Getting Started with the Atmel Mega2560
Questions?
Quiz
Port-Related Registers

Some of the C-accessible registers for controlling digital I/O:

<table>
<thead>
<tr>
<th></th>
<th>Directional control</th>
<th>Writing</th>
<th>Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port B</td>
<td>DDRB</td>
<td>PORTB</td>
<td>PINB</td>
</tr>
<tr>
<td>Port C</td>
<td>DDRC</td>
<td>PORTC</td>
<td>PINC</td>
</tr>
<tr>
<td>Port D</td>
<td>DDRD</td>
<td>PORTD</td>
<td>PIND</td>
</tr>
</tbody>
</table>
Arduino Mega Board

(see schematic)
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”
• Download libou_atmega2560.a, oulib.h from the Atmel HOWTO page
• Inside of your class folder, create folders “oulib”, “oulib\lib” and “oulib\include”
  • Place libou_atmega2560.a in oulib\lib\n  • Place oulib.h in oulib\include
Compiling and Downloading

Preparation (unix only):

• Create a project folder, e.g. “testproject” in your class folder

• Download makefile into this directory:
  • Modify the “TARGET” line to be the name of the C file that you are about to create
  • Modify the “OULIB_DIR” line as necessary. In this example, it should be “../oulib/”

• Create your C file in testproject
Compiling and Downloading
(the Unix way)

At the command line:
• “cd” to your project directory

• 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

Choose a folder name for your project
Select the ATmega2560
Project ➔

*<Project Name>* Properties (Alt+F7)

1. Tools
2. Compiler Settings
3. Compiler Options
4. Add: F_CPU=16000000 atmega2560

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

1. **Optimization**
2. Optimization Level: **Optimize most (-O3)**
Add Directories

1. Click on Directories
2. Browse to your Include folder
3: Browse to your Include folder
4. If relative path causes a crash, then uncheck the box
Add Libraries

1. Select "Libraries" in the toolchain settings.
2. Add your lib folder.
3. Add "ou_atmega2560"
Add Header Files

4: Add oulib.h. Add other header files here as well (as needed)
Now for the code...

```c
#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

![AVRISP mkII Configuration](image)

1. **Tool**: Select **AVRISP mkII**
2. **Device**: Select **ATmega2560**
3. **Interface**: Select **ISP**
4. **Apply**: Apply the selected settings
5. **ISP Clock**: Ensure the frequency is lower than 1/4 of the device's operating frequency
6. **Set**: Set the clock frequency to **2 MHz**

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

Bit-wise operators for digital input/output

Dr. Sesh Commuri is lecturing