Getting Started with the Atmel Mega2560
Questions?
Solderless Breadboards

- Power bus (red)
- Ground bus (blue)
- Component bus

Note that the two sides are not connected.
Wiring Standards

When possible, use wire colors for different types of signals:

- Black: ground
- Red: power
- Other: various signals
Clean Wiring
A clean breadboard will make debugging easier – and it makes circuits more robust

www.linefollowing.com
tangentsoft.net
Care with Power

• Only insert components and wires into the breadboard when power is disconnected
• “Wire, check-twice, then power”
  • Never reverse power and ground (this is a very common mistake)
• Most chips that we will use expect +5V
  • More can destroy the chips
  • We will use DC/DC converters to step battery voltages down to +5V
Suggested Wiring Procedure

• Power supply
• Power/ground buses
• Insert primary components
• Wire power/ground for components
• Add signals and remaining components
• Test incrementally
Debugging Techniques

• Test incrementally
• Test intermediate sub-circuits
Physical Interface for Programming

AVR ISP
Physical Interface for Programming
AVR ISP
USB connection to your laptop
Physical Interface for Programming
AVR ISP

Header connection will connect to your circuit (through an adapter)

Be careful when you plug your circuit in (check before powering)
AVR ISPs are Cranky

• When things are plugged in and powered, you should see two green LEDs on the ISP (on most units)
• One red: usually means that your circuit is not powered
• Flashing orange: connector is backwards!
• Orange: the programmer is confused
  • Could be due to your circuit not being powered at 5V
  • Could be due to other problems
  • Check power and reboot the ISP
Compiling and Downloading Code

Once the chip is programmed, the AVR ISP will automatically reset the processor; starting your program
Hints

• Use LEDs to show status information (e.g., to indicate what part of your code is being executed)

• Remember: on the Arduino boards, there is a LED connected to port B, pin 7

• Have one LED blink in some unique way at the beginning of your program

• Go slow:
  • Implement and test incrementally
  • Insert plenty of pauses into your code (e.g., with delay_ms())
Project 0

• Summary:
  • Write program that flashes the LED attached to PORTB, pin 7 at a chosen (visible) frequency.
  • Connect 4 LEDs and a switch to your Arduino board
  • Write a program that: waits for the switch to close, then displays an interesting LED flashing pattern

• Details are on the class web page
Compiling and Downloading

Preparation:

• Create a class folder to work in: e.g., “ame3623”
• Check out your group’s svn tree into this folder:
  • [http://www.cs.ou.edu/~fagg/classes/ame3623/svn.html](http://www.cs.ou.edu/~fagg/classes/ame3623/svn.html)
Compiling and Downloading

Preparation (unix only):

• You will work in: csesX/project0/project0/

• Makefile:
  • Copy csesX/makefile to project0/project0
  • No changes need to be made now, but the key lines are:
    • “TARGET” line is the name of the C file with the main function. Here, we have chosen “main”
    • “OULIB_DIR” references csesX/oulib/. In this example, it should be “../../oulib/”

• Create your C file in project0/project0/main.c
  • Most of you are using XCode for this
Compiling and Downloading (the Unix way)

At the command line:

• “cd” to project0/project0/

• Type “make”
  • You should see no errors
  • If there are errors, then you must fix them before moving on

• Type “make program”
  • This will download your code to the processor
  • Again, you should see no errors
Windows: Getting Started
New Project

Browse to your `csesX\project0` folder
Solution name: uncheck “create directory for solution”
Select the ATmega2560
Project ➔

<Project Name> Properties (Alt+F7)

1. Toolsheet
2. Macros
3. Add: F_CPU=16000000
4. Add: atmega2560

Andrew H. Fagg: Embedded Real-Time Systems: Atmel
Compiling
Compiler Optimization

1. Select the Optimization option.
2. Set the Optimization Level to "Optimize most (-O3)".
Add Directories

1. Select "Directories" from the Toolchain menu.
2. Click the "Add Include Paths (-I)" button.
3. Browse to your Include folder.
4. You should end up with something like: ..\..\..\oulib\include
Add Libraries

1. Enable libraries.
2. Add your lib folder.
3. Add “ou_atmega2560”
Now for the code…

#include "oulib.h"

int main(void)
{
    DDRB = 0x80;       // port B, pin 7

    while(1) {
        // Your code here
    }
}
Build menu: Build Solution
You should get this
Now We Are Ready…

• Plug the programmer into your computer and into the Arduino board (If it is not already)
• Make sure your Arduino board has power
  • Either from USB or batteries
• And download the program…
  • Tools Menu: Device Programming
Select the AVR Mk II

NOTE 1: you only need to do steps 5 & 6 the first time you use a particular blue box

NOTE 2: if you are asked to upgrade the firmware, then do so
2: Find the `<Project Name>.elf` file. It will be in your Debug folder.

3: Press to program.
Flashing?

Your program will start executing as soon as the download is complete …

Your on-board Light Emitting Diode should be blinking
Next Time

Finite State Machines