# **AVR FAQ**

This document’s purpose is to address some of the common problems encountered when programming Atmel AVR microcontrollers. A list of symptoms are shown, and the most likely or obvious solutions are presented. In most cases, reference to this document should solve the vast majority of problems faced by beginning embedded programmers. If this document does not address your specific issue, it is advisable to still read the document and consider the techniques used and how they might apply to your special case.

Frustration is a part of a programmer’s life - patience is your best virtue. This is doubly true in the realm of embedded systems, where one must deal not only with the intricacies of software development, but also with hardware circuit designs that many computer science majors are not yet comfortable with. This document should aid the troubleshooting of your hardware circuit, and to a lesser degree, common software mistakes. Your approach to debugging must always be a methodical system consisting of three repeating steps:

**i) Develop hypothesis (e.g. No power to circuit).**

**ii) Test hypothesis (e.g. Multimeter measures 0 volts)**

**iii) Address result (e.g. Turn on power, or develop new hypothesis)**

## **COMMON SYMPTOMS**

[1. AVR Studio can’t connect to programmer.](#kix.xyzt5mggtwnf)

[2. The chip is programmed, but nothing happens.](#kix.uphnjaiyfr2c)

[3. The timer doesn’t give the right interval.](#kix.u2lcsuy93pm9)

[4. The timer doesn’t work at all.](#kix.x0i0hllworqw)

[5. LEDs not working.](#kix.s718l55ey02s)

[6. Not detecting button presses.](#kix.bi5necjhk1oe)

[7. AVR Studio Warnings.](#kix.e7s49ltz2cr4)

[8. Input pin has voltage (Pull-up resistor)](#kix.f2iri23qvyun)

[9. Interrupts aren't working!](#kix.cfd2gdcc9f4m)

[10. Microcontroller keeps resetting!](#kix.axjbm48g062e)

[11. Installing the AVRISP](#kix.gbj840aosyc9)

[12. Firmware upgrade for the AVRISP](#kix.gsialmhvucqu)

#### **1. AVRDUDE can’t connect to programmer**

This is the most common issue that arises when attempting to program the AVR, and there are quite a few reasons as to why the programmer does not respond.

Observe the LED on the programmer:

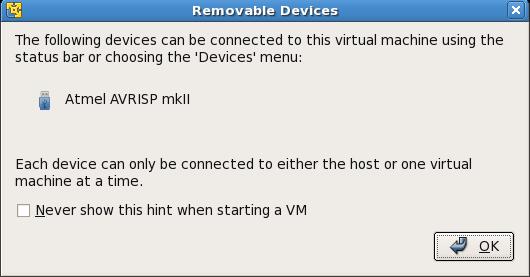
**OFF)** No power is reaching the programmer. Ensure that it is plugged into a USB port, and that the USB port is enabled and powered on. If using VMWare, ensure that VMWare has enabled (captured) the device.

**RED)** The programmer itself has power, but does not detect a connected microcontroller. This always implies that the microcontroller does not have power. Double check your circuit to ensure that VCC (pin 10 on ATMEGA32), and GND (pins 11 & 31 on ATMEGA32) are connected to power sources. Ensure that the power sources are providing enough power. It is possible that batteries have been drained and can no longer provide the required voltage, in which case you should change the batteries.

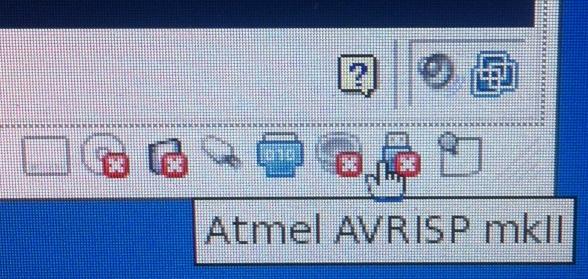
In some cases, your AVR may be dead. Replace the AVR and see if the behavior changes.

**Yellow)** The AVR has power, but connection can not be established. The most common cause of this is misconfigured SPI pins between the AVR and programmer. Double check to make sure the connections are in the right place. Also, double check there are no short circuits in your hardware circuit’s power busses. Your AVR may possibly be dead, replace the AVR and determine if new behavior occurs. In rare cases, the programmer itself may be broken. You may update the programmer firmware from within AVR Studio, or forcibly overwrite it to determine if new behavior occurs.

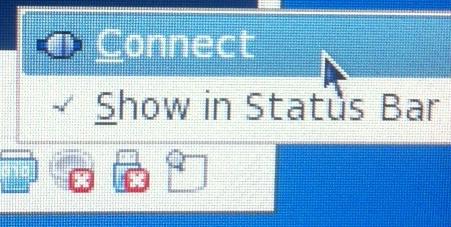
**Green)** Your AVR is usually connected successfully. If you are running from VMware, you must make sure that the virtual machine has also captured the USB device. Both LEDs can be green even if the AVRISP isn’t captured by the virutal machine. Usually this capture occurs automatically:



If this did not occur automatically you must manually capture the USB device. Non-capture is indicated by an **X** on the USB icon in the lower right hand corner of the VMWare window:



And in the devices menu, if the **X** is present connect the Atmel AVRISP mkII from *either* location:



If AVR Studio still has has problems connecting to the programmer while the light is green, try restarting AVR studio.

#### **2. The chip is programmed, but nothing happens.**

AVR Studio defaults to the use of the previous .elf file when programming. Ensure that you have selected the correct file to download when utilizing the program AVR window in AVR Studio.

If you are sure you are downloading the correct file to the AVR, then try to make a very simple program to determine whether the problem is in hardware or is in software.

i). Connect an LED between pin 40 and ground (with the long end of the LED connected to pin 40).

ii) Write software to turn the light on at the start of the program. Set DDRA = 0xff, and set PORTA = 0xff within the main software function, performing no other actions.

iii) If the light turns on, then the tools and microcontroller are fine, and something is wrong with your original software and/or hardware.

Use your original hardware to perform similar simple tests (turn a light on, beep the speaker, etc), to determine the source of the bug. If software is the problem, the best solution is usually to iteratively remove or add code components until you identify the source.

#### **3. The timer does not count correctly.**

#### The timer is based on the internal clock of the AVR and on the timer configuration registers. Ensure that the prescalar value is correct in the TCCR register. If your code for configuring the timers is correct but the timer still counts at the wrong interval, you likely need to set the fuses of the AVR. The AVR Studio programming dialog box gives access to the fuses which control the microcontroller behavior. Ensure that you have selected the appropriate clock configuration fuse setting (Internal 8MHz oscillator is the most common used).

If the fuses and timer are set correctly, then the issue is with the software used to keep track of time. Consult [4. The timer does not work.](#kix.x0i0hllworqw)

#### **4. The timer does not work.**

#### Before using the timer, you must correctly configure the timer control registers. Consult the datasheet of the ATMega32 for information (h[ttp://www.atmel.com/dyn/resources/prod\_documents/doc2503.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc2503.pdf)). Timers use interrupts to signal that it’s interval has expired. Most commonly, the timer is configured to count up continuously until it overflows (8 bit timers count from 0 to 255, then overflowing back to 0). It is up to you to write code within the ISR (interrupt service routine) that keeps track of time such that your application can utilize it.

i) Ensure that the timer overflow interrupt bit has been enabled in register TIMSK.

ii) Ensure that global interrupts have been enabled in the SREG register.

iii) Declare the function *ISR(TIMER0\_OVF\_vect)* as your ISR. This function will be called whenever timer 0 overflows. You should utilize a counter and flag to track the number of times this function is called, setting the flag when a desired interval is reached.

#### **5. The LED is not turning on.**

The polarity of the LED is important. The LED wire leads have different lengths. You should connect the longer lead to the power source of the LED, and the shorter lead to GND. You should also connect a resistor between the short end of the LED and GND.

Ensure that the appropriate bit in the Data Direction Register for the port connected to the LED is set to a ‘1’ to enable the pin as an output. Also ensure that that the bit in the PORT register has been set to ‘1’ to turn the LED on.

**6. Not detecting button presses.**

To retrieve input from the circuit, you must sample the data found within the corresponding PIN register of the port (e.g. PINA,PINB,PINC, or PIND). The PIN register latches the inputs every clock cycle for pins whose DDR is set to ‘0’(input), and sets those pins whose DDR value is ‘1’(output) to the value of the corresponding bit in the PORT register.

**7. AVR Studio Warnings.**

Take warnings of AVR Studio seriously. For example, the warning: “'TIMER0\_ovf\_vect' appears to be a misspelled signal handler” will cause the microcontroller to function incorrectly. Specifically, the timer overflow interrupt vector will never be called, because it’s correct name is “TIMER0\_OVF\_vect”, where capitalization matters. Important: To see the list of all available I/O Registers, Interrupt Vectors, and other useful information for an ATmega32 microcontroller, see the header file “iom32.h” under “External Dependencies” on the left side of AVR Studio. The names of the registers and interrupt vectors should be the same as in the datasheet, but in case they are not, the names in header file are exactly what the microcontroller uses.

**8. Input pin has voltage/always reads “high”.**

You have enabled pull-up resistors. A pull-up resistor “pulls-up” the value of the pin when it is disconnected to act as a logic “yes”. Then to switch to a logic “no”, we apply 0V, and since current takes the path of least resistance, the microcontroller now reads a logic “no”. For example, say you want to “read” a button press. When DDRA = 0x00, we have configured port A (not to be confused with PORTA) to be an input. To enable pull-up resistance, simply set PORTA = 0xff (and conversely PORTA = 0x00 to disable pull-up resistance). With pull-up enabled and port A set to input, PINA will always read a logic 1 until it is driven low-- this is known as active-low, because the act of pressing a button drives the pin low.

**9. Interrupts aren’t working!**

Make sure you have #include <avr/interrupt.h>. It’s happened to the best of us.

**10. Microcontroller keeps resetting!**

If you have enabled any interrupts, make sure to declare them all within your code, otherwise an interrupt without an interrupt handler will by default reset the microcontroller. From AVR-LIBC: “In rare circumstances, [a]n interrupt vector does not need any code to be implemented at all. The vector must be declared anyway, so when the interrupt triggers it won't execute the BADISR\_vect code (which by default restarts the application).”

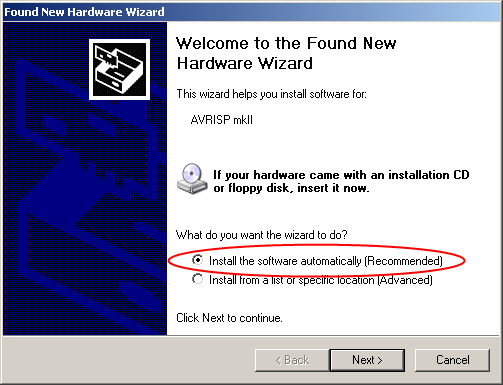
**11. Installing the AVRISP**

**AVR Studio must be installed prior to installing the AVRISP.**

The first time you plug in the USB cable from the blue AVRISP device to the PC you will see a “Found New Hardware” bubble:



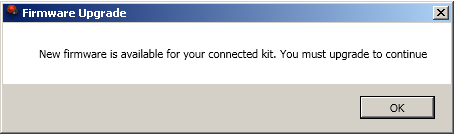
Windows will then launch the “Found New Hardware Wizard” to complete the installation of the drivers for the AVRISP. Ensure you have installed AVR Studio first, and then select the “Install the software automatically” option. You will then proceed to next and Windows will set up and install the drivers for the AVRISP.



Assuming all went well you can finish and use your AVRISP with AVR Studio. If there was an issue finding the drivers you may need to update or re-install AVR Studio.

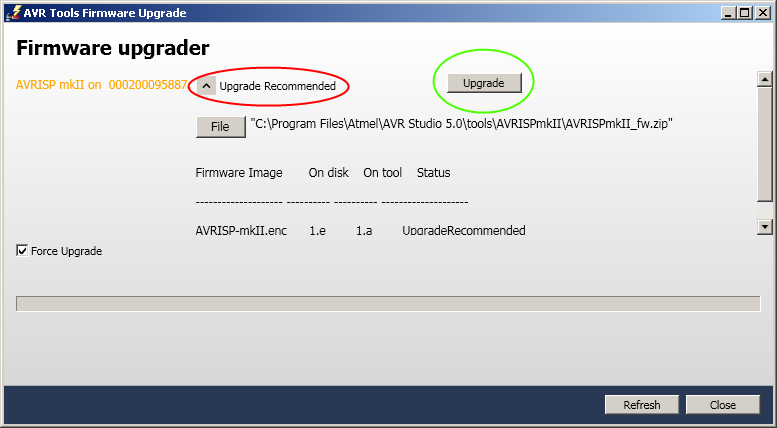
**12. Firmware upgrade for the AVRISP**

In order to program your microcontroller, AVR Studio requires that you use the most current version of the firmware on your blue AVRISP device. If you try to program your micro-controller with outdated firmware, AVR Studio will force you to update:



AVR Studio will automatically search online for the latest version of your firmware, follow its recommendation to upgrade by clicking the “Upgrade” button.

**During the update process DO NOT disconnect the AVRISP or close the Firmware Upgrade application.**



Upon successful completion you may close the Firmware Upgrade window; you should now be allowed to program your microcontroller.