**Burning sketches to the Arduino board with an external programmer**

If you have an external programmer (e.g. an AVR-ISP, STK500, or [parallel programmer](http://arduino.cc/en/Hacking/ParallelProgrammer)), you can burn sketches to the Arduino board without using the bootloader. This allows you to use the full program space (flash) of the chip on the Arduino board. So with an ATmega168, you'll get 16 KB instead of 14 (on an ATmega8 you'll get 8 KB instead of 7). It also avoids the bootloader delay when you power or reset your board.

**Editing the Arduino preferences file**

To use an external programmer for all boards you need to make some small changes to the Arduino [preferences.txt](http://arduino.cc/en/Hacking/Preferences). Make sure to edit this file when the Arduino IDE is not running.

This preferences file is found in this folder:

\* /Users/<USERNAME>/Library/Arduino/preferences.txt (Mac)

\* c:\Documents and Settings\<USERNAME>\Application Data\Arduino\preferences.txt (Windows)

\* ~/.arduino/preferences.txt (Linux)

Note that when you upgrade Arduino, this folder may not be deleted. If you are having trouble, delete this folder. Upon launching Arduino a new file and preferences.txt will be created.

Change: upload.using from bootloader to the identifier of one of the programmers in hardware/programmers.txt (e.g. avrispmkii).

If you would rather use an external programmer for only an individual board, you can edit the boards.txt file in the hardware/ sub-directory of the Arduino application directory. Set the *board*.upload.using parameter to the identifier of one of the programmers in programmers.txt.

**Burning the sketch**

After you've made these changes, you can burn sketches to the board with the normal upload button or menu item in the Arduino environment. The board needs to be powered, but you don't have to press the reset button before uploading.

You must specify which Arduino board you are using, but you do not have to specify a serial port (on Mac at least).

**Going back to normal USB/serial upload**

To go back to uploading sketches over the USB or serial port without an external programmer, you'll need to set the upload.using preference in preferences.txt back to bootloader; or remove the *board*.upload.using parameter from boards.txt. You'll also need to burn the [bootloader](http://arduino.cc/en/Hacking/Bootloader) back onto the chip on the Arduino board.

**Burning Sketches to the Arduino Mini**

To burn sketches to the mini you must first make a cable to interface between your programmer and the mini board. Gian Pablo Vilamil has written a very good tutorial on [Bootloading the Mini](http://www.arduino.cc/en/Hacking/MiniBootloader). To burn sketches, follow the instructions on bootloading the mini.

Once you have made the proper cable, instead of choosing bootloading from the arduino menu, you can use the instructions on this page to burn sketches directly to the Mini.

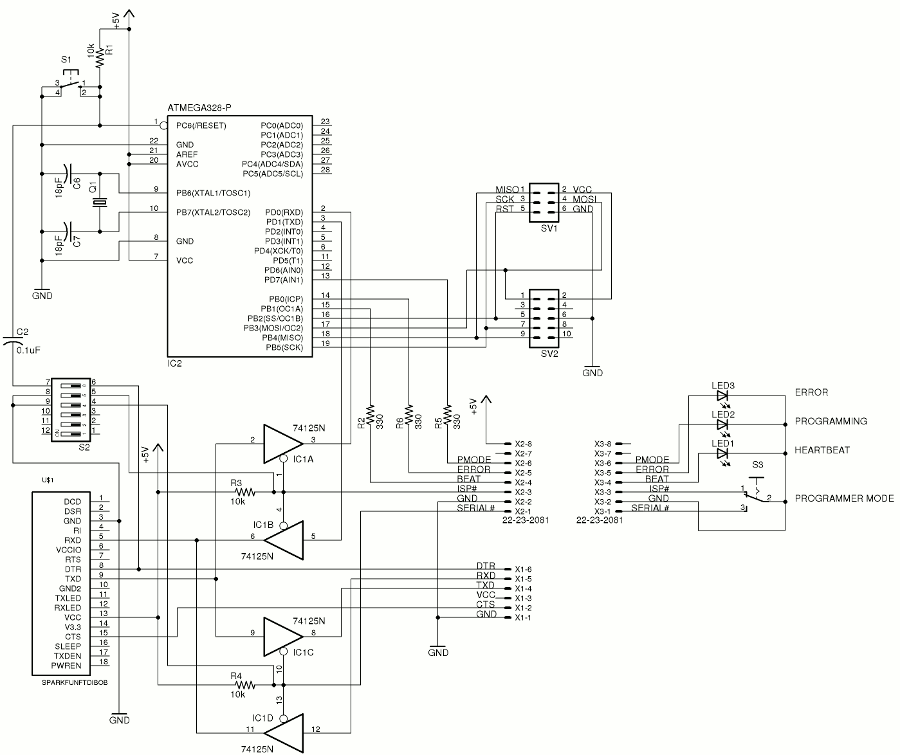
# USB "ArduinoISP" Programmer and Loader using an ATmega8

This circuit is an Arduino without the Arduino, in that it has an ATmega8, ATmega88, ATmega168, or ATmega328 programmed with the Arduino USB programmer code using the [direct programming method outlined here](http://www.arduino.cc/en/Hacking/Programmer). It wasn't obvious to me, but the programmer identifier from programmers.txt is not the name, but the programmer prefix:

* "arduinoisp.name=Arduino as ISP"

In the above example, the programmer name it is looking for is "arduinoisp". The board selected is any board that supports the ATmega you are using. In my case, the ATmega8 is supported by the entry "Arduino NG or older w/ATmega8". Put the Atmega in pretty much any board that has an ISP connector, and program the part as if it were an Arduino. It will burn the sketch without the bootloader into the ATmega and then you can put it in your new programmer. The way I did it was to make a small board that has a 10-pin ISP socket. I programmed the ATmega in the board, then used the 10-pin connector as the programming output connector. The pinout is the same, and the pins are the same, with the exception of one signal - pin 5 of the ISP connector, which is normally connected to reset, is to be connected to pin 16 on the ATmega.

### Schematic Diagram



### Parts

* ATmega8, ATmega88, ATmega168, or ATmega328
* 10 conductor ribbon cable
* 10 conductor ribbon connector
* 6 position 0.1" single row header connector
* Any of the dozens of available USB to TTL converters using the FTDI pinout
* Three LEDs
* 7805 regulator
* Reset switch
* 28-pin IC socket
* Misc resistors, capacitors, diode(s)
* Power connector

### Software

The software needed is the Arduino environment, which comes with the Arduino ISP as an example. It will work out of the box.

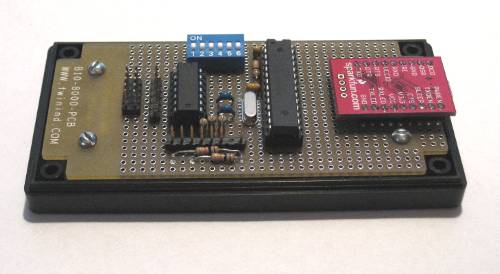
### Notes

1. The target microcontroller may provide it's own power, or get it from the USB bus. I wired this one so the target must supply its own power.
2. S2-6 Controls auto-reset, and is ON to reprogram the ATmega328, OFF in normal operation.
3. S2-5 Enables the serial mux to ATmegax8, and is ON to reprogram the ATmega328 (with the lid removed), OFF in normal operation.
4. S2-4 Enables the serial mux to an external bootloaded AVR, and is OFF in normal operation.

### Assembly

I mounted the breakout board upside down because the pin identification is on the bottom, except for the four that duplicate TX, RX, and a couple more, which are on the top with the LEDs. Keep that in mind when you put the headers on. It took me about four hours to assemble the ArduinoISP programmer. About half an hour was spent hogging out the little openings in the box.

If you decide to build it with just a board, and no external parts, you will want to put the LEDs and switch on the board. In that case you can use a SPDT slide switch to select between the ISP and Serial modes.

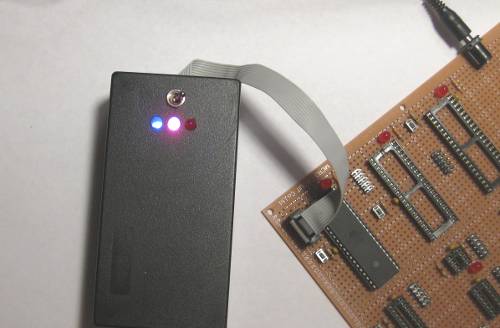


Complete unit without the cables

### Configuration

1. Build the programmer as shown in the schematic.
2. Setup the Arduino environment.
3. Turn dip switches 5 (Serial to the 328) and 6 (Auto Reset) both on to allow uploading to the ATmega328.
4. Hook up a USB cable to the Sparkfun USB to TTL BoB.
5. Load and build the AurduinoISP sketch from the examples.
6. Upload it. It should just say the sketch size when it finishes. Watch the lights on the BoB for activity.
7. Flip dipswitches 5 and 6 off.
8. Plug in the appropriate 6 or 10 pin cable to SV1 or SV2, and to a target.
9. Power up the target.
10. Use AVRDUDE to burn a program into the target.
11. The target will reset and begin execution.

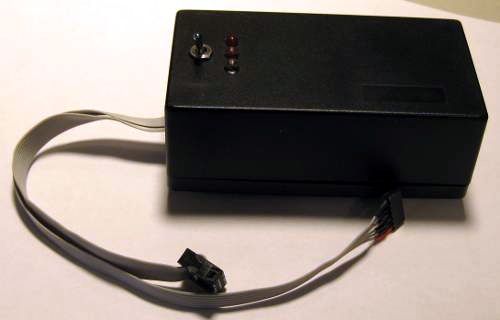
If you have built it without the external switch to select between ISP and Serial downloading, leave dipswitch 5 on, so the ArduinoISP code will see the serial data from the PC. It will then act as an AVR ISP. Open dipswitch 5 and close dipswitch 4 and the serial data will go to the 6-pin sip, for programming an external Arduino.



Programming an ATmega32

If you look at pins 1, 2, and 3 on X2 you will see ISP, GND, and Serial. These go to a toggle switch that selects whether the ArduinoISP gets the serial data from the PC, or the data goes out the 6-pin SIP to another board with a downloader installed. If you choose to use that switch, make sure you turn DIP switch 4 & 5 OFF, or you will be running two outputs into the same wire. To use the serial as an external Arduino sketch downloader, ground "Serial#". To use it as an ISP programmer, ground "ISP#". Do not ground both unless you have extra 74HC125's around. Trust me.

Also on that 8-pin SIP are three lines for LEDs to indicate power, programming, and error. The power LED "throbs" when the power is on - a cute feature of the Arduino ISP. The LEDs are on when the line is pulled high, so ground the cathodes.



Finished USB Multi-mode AVR Programmer

If you have access to an ATmega168 or ATmega328 with the Arduino bootloader in it already, you won't have to write one line of code, nor will you need another programmer to program this one. I picked up a pair of ATmega328's with the Arduino bootloader in them at [Evil Mad Science](http://evilmadscience.com/productsmenu/partsmenu/132). The ATmega328 is getting hard to find right now, but the ATmega168 will do fine. Just make sure to select the right microcontroller in the Arduino environment when you flash the programmer with the Arduino ISP sketch.

You may get the error:

avrdude: Can't find programmer id "arduino"

Don't try to change your local Arduino preference file. Although the local file is supposed to override the global one, the Arduino environment will just change it back. Instead, go to the [AVRDUDE](http://www.nongnu.org/avrdude/) site and get the latest version and install it. It supports this programmer directly, as "arduino".

### Converters:

[Main Site](http://arduino.cc/en/) [Blog](http://blog.arduino.cc/) [Playground](http://playground.arduino.cc/) [Forum](http://arduino.cc/forum/) [Labs](http://labs.arduino.cc/) [Store](http://store.arduino.cc/ww/)

[Sign in](https://id.arduino.cc/auth/login/?returnurl=http%3A%2F%2Fwww.arduino.cc%2Fen%2FHacking%2FMiniBootloader) or [Register](https://id.arduino.cc/auth/register/?returnurl=http%3A%2F%2Fwww.arduino.cc%2Fen%2FHacking%2FMiniBootloader)

[Arduino](http://arduino.cc/en)

Top of Form



Bottom of Form

[Buy](http://arduino.cc/en/Main/Buy) [Download](http://arduino.cc/en/Main/Software) [Getting Started](http://arduino.cc/en/Guide/HomePage) [Learning](http://arduino.cc/en/Tutorial/HomePage) [Reference](http://arduino.cc/en/Reference/HomePage) [Products](http://arduino.cc/en/Main/Products) [FAQ](http://arduino.cc/en/Main/FAQ) [Contact Us](http://arduino.cc/en/Main/ContactUs)

**Learning**   [Examples](http://arduino.cc/en/Tutorial/HomePage) | [Foundations](http://arduino.cc/en/Tutorial/Foundations) | [Hacking](http://arduino.cc/en/Hacking/HomePage) | [Links](http://arduino.cc/en/Tutorial/Links)

## Bootload the Arduino Mini

by Gian Pablo Vilamil

This document explains how to bootload an Arduino mini.

If you know what a [bootloader](http://arduino.cc/en/Hacking/Bootloader) is, why you want to do it, and are just looking for specific instructions for the Arduino Mini, then skip ahead to the section called "bootload the Arduino Mini".

### What is a bootloader, and what is bootloading?

The bootloader is the little program that runs when you turn the Arduino on, or press the reset button. Its main function is to wait for the Arduino software on your computer to send it a new program for the Arduino, which it then writes to the memory on the Arduino. This is important, because normally you need a special device to program the Arduino. The bootloader is what enables you to program the Arduino using just the USB cable.

When we refer to "bootloading" the Arduino, we are talking about using a special device (called an In-System Programmer or ISP) to replace the bootloader software.

### Why do I need to bootload the Arduino?

There are at least four reasons why you might need to do this:

1. Your Arduino does not have a bootloader on it. This might happen if you have replaced the microcontroller with a new chip. This is unlikely to be the case with a Mini, since the chip is not replaceable, and all Minis have the bootloader installed when shipped.
2. Replace the existing bootloader with a different bootloader. The bootloader on Arduinos prior to the Diecimila imposes a 10 second delay after resetting, while it waits for information from the Arduino software. This delay also occurs when the Arduino is powered up, or after programming. There are alternative bootloaders that eliminate these delays, including the Diecimila bootloader, the Adaboot bootloader, the LilyPad bootloader, and others. You can also download the source code and modify the bootloader yourself (see [this page](http://arduino.cc/en/Hacking/Bootloader) for more details and a link to the source). This is useful if you want your project to start up immediately, and especially useful if it receives serial input. If the bootloader receives any input during startup, it will think the Arduino software is trying to talk to it, and delay, possibly indefinitely, starting your program. Here is [some info on alternative bootloaders](http://www.ladyada.net/library/arduino/bootloader.html).
3. Not use a bootloader at all. The process for replacing the bootloader on the Arduino also lets you replace it entirely, with your own application. The advantages of doing this are that your application will start immediately, and it will give you an extra 2K of program memory. The disadvantage is that you will no longer be able to program your Arduino over USB - you will have to use the ISP (at least until you replace the bootloader).
4. The existing bootloader has somehow become corrupted. This is extremely unlikely. However, it is possible that under some circumstances the Arduino flash memory may become corrupted, and you may not be able to program the Arduino normally. Reinstalling the bootloader may fix these problems. It's kind of like reformatting your computer's hard drive.

## Bootload the Arduino Mini

### How is bootloading the Arduino Mini different from programming other Arduinos?

Regular-size Arduinos have a 6 pin connector for the ISP. The Arduino Mini does not have this, so you have to make a connector cable. Other than this, the procedure is the same. With slight variations on the process, you can bootload many different configurations of Atmel-based projects.

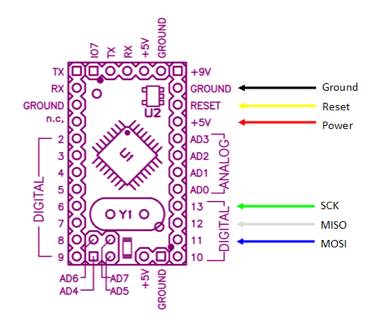
### What do I need?

* An ISP (eg. AVR ISP MkII)
* 6 pieces of wire
* One piece of male header strip, with 12 pins
* Two pieces of male header stip, 3 pins each

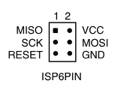
8 AVR Studio software (from Atmel, [free download](http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725))

### How do I make the cable?

You need to connect 6 pins on the Arduino Mini to six pins on the ISP. The six pins are called MISO, MOSI, SCK, RESET, Power and Ground. Three of them correspond to digital I/O pins, numbered 11 - 13. The other three are the Reset, +5V and Ground pins you have probably already been using. (Note that you want to use the +5V pin, not the +9V pin. This is used to supply power to the ISP.)



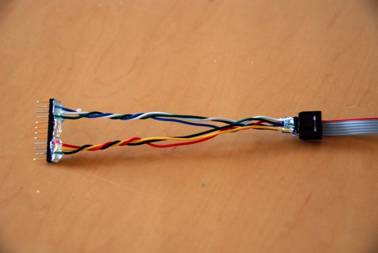
These pins on the Arduino have to be connected to the corresponding pins on the 6-pin connector coming from the ISP, shown below.



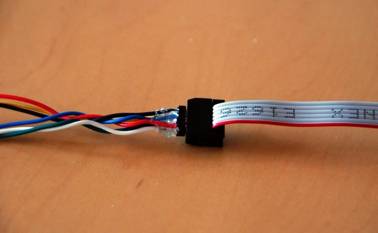
Note that this is the view from the back of the connector. If you hold the connector by the ribbon, with the red stripe uppermost, and the connector facing away from you, this is what you will see.

A simple way to make the connecting cable is to start with 6 pieces of wire, a piece of male header strip with 12 pins, and two pieces of header strip with 6 pins. Remove the 2nd through the 5th pins of the long strip using needle nose pliers. Remove the 9th through the 11th pins as well. Remove all the pins from the two smaller sections of male header.

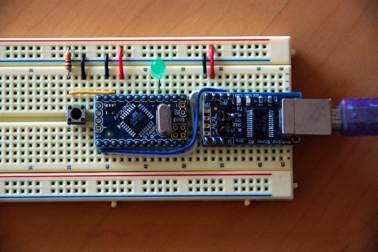
Insert the wires corresponding to ground, reset and power (in that order) in the holes left by the 2nd through 5th pins. Insert the wires corresponding to SCK, MISO and MOSI (in that order) into the holes left by the 9th through 11th pins. Then, take the other ends of the wires, and put MISO, SCK, and RESET into piece, and Power, MOSI, and Ground into the other pieces. (Note that MOSI and Reset "swap sides", so to speak.) Trim the wires so they are all the same length. It is a good idea to use some hot glue to keep everything in place. It is easiest do this by plugging the two segments with 3 headers into the ISP cable (double-check they are correct!) and adding a drop of glue. (Don't glue the ISP cable itself.) You should have a cable that looks like this:



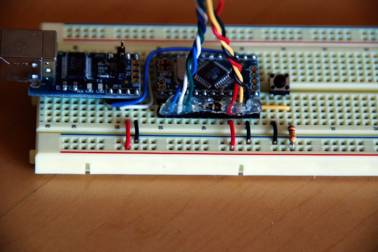
Note that the pins on the ends are not connected to anything - I did this so that the header would be exact length of the Arduino Mini, making it easier to align. Note the order of the wires relative to the red strand on the programmer cable. When the red strand is on the top, and the cable is on the far side of the connector, the colors are white (MISO), green (SCK) and yellow (RESET). In the picture below, with the red strand on the bottom, and the ribbon cable on the near side, the cables are black (GND), blue (MOSI) and red (+5V):



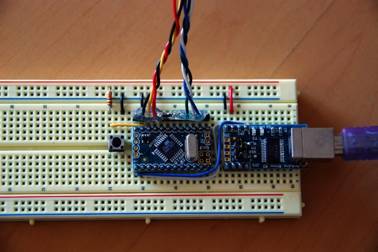
Mount your Arduino Mini on a breadboard, as shown (note that you will have to remove the LED on pin 13 during programming):



Line up the male header with 12 pins along the side of the Arduino Mini, so that it lines up with the pins. The Power, Reset and Ground wires should line up. (Note that the LED has been removed.) It should look like this:



Here is a view from above:



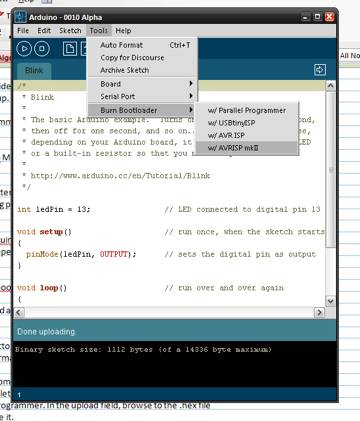
If you are using a USB adapter with your Arduino Mini, plug it into your computer now. If you are not, plug in whatever power source you are using. The Arduino must be powered for ISP to work.

Plug the USB cable from the ISP into your computer now. You should see two green LEDs light up on the programmer. Note: you should not have anything plugged into pins 11-13 of the Arduino. This may interfere with the operation of the programmer.

From this point on, the instructions are exactly the same as those for bootloading any Atmel 168 microcontroller. You can use [the instructions elsewhere on this site](http://arduino.cc/en/Hacking/Bootloader), or continue on.

### How to load the bootloader using Arduino environment?

You can now bootload the Arduino using the Arduino environment. Open Arduino on your computer, set the type of board you are using (Arduino Mini), open the Burn Bootloader menu and choose your programmer. (In this example, AVRISP MkII.)

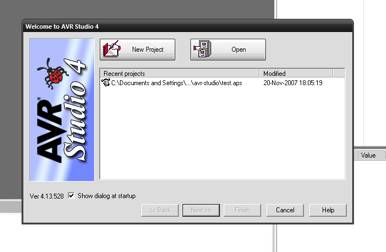


Unlike regular Arduino programming, you don't have to press the reset switch. The programmer resets the Arduino for you. This will install (or restore) the original Arduino bootloader. However, this is of little benefit, unless you are trying repair the bootloader, or your mini somehow came without one.

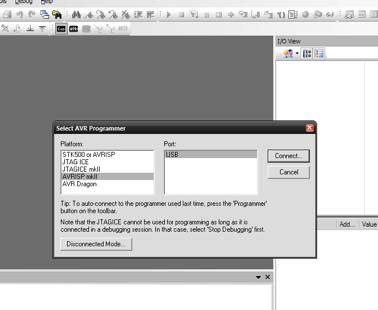
### How to load the bootloader using AVRStudio?

To install an alternative bootloader, you will need a piece of software called AVR Studio. Download it from [here](http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725). Unfortunately, AVR Studio only exists for the Windows environment. However, you can use the instructions for bootloading the Atmel168 on Mac OS/X.

Run AVR Studio. Press cancel to skip the opening screen.

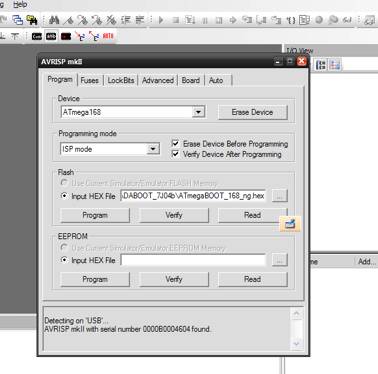


Press the Con button. It is the tiny little button labeled "CON" that looks like a little chip. Then, select the type of ISP that you are using: in this example, AVR ISP Mk II.



You will see the firmware upload dialog box, as shown below. Confirm that all the settings are as shown in the dialog. In the field "input HEX file", browse to the file you want to upload. This can be either:

1. A new bootloader
2. A program of your own. The .hex file will be stored in the applet folder within your Arduino sketch's folder.



Press the "Program" button. You should see some activity in the status window. Press "Verify" if you want to confirm that everything is OK. Disconnect the ISP by removing its USB cable from the computer. If you want to play it safe, remove power from your Arduino. Disconnect the ISP header from your breadboard.

Your Arduino Mini is now bootloaded!

[Bookmark and Share](http://www.addthis.com/bookmark.php?v=300&pubid=xa-50f052251629971d)

©Arduino | [Copyright Notice](http://www.arduino.cc/en/Main/CopyrightNotice) | [Edit Page](http://arduino.cc/en/Hacking/MiniBootloader?action=edit) | [Page History](http://arduino.cc/en/Hacking/MiniBootloader?action=diff) | [Printable View](http://arduino.cc/en/Hacking/MiniBootloader?action=print) | [All Recent Site Changes](http://arduino.cc/en/Site/AllRecentChanges)