Creating and Simulate/Emulating an ASM Project in Atmel

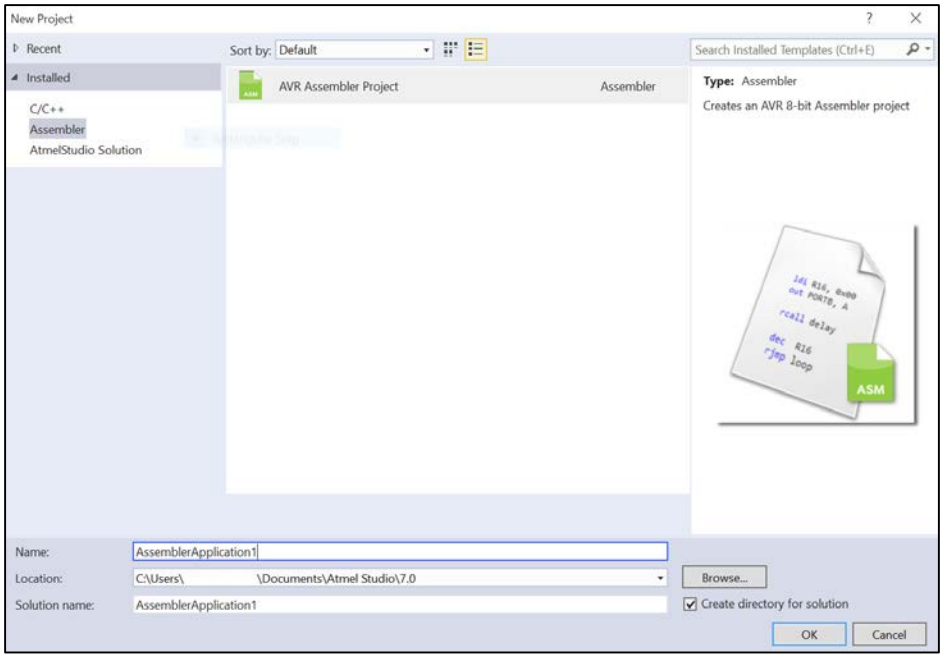
Introduction

The purpose of this document is to enable a student to quickly create a project under Atmel Studio, to simulate the program, and then to emulate the program. To complete this tutorial you will need one additional file, *GPIO_Output.asm*.

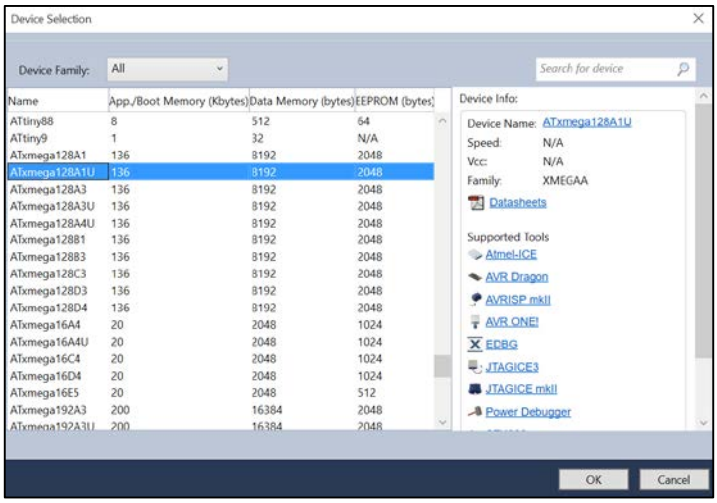
Go to the last page of the document to learn variations using C. Note that when using C, you will need to do something special to work across a network.

Procedure

- This tutorial assumes that you already have Atmel Studio installed and have set your workspace folder to a known location.
- Open Atmel Studio 7 and create a new project **File → New → Project**.
- Under "Installed Templates" choose an AVR **Assembler** Project
- Give the project a name and browse to a desired location. For this tutorial we will call the project "GPIO_Output". (Browse to the location that you want to save your project folder and then Name your project. I usually use the same project name as folder name.) Click OK.

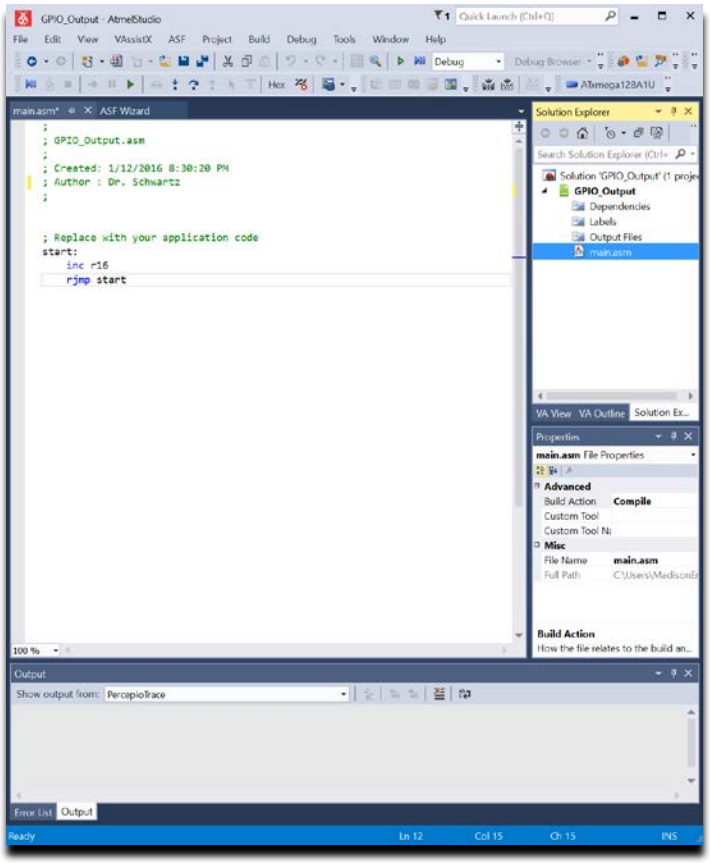


- Select the correct device and click OK. We are using the ATxmega128A1U.



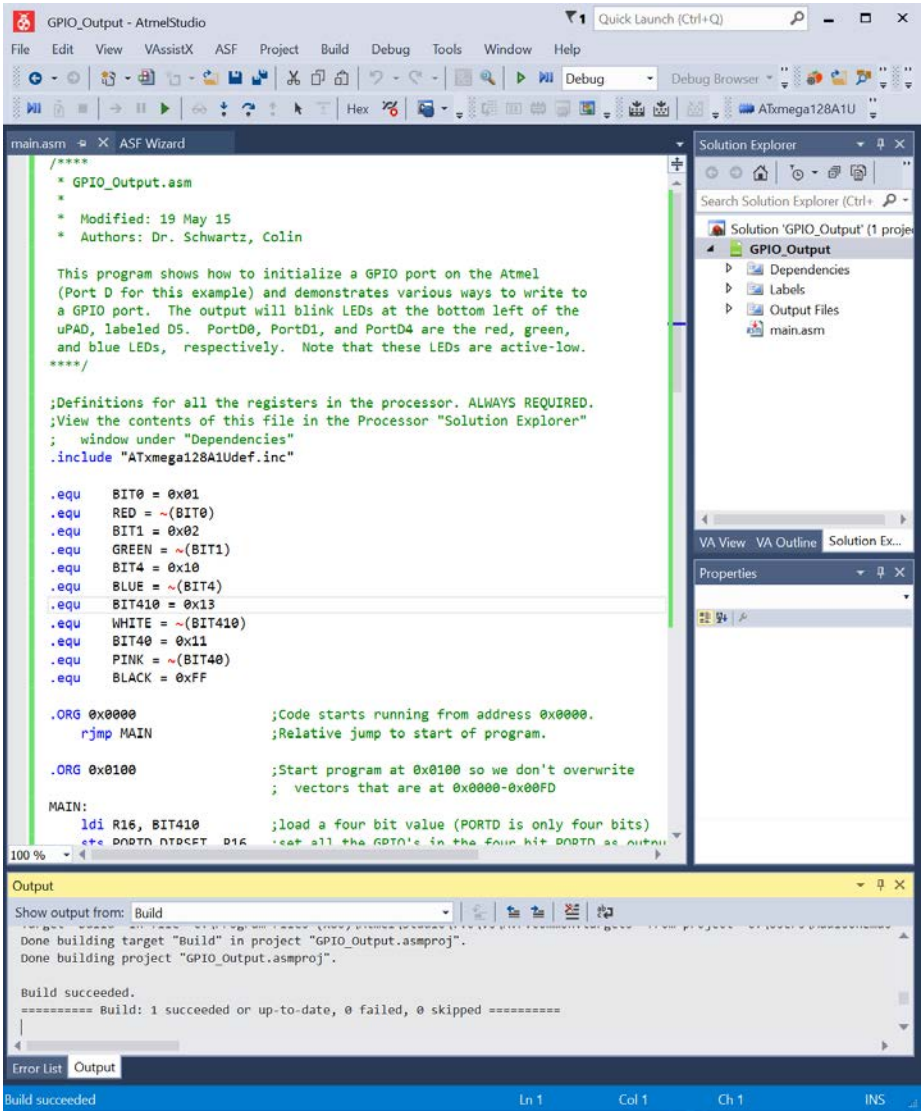
Creating and Simulate/Emulating an ASM Project in Atmel

- You should now see the following:



Creating and Simulate/Emulating an ASM Project in Atmel

- Copy the code from the accompanying GPIO_Output.asm file on the website into the GPIO_Output.asm file. Build the project (which in Atmel is called a **Build Solution**) by going to **Build → Build Solution** (or use function key F7). The output window at the bottom should say "Build succeeded," as shown below.

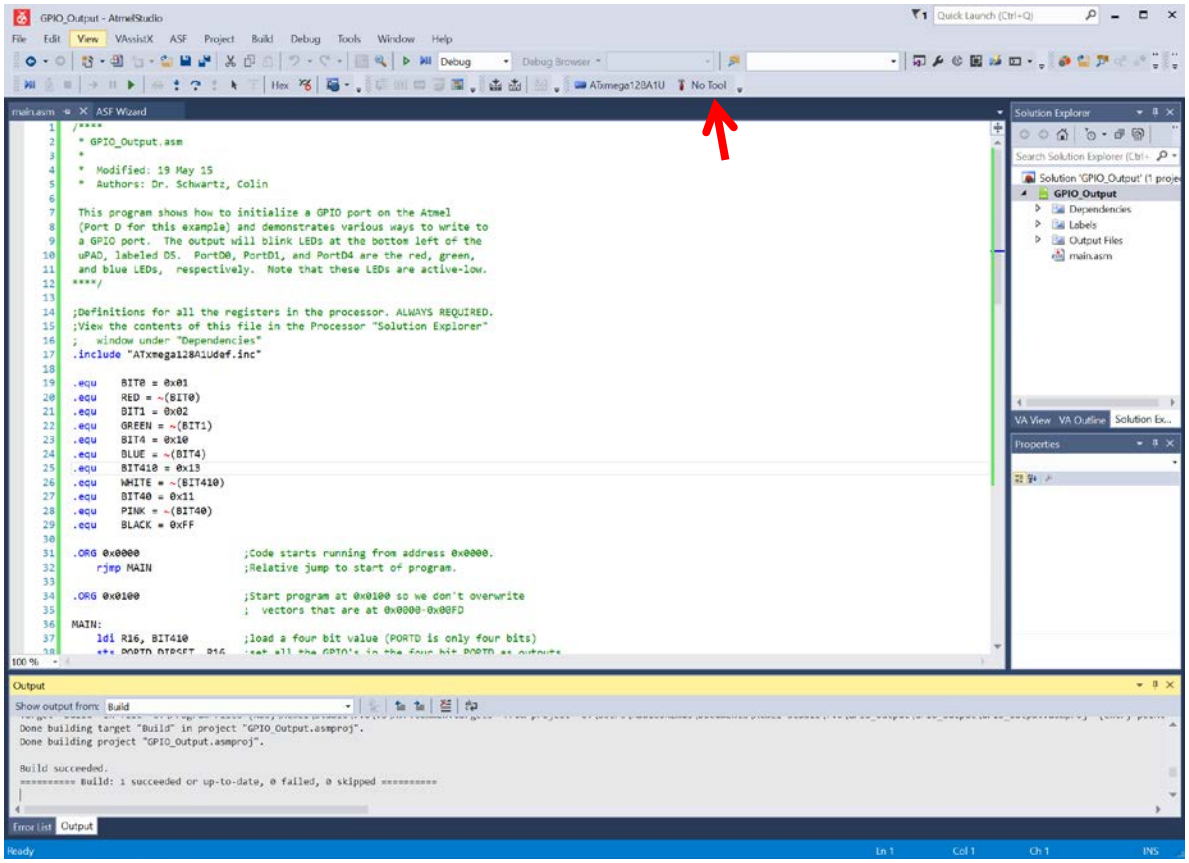


If you have an error or warning (or several of each), you can double-click on the error or wanting and your cursor will be immediately brought to the offending line.

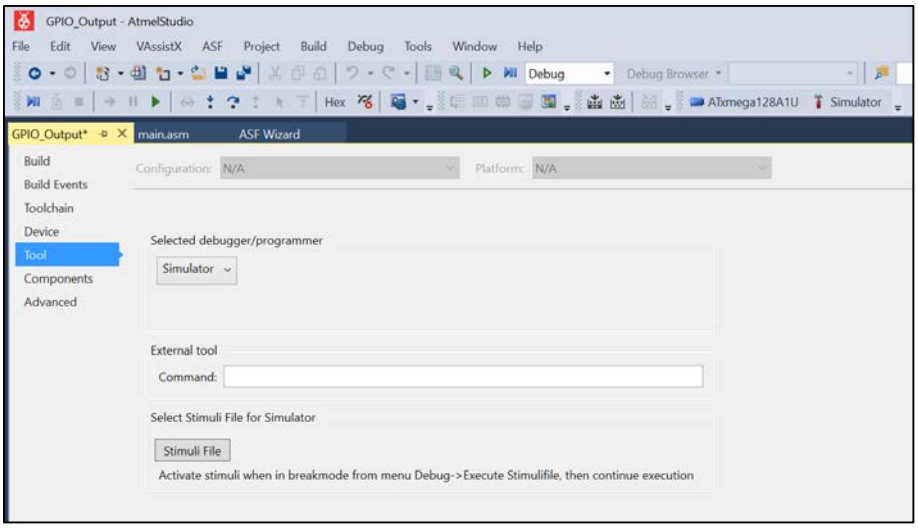
If you would like to add line numbers in the asm file editor (to find warning and errors yourself), go to **Tools → Options → Text Editor → All Languages** and select **Line Numbers** under the **Display** heading.

Creating and Simulate/Emulating an ASM Project in Atmel

- You are now ready to test your code in software (**simulate**) and on the board (**emulate**). First select the debugging tool that you will be using. To do this, click the "No Tool" item. When you put your mouse on the "No Tool," item, the second figure below will appear. Under "Selected debugger/programmer" use the dropdown menu to select "Simluator" (not Atmel ICE). The "No tool selected" box in the first figure below should now display "Simulator" (as shown).

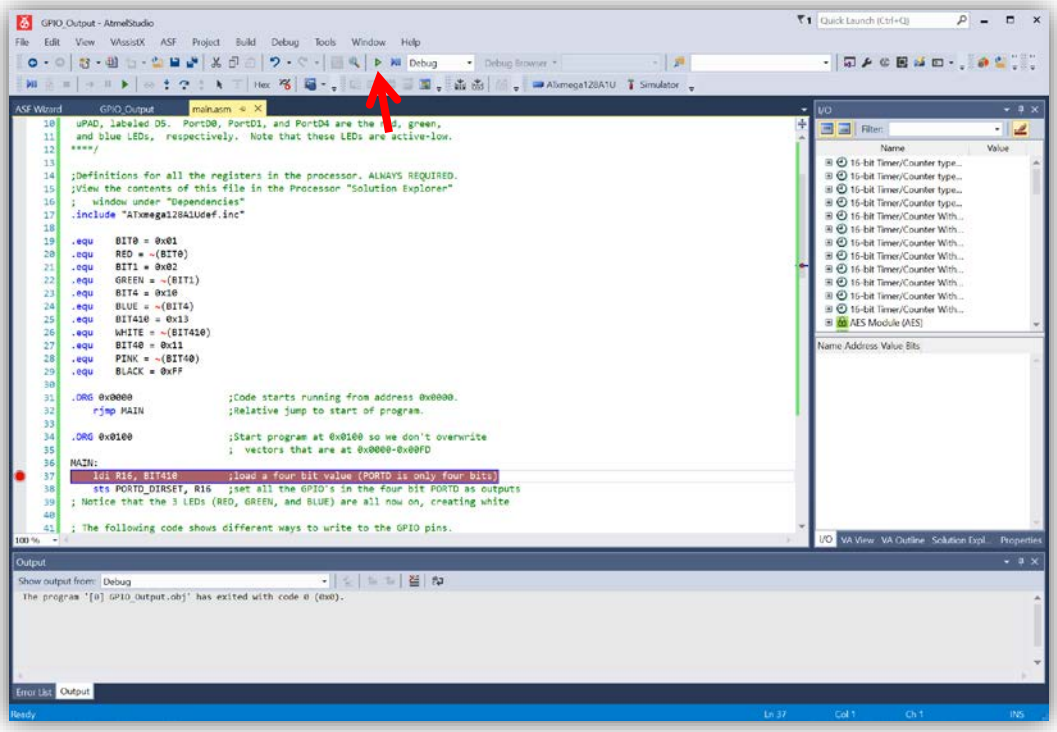


The "Simulator" window should appear as below.



Creating and Simulate/Emulating an ASM Project in Atmel

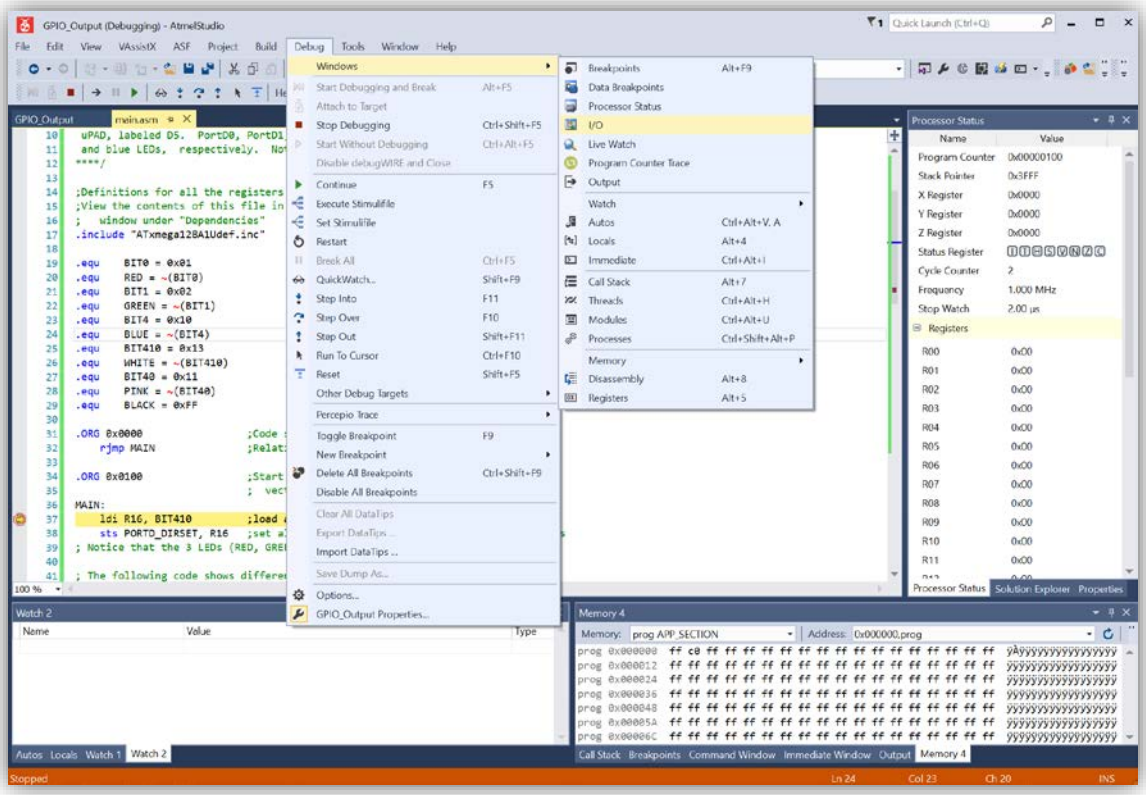
- Place a breakpoint in your .asm window on the first instruction by using your mouse to select a point to the far left of an instruction like below, i.e., `ldi R16, BIT410`. Breakpoints will automatically halt your program *before* executing the instruction at the breakpoint. A red dot will appear where you placed your breakpoint, as shown below.



Select the green arrow at the top (see the red arrow in the above figure), or go to **Debug → Continue,** or simply click **F5** on the keyboard to start the simulation. The program should immediately stop execution at the breakpoint you placed. The window on the right should now display the "Processor" view instead of the "Solution Explorer".

Creating and Simulate/Emulating an ASM Project in Atmel

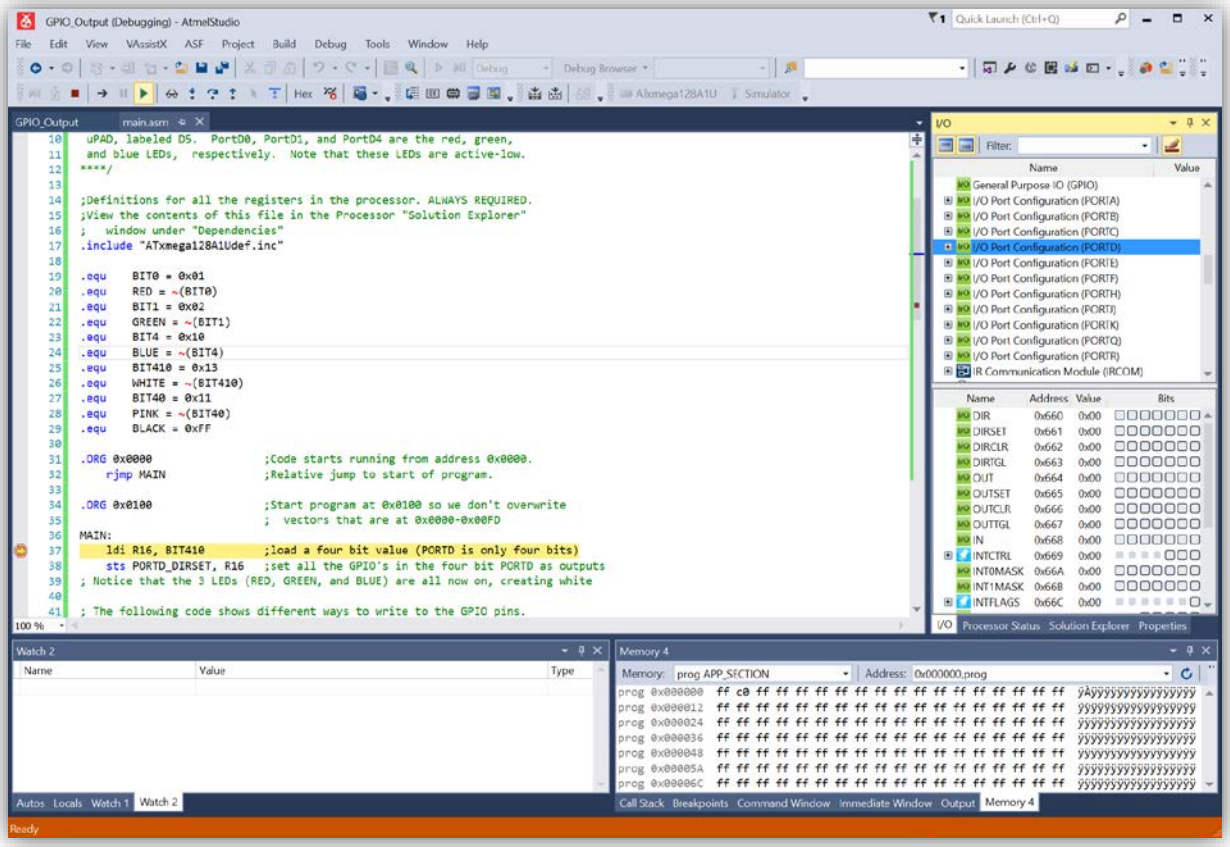
- While debugging, you can view any of the processor's registers, I/O Ports, memory locations, etc. Since we will be using PORTD in this example, we will explore the I/O view of the debugger. Go to **Debug → Windows → I/O View**, as shown below. Also note the **Processor Status View** (above **I/O View**), which is shown in the window on the right. If you do not see Processor View, go to **Debug → Windows → Processor Status.**



- If you want the simulated speed to match the default speed of the actual uPAD board (and the speed when you emulate), select the value next to Frequency in the Processor window and change the 1.000 MHz to 2.000 MHz, which is the built-in default oscillator frequency for our ATxmega128A1U (and is shown changed above).

Creating and Simulate/Emulating an ASM Project in Atmel

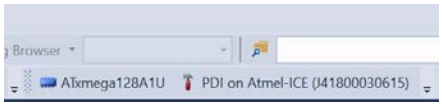
- Now in the I/O View window to the right, find and select **I/O Port Configuration (PORTD).** After selecting it, you will be able to view all of the registers associated with PORTD below. (NOTE THAT THE FIGURE BELOW SHOWS PORTQ, not PORTD, i.e., it needs to be updated.) Step through the code you are debugging by hitting **F11** on the keyboard. You may also step through by selecting **Debug → Step Into**. As you step through the code, notice the changes that occur in the registers of PORTD. Notice the grey box that gives some info on **DIRSET**. Boxes like this appear when you hover your mouse over one of the register names.

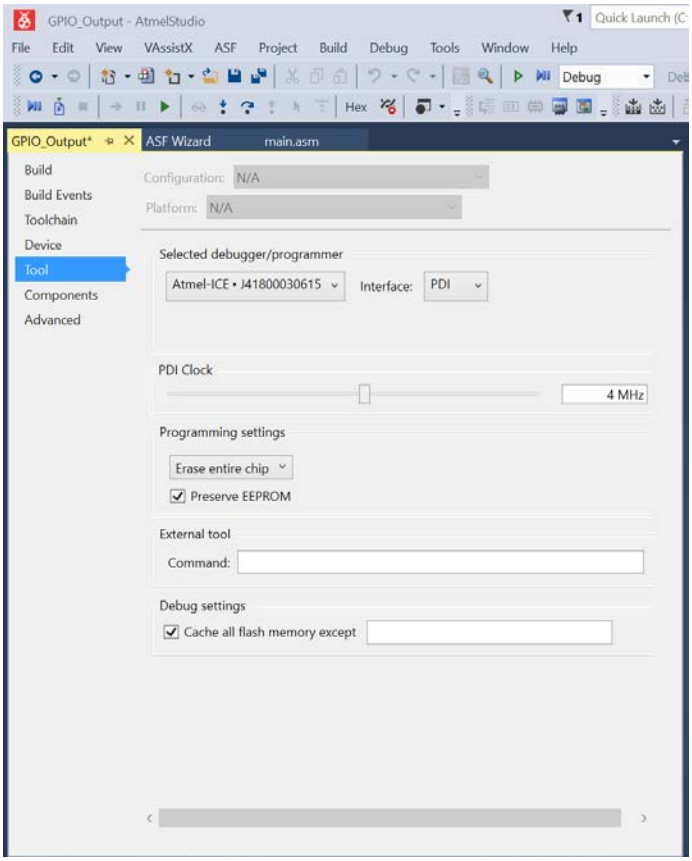


Creating and Simulate/Emulating an ASM Project in Atmel

- Now we will repeat steps 10-12, but the code will now be executing in hardware rather than the software simulator; this is called **emulation** instead of simulation. To stop debugging you click **Debug → Stop Debugging,** or click on the blue square in the toolbar.

- Make sure the Atmel ICE and PCB board are both connected to the computer with your two USB connectors and that the Atmel ICE connector is connected to J1 (next to the pushbutton at the top of the uPAD) For the ICE, use the cable with three female headers. Connect the header at the end of the cable that has 8-pins to the ICE (with the red wire adjacent to the notch under AVR). Connect the middle header, which also has eight pins, to the uPAD (with the red wire adjacent to the dot on the uPAD PCB). Click the box that now says **Simulator** and change the "Selected debugger/programmer" to the **Atmel ICE**. Also select the "Interface" to **PDI** (as shown below). Save the changes and verify that the debugging tool at the top right now says AVR Ice (as shown to the right) and not Simulator. Now you are ready to **emulate** with our uPAD board.





- Repeat steps 10-12. The code will now be running (emulation) on the actually processor. You should see the LEDs at the top of your board (labeled D6, D7, D8, D9) reflect the changes you make to PORTD. These LEDs are connected to PORTD of your processor.

More notes

- In an assembly file, use Ctrl+space to bring up a list of assembly instructions. If you type a letter or two first, it will start at the instruction that starts with those letters. This is sometimes known as **Intellisense**, **Autocomplete**, **Auto Competition**.

Creating and Simulate/Emulating an ASM Project in Atmel

Special Considerations for using C in AtemelStudio

Setting up a Network Drive for C

AtmelStudio can not work accoss networks without using a network drive. Instructions on how to create a network drive depend on which version of Windows you are using.

Windows 10:

- Type **Windows-E** (i.e., hold down the windows key and then type E) to open and Windows Exporer
- In Windows Explorer, select **This PC**
- Select **Computer → Map network drive**
- Select a drive letter, e.g., **Z:**
- Put the path to the folder that you want to use, e.g., \\mil.ufl.edu\tebow\3744\labs\
- Select **Reconnect at sign-in**
- Select **Finish**

Pre Windows 10:

- Type **Windows-E** (i.e., hold down the windows key and then type E) to open and Windows Exporer
- In Windows Explorer, select **This PC**
- Select **Map network drive**
- Select a drive letter, e.g., **Z:**
- Put the path to the folder that you want to use, e.g., \\mil.ufl.edu\tebow\3744\labs\
- Select **Reconnect at logon**
- Select **Finish**

Procedure to Create, Simulate, and Emulate in C

- Open Atmel Studio 7 and create a new project **File → New → Project**.
- Under "Installed Templates" choose an AVR **GCC C Executable** Project
- Find the path to the proper location. If necessary, put in the network drive (e.g., Z:) and then the folder name. Browse if necessary.
- Everything else should work **JUST AS IT DID** in Assembly. The only limitation is the to step through your program, you will need to turn off optimization.
  - Locate the device part number at the top right (look for the picture of the chip) of the Atmel Studio screen: **ATxmega128A1U → Toolchain → AVR/GNU C Complier → Optimization → Optimization Level: None (-O0)**.
  - Don't forget to optimization back on when you are finished debugging.