# Atmel Programming Cheat Sheet

## ATmega328P + Atmel-ICE-Basic Programmer + Atmel Studio 7

Note: This Cheat Sheet is intended to be used in conjunction with the accompanying video and the related contents on the associated GitHub site:
(video link here when done)
(GitHub site here when done)

1) Get an Atmel chip, programmer, and necessary common electronic components. This video specifically uses an Atmel ATmega328P-PU and Atmel-ICE-Basic, applicable changes may be necessary if you are using a different chip or programmer. See **Parts List.pdf** and the video for further details.

2) Have a look at **ATmega_328P_ pinout.pdf** to get a general familiarity with the pinout of the ATmega328P.

3) Breadboard **MyAtmelBlink circuit with jumper wires.png**

4) Download and install the latest version of Atmel Studio (Atmel Studio 7.0 is used in the video), the default install settings are ok.

5a) Start Atmel Studio and connect the Atmel-ICE to your computer
5b) Choose the Device Programming icon on the toolbar (looks like a blue chip with a yellow lightning bolt) or choose Tools -> Device Programming. Set "Tool" to "Atmel-ICE", "Device" to "ATmega328P", and "Interface" to "ISP", then choose "Apply"
5c) The first time you connect to the Atmel-ICE (and periodically thereafter) you will be prompted to allow a firmware update, choose to allow the update
5d) When the Atmel-ICE firmware update is complete, return to the Device Programming screen, choose "Tool", "Device", and "Interface" again if they were reset during the Atmel-ICE update, then choose "Apply" again if necessary
5e) Verify your circuit is powered, check voltage across power and ground, verify it's 5V
5f) On the Device Programming Screen, under Target Voltage, choose "Read", the adjacent text box should show "5.0 V"
5g) Under Device Signature, choose "Read", the adjacent text box should show "0x1E950F". The specific device signature is not important to us at the moment, rather we are verifying that the Atmel-ICE can successfully communicate with the chip. Once Target Voltage and Device Signature have been verified close the Device Programming screen

6a) Choose File -> New -> Project, GCC C Executable Project, choose Name and Location as preferred (unchecking "Create directory for solution" is recommended) then choose OK

6b) Choose your chip, ex. ATmega328P
6c) Copy/paste in the additional code from **MyAtmelBlink.c**
6d) Next define F_CPU to inform the compiler of our clock speed; this can be done in either of two ways: in the Atmel Studio menus go to Project -> (Project Name) Properties -> Toolchain -> AVR/GNC C Compiler -> Defined symbols -> Add Item -> add "F_CPU=1000000", or at the very beginning of your program (even before your includes) add the line "#define F_CPU 1000000UL". In **MyAtmelBlink.c** a conditional define is used so the F_CPU define in the code will be used if the menu entry is not made.
6e) Choose the Start Without Debugging icon (hollow green triangle) or choose Debug -> Start Without Debugging, you will be prompted to choose your programmer. For "Selected debugger/programmer" choose "Atmel-ICE", for "Interface", choose "ISP", for the other choices the defaults are ok. Alternatively, to select your programmer you can choose Project -> (Project Name) Properties -> Tool, or choose the hammer icon on the toolbar.
6f) Choose Start Without Debugging again, Atmel Studio will build your project and load it into the chip
6g) The LED on your board should now be blinking

7) As of Atmel Studio 7.0, you may experience valid code being underlined red by Atmel Studio. Various steps can be used to resolve this:
7a) Macros contained in .h files may be underlined red, to resolve this, open the .h file with the macro that Atmel Studio is underlining red. To open .h files, go to View -> Solution Explorer -> expand your project name -> expand Dependencies -> double click on the applicable .h file. The 1st .h file to open to attempt to fix red underlines of valid code will usually be io.h.
7b) If you would rather not have spell checking on for comments, in the Atmel Studio menus go to VAssistX -> Visual Assistant Options -> Underlines -> uncheck "Underline spelling errors in comments and strings". Note that "VAssistX" gets its own menu item on the menu bar, 4th from the left (in between "View" and "ASF")
7c) If the above and the other VAssistX options do not resolve the concern, try VAssistX -> Enable/Disable Visual Assist.
7d) If the concern persists, at your own risk try Tools -> Extensions and Updates -> Visual Assist for Atmel Studio -> Uninstall
7e) If you would like to turn off the code folding option, also known as "Outlining", choose Edit -> Outlining -> Stop Outlining, or alternatively press Ctrl+M, Ctrl+P

8) To make programming much easier in the future, make the Atmel-ICE to breadboard adapter, as shown in **Atmel-ICE to breadboard adapter pics.png**. To clarify the wiring for this, refer to **MyAtmelBlink circuit with breadboard adapter.png.**

9) Breadboard **MyAtmelBlink circuit with breadboard adapter.png**, then load and run **MyAtmelBlink.c** again with the Atmel-ICE to breadboard adapter