**Table of Contents**

1. **Getting Started**
2. **A Microcontroller Program in C**
3. **Hardware Setup – How to Not Destroy the Xmega**
4. **How to Program an Xmega (an example)**
5. **Description of Code for Direct Programming “A1U Manual Position Mode R04” Functionality**
6. **Special Topics**
   1. **Printing Floating Point Numbers**
7. **Getting Started**

Install Atmel Studio. This is the programming environment for programming Atmel (now owned by Microchip) AVRs. It is necessary for reprogramming the Xmega or any other Atmel controller. Atmel Studio supports programming in C or C++ and provides access to many useful C libraries and AVR specific macros. An existing Atmel program will work only for the controller for which it was written, as the controller is specified upon creation of a new program file.

<http://www.atmel.com/microsite/atmel-studio/>

Install the latest version of Tera Term. This is an open source software which permits 2-wire serial communication (USART) with a PC. It is a critical debugging tool.

<https://osdn.net/projects/ttssh2/releases/>

1. **A Microcontroller Program in C**

The code used for microcontroller programming is a combination of C in our case (or C++) and code to access the microcontroller, which varies by microcontroller, and is specified in the microcontroller’s manual, datasheet, and supporting documentation from Atmel in the form of Application Notes.

The manual, datasheet, and relevant application notes used in the programming for the robot balancing platform are provided in the project’s file repository (catalog).

1. **Hardware Setup – How to Not Destroy the Xmega**

If the Xmega is alive and well, it has a green power light up near the power input pin.

The number 1 way to destroy the Xmega is to reverse the polarity on the power input pin to the Xmega. The Xmega has an internal voltage regulator, and can accept POSITIVE voltage inputs between 5V and 12V. If a NEGATIVE voltage is applied to the power input pin, the Xmega will die.

* *The current system is not equipped with a diode, but a diode placed on the input to the breadboard can prevent polarity reversal. However, diodes usually cause a small voltage drop, so slightly more than 5V will need to be fed into the board to get a minimum of 5V on the input to the microcontroller.*

The number 2 way to destroy the Xmega is to apply too much voltage (or too much current) to any individual pin on the board. The Xmega pins are 3.3V logic pins. If 5V are applied to a pin, at least that pin will be destroyed, and it’s possible that the Xmega will die.

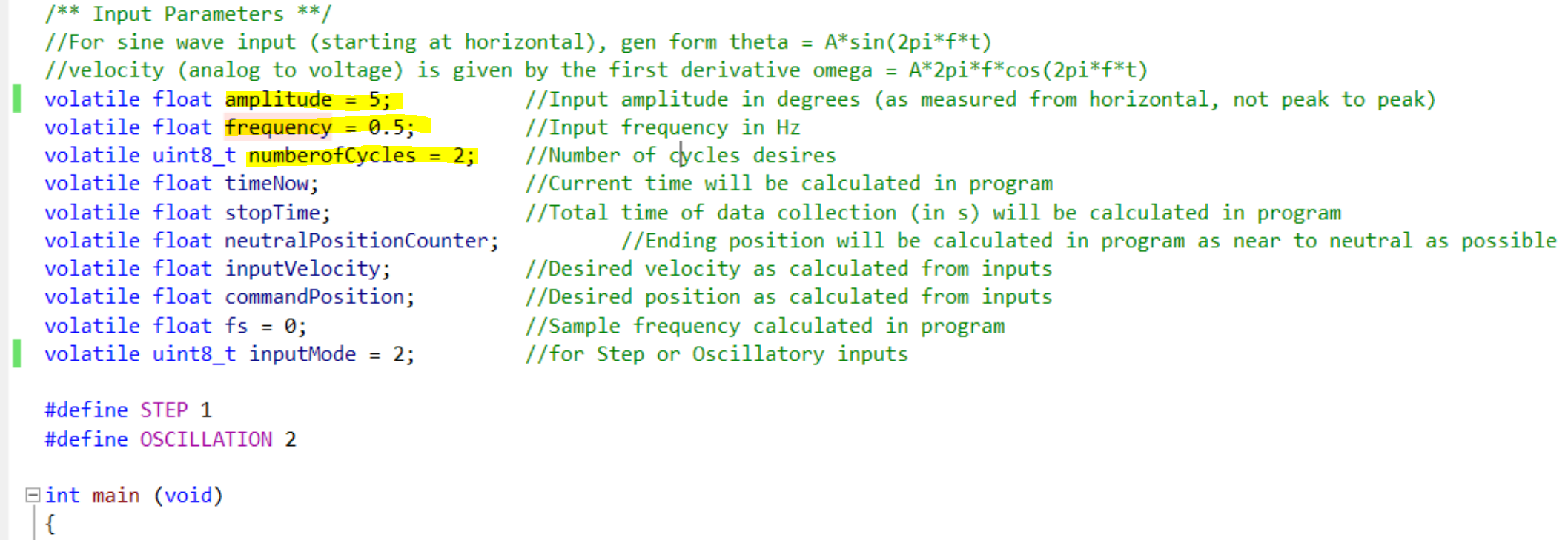
* *Bypass capacitors (0.1uF, 10uF, 100uF) have been placed on the input to the breadboard to prevent current peaks or valleys. Just as too much current can kill a pin (or the whole board), so can too little current.*

The key here is to be sure you know EXACTLY how much voltage you’re putting into first the breadboard, and then into the microcontroller, BEFORE you turn anything on. Best practice is to disconnect all the power connections and measure at every stage leading up to the microcontroller power input. This exercise should be repeated any time there is a possibility that something has changed. For instance, if a lab power supply is being used to power the project, there is always a chance that someone used it for something else. If a wall wart, and voltage regulators are being used, it is much less likely that something has changed while you were away.

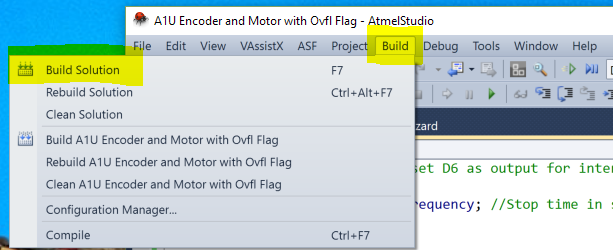
1. **How to Program the Xmega (an example)**
2. Update the code by selecting user inputs

In this example, the user inputs are located directly above the “main” function. The user defines the following characteristics of a sine wave to create a position command signal for the system.

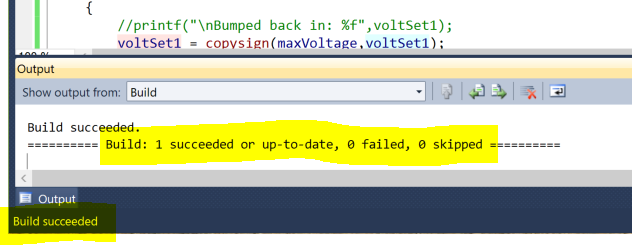
* Amplitude in degrees (as measured from horizontal, not peak to peak)
* Desired oscillation frequency in Hz
* Total number of oscillation cycles desired



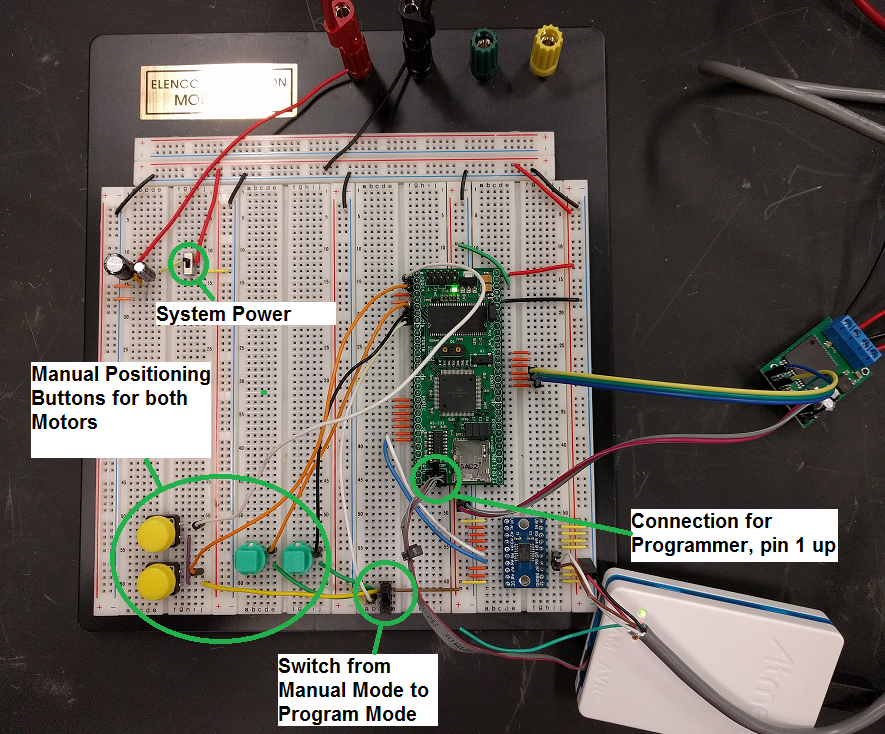
1. “Build Solution” to compile the code



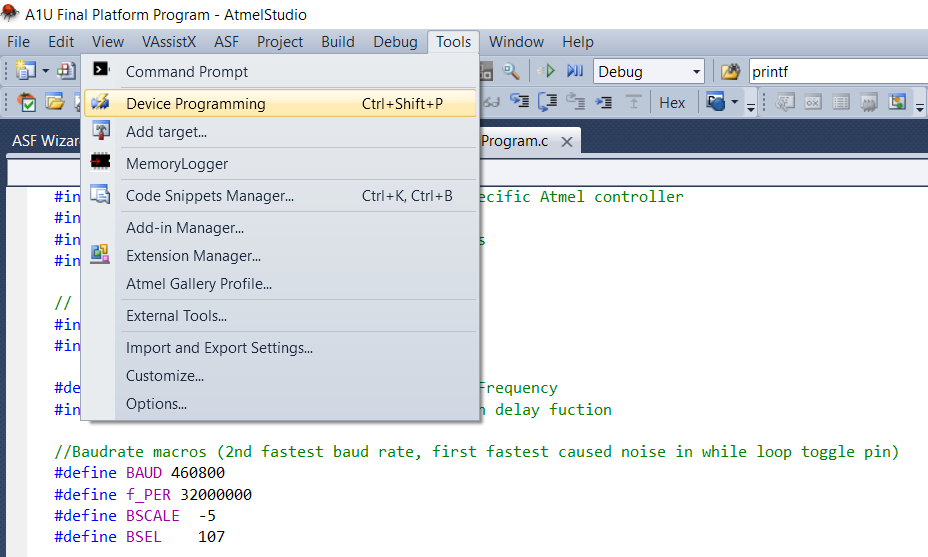
1. Check that your build succeeded in the “Output” section, typically at the bottom of the screen. If the build failed, something is wrong with the code that is preventing it from compiling.



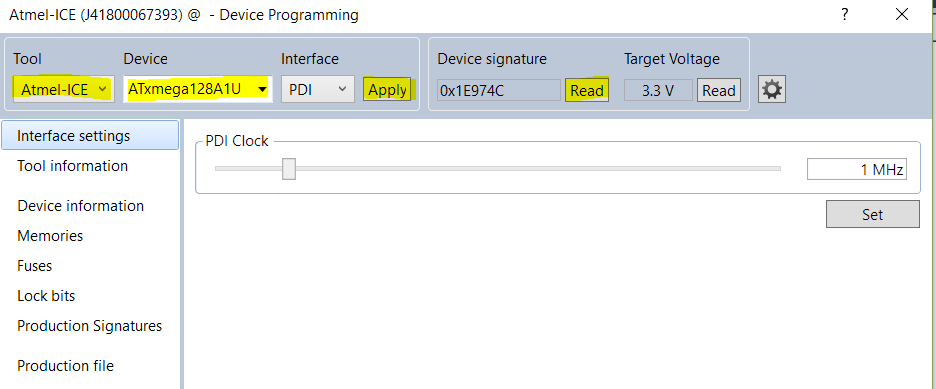
1. Program the Microcontroller
   1. The microcontroller must be on, by flipping System Power switch and the Atmel-ICE programmer must be plugged in



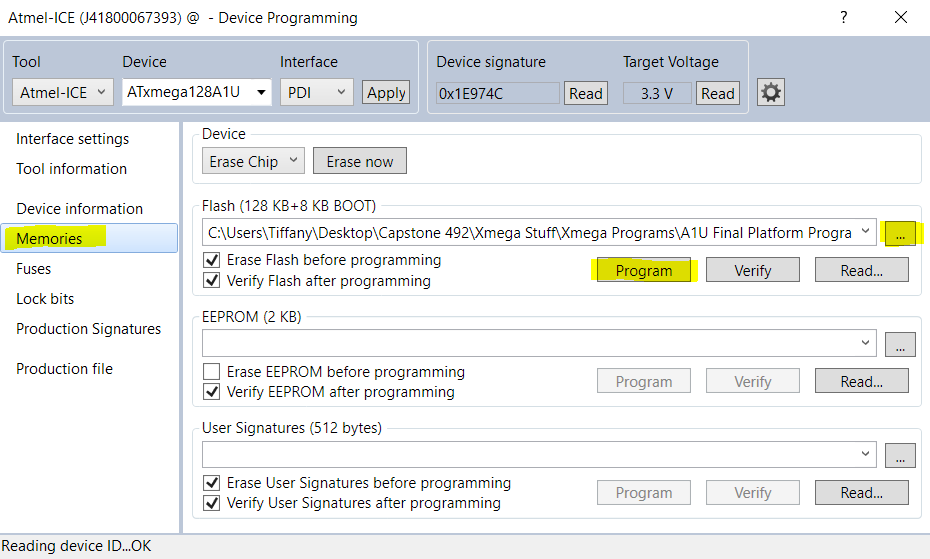
* 1. Go to Tools -> Device Programming



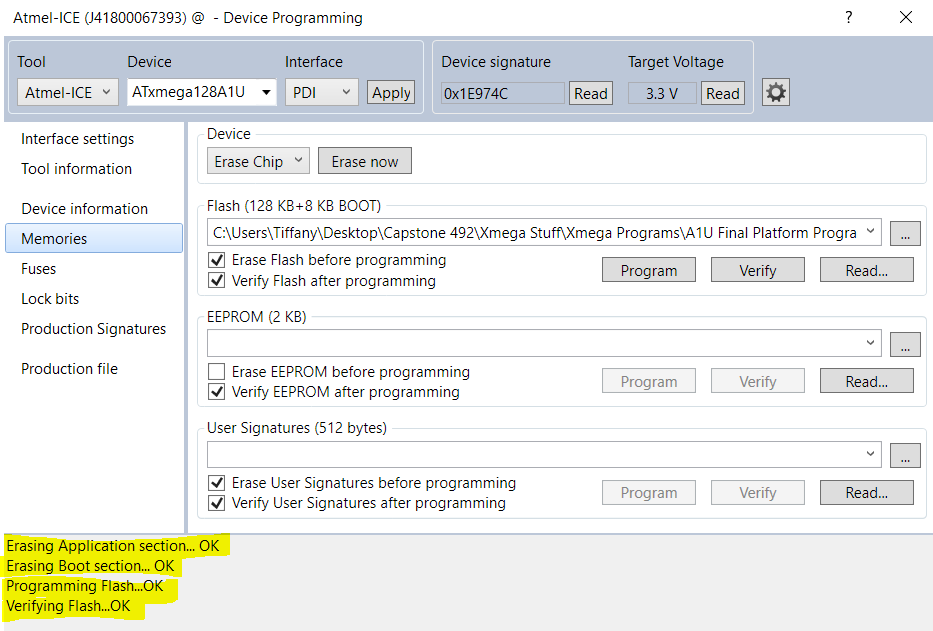
* 1. Select:
     1. Tool Atmel-ICE
     2. Interface PDI -> Apply
     3. Select “Read” next to Device signature
     4. If you receive an error message when reading, check that System Power is on



* 1. In Memories menu:
     1. Browse to find the correct file (always defaults to whatever file was last programmed, not the file you are currently working on.
        1. Select the “.hex” file for programming
     2. Press Program



* 1. If programming was successful, these messages will appear at the bottom of the Memories screen



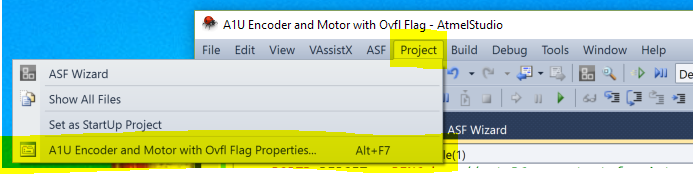
1. **Description of Code for Direct Programming “A1U Manual Position Mode R04” Functionality**

* Upon entering user inputs and programming, code will enter “manual position mode” for both motors, assuming they are both hooked up to a power supply and h bridge
  + In this mode, the buttons on the breadboard may be used to manually position each platform
* When ready to initialize program, flip the small switch toward the bottom of the breadboard near the buttons. This will launch the oscillation or step response that was input prior to programming.
* Use tera term to print the statement at the end of the interrupt to obtain data regarding the command position and actual position of the platform
* Limitations: Currently this program provides manual button positioning for 2 motors, but only inputs a sine wave for motor 1. As all the hardware and software functions for motor 2 are provided in code, only the software must be added to provide sine waves or step inputs for motor 2.

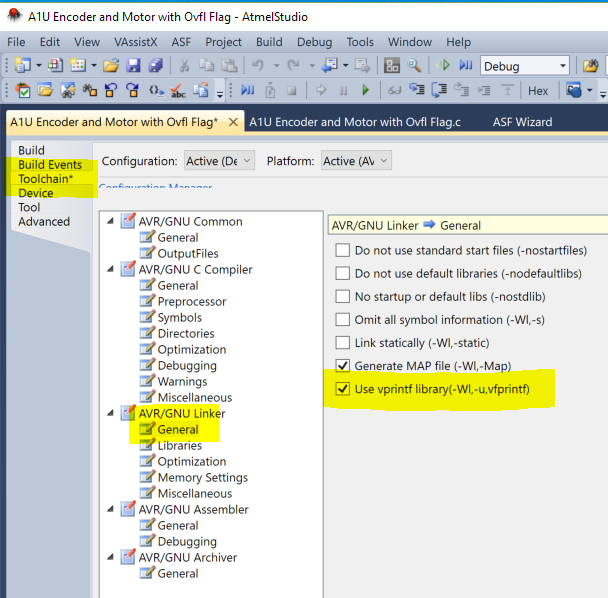
1. **Special Topics**
   1. **Printing Floating Point Numbers**

In Atmel Studio, special steps must be taken in EVERY PROGRAM for which you need floating point print statements to be available. Take these steps prior to compiling the program.

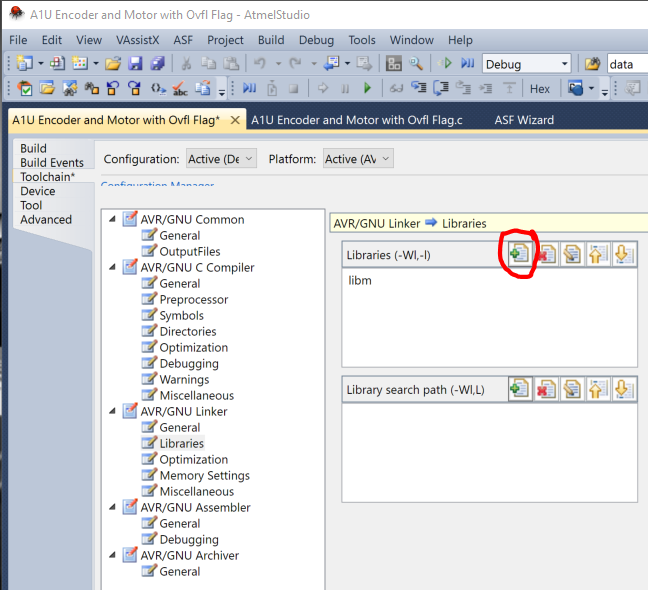
1. Go to the program’s Properties section



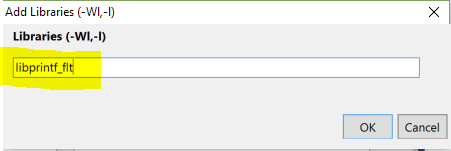
1. In Toolchain -> General, check the “vprint library” box



1. In Toolchain -> Libraries, click the “+” box to add a library



1. Add the floating point library -> “libprintf\_flt” and click OK



1. Complete the program and build as usual. Test that you can print floating point numbers.