**About**

With these configuration files you can program ATmega 168/328p microcontrollers from the [Arduino IDE](http://arduino.cc), without using the Arduino bootloader. It supports chips using external or internal clocks.

Skipping the Arduino bootloader means sketches start immediately after power-on, without any delay, and you have a little extra flash memory available to your programs. Using the (optional) slower internal clock options means you can save on components, but also on power (since a slower chip draws less current).

**Install**

Create an hardware folder inside your Arduino sketch folder if it doesn't already exist. Then place the contents of this repository in a sub-folder called atmega. You should end up with an hardware/atmega/avr folder containing the actual configuration files. Restart the Arduino IDE and the new board options should appear in the Tools > Board menu.



**Programming**

To program the microcontroller you will need an ISP programmer. An [Arduino as ISP](http://arduino.cc/en/Tutorial/ArduinoISP) works just fine:



Choose your ISP programmer in the Tools > Programmer menu. Then choose your ATmega microcontroller from Tools > Board and your choice of clock source and clock frequency from Tools > Clock.

To set the ATmega configuration fuses, use the Tools > Burn Bootloader menu item. This doesn't actually burn an Arduino bootloader onto the chip, it only sets the fuses for the chosen clock settings.

To load programs into the microcontroller, use the Upload button as usual. The IDE will upload the code using the selected ISP programmer.

**Pin Mapping**

The ATmega168 and ATmega328p have identical pin configurations. Check [this diagram](http://arduino.cc/en/Hacking/PinMapping168) for their correspondence to Arduino pin numbering:



**Arduino IDE Versions**

These configuration files are meant for Arduino IDE 1.6 or later. If you're still using version 1.0 of the IDE, you'll need to get the [ide-1.0.x](https://github.com/carlosefr/atmega/releases/tag/ide-1.0.x) release instead.

**Tips and Caveats**

If you're using an ATmega chip that already has the Arduino bootloader inside or has otherwise been configured to require an external clock source, you may get an avrdude: Yikes! Invalid device signature. error. In this case, connect an appropriate external clock source to it (most likely 16 MHz) and try again. Once the ATmega has been configured to use its internal clock source, you can remove the external one and the error shouldn't happen again.

The core delay() function is not very precise for clock rates other than external 8 and 16MHz. The internal clock is not so bad but external 12 and 20MHz drift very noticeably. If this is a problem for you code, beware.