Atmel Studio 7 Part I

Advanced Developer Software

Deciding Development Software:

- Application Dependent:
  - Embedded Systems
  - Front end GUI
  - Databases
  - AI
  - PCB fabrication
  - Simulation
- Vendor Dependent:
  - ARM
  - Atmel
  - Etc.

Arduino IDE

- Pros:
  - Light Weight
  - Works “out of the box”
  - Comes with examples and samples to help get started
  - Easy to configure with Hardware
  - Open Source
  - Add-ons for additional hardware
    - Ada Fruit
    - Sparkfun
    - Etc
- Cons:
  - Not an engineer's tool
  - Anything more complex than a Hobbyist's breadboard device can be hard to manage

Atmel Studios 7

Pros:

- More professional level tools
  - Autocomplete
  - Project hierarchy
  - Simulator
  - In-System Programming and In-Circuit Emulator (ICE) support

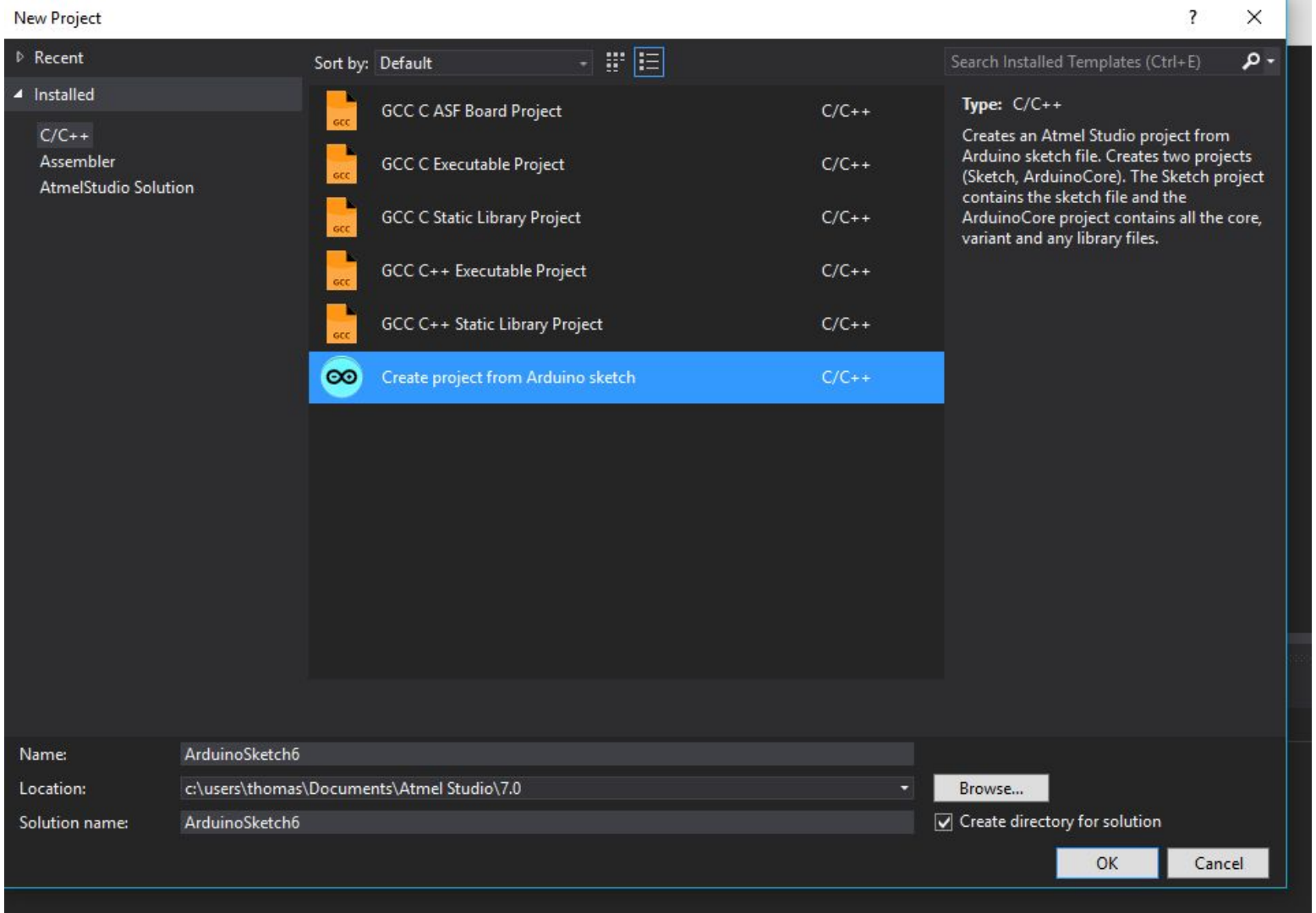
Cons:

- Much more bulky
  - Built on top of a Microsoft Visual Studios Shell. (~3GB in size)
- Another software to learn and feel familiar with.

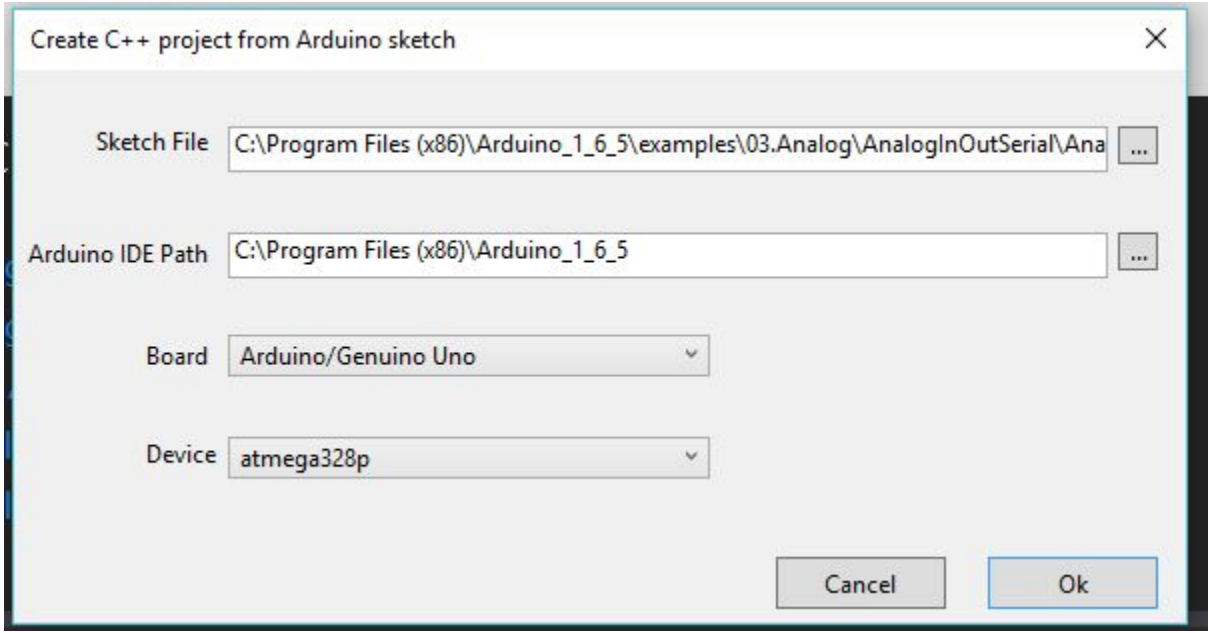
Importing Arduino Script

Process:

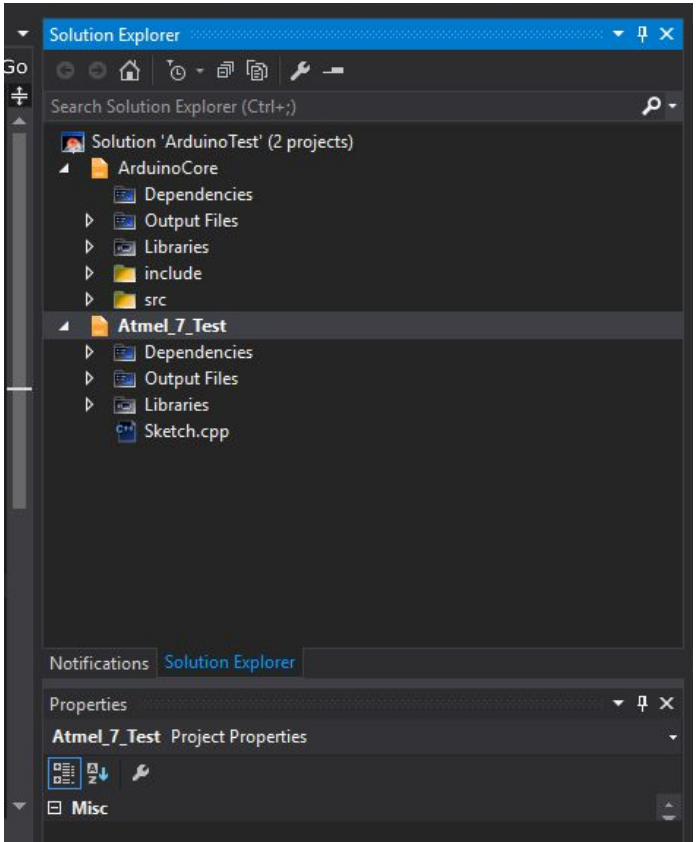
- Make an Arduino script or take a blank one
- Go to “file” -> ”New project” -> “import Arduino Script”
    - Pick the path of desired script and Arduino IDE install path.
  - Select Board type and Device type
    - For our projects these will either be “Leonardo” or “Lillypad USB”


New Project
Recent
Installed
C/C++
Assembler
AtmelStudio Solution
Sort by: Default
Search Installed Templates (Ctrl+E)
GCC C ASF Board Project C/C++
GCC C Executable Project C/C++
GCC C Static Library Project C/C++
GCC C++ Executable Project C/C++
GCC C++ Static Library Project C/C++
Create project from Arduino sketch C/C++
Type: C/C++
Creates an Atmel Studio project from Arduino sketch file. Creates two projects (Sketch, ArduinoCore). The Sketch project contains the sketch file and the ArduinoCore project contains all the core, variant and any library files.
Name: ArduinoSketch6
Location: c:\users\thomas\Documents\Atmel Studio\7.0
Browse...
Solution name: ArduinoSketch6
Create directory for solution
OK
Cancel

Arduino Import



Hierarchy


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ArduinoTest' (2 projects)
ArduinoCore
Dependencies
Output Files
Libraries
include
src
Atmel_7_Test
Dependencies
Output Files
Libraries
Sketch.cpp
Notifications
Solution Explorer
Properties
Atmel_7_Test Project Properties
Misc

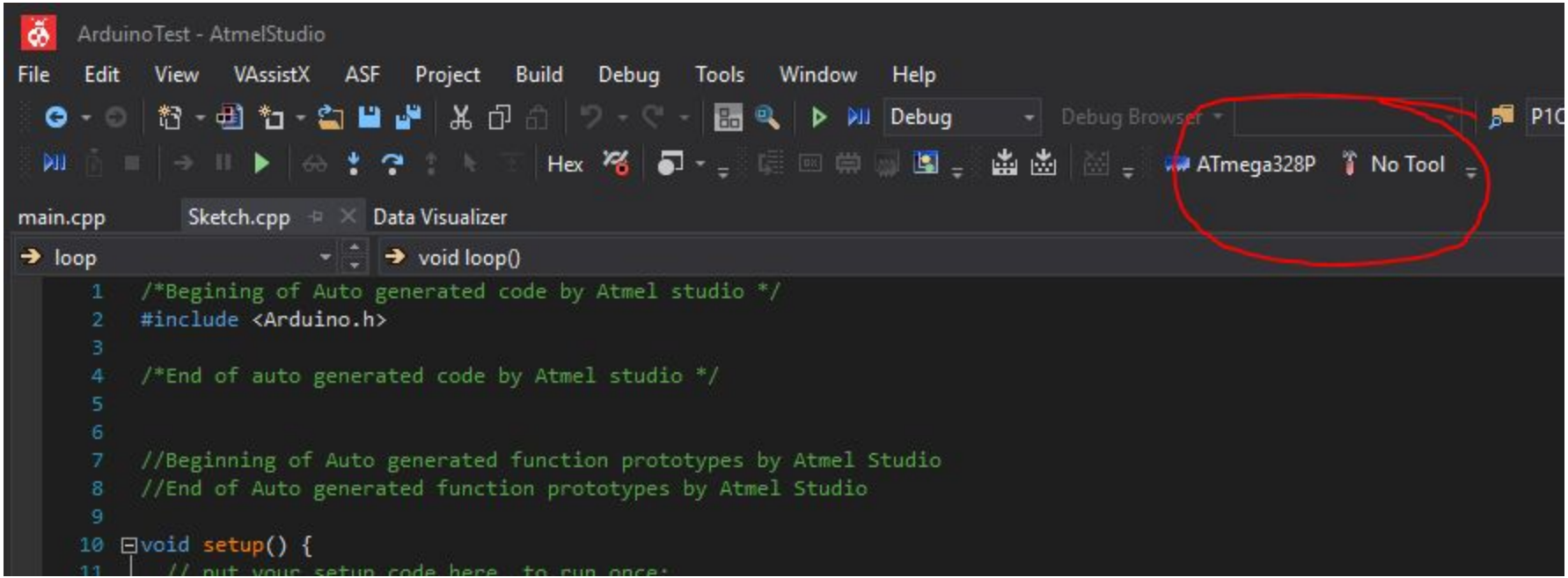
Hierarchy Cont.

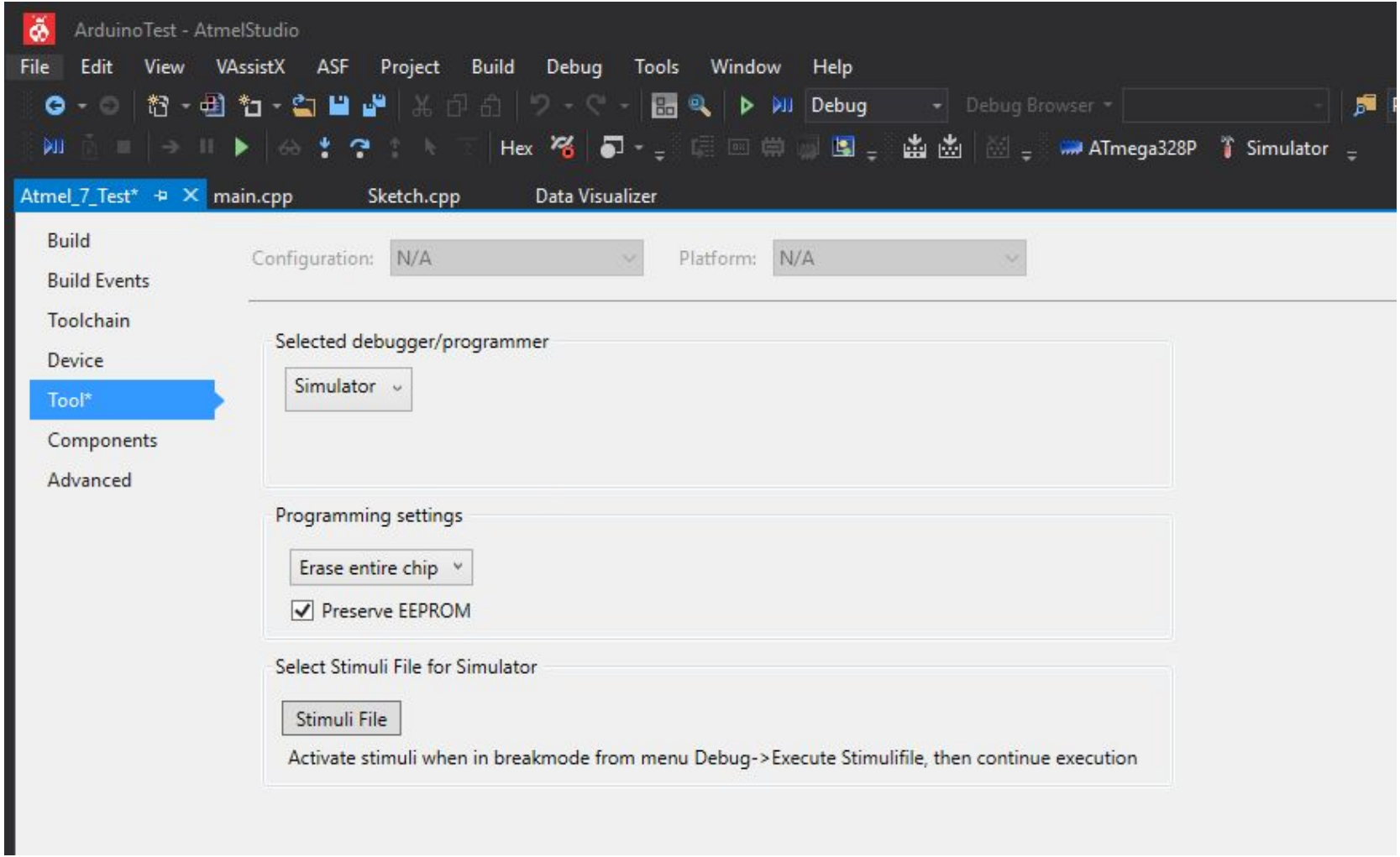
For Arduino projects the Hierarchy is important for showing two different things:

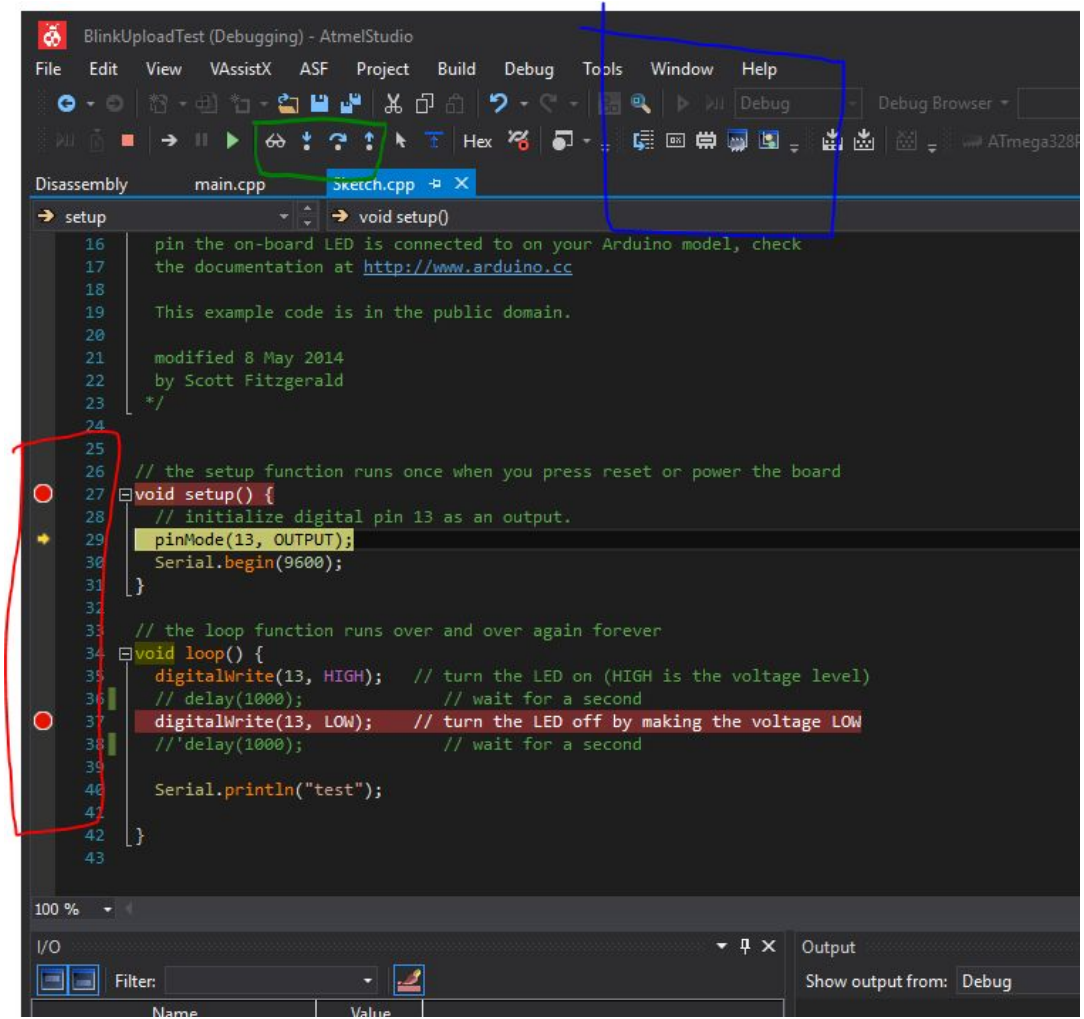
- Arduino Base Code
  - Looking into the base code is important for finding microcontroller specific defaults.
- Better Organize your project source files
  - Main Script
  - Headers/ Src.

Simulator

- Similar to AVR Studio 4’s simulator, used in EE346, the Atmel Studio 7 simulator provides a quick method of verifying your code.
  - Go to “tools” as seen in the toolbar the select “debugger” -> “simulator”
  - Same as other simulators, utilize the stepping tools as needed.
  - Warning: Delays are not modeled by the simulator and therefore will not reflect actual timing within the target application.
    - Easier to set breakpoints than delays (comment out if possible).
  - Also Note: Serial.print commands inconsistent also (due to not having a port connection)
- Pin simulation
  - As opposed to software development, our simulators allow us to see IO register, General Purpose Registers, as well as PIN states (HIGH, LOW)
  - This gives us more flexibility to debug specific sections of the program.


ArduinoTest - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Debug Browser
Hex
ATmega328P
No Tool
main.cpp
Sketch.cpp
Data Visualizer
loop
void loop()
1 /*Begining of Auto generated code by Atmel studio */
2 #include <Arduino.h>
3
4 /*End of auto generated code by Atmel studio */
5
6
7 //Beginning of Auto generated function prototypes by Atmel Studio
8 //End of Auto generated function prototypes by Atmel Studio
9
10 void setup() {


ArduinoTest - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Debug Browser
Hex
ATmega328P
Simulator
Atmel_7_Test*
main.cpp
Sketch.cpp
Data Visualizer
Build
Build Events
Toolchain
Device
Tool*
Components
Advanced
Configuration: N/A
Platform: N/A
Selected debugger/programmer
Simulator
Programming settings
Erase entire chip
Preserve EEPROM
Select Stimuli File for Simulator
Stimuli File
Activate stimuli when in breakmode from menu Debug->Execute Stimulifile, then continue execution


BlinkUploadTest (Debugging) - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Debug Browser
Hex
ATmega328P
Disassembly
main.cpp
Sketch.cpp
setup
void setup()
pin the on-board LED is connected to on your Arduino model, check
the documentation at http://www.arduino.cc
This example code is in the public domain.
modified 8 May 2014
by Scott Fitzgerald
*/
// the setup function runs once when you press reset or power the board
void setup() {
// initialize digital pin 13 as an output.
pinMode(13, OUTPUT);
Serial.begin(9600);
}
// the loop function runs over and over again forever
void loop() {
digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
// delay(1000); // wait for a second
digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
//'delay(1000); // wait for a second
Serial.println("test");
}
100 %
I/O
Filter:
Name
Value
Output
Show output from: Debug

Simulator operation:

- Red: setting breakpoints on the left margin of line numbers
- Blue: Memory and register inspector windows
  - Left to right: Disassembly, registers, Memory 1, Processor Status (SREG), IO
  - For our purposes, IO will be the primary concern
- Green: Simulator control
  - Left to right:
    - Add watch: look at variable address and value
    - Step into: look “into” function and step through.
    - Step over: execute function in its entirety (or until breakpoint/ inf. loop)
    - Step out: Done to leave a “step in”

Live Debugging

Atmel ICE programmer:

- Provides: ISP (in-system programmer), fuse settings, .hex- uploading, and bootloader configuration.
  - ISP enables a deeper level of simulator where the hardware is running while connected to the PC.
  - Use this to further understand complex problems and pin-point if it requires hardware or software changes.

Secondary Discussion: Bootstrap vs. Bootloader

- Bootloader:
  - Used on most (all) computers from embedded systems to desktop computers to ensure successful power-up sequence.
  - Bootloaders are hardware specific as they operate at a C/ assembly level to configure IO based on application
  - For example: Arduino uses a bootloader and on power-up it checks if the arduino IDE/ AVR dude is uploading code via serial connection.
  - Resides in FLASH program memory at specific allocated locations.
- Bootstrapping:
  - Used in applications that require an operating system.
  - The bootstrap ensures the kernel (“OS”) is uncompressed and not corrupted before handing the device over to the OS.
  - Stored in non-volotile memory (FLASH, ROM, EEPROM)