Getting Started

Unpacking the System

The AVRISP Product contains the following items:

- AVRISP Programmer
- AVRISP User Guide
- 10-pin ISP cable (connected to AVRISP)
- 6-pin ISP cable (optional)
- 9-pin RS-232 Cable

System Requirements

The minimum hardware and software requirements are:

- 486 Processor (Pentium is recommended)
- 16 MB RAM
- 15 MB Free Hard Disk Space
- Windows® 95, Windows® 98, Windows NT® 4.0, or Windows® 2000
- 115200 baud RS-232 port (COM port).

Other topics of interest:

- Hardware description
- AVR Studio Software Front-end
- Command Line Software
- Special Considerations
- Troubleshooting Guide

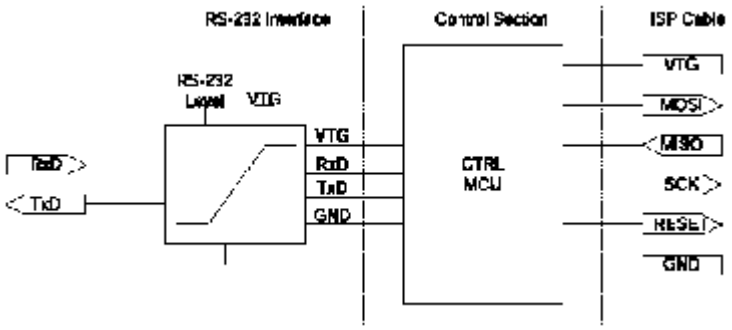
Return to introduction

Hardware Description

General Board Description

A block diagram of the AVRISP is shown in Figure 1. The AVRISP can be divided in 3 sections: The RS-232 interface, the Control section and the ISP cable. In this section a brief overview of the different blocks will be given.

Figure 1: Simplified AVRISP Block Schematics



RS-232 Serial Interface

The AVRISP uses a standard female DSUB, RS-232 port for communication with the front-end software (AVR Studio). It supports communication of 115200 baud.

Control Section

The control MCU handles all communication between the target AVR and the front-end software. The AVRISP is completely software controlled from AVR Studio. No manual configuration of the AVRISP is needed.

Status Led.

Led Color	Description
Red-Yellow-Green-Off-Green cycle	Power on Sequence
Yellow	Busy - Programming
Red	Programming failed
Green	Ready - OK

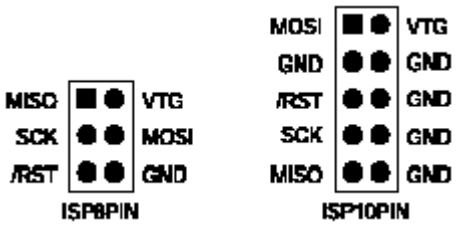
A 3-color LED indicates the status of the AVRISP. During programming the LED is yellow. When the target AVR is successfully programmed, the LED will turn green. If programming fails, the LED will turn red to indicate that programming (or verification) failed. If programming fails, check the Troubleshooting Guide. During start-up the status LED cycles through red, yellow to green to indicate that the master MCU is ready.

ISP Interface Cables

AVRISP supports both the normal 6-pin header connector pinout and the 10-pin header connector used by the older STK200 and STK300 development boards. Figure 2 shows the pinouts for the 6-pin and 10-pin ISP connectors.

AVRISP is delivered with one 6-wire and one 10-wire ISP cable. Use the one that match the pinout of the target ISP connector. Note however that only one cable should be connected, and used, at any given time. By default the 10-pin header connector is mounted.

Figure 2: AVRISP Connectors (Top View)



VRISP Connector pinout

Signal	6-Pin	10-Pin	I/O	Description
VTG	2	2	-	Power is delivered from the target board
GND	6	3,4,6, 8,10	-	Ground
MOSI	4	1	Output	Commands and data from AVRISP to target AVR
MISO	1	9	Input	Data from target AVR to AVRISP
SCK	3	7	Output	Serial Clock, Controlled by AVRISP
RESET	5	5	Output	Reset. Controlled by AVRISP

AVRISP Power Requirements and Considerations

Since the AVRISP draws power from the target, it is important that the target board is able to provide enough power to ensure correct operation. The AVRISP will draw maximum 50mA @ 5.5V. The current is drawn through the VTG line. The target should thus be able to supply at least this amount of power in addition to the requirements of the target board itself.

The AVRISP is not equipped with a power switch. Power is turned on when AVRISP is connected to the target application and turned off when disconnected.

Connecting AVRISP to Target Board

The AVRISP connects to the target board through a 3x2 or 5x2 male header connector with 2,54mm (0.1") spacing. Depending on if there is a 6-pin or 10-pin ISP connector on the target board, the cable on the AVRISP has to be changed accordingly.

The power of the target board should be turned off when connecting or disconnecting the header connector. Hot-swapping is not supported, and might damage the programmer.

To change the cable the AVRISP box must be opened and the correct cable must be connected. See Figure 3 and 4. Note that only one cable should be connected at any given time. Make sure that the cable is mounted in the correct orientation.

Figure 3: AVRISP with 10-pin ISP Connector

Figure 4: AVRISP with 6-pin ISP Connector

Handling the ISP Lines

When connecting the AVRISP to an external target some precautions should be taken. In this section a few hints and tips will be given that should assure problem free communication between the AVRISP and target device.

The part can be programmed in system from AVR Studio with In-System Programming (ISP) programming mode, running at the parts normal supply voltage.

For instruction on using the AVR studio programming software, please refer to the AVR Studio Section.

VCC and GND

Connect the AVRISP ISP power lines to the appropriate pins on the AVR device (preferably through a 6 or 10-pin connector on the target board). Make sure the target Voltage is within specified range of the programmer (2.7V - 5.5V). Make sure that the target power supply can deliver the additional power required to power the AVRISP at the given voltage.

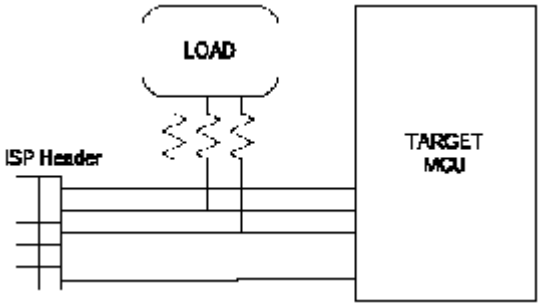
SCK

The target AVR samples the clock signal generated from the AVRISP. To make the sampling robust, a target XTAL1-frequency four times higher than the programming clock (SCK) is required. By selecting the correct target speed in AVR Studio, the correct SCK is automatically generated. The lowest supported target frequency is 8kHz.

MOSI/MISO/SCK

If the MOSI, MISO or SCK line are used as general I/Os in the application, it is recommended to use series resistors between the load and the AVR as shown in Figure 5. The AVRISP lines should be connected directly to the AVR pins, without any series resistors.

Figure 5: Loading on the MISO/MOSI/SCK lines



RESET

To enter programming mode AVRISP needs to pull RESET low. It is important that the external pull-up resistor on RESET pin is not so strong that it forces (holds) the pin high. To avoid this problem it is recommended that the RESET pull-up resistor should be no less than 4.7 kOhm.

Other topics of interest:

Return to introduction

AVR Studio Software Front-end

Introduction

As for all Atmel AVR Tools, AVR Studio with its Integrated Development Environment (IDE) is the ideal software for all AVR development. It has an editor, an assembler, a debugger and is front-end for all AVR emulators, STK500 and the AVRISP In-System Programmer. AVRISP uses the same programming interface as the STK500. Please check regularly for updates. AVR Studio is maintained and new releases are made available quite frequently. The AVR Studio releases also carry firmware upgrades for the tool itself, if available. The latest version can always be downloaded from www.atmel.com.

AVR Studio is continuously updated to support new devices and to add functionality. The latest version of AVR Studio can be downloaded from www.atmel.com

Using AVR Studio

It is assumed that the reader has general knowledge of how to use AVR Studio. AVR Studio use is covered by a separate book in the on-line help system. In this section the supporting software for AVRISP will be presented, and an in-depth description of the available programming options given.

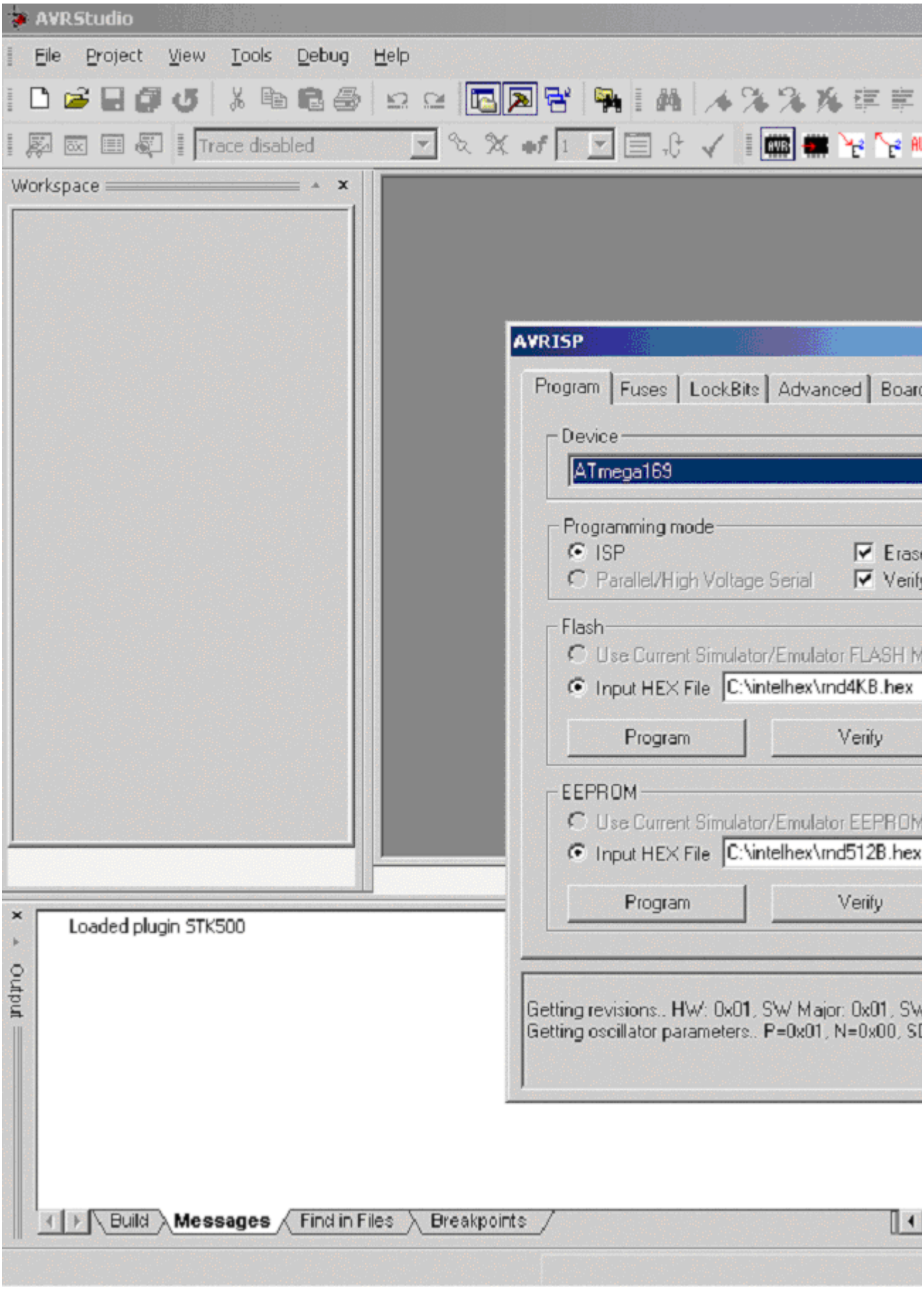
Starting the Windows Software

The software used for communicating with the AVRISP programmer is included in AVR Studio. Once installed, double clicking on the  icon starts AVR Studio.

Starting AVRISP Interface

Pressing the  button on the AVR Studio toolbar will start the AVRISP user interface as shown in .

Figure 1: AVR Studio with AVRISP User Interface



Note that the same interface is used for both STK500 and AVRISP. Since STK500 includes features that are not supported in the AVRISP, some features are not selectable when using the AVRISP

interface. Only supported features are selectable.

Using AVRISP and STK500 Simultaneously

When AVR Studio is scanning for connected devices it searches through the COM ports in a sequential manner. The first device encountered, will gain control over the COM port. It is not possible to control both a STK500 and an AVRISP from AVR Studio simultaneously. To do this two instances of AVR Studio must be executed simultaneously. The title bar on the Programming interface will indicate whether it controls the AVRISP or the STK500.

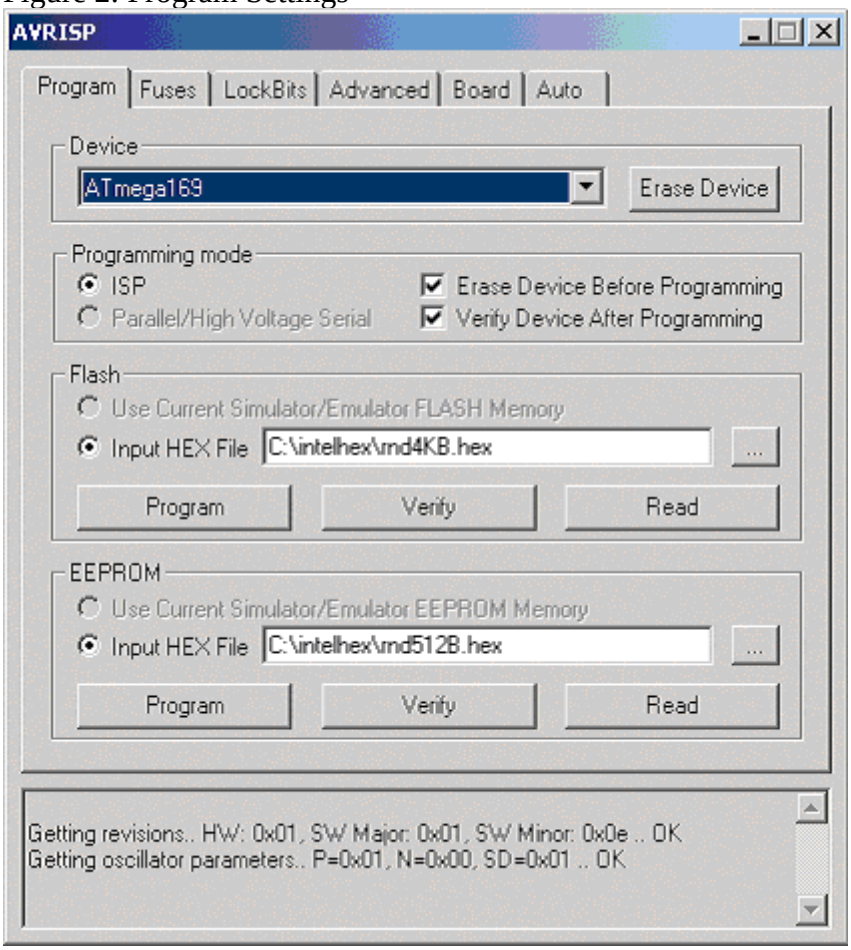
AVRISP User Interface

The AVRISP User Interface includes a lot of powerful features for the AVRISP In-System Programmer. The available settings are divided into six groups, each selectable by pressing on the appropriate tab. Since different devices have different features, the available options and selections will depend on which device is selected. Unavailable features are grayed out.

"Program" Settings

The program settings are divided into 4 different sub groups.

Figure 2: Program Settings



Device

Device is chosen by selecting the correct device from the pull-down menu. This group also includes a button that performs a chip-erase on the selected device, erasing both the FLASH and EEPROM memories.

Programming Mode

This group selects programming mode. AVRISP only supports the ISP low voltage mode. Checking the "Erase Device Before Programming" will force AVRISP to perform a chip-erase before programming the device. Checking the "Verify Device After Programming" will force AVRISP to perform a verification of the memories after programming.

Flash

If the AVRISP User Interface is opened without a project loaded in AVR Studio, the "Use Current Simulator/Emulator FLASH Memory" option will be grayed out. When a project is open this option allows programming of the Flash memory content currently present in the Flash Memory view of AVR Studio (For more information about AVR Studio memory views, please take a look in the AVR Studio help system.).

If no project is running, or the source code is stored in a separate HEX file, select the "Input HEX File" option. Browse to the correct file by pressing the  button, or write the complete path and filename in the text field. The selected file must be in "Intel-hex" format or "extended Intel-hex" format.

EEPROM

If the AVRISP User Interface is opened without a project loaded in AVR Studio, the "Use Current Simulator/Emulator EEPROM Memory" option will be grayed out. When a project is open this option allows programming of the EEPROM memory content currently present in the EEPROM Memory view (For more information about AVR Studio memory views, please take a look in the AVR Studio help system.).

If no project is running, or the source code is stored in a separate HEX file, select the "Input HEX File" option. Browse to the correct file by pressing the  button, or write the complete path and filename in the text field. The selected file must be in "Intel-hex" format or "extended Intel-hex" format.

"Fuses" Settings

On the "Fuses Settings" tab an overview of accessible fuses are presented. Some fuses are only available during Parallel /High-Voltage programming. These will be displayed, but are not accessible when operating in ISP programming mode. Press the "Read" button to read the current value of the fuses, and the "Write" button to write the current fuse setting to the device. Checking one of these check-boxes indicates that this fuse should be enabled/programmed, which means writing a "zero" to the fuse location in the actual device. Note that the selected fuse setting is not affected by erasing the device with a chip-erase cycle (i.e. pressing "Chip Erase" button in the "Program" settings).

- Check Box Description

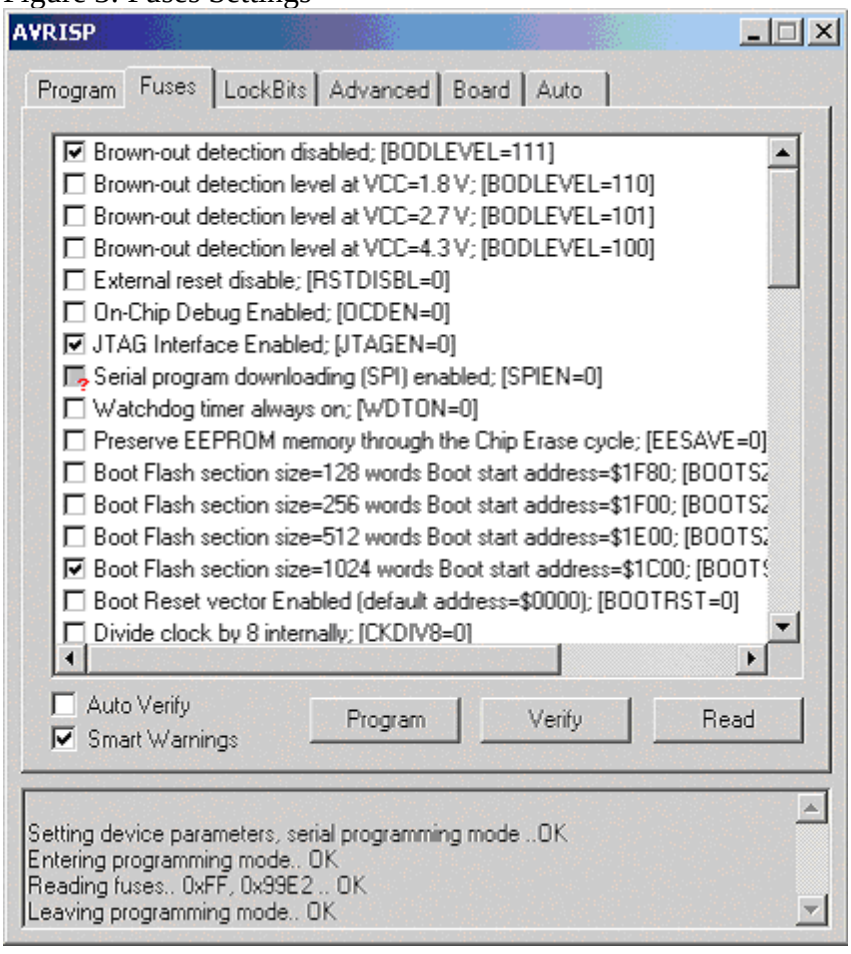
Icon	Description
	Unprogrammed fuse or lockbit.
	Programmed fuse or lockbit

☑?	Readback of current state is impossible, but fuse or lock can be programmed. (Set to be programmed.)
☐?	Readback of current state is impossible, but fuse or lock can be programmed. (Not set to be programmed.)
☐	Readback of current value indicated unprogrammed lock or fuse bit, but no access is possible. i.e. can not be changed in serial mode
☑	Readback of current value indicated programmed lock or fuse bit, but no access is possible. i.e. cannot be changed in serial mode
☐?	Fuse or lock bit is not accessible, and read back is impossible.

Detailed information on which fuses are available in the different programming modes and their functions can be found in the appropriate device datasheet. By checking the "Auto Verify" check box a verification will be automatically preformed after each programming.

Please if you plan to change the RSTDISBL or SPIEN fuse be aware of the fact that further ISP-programming of the device will not be possible.

Figure 3: Fuses Settings

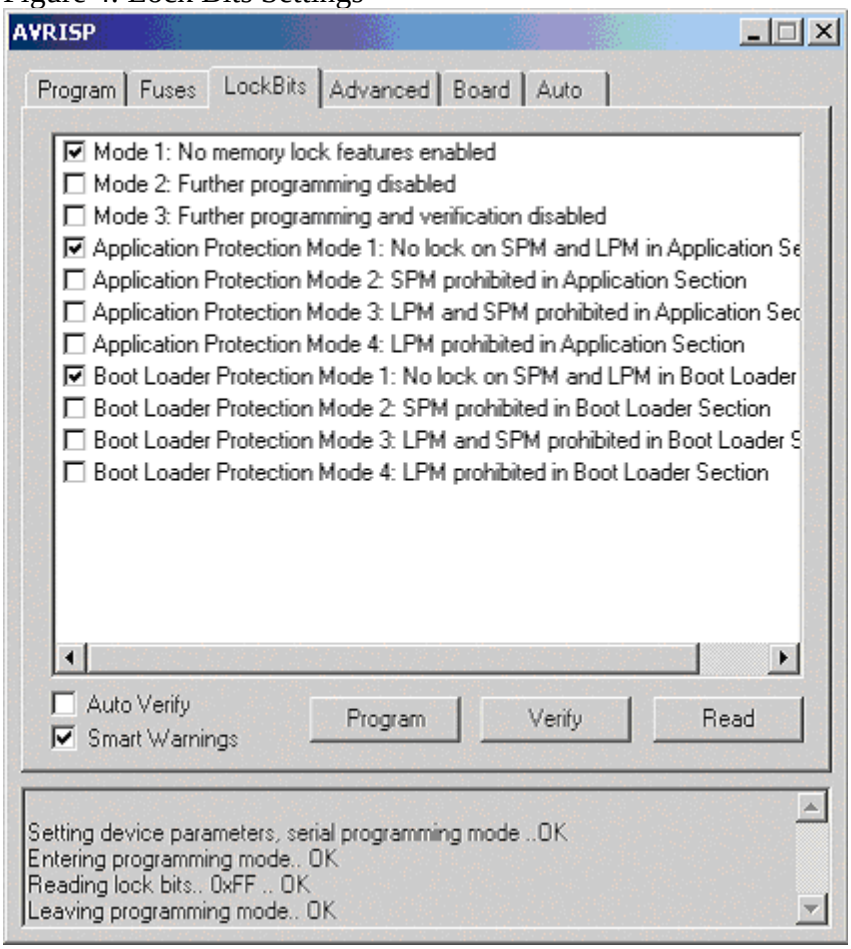


"Lock Bits" Settings

Similar to the "Fuses" settings, the "Lock Bits" tab shows which lock modes are applicable to the selected device. All lock bits are accessible in ISP programming mode. A lock mode may consist of a combination of multiple lock bits. The AVRISP User Interface handles this, and the correct lock bits are programmed automatically for the selected Lock mode. Once a Lock mode protection level

is enabled it is not possible to lower the protection level by selecting a lower degree of protection by setting a different Lock mode. The only way of removing a programmed lock bit is to do a complete chip-erase, erasing both Program and EEPROM memories. One exception exists: If the target device has a programmed "EESAVE" fuse, the contents of the EEPROM will be kept even when a complete chip erase on the device is performed. By checking the "Auto Verify" check box a verification will be automatically preformed after each programming.

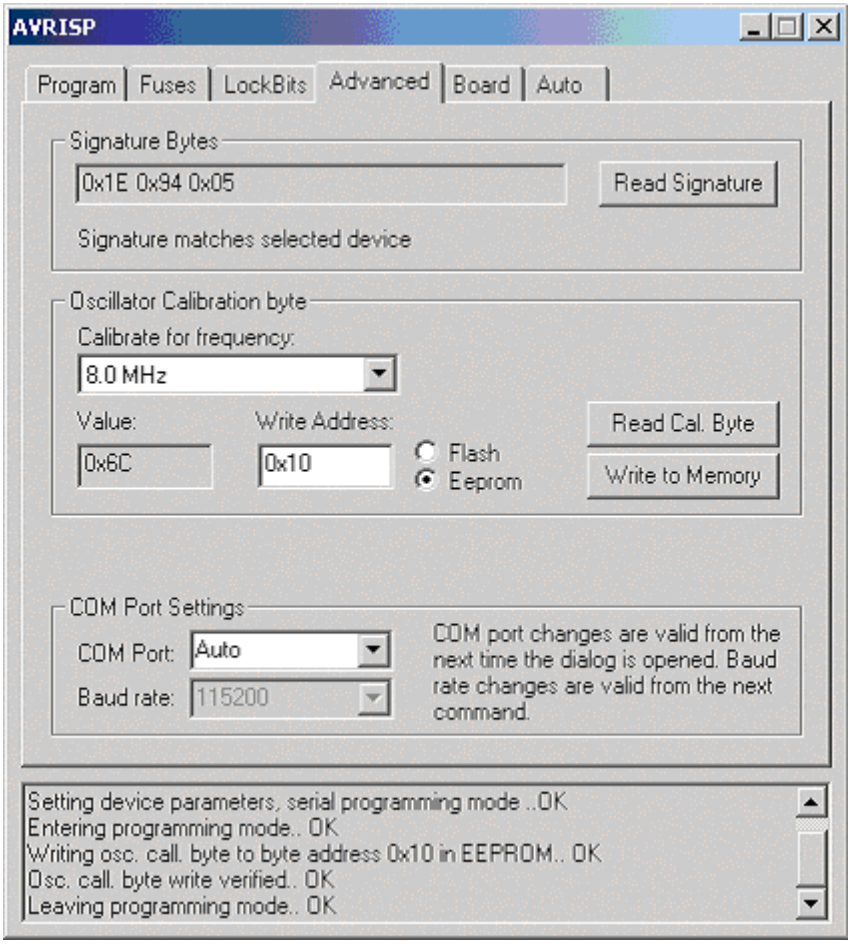
Figure 4: Lock Bits Settings



"Advanced" Settings

The Advanced tab is currently divided into two sub groups.

Figure 5: Advanced Settings



Signature Bytes

By pressing the "Read Signature" button, the signature bytes are read from the target device. The signature bytes act like an identifier for the part. Please refer to the AVR datasheets to read more about signature bytes.

Oscillator Calibration Byte

For devices with calibratable Internal RC Oscillator, the oscillator calibration byte is written to the device during manufacturing, and can not be erased or altered by the user. The calibration byte is a tuning value that should be written to the OSCCAL register in order to tune the internal RC to specified frequency.

Reading Oscillator Calibration Byte(s)

By pressing the "Read Cal. Byte" button, the calibration value is read from the device and shown in the "Value" text box. Note that on some devices the calibration byte is not directly accessible during program execution and must be written to a memory location during programming if the program should use it. If this option is grayed out, the selected device does not have a tunable internal RC Oscillator. On selected devices, the RC oscillator is self-calibrating for the default RC oscillator clock source. On these devices there is no need to handle the Calibration byte manually (for more information see appropriate device datasheet).

Writing Oscillator Calibration Byte

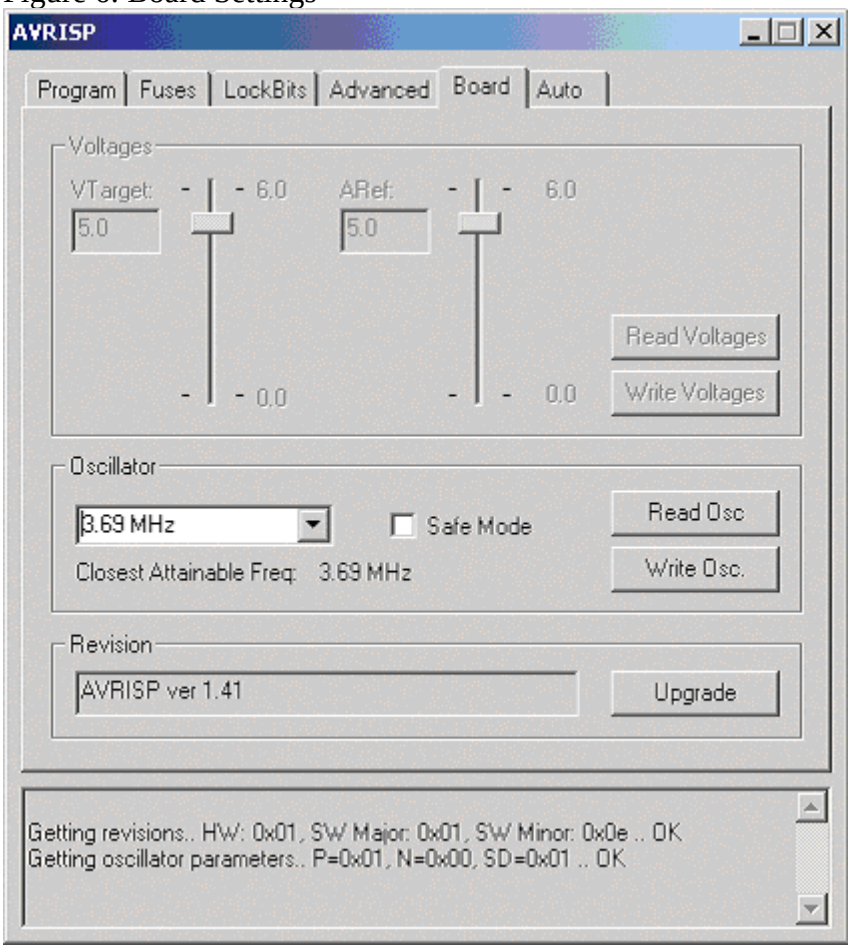
Since the calibration byte is not directly accessible during program execution, on devices without automatic RC calibration the user should write the calibration byte into a known location in Flash or

EEPROM memory. Do this by writing the desired memory address in the "Write Address" text box and then press the "Write to Memory" button. The calibration byte is then written to the memory indicated by the "Flash" and "EEprom" radio buttons.

"Board" Settings

The Board tab allows changing the operating conditions for the AVRISP programmer. The AVRISP allows modification of the Oscillator frequency.

Figure 6: Board Settings



Oscillator

The frequency given here should be lower or equal to the frequency of the target AVR to be programmed. Based on this number, the AVRISP User Interface calculates the communication speed between the AVRISP and the target AVR.

Revision

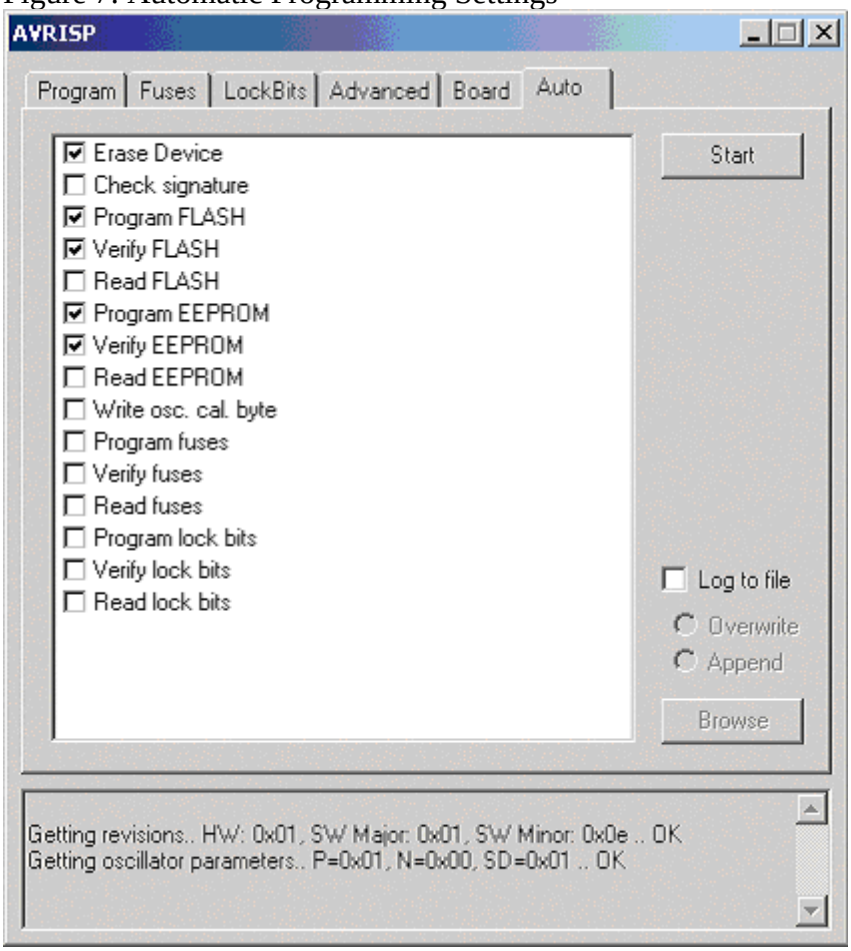
In the revision box the current revision AVRISP revision number is shown. If AVR Studio discovers that the AVRISP contains an older version than the one distributed with AVR Studio, it will automatically ask for permission to upgrade the programmer.

"Auto" Settings

When programming multiple devices with the same code, the Auto tab offers a powerful method of automatically going through a user-defined sequence of commands. The commands are listed in the order they are executed (if selected). To enable a command, the appropriate check box should be checked. E.g. If only "Program FLASH" is checked, by pressing the "Start" button the FLASH

memory will be programmed with the HEX file specified in the "Program" settings. All commands depend on, and use, the settings given in the AVRISP User Interface. It is possible to log the command execution to a text file by checking the "Log to file" check box.

Figure 7: Automatic Programming Settings



Setting up the System for Auto Programming

Click on the check boxes for the commands the AVRISP User Interface should perform. A typical sequence where the device is erased and then programmed is shown in Figure 7. The chip is erased, both memories programmed and verified.

Once configured, the same programming sequence is executed every time the "Start" button is pressed. This reduces both work and possibilities for errors due to operational errors.

Logging the Auto Programming to a File

By clicking on the "Log to file" check box all output from the commands are written to a text file. The file is selected/created by pressing the "Browse" button. Navigate to the location where the file is placed, or should be created. The output is directed to this file, and can be viewed and edited using a standard text editor. An existing file will be overwritten.

History Window

The History window is located at the bottom of the AVRISP view. In this window the dialog between AVR Studio and AVRISP is shown. For every new command performed, the old dialog is replaced with the new one.

Other topics of interest:

Command Line Software

The DOS command line of the programming interface is the same as for the STK500 starter kit. It is named "stk500.exe" and allows simple batch files for automatic programming.

In the following text it will be shown how to make simple batch files for automating the programming steps for a device.

Synopsys: STK500

[-d device name] [-m s|p] [-if infile] [-ie infile] [-of outfile] [-oe outfile] [-s] [-O index] [-Sf addr] [-Seaddr] [-e] [-p f|e|b] [-r f|e|b] [-v f|e|b] [-l value] [-L value] [-y] [-f value] [-F value] [-q] [-x value] [-af start,stop] [-ae start,stop] [-c port] [-ut value] [-ua value] [-wt] [-wa] [-j value] [-b h|s] [-! freq] [-&] [-t p|t] [-t] [-n] [-g] [-z] [-U] [-h|?]

Note: There should be no space between the parameter literal and the argument for any of the command line parameters that takes sub arguments. I.E. it should be -dAT90S8518 and not -d AT90S8515.

Parameters:

d - Device name. Must be applied when programming the device.

m - Select programming mode; serial (s) or parallel (p). Serial progr. mode is the default, and is used if this parameter is not applied.

if - Name of FLASH input file. Required for programming or verification of the FLASH memory. The file format is Intel Extended HEX.

ie - Name of EEPROM input file. Required for programming or verification of the EEPROM memory. The file format is Intel Extended HEX.

of - Name of flash output file. Required for readout of the FLASH memory. The file format is Intel Extended HEX.

oe - Name of EEPROM output file. Required for readout of the EEPROM memory. The file format is Intel Extended HEX.

s- Read signature bytes.

O - Read oscillator calibration byte(s). "Index" is otional.

Sf - Write oscillator cal. byte to FLASH memory. 'addr' is byte address

Se - Write oscillator cal. byte to EEPROM memory. 'addr' is byte address

e - Erase device. If applied with another programming parameter, the device will be erased before any other programming takes place.

p - Program device; FLASH (f), EEPROM (e) or both (b). Corresponding input files are required.

r - Read out device; FLASH (f), EEPROM (e) or both (b). Corresponding output files are required

v - Verify device; FLASH (f), EEPROM (e) or both (b). Can be used with -p or stand alone. Corresponding input files are required.

l - Set lock byte. 'value' is an 8-bit hex. value.

L - Verify lock byte. 'value' is an 8-bit hex. value to verify against.

y - Read back lock byte.

f - Set fuse bytes. 'value' is a 16-bit hex. value describing the settings for the upper and lower fuse.

F - Verify fuse bytes. 'value' is a 16-bit hex. value to verify against.

q - Read back fuse bytes.

x - Fill unspecified locations with a value (0x00-0xff). The default is to not program locations not specified in the input files.

af - FLASH address range. Specifies the address range of operations. The default is the entire FLASH. Byte addresses.

ae - EEPROM address range. Specifies the address range of operations.The default is the entire EEPROM. Byte addresses.

c - Select communication port; 'com1' to 'com8'. If this parameter is omitted the program will scan the comm. ports for the AVRISP.

ut - Set target voltage VTARGET in Volts. 'value' is a floating point value between 0.0 and 6.0, describing the new voltage.

ua - Set adjustable voltage AREF in Volts. 'value' is a floating point value between 0.0 and 6.0, describing the new voltage.

wt - Get current target voltage VTARGET.

wa - Get current adjustable voltage AREF.

b - Get revisions; hardware revision (h) and software revision (s).

! - Set oscillator frequency; 'freq' is the frequency in Hz.

& - Get oscillator frequency.

t - Get currently selected device parameters.

n - Get current programming mode.

g - Silent operation.

z - No progress indicator. Eg. if piping to a file for log purposes, use this option to avoid the non-ascii characters used for the indicator.

h|? - Help information (overrides all other settings).

Since the interface is also used for the STK500 Starter Kit, not all listed switches are applicable to the AVRISP. STK500 software will give an error-message if an unsupported switch is used.

Sample Usage

Erase, program and verify the flash of an AT90S8515

STK500 -dAT90S8515 -ms -e -pf -vf -ifTest.hex

Erase, program and verify the EEPROM of an AT90S/LS4433

STK500 -dAT90S4433 -ms -e -pe -ve -ieTest.hex

Other topics of interest:

- Getting Started
- Hardware description
- AVR Studio Software Front-end
- Special Considerations
- Troubleshooting Guide

Return to introduction

Special Considerations

There are a few special considerations that should be noted when using this AVRISP programmer to In-System program AVR Devices.

Fuse Programming

Some devices have fuses not accessible in ISP mode. To program these fuses a parallel programmer is needed. Some of the AVR devices allow access to the SPIEN and RSTDISBL fuses. Un-programming / programming these fuses will disable further ISP programming.

RESET Used as General IO port.

If the RESET pin is used as a general purpose IO, In-System Programming is not possible. The reason is that the RESET pin must pulled to 12V to enter High Voltage Serial or Parallel Programming mode (HVSP or HVP). HVSP or HVP must be used to change the RSTDSBL fuse.

AVR Devices with no ISP Option

Some devices do not have an ISP programming option. (e.g. ATtiny28) To program these devices, a parallel programmer is required. (e.g STK500 Starter Kit). Only devices with low voltage ISP programming mode are supported by the AVRISP.

Devices without RC Oscillator Calibration

Not all devices with internal RC clock option feature Oscillator calibration. For these devices the internal RC will run at the default speed as indicated by the appropriate datasheet.

Other topics of interest:

- Getting Started
- Hardware description
- AVR Studio Software Front-end
- Command Line Software
- Troubleshooting Guide

Return to introduction

Troubleshooting Guide

Problem	Reason	Solution
	AVRISP is not connected to target	Connect ISP cable to target board
The LED is not lit	ISP pinout is not correct	Verify pinout on target ISP header connector
	Target does not provide enough power	Verify that the target power supply can deliver enough power
	ISP pinout is not correct	Verify pinout on target ISP header connector
	Device does not support ISP programming mode	Verify that device supports ISP mode, and that correct IO pins are connected.
	Heavy loading on ISP pins	Connect series resistors between load and IO pins as shown here.
	Too strong pullup on RESET pin	Reset pullup resistor should be more than 4.7kOhm.
Can't get any communication with target device	Target frequency set wrong in AVR Studio	Reduce the frequency in AVR Studio to match the target board frequency
	Target does not provide enough power	Verify that the target power supply can deliver enough power to source both application and AVRISP
	SPIEN fuse disabled	Enable SPIEN fuse with a Parallel /High Voltage Serial programmer
	Reset used as general IO	Use a High Voltage Serial Programmer /Parallel programmer to Change the RSTDISBL fuse.
AVR Studio does not find AVRISP	Old version of AVR Studio	Download AVR Studio 4 from www.atmel.com
	Other Device or service controls the COM port	Disable Mouse Drivers IRDA drivers or other devices that takes control of the COM port.

Other topics of interest:

- Getting Started
- Hardware description
- AVR Studio Software Front-end
- Command Line Software
- Special Considerations

Return to introduction

AVRISP Manual Firmware Upgrade

Firmware Upgrading is usually done either automatically by AVR Studio, if AVR Studio detects that the Firmware distributed with AVR Studio is newer that the Firmware present in the AVRISP, or by selecting "upgrade" in the STK500/AVRISP dialog window opened from the Tools Menu in AVR Studio. However, if the communication between the AVRISP and the PC is broken during Firmware Upgrading the AVRISP stops responding, both Firmware Upgrading and Programming will not work. The AVR ISP is NOT broken, but it is no longer able to automatically enter Programming mode – Programming mode have to be forced to do a firmware upgrade. Once this is done the AVRISP is up and running again.

The problem can occur if removing the power from the target application (which provide the target voltage for the AVR ISP) or if the serial cable is removed during the Programming.

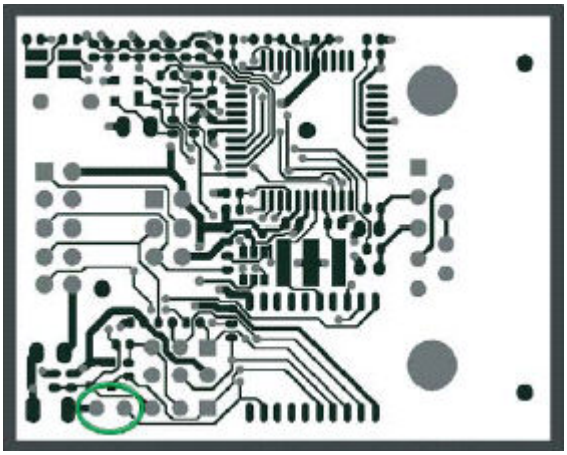
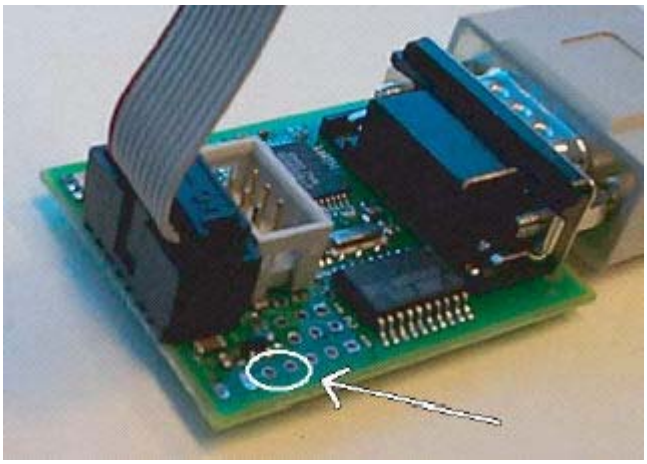


Figure 1: AVR ISP PCB Layout – Pin Holes Marked with Green Circle



Fugure 2: AVR ISP without Housing – Pin Holes Marked with White Circle

Procedure for manually upgrading the AVR ISP Firmware:

- Power off the AVRISP by powering off the target board that supplies the power to the AVRISP.
- Short the pin holes marked on Figure 1 (PCB layout) and Figure 2 (photo).
- Power on the AVRISP by powering the target board that supplies the power to the AVRISP (1).
- Wait for 5 seconds.
- Remove the short.
- Start AVR Studio.
- Start the application “AVR Prog” located in the Tools menu.
- Click “Browse” button in Hex file window. Locate and select the “STK500.hex” (STK500.ebn in AVR Studio version 3.54 and later) in the AVR Studio subfolder named STK500. typical path is “C:\Program Files\Atmel\AVR Studio\Stk500” in AVR Studio 3.x and “C:\Program Files\Atmel\AVR Tools\STK500” in AVR Studio 4.x.
- Click “Program” button in Flash window.
- Close “AVR Prog” after Programming and verification is completed.
- Power off the AVRISP.

The Firmware of the AVRISP is now upgraded and the programmer is ready for usage.

Note: 1. If the STK500 is used to provide power to the AVRISP, make sure to use the headers SPROG1/2/3 and remove any devices present in the STK500. If a custom target board is used to supply power to the AVRISP, make sure to erase the device prior to the AVRISP upgrade to ensure that the device is not driving the ISP Lines.