TECH 3233
Lab #4a

Using what you saw in last week's code, you might have figured out that DDRB and PORTB were used to blink the LED. DDRB set the pins to either input or output, and PORTB sets the output status (high or low). Here is a full description of those registers:

18.4.3. Port B Data Direction Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: DDRB
Offset: 0x24
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x04

Bit	7	6	5	4	3	2	1	0
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DDRBn: Port B Data Direction [n = 7:0]

18.4.2. Port B Data Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PORTB
Offset: 0x25
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x05

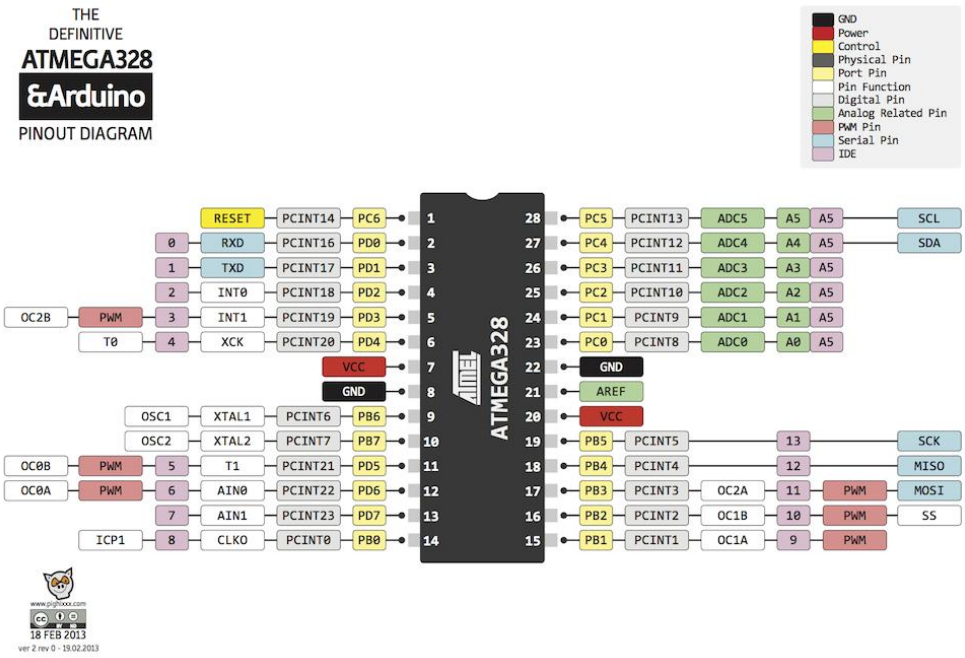
Bit	7	6	5	4	3	2	1	0
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – PORTBn: Port B Data [n = 0:7]

Write a program that will create the following sequence of lights on PB0..PB3

PB3	PB2	PB1	PB0
○	○	○	●
○	○	●	○
○	●	○	○
●	○	○	○

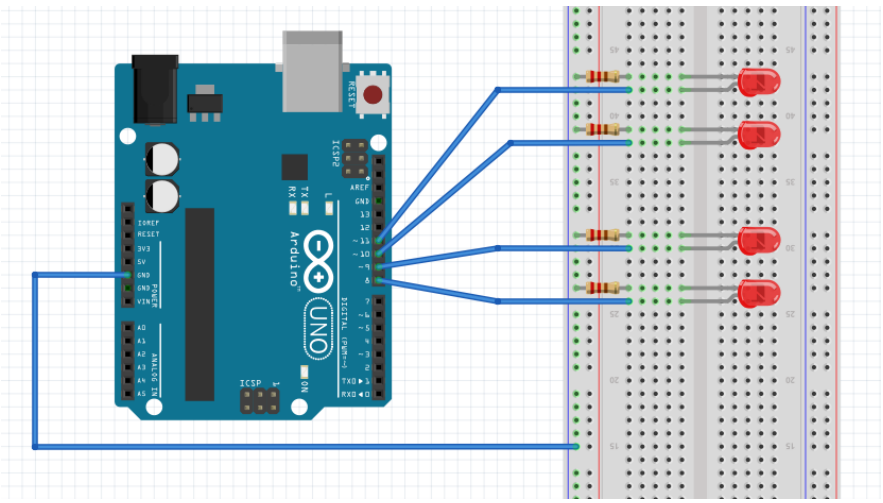
Once PB3 is on the next output should go back to PB0 being on. There should be a 300ms delay between each step. The program should repeat indefinitely. All outputs should be sent one at a time. Use an 8 bit (char) array to hold the values to be sent the port.



Remember that Arduino pins and Atmega pins are NOT the same. PB0 is on the IC on pin 14 but on the Arduino headers it is IO Pin 8 (for example).

To test, use four LED's and four 220Ω resistors. Connect the LED's + lead to Arduino Digital pins 8,9,10,11 respectively, connect each of the – leads to one side of the 220Ω resistor and connect the other side of the resistors to the any of the Arduino's ground pins.

So the circuit will look like:



[We will use a 220Ω sip for the resistors shown above (the common pin of the sip will go to ground]

Once the code works, transfer the program to the Arduino with the pre-built stepper motor driver to test its operation.

Demo the stepper motor driver to the instructor and submit the FULL PROJECT as a .zip file as Lab #4a.

Please make sure you **KEEP THIS PROGRAM**, the next lab will be to add functionality to the program to read switches to start/stop the motor, drive the motor forward / reverse, and to choose between half and full step mode.