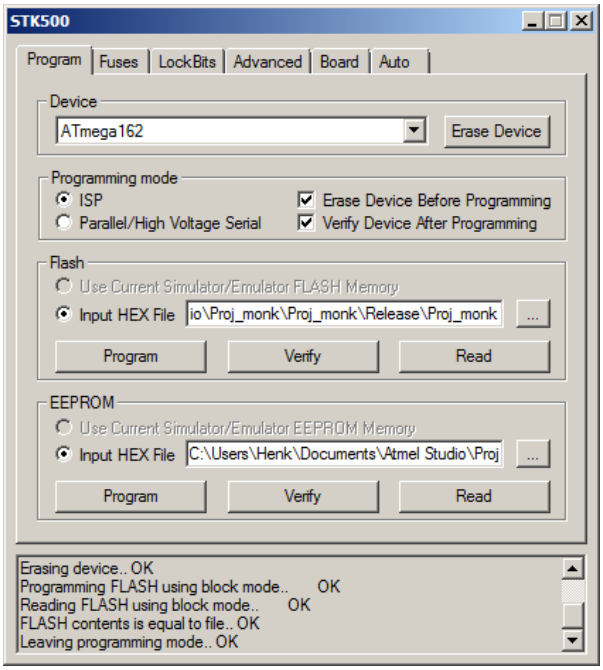
2 ATmega 162

2.1 Programming the device with STK500

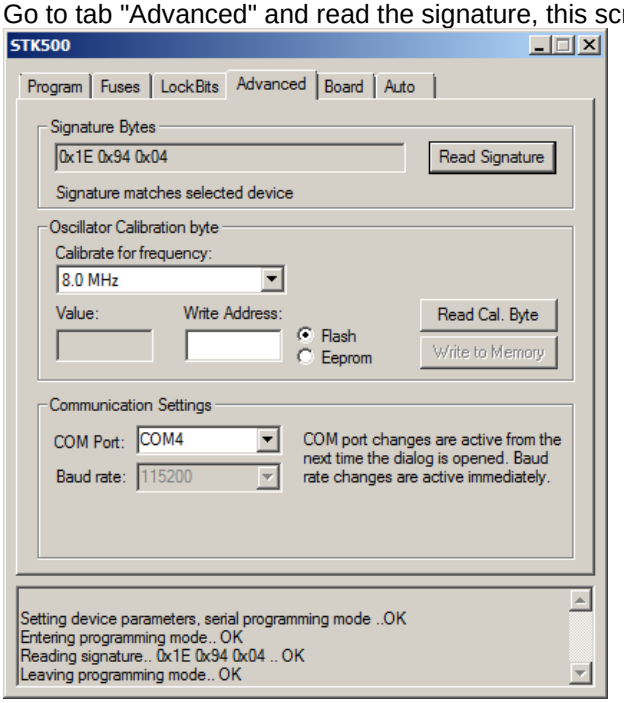


With the setup as shown above, it is possible to program the board and use the spare RS232 interface for debugging purposes without changing board settings.
- Board supply=+15VDC
- Jumpers VTARGET, AREF, RESET, XTAL1, OSCSEL[3-2], BSEL2 are mounted
- Crystal=4MHz
- Yellow/Green cable for serial communication
- Flat-cable for programming ISP6PIN <-> SPROG3
- Socket SCKT3000D3 (red)

AVR Studio:
Put programming mode to ISP.



Go to tab "Advanced" and read the signature, this screenshot should be the result:



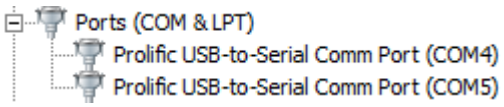
2.2 Fuses

All fuses are “Off” except for these settings:
- Brown-out detection level at VCC=4.3V; [BODLLEVEL=100]

- Boot Flash section size=128
- Clock output on PORT0 (this can be turned on/off to test if fuse programming operates correctly)
- Ext. Crystal Osc.: Frequency 3.0-8.0 MHz; Start-up time 1K CK + 0ms.

2.3 Terminal program

Tool: Putty.exe
Select: Serial
COM: Open “device manager” and open Ports to find which ports are used
In this case COM5 is the port used for serial communication, while COM4 is used to program STK500.

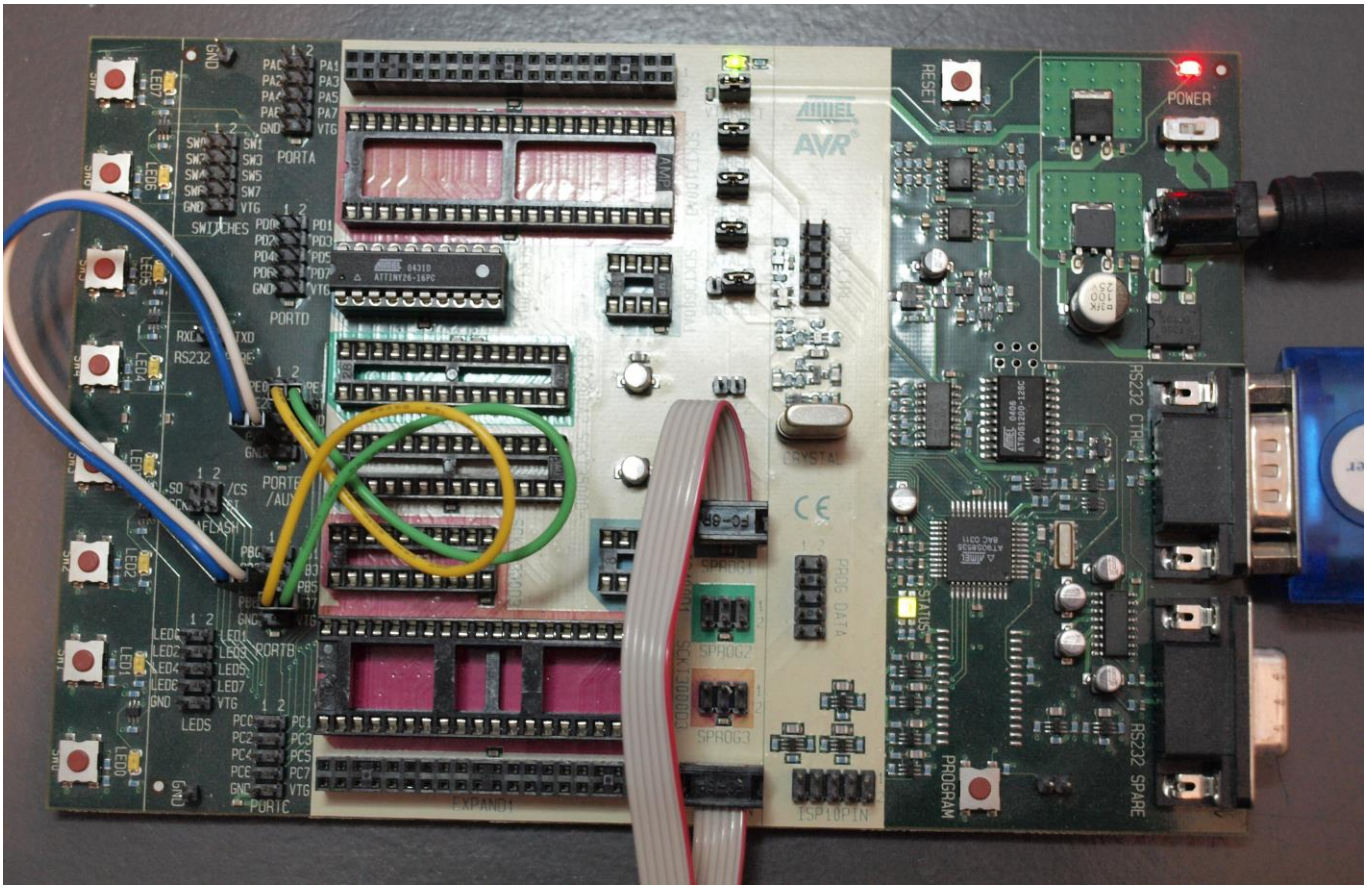


Baud: 19k2,8,N,1

STK500 setup:
Remove PD0..7 cable
Connect RS232 spare RXD with PD0, TXT with PD1

3 ATtiny 26

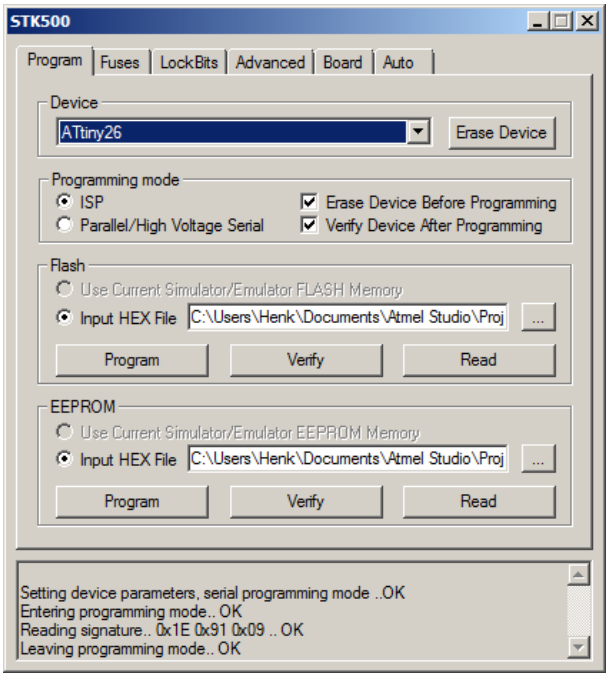
3.1 Programming the device with STK500



With the setup as shown above, it is possible to program the board and use the spare RS232 interface for debugging purposes without changing board settings.
- Board supply=+15VDC
- Jumpters VTARGET, AREF, RESET, XTAL1, OSCSEL[2-1] are mounted, BSEL2 is not mounted
- Crystal=4MHz
- Connect PORTE/AUX pin RST with PORTB pin PB7
- Connect PORTE/AUX pin XT1 with PORTB pin PB4
- Flatcable for programming ISP6PIN <-> SPROG1
- Socket SCKT3700A1 (blue)

Note: When PB7 is functionally used, a fuse has to be programmed to indicate this. If this fuse is programmed the ISP mode is no longer availale. Meaning: prevent as long as possible to use PB7 as a functional port.

AVR Studio:
Put programming mode to ISP.



Go to tab "Advanced" and read the signature, this screenshot should be the result: